



COST EFFECTIVE SENSOR PACKAGE FOR ROAD MONITORING

Benjamin Wehus Knutsen and Omer Zec



Supervisor

Kristian Muri Knausgård

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

Abstract

The companies responsible for road maintenance are expecting an increase in number of roads. It is of interest to find effective solutions to adapt to the increased work in road maintenance. Most of the road monitoring has to be done manually. This thesis is aimed to do research on how road inspection effectively can be monitored at a low cost. The goal is to create and implement a sensor package in a vehicular platform. The group has provided concepts, produced a sensor package for testing, software development and tested chosen areas of interest.

Traffic sign recognition, lane mark quality, vibration measurements and light intensity is researched and tested. Results are uploaded to a database with associated GPS coordinates and illustrated on a map service program. The maintenance staff can have access to the database and a complete overview of the detected states. The sensor package is designed for mounting in several vehicles, to avoid special made vehicles for road monitoring. *Robot Operating System* is used to launch and operate the road monitoring system. Most of the system software is written in *Python*.

The method for sign recognition with red color extraction gave positive results based on the testing. Lane mark measurement using a 2D camera is shown to work with correct installation. The communication for retrieving and writing data to geospatial database is shown to work properly. The results indicate that lightening conditions and other vehicles affect the results in a negative way. In addition, the results indicates that with further work on this topic, it is possible to monitor road conditions from a low cost and compact sensor package installed in a vehicular platform.

Acknowledgements

First of all the group would like to thank the supervisor, Kristian Muri Knausgård for help and guidance during this thesis. We are grateful for all the lab engineers at University of Agder for help during this spring.

Additionally the group is grateful for HK Motorservice AS for letting us borrow their car sporadically during the afternoons which allowed us for testing the sensor package.

Lastly the group will thank Nye Veier AS and Tor Alf Høyve, academic responsible at Nye Veier AS for meeting us and identifying their needs.

Benjamin W. Knutsen

Benjamin Wehus Knutsen

24.05.2019

Date

Omer Zec

Omer Zec

24.05.2019

Date

Contents

Contents	I
List of Figures	V
List of Tables	X
1 Introduction	1
1.1 Motivation	1
1.2 State of the Art	1
1.3 Problem Statement	2
1.4 Requirement and Standards	2
1.4.1 Illumination Requirements	3
1.4.2 Lane Mark Quality Requirements	4
1.5 Report Outline	4
1.6 Acronyms and Abbreviations	5
2 Theory	6
2.1 Software	6
2.2 Libraries	8
2.3 Hardware	9
2.4 Machine Vision	12
2.5 Thresholding	14
2.6 HSV, HSL and LAB Color Space	15
2.7 Canny Edge Detection	17
2.8 Hough Transform	19
2.9 Template Matching	21
2.10 2D Convolution	23
2.11 Sensors	23
2.11.1 Inertial Measurement Unit	23
2.11.2 Photoconductive Sensor	24
2.11.3 GPS Receiver	26
2.11.4 Infrared Temperature Measurement	27
2.12 Edge Computing Vs. Cloud Computing	28
2.13 GPU Vs. CPU Computing	29
2.14 Geospatial Information System	30
2.15 Diffuser for Light Sensor	30
2.16 Camera Calibration	30
2.17 Thermal Camera Calibration	34
2.18 IMU Calibration - Intel Real Sense D435i	34

3	Concept Development	35
3.1	Product Specification	35
3.2	Concept Generation	36
3.2.1	Concept 1 - Standard Road Detection with Temperature Measuring	37
3.2.2	Concept 2 - Hybrid Camera with Light Intensity Measurements	38
3.2.3	Concept 3 - Hybrid Camera with Light and Surface Temperature Measurements	39
3.3	Concept Evaluation & Selection	40
3.4	Structure & Shape Variations	43
3.4.1	Power Supply	43
3.4.2	Mounting Location Temperature Sensor	43
3.4.3	Mounting Location Light Intensity Sensor	44
3.4.4	Light Intensity Detection	44
3.4.5	Hardware	44
3.4.6	Temperature Sensor	45
3.5	Final Concept	46
4	Methods	47
4.1	System Design	47
4.1.1	Design Detailing	47
4.1.2	Heat Dissipation Up2	52
4.1.3	Sensor Package Frame	54
4.1.4	Power Supply	56
4.2	Data Flow	57
4.3	Sensor Package System	58
4.3.1	Create Workspace	58
4.3.2	Create Package	59
4.3.3	Camera Node	59
4.3.4	Sensor Node	61
4.3.5	GPS Subscriber Node	63
4.3.6	IMU Subscriber Node	64
4.3.7	Database Communication	65
4.3.8	Dropbox Communication	67
4.3.9	System Launch	69
4.4	Calibration	70
4.4.1	Camera Calibration - Intel RealSense D435i	70
4.4.2	Light Intensity Sensor Calibration	73
4.4.3	IMU - Intel Real Sense 435i	77
4.5	Electrical Connections	79
4.6	Speed Limit Sign Recognition	80
4.7	Lane Mark Quality	86
4.7.1	Determine Lane Mark Length	92
4.7.2	Alignment Error on Line Length Measurement	97

4.8	Lane Curvature, Curve Radius and Vehicle Position	99
4.9	Light Intensity Detection	106
4.10	Prototype Testing	109
4.11	Vibration Measurements	110
4.12	Communication in ROS	112
5	Results	115
5.1	Traffic Sign Recognition	115
5.2	Vibration Measurements	117
5.3	Database	118
5.4	QGIS	119
5.5	Dropbox	121
5.6	Lane Mark Quality	123
5.7	Light Intensity Detection	126
5.8	Lane Curvature	129
6	Discussion	130
6.1	Bright Conditions	130
6.2	Light Intensity Sensor	132
6.3	Lane Curvature	132
6.4	Camera Position Error	133
6.5	Power Supply	134
6.6	IMU Coordinate System	135
6.7	Machine Learning Vs. Template Matching	136
6.8	Light Intensity Sensor - GPS Problem	136
6.9	ROS, QGIS and PostgreSQL	136
6.10	Lane Mark Detection	136
6.11	Dropbox	137
6.12	GPS Antenna	137
7	Conclusion	138
8	Suggestions for Further Work	139
8.1	Speed Limit Sign Recognition	139
8.2	Camera	139
8.3	Hardware	139
8.4	Light Intensity Algorithm	139
8.5	Touch Monitor for Up2	140
8.6	Lane Mark Quality	140
8.7	Improve Algorithm for Bright Conditions	140
	Bibliography	141
	Appendices	146

A Cost	147
B Solidworks Drawings	148
B.1 Sensor Package	148
B.2 Frame for Power Supply	154
C Gantt Chart	158
D Verification of Test Setup During Length Measurement	159
E Data Sheets	162
E.1 Intel RealSense 435i	162
E.2 MLX 90640 - Thermal Camera	162
E.3 Photocell	163
E.4 GPS Sensor	164
E.5 Intel Up Ai Squared Computer	167
E.6 Arduino Uno R3	169
E.7 Biltema Power Supply	178
F Python Scripts	188
F.1 Finding Lane Curvature, Curve Radius and Vehicle Position	188
F.2 IMU Calibration Script - Intel RealSense 435i [1]	203
F.3 Traffic Sign Detection Script	220
F.3.1 Speed Limit 100	220
F.3.2 Speed Limit 90	223
F.4 Lane Mark Detection Script	227
F.5 Vibration Measurement Script	230
F.6 Upload to Dropbox Script	232
F.7 Communication Arduino and ROS	233
G ROS - Launch File	235
H MatLab Scripts	236
H.1 Script for Reading ROS bag files	236
I Arduino Scripts	239
I.1 GPS and Light Intensity Script	239

List of Figures

1.1	Tunnel Broken Down Into 4 Parts [2]	3
2.1	ROS Overview [3]	7
2.2	Intel UP Squared AI Vision X Developer Kit	9
2.3	Arduino Uno R3	10
2.4	Intel RealSense D435i	11
2.5	Image Acquisition [4]	12
2.6	Original Image	14
2.7	Grayscale Image	14
2.8	Binary Image	14
2.9	Hue - Color Representation [5]	15
2.10	RGB Image	15
2.11	HSV Image	15
2.12	HSL - Color Space	16
2.13	LAB - Color Space	16
2.14	Non Maximum Suppression [6]	17
2.15	Determine Edge with Hysteresis [7]	18
2.16	Canny Edge Detection - Illustration	18
2.17	Polar Coordinates [8]	19
2.18	Illustration of the Function [8]	20
2.19	Template Matching - Illustration	21
2.20	2D Convolution [9]	23
2.21	Valence and Conduction Band [10]	24
2.22	Change in Resistance Vs. Change in Lux [11]	24
2.23	Circuit of the Voltage Divider [11]	25
2.24	Trilateration - GPS Positioning [12]	26
2.25	Infrared Thermometer [13]	27
2.26	Edge and Cloud Computing Visualized [14]	28
2.27	CPU Vs. GPU Cores [15]	29
2.28	From 3D World Coordinates to 2D Image Coordinates [16]	30
2.29	The Pixel Skew Illustration [17]	31
2.30	Tangential Distortion [17]	32
2.31	Radial Distortion [17]	32
2.32	Reprojection Error of Images [18]	33
3.1	Concept Phase	36
3.2	Concept 1 - Dashboard View	37
3.3	Concept 2 - Dashboard View	38

3.4	Concept 3 - Dashboard View	39
3.5	Final Concept [19]	46
4.1	Intel Real Sense D435i [20]	47
4.2	MLX90640 Thermal Camera [21]	48
4.3	Joby Suction Cup	49
4.4	NEO 6M - GPS Receiver	50
4.5	Photocell	50
4.6	Up2 Dimensions	52
4.7	Temperature Rise Above Ambient [22]	53
4.8	Base Frame	54
4.9	Top Cover	54
4.10	Bracket for Light Intensity Sensor	55
4.11	Base Frame with Hardware	55
4.12	Sensor Package - Assembly	55
4.13	Step-Down Module - Inputs and Outputs	56
4.14	Frame for Power Supply	56
4.15	Illustration of the Data Flow	57
4.16	Calibration Setup	70
4.17	Calibration Grid	70
4.18	Intel RealSense Dynamic Calibrator - Start Up Screen	71
4.19	Intel RealSense Dynamic Calibrator - Initialization Step	71
4.20	Intel RealSense Dynamic Calibrator - RGB Camera	72
4.21	Intel RealSense Dynamic Calibrator - Depth Camera	72
4.22	Certified RS Light Meter	73
4.23	Setup - Light Intensity Calibration	73
4.24	Measurements - Graphical Illustration	74
4.25	Measurements - Logarithmic Conversion	75
4.26	Position 1 - Upright Facing Out	77
4.27	Position 2 - USB Cable Up and Facing Out	77
4.28	Position 3 - Upside Down Facing Out	78
4.29	Position 4 - USB Cable Down and Facing Out	78
4.30	Position 5 - Facing Down	78
4.31	Position 6 - Facing Up	78
4.32	Calibration Script - Command Prompt	78
4.33	Circuit Diagram	79
4.34	Speed Limit Signs	80
4.35	Flowchart - Traffic Sign Recognition	80
4.36	Speed Limit Sign Recognition Test Setup	81
4.37	Threshold for Red Circles	82
4.38	HSV Color Range [23]	83
4.39	Bounded Box around Circles	83
4.40	Template Score and Match	85

4.41	Flowchart - Lane Mark Quality	86
4.42	Field Of View - From E18	87
4.43	Lane Mark Dimension - E18 with Speed Limit Over 90 $\frac{km}{h}$	87
4.44	Warped Frame	88
4.45	Binary Output	89
4.46	Warped Output	90
4.47	Test Setup Line Length	92
4.48	Geometrical Model to Determine Line Length	92
4.49	Floor Level 1	93
4.50	Floor Level 2	93
4.51	Camera Level 1	93
4.52	Camera Level 2	93
4.53	Binary Image - Line Length Test 1	95
4.54	Binary Image - Line Length Test 2	95
4.55	Result - Length Test 1	96
4.56	Result - Length Test 2	96
4.57	Camera Angle at 0 Degrees	97
4.58	Result with Different Angles	97
4.59	0 Degree Angle	98
4.60	2.5 Degree Angle	98
4.61	5 Degree Angle	98
4.62	Library Function - Display Images	99
4.63	Library Function - Image Warping	100
4.64	Input Image	100
4.65	Warped Image	100
4.66	Library Function - HLS Thresholding	100
4.67	HLS Image and the Binary HLS Image	101
4.68	Library Function - LAB Thresholding	101
4.69	LAB Image and the Binary LAB Image	101
4.70	Library Function - LAB Combined with HLS Thresholding	102
4.71	Warped RGB Image and the Combined Thresholded Image	102
4.72	Library Function - Sliding Window Method Part 1	103
4.73	Library Function - Sliding Window Method Part 2	103
4.74	Sliding Window Method - Output	104
4.75	Library Function - Polynomial Previous Fit	104
4.76	Polynomial Previous Fit - Output	105
4.77	Curve Position - Function	105
4.78	LDR Circuit Wiring	106
4.79	Making Diffuser	106
4.80	3D Model of Diffuser Frame	107
4.81	Diffuser Frame, Diffuser Dome and Photo Cell	107
4.82	Assembled Diffuser	107

4.83	Test Setup	109
4.84	Cameras Coordinate System	110
4.85	Measurement from Car with Soft Suspension	111
4.86	Measurement from Car with Stiff Suspension	111
4.87	ROS Environment	114
5.1	Detected 90 Sign Nedenes-Grimstad	115
5.2	Detected 90 Sign Grimstad-Nedenes	115
5.3	Detected 50 Sign After Testing	116
5.4	Detected 100 Sign After Testing	116
5.5	Detected Peaks in Z-direction	117
5.6	Detected Peak	117
5.7	Pushed Road Condition Data to PostgreSQL	118
5.8	Pushed Sign Data to PostgreSQL	118
5.9	90 Sign Illustrated in QGIS	119
5.10	IMU Peaks Illustrated in QGIS	119
5.11	Information about GPS Point	120
5.12	Uploading Detected Images to Dropbox	121
5.13	Uploaded Detected Images to Dropbox Folder	121
5.14	Detected Images to Dropbox From another Computer	122
5.15	Lane Mark Detection Between Grimstad and Kristiansand	123
5.16	Shadow from Lamp Post Crossing the Lane Mark	124
5.17	Lane Mark Detection - Shadows from Vehicles	124
5.18	Lane Mark Detection - Worn Lane	125
5.19	Test Setup - Heavy Rain	126
5.20	Test Route	126
5.21	Light Intensity Measurements - Rainy Weather	127
5.22	Test Setup - Sunny Weather	128
5.23	Light Intensity Measurements - Sunny Weather	128
5.24	Final Output - Lane Detection	129
6.1	Dark Frame - Not Detected Sign	130
6.2	Detected Sign	130
6.3	Lane Mark Detection - Worn Lane	131
6.4	Diffuser with Conical Shape	132
6.5	Coordinate System for Object and Camera	133
6.6	Voltage Converter from Biltema [24]	134
6.7	Power Supply with Ignition Off	134
6.8	Z-Axis for Vertical Measurement	135
6.9	Y-Axis for Vertical Measurement	135
C.1	Gantt Chart - Part 1	158
C.2	Gantt Chart - Part 2	158

D.1	Height Table	159
D.2	Height Box	160
D.3	Height Camera	160
D.4	Length Long Line	161
D.5	Length Short Line	161

List of Tables

1.1	Dimension Class for Illumination [25]	3
1.2	Average Illumination Requirements in Lux [25]	3
1.3	Requirements Illuminance tunnels [2]	4
1.4	Description of Acronyms and Abbreviations	5
2.1	Intel UP Squared AI Vision X - Specifications [26]	9
2.2	Cloud Computing	28
2.3	Edge Computing	28
3.1	Evaluation - Functionality	41
3.2	Evaluation - Cost	41
4.1	Cost for Consept 3	51
4.2	3D Printer Specifications	55
4.3	Measurements - Numerical	74
4.4	Measurements - Numerical [log]	75
A.1	Cost	147

1. Introduction

1.1 Motivation

Today most of the road monitoring is manually inspected. Human inspectors drive in their inspection vehicles and report areas that are not sufficient. This way of road inspection is inefficient and time consuming.

It is of interest to apply disciplines within machine vision, automation and digitization for efficient road monitoring and data collection. There is placed stationary road monitoring devices along the road today, but these sensors provide information from the stationary point. The solution is to implement a compact sensor package on a vehicular platform that can monitor the road in real time and send information to a database, where the owners of the road can access the stored information.

1.2 State of the Art

In paper [27] authors have developed a sensor package on a vehicular platform for road maintenance during winter. The main purpose of the sensor package is to support roads or highways operators. This platform allows real time exchanges between the maintenance vehicles and the operating server in order to improve the maintenance and inform clients in real time about the road condition. In the research article [28], an automatic visual inspection system using *onboard in-car* camera is developed. The paper presents the method for detecting potholes, pavements and lane quality on the road using a camera. When the unaccepted states occurs the camera marks the spot on the road describing the fault.

The papers [27] and [28] are compact sensor packages that can be mounted inside a vehicular platform. In other papers the focus is on one topic using one sensor. In paper [29], a temperature sensor is used for measuring road surface temperature on a vehicular platform. This product indicates the surface temperature while driving and it is mounted outside the vehicle. In paper [30] the author presents how salt concentration is measured on the highway roads. A refractometer is used to probe the salt concentration. The refractometer gives a statement about the salt concentration on the road. Author in paper [30] does not use a vehicle when measuring the salinity, but in paper [31], it is shown that a refractometer can be used behind the rear bumper, where the water from the roads splashes in a refractometer. Other sensors that are made for road maintenance are the sensors provided by *Pavemetrics*. The sensor package features terrain models, detection of road edges, lane markings and curbs. The equipment uses dual laser profiles to map the road surface and the road shape can be inspected.

In the papers above, two papers include road maintenance as a package with several sensors. In paper [27], there is used temperature, humidity sensor, camera and other heterogeneous sensors. In the other references and research papers there is used individual sensors to detect the road quality on a vehicular platform. When companies and contractors provide road maintenance services, the focus is scoped to one topic. The sensors used to provide information about single topics are custom designed and using components with high accuracy to ensure better results. This also results in expensive equipment and does not make it a low cost sensor package for vehicles patrolling on the highways.

1.3 Problem Statement

The number of highways in Norway increases and due to this the road maintenance increases. By digitizing the road inspection and monitoring, it decreases cost and increases the efficiency for maintaining the roads. The interest is how the road monitoring can be digitized and the goal is to develop a cost-effective sensor package that can be installed in several vehicles which is already trafficking the roads.

Examples on conditions that can be monitored and/or do research at are listed below:

- Speed Limit Sign Recognition
- Quality of the Lane Markings
- Road Condition
- Light Intensity
- Detect Road Surface Temperature
- Detect Lane Curvature

When a sensor detects a state, a probe of the state is documented by the sensor package and sent to a map service application. Therefore a task of this project is to investigate how the sensor package can communicate with the map service program.

Which sensors and methods can monitor and detect these conditions? The group should provide concepts, system design, develop a prototype and also show test results from a chosen area of focus.

1.4 Requirement and Standards

In this section the requirements for the roads that Nye Veier is responsible for is presented. The relevant roads are highways with speed limits 90, 100 and 110 $\frac{km}{h}$. The roads with speed limits at 100 $\frac{km}{h}$ are classified as H8 and H9, while roads with speed limits 90 $\frac{km}{h}$ are classified as H5. As a new manual is being released, the new classification is H2 and H3 for the roads with speed limit 90 and 110 $\frac{km}{h}$.

1.4.1 Illumination Requirements

By driving in the dark, the risk of accidents is 1.5 to 2 times higher than driving in the daylight. These statistics are for serious accidents and young drivers. The illuminance on the highways is to help drivers holding the course steady and handle unexpected situations in a best way. [25]

The dimension class table for illumination is illustrated in table 1.1. Since the relevant roads are roads with middle railings and the average daily traffic is more than 4000 vehicles, the class is MEW3.

Table 1.1: Dimension Class for Illumination [25]

	ADT < 1500	ADT 1500 - 4000	ADT > 4000
Roads with middle railing		MEW3	MEW3
Roads with speed limit over $40 \frac{km}{t}$	MEW4	MEW3	MEW2
Roads with speed limit $30 \frac{km}{t}$		CE3	CE3

Table 1.2: Average Illumination Requirements in Lux [25]

Average luminance in $\frac{cd}{m^2}$		2	1.5	1	0.75	0.5			
Class	CE0	MEW1 CE1	MEW2 CE2	MEW3 CE3 S1	MEW4 CE4 S2	MEW5 CE5 S3	S4	S5	S6
Average luminance in Lux	50	30	20	10	7.5	5	3	2	

As seen from table 1.2, the average luminance in the roadway has to be minimum 20 Lux.

Requirements for tunnels are different. If the tunnels are longer than 100 meter it is required to have illuminations. The tunnels can be broken down into 4 parts, which is shown in Figure 1.1.

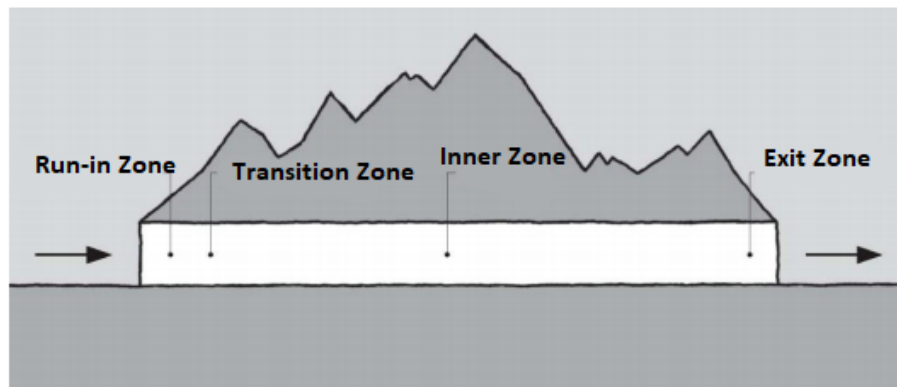


Figure 1.1: Tunnel Broken Down Into 4 Parts [2]

In table 1.3 the requirements for illuminance in the tunnels are shown. For roads with a speed limit of $110 \frac{km}{h}$ the lux needs to be minimum 1 between 00:00 A.M and 05:00 A.M, else 2 Lux. During the day the Lux needs to be minimum 4 at the inner zone. The *run-in* zone needs to be at least 7% of the adaption luminance. By adaption it means the ability for the eye to adjust to various levels of light. The adaption luminance is the average luminance of objects and surfaces in the vicinity of the observer estimating the visual range. [32]

Table 1.3: Requirements Illuminance tunnels [2]

ADT Speed Limit	<4000		4000 - 12000		>12000	
	$60 \frac{km}{t}$	$80 \frac{km}{t}$	$60 \frac{km}{t}$	$80 \frac{km}{t}$	$80 \frac{km}{t}$	$110 \frac{km}{t}$
Run-in zone	2%	3%	3%	4%	5%	7%
Inner zone day	$2 \frac{cd}{m^2}$	$2 \frac{cd}{m^2}$	$2 \frac{cd}{m^2}$	$2 \frac{cd}{m^2}$	$4 \frac{cd}{m^2}$	$4 \frac{cd}{m^2}$
All zones night	$1 \frac{cd}{m^2}$	$1 \frac{cd}{m^2}$	$1 \frac{cd}{m^2}$	$1 \frac{cd}{m^2}$	$2 \frac{cd}{m^2}$	$2 \frac{cd}{m^2}$
All zones night between 00 - 05 A.M.	$0.5 \frac{cd}{m^2}$	$0.5 \frac{cd}{m^2}$	$0.5 \frac{cd}{m^2}$	$0.5 \frac{cd}{m^2}$	$1 \frac{cd}{m^2}$	$1 \frac{cd}{m^2}$

1.4.2 Lane Mark Quality Requirements

The lane markings plays an important role regarding the traffic safety. There was not provided any requirements for lane mark quality. Therefore, during this project, it is categorized as good lane or bad lane. This is set by thresholds which can be modified for further requirements or standards that comes in the future.

1.5 Report Outline

This report contains 7 chapters. Chapter 1 includes introduction, problem statement and the state of art. Chapter 2 contains the theory which is used to solve the problems. Chapter 3 includes the concept phase which contain concept generation and concept selection. Chapter 4 includes the methodology that is used in this project. In chapter 5 the results is presented. In chapter 6 discussions about certain subjects and the groups opinion is presented. In chapter 7 the conclusion of the project is presented. Suggestions for further work is discussed in chapter 8.

1.6 Acronyms and Abbreviations

The acronyms and abbreviations mentioned in the report are listed below in table 1.4.

Table 1.4: Description of Acronyms and Abbreviations

Acronyms and Abbreviations	Description
up2	Intel Up AI Squared - The computer
LDR	Light Dependant Resistor
IMU	Inertial Measurement Unit
GPS	Global Positioning System
ROS	Robot Operating System
MEW	Illumination Classes for roads with speed limit over $40 \frac{km}{h}$
CE	Illumination Classes for roads with speed limit at $30 \frac{km}{h}$
S	Illumination Classes For pedestrian and cycle paths
Lux	Unit of illumination
GIS	Geospatial Information System
OpenCV	Open Source Computer Vision
API	Application Programming Interface

2. Theory

In this chapter the needed theory to solve the problems is presented. In addition theory and information about the products and software which was used is included as well.

2.1 Software

Python

Python is a programming language for developers, and it is an open source programming language made to be *easy-to-read* and powerful. In Python there is no compilation step and it is easy to debug. Python is a high-level programming language and because of its simplicity, writing scripts are less time consuming than in other languages. When Python was created, the inspiration came from programming languages such as C and C++. Python was written in the programming language C. [33]

Robot Operating System

Robotic Operating System, referred to as *ROS* is a system which contains a number of independent nodes. Each node can communicate with each other. For example, one node could be a camera and another node could be a script for using camera to detect objects. The camera subscriber script subscribes to the camera publisher node to receive a video frame. The nodes are independent and does not need to have same programming languages to run.

There is a ROS master node and ROS nodes, the ROS master node handles the communication between nodes. If for example "*node1*" requests information from "*node2*" the ROS master node makes the communication go through. The way these nodes are communicating is publishing and subscribing to topics. If a node is a subscriber to another node it receives information from it. Then the subscriber node requests information from the publisher node.[3] In Figure 2.1 a basic overview over the ROS master and ROS nodes is illustrated.

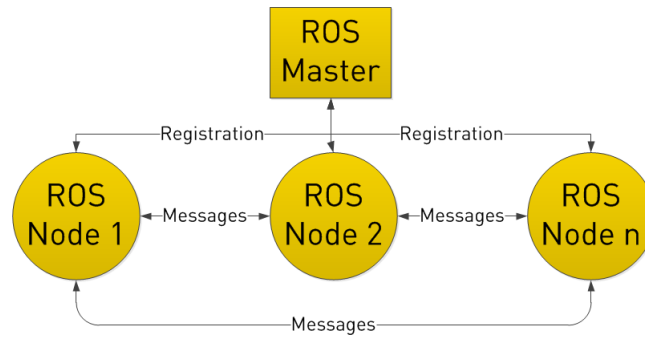


Figure 2.1: ROS Overview [3]

Quantum GIS

Quantum GIS known as QGIS is an open source geographic information system application that gives users the ability to view, edit and analyze geospatial data. QGIS supports vector layers, where the vector data is either stored as a point, polygon or line features. QGIS has integrated software support for PostGIS, MapServer and GRASS. For more flexibility QGIS is equipped with C++ and Python to execute individual operations on the geospatial data. The application is written in C++ and Python, and the user interface is developed in Qt. The application can be installed on any device that operates Microsoft Windows, Linux, macOS or UNIX. [34]

PostgreSQL

PostgreSQL, known for being referred to as Postgres is an open source object-relational database management system. Postgres has focus on extensions and standards compliance. Postgres can handle workloads on local memory, but it handles larger tasks such as internet-facing applications and data warehousing. To show an example, Postgres is the default database on the macOS Server. The Postgres database system supports a large amount of extensions, one extension that is important for this project is PostGIS. This extension allows Postgres to store geographic information data types. The data can then be imported to a GIS application as a layer and view the geospatial information. There is a long list of interfaces for Postgres, in this project psycopg2 was used for Python interface. Postgres has built-in support for processing languages. Processing languages give the developer opportunities to extend and modify databases for own specifications. There are three supported languages in Postgres, and those are SQL, PL/pgSQL and C. The first two procedural languages are safe to use, but C is not safe. It is experienced that functions written in C have the best performance, but bugs can occur and this results a crash and corrupting the database. [35]

2.2 Libraries

Open Source Computer Vision

Open Source Computer Vision, OpenCV, is an open source computer vision library. The library consists of different algorithms that perform given tasks within image processing and machine vision. Examples of tools found in the OpenCV library is edge detection by using the canny edge method, line identifications by using Hough Transform, converting images to different types such as grayscale and HSV images. This is a few examples of what the library has available. OpenCV supports the common programming languages such as C++, Python and Java. [36]

Dropbox API

Dropbox is a file hosting service and can be synced with computers, cell phones and other devices. To use Dropbox as an image storing place, the Dropbox application has to be downloaded on the local computer. It is not possible to upload images to Dropbox from scripts without downloading and including the Dropbox SDK for a specific programming language. The images are stored in desired folders on the local computer and when the internet is available, the images are automatically synced to the Dropbox.

Psycopg2

Psycopg2 is a library that establishes communication between Python scripts and PostgreSQL. This library ensures to access different features for handling data between scripts and databases. Psycopg2 is made for Python scripts only and cannot be used in other programming languages. Other programming languages has own libraries for communication with PostgreSQL.

2.3 Hardware

Intel UP Squared AI Vision Kit

Intel UP AI Vision Kit is a compact computer which is designed for computer vision related work. This computer includes pre-installed software and comes with Ubuntu operating system. In addition, it includes the library OpenVINO, which is an AI based library that uses pre-trained models to detect different objects in a camera frame. This hardware is made for advanced applications with edge computing. The kit comes with a USB camera with resolution up to 1920p x 1080p at 30 FPS. [26] In table 2.1 the specifications of the computer are presented. In Figure 2.2 the kit is shown.



Figure 2.2: Intel UP Squared AI Vision X Developer Kit

Table 2.1: Intel UP Squared AI Vision X - Specifications [26]

VPU	Intel Movidius Myriad X
Graphics	Intel HD Graphic 505
System Memory	8GB LPDDR4
Storage Capacity	64GB eMMC
Connectivity	WiFi (option), LTE (optional)
Video Output	HDMI and Displayport
Power input	5V at 6A

Arduino Uno

Arduino Uno is a microcontroller that is operated by its own software. The Arduino software is used to write code to the Arduino microcontroller. Arduino is an open source platform and uses a simplified version of C++ as a programming language, which makes this a vast platform for sensor applications. Since there is high number of Arduino users, developers have made wrappers for using other programming languages to operate the Arduino platforms. The Arduino Uno is equipped with an ATmega328 microcontroller, 14 digital I/O and 6 analog I/O. The Arduino Uno is powered by a computer or an external battery source. In Figure 2.3 the Arduino Uno R3 is shown.[37]



Figure 2.3: Arduino Uno R3

Intel RealSense D435i

Intel RealSense D435i is the latest model of the D400 series. The camera is a depth camera which can calculate depth by using stereo vision. In addition the camera is equipped with an inertial measurement unit. The camera produces high resolution images in RGB and depth. The camera delivers depth images with maximum resolution 1280 x 720 and RGB images with maximum resolution 1920 x 1080 pixels. [20] In Figure 2.4 the camera is shown.



Figure 2.4: Intel RealSense D435i

2.4 Machine Vision

Machine vision is the process of applying algorithms and methods to an image, so the computer gets the ability to *see*. Machine vision is used in industrial applications such as pattern recognition, hand writing recognition, object detection etc. Machine vision can be broken down into the 5 steps which is described below.

Image Acquisition

The first stage of vision algorithms is the image acquisition. This converts from analog to digital. It is required to perform before other operations and algorithms are applied. In Figure 2.5 a graphical illustration of the image acquisition process is shown.

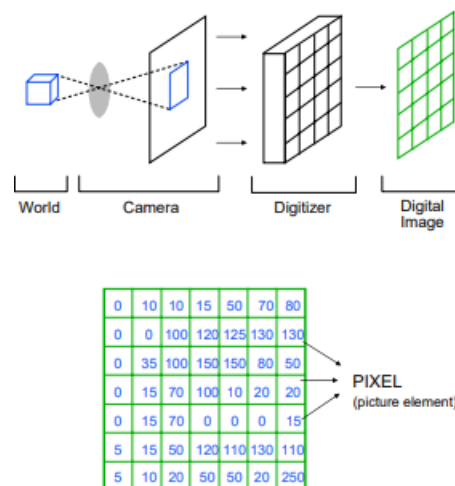


Figure 2.5: Image Acquisition [4]

Image Processing

Image processing is a way to perform operations at an image. Reducing noise and improving contrast are examples of image processing. The purpose of this step is to extract useful information by suppressing unnecessary information. By removing unnecessary information, the computer needs less power to compile the program. There are two types of image processing, digital and analog. Analog image is for example a photography, and digital image is the image the computer sees. Digital image processing is the most relevant. With image processing the images are represented as 2D signals. Reasons for performing image processing are for example: [38]

- Improve Regions of Interest
- Improve Quality of an Image
- Reduce Noise
- Remove Unnecessary Information of an Image

Image Segmentation

Image segmentation is the process where the image is divided into several segments, which is several sets of pixels. The purpose is to simplify or amplify for example changes in an image which makes it easier to analyze. Image segmentation is used for locating lines, boundaries etc. For example, Canny Edge detection algorithm. Image segmentation is used to identify objects in an image. The most common image segmentation is the threshold method [39]. Thresholding is further explained in chapter 2.5.

Image Analysis

Image analysis means extracting useful information of an image, it is the ability to recognize attributes in the image. For example, taking measurements of objects or relationships, reading a QR-code or recognizing a face of a human.

Pattern Recognition

Pattern Recognition is an automated process which recognize patterns in an image. Machine learning and AI is relevant subjects for pattern recognition. The pattern recognition is to verify for example if the face or code from an image analysis is the correct one.

2.5 Thresholding

The input image to a threshold operation can for example be a grayscale or a color image. The output image after a threshold operation is for example a binary image. In a binary image the black pixel represent the background while the white pixels represent the foreground. In an implementation such as intensity threshold, each image pixel is compared to the threshold value. If the intensity of the pixel is higher than the threshold, the pixel is white, and if the pixel intensity is below the threshold value then it is black.

Thresholding is widely used within machine vision to remove unnecessary information. In Figure 2.6 the original image is shown. In Figure 2.7 the original image is thresholded into grayscale and in Figure 2.8 it is further thresholded into binary.



Figure 2.6: Original Image

Figure 2.7: Grayscale Image

Figure 2.8: Binary Image

2.6 HSV, HSL and LAB Color Space

HSV, HSL and LAB are three different color spaces which are used to represent an image. By changing the color space in an image, it can help extracting useful information. HSV, HSL and LAB are further explained below. [40]

HSV is short for hue, saturation and value. Hue is the color model which goes from 0 to 360 degrees which determine the portion of a certain color. In Figure 2.9 it is illustrated which color that belongs to what value. Saturation is the amount of gray in the color and can be adjusted from 0 to 100%. Value goes from 0 to 100% and is the brightness.

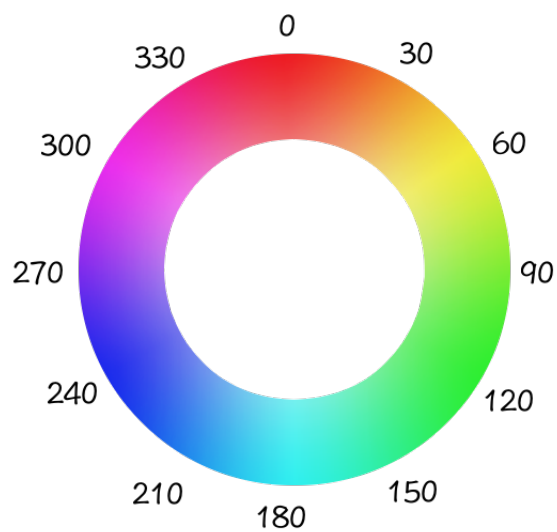


Figure 2.9: Hue - Color Representation [5]

In Figure 2.10 and 2.11 the same image is illustrated in RGB and HSV.



Figure 2.10: RGB Image



Figure 2.11: HSV Image

HSL is short for hue, saturation and lightness. Hue is the same as in HSL as for the HSV color model. S is the saturation of the color and L is the lightness of the color.

In Figure 2.12 the same image as shown in Figure 2.10 is illustrated in HSL color space.



Figure 2.12: HSL - Color Space

LAB is a three axis color space. *L* is short for lightness, which has a range from 0 to 100, where 0 is black and 100 is white. *A* goes from cyan(-100) to magenta(100). Cyan and magenta is 2 different colors, cyan is a light blue color and magenta is a light pink color. *B* is short for blue(-100) to yellow(100). [41]

The same RGB image as shown in Figure 2.10 is illustrated in the LAB color space in figure 2.13.



Figure 2.13: LAB - Color Space

2.7 Canny Edge Detection

Canny edge detection is an algorithm for detecting edges in images. The algorithm consist of 5 steps which is further explained below [42] [6] :

1. **Apply Gaussian filter.** In order to reduce false lines the noise has to be filtered out. This is done by a Gaussian filter. The size of the Gaussian kernel will give different results for edge detection.
2. **Find intensity gradients of the image.** Since an edge in an image can be pointed in any direction, a sobel filter is used to detect the horizontal and vertical direction. Furthermore, the first derivative in horizontal direction (G_x) and in the vertical direction (G_y) is found. From this the edge gradient and direction for each pixel is found. The gradient is perpendicular to the edges, so θ can be four different angles. In Equation (2.1) the formula for edge gradient is shown, and in Equation (2.2) the formula for calculating the direction is shown. This stage of the the algorithm amplifies the changes in an image.

$$G = \sqrt{(G_x)^2 + (G_y)^2} \quad (2.1)$$

$$\theta = \tan^{-1} \left(\frac{G_x}{G_y} \right) \quad (2.2)$$

Where:

G: The gradient of the edge.

G_x : The first derivative in the horizontal direction.

G_y : The first derivative in the vertical direction.

θ : The direction of the edge.

3. **Apply non maximum suppression to remove outliers.** This stage finds the line with the highest gradient. As seen in Figure 2.14, point A is on the edge. Point B and C are on the direction of the gradient, but not on the edge. Then point A is tested against point B and C to see if they create a local maximum. If that is the case it can proceed to the next stage, otherwise it is suppressed.

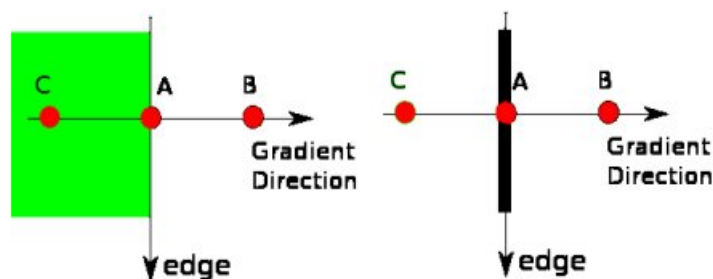


Figure 2.14: Non Maximum Suppression [6]

In the end the output is a binary image with thin edges, it is illustrated in the right image in Figure 2.16.

4. **Apply double threshold to locate lines.** Then apply a low and a high threshold. If an edge pixel gradient is higher than the high threshold it is marked as a strong edge. If the edge pixel gradient is between the higher and lower threshold it is marked as possible edge. If the edge pixel gradient is below the low threshold it gets suppressed. For visualization see Figure 2.15.

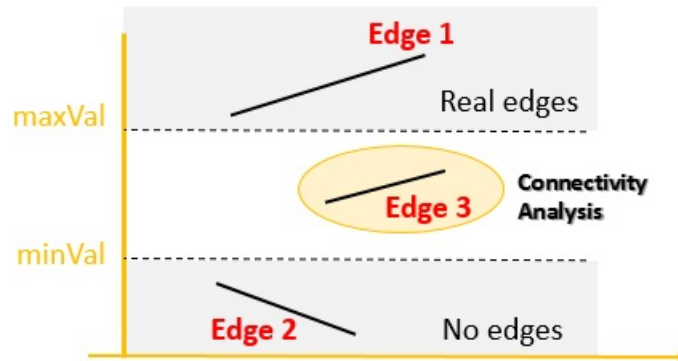


Figure 2.15: Determine Edge with Hysteresis [7]

5. **Locate edges by using hysteresis.** Then the strong edge pixels are in the final image. The weak edge pixel is in the final image only if the previous pixel is a part of the line.

In Figure 2.16 an illustration of the Canny Edge detector is shown.

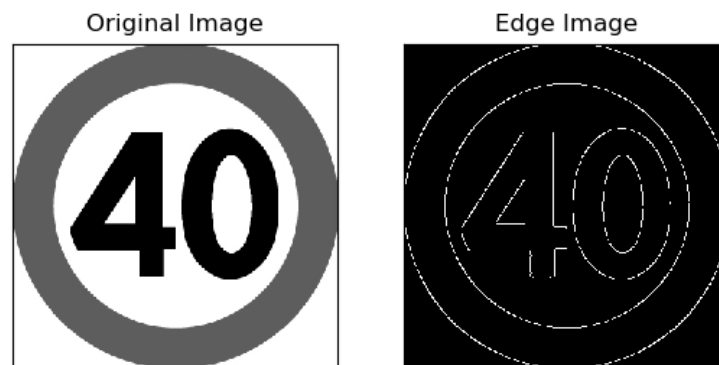


Figure 2.16: Canny Edge Detection - Illustration

2.8 Hough Transform

Hough Transform is a technique used in image processing to detect lines and identify shapes. Hough Transform is a way of finding the values that indicate a line, circle or other parametric curve. The first step in Hough Transform is to detect all the lines in an image. The Canny Edge detector is used. Furthermore, the resulted edge detection is the input for the Hough Transform algorithm. Lastly a set of pixels describes the boundary to an object.

The general formula for a line is described in Equation (2.3).

$$y = ax + b \quad (2.3)$$

The problem with the general formula is that the vertical lines can not be described, therefore it is described by polar coordinates which is shown in Figure 2.17. The distance from the origin and to the point P is r , and θ is the angle of the vector.

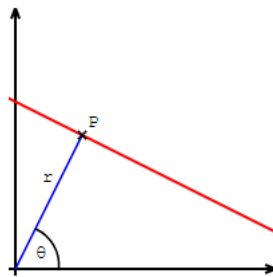


Figure 2.17: Polar Coordinates [8]

Further the line is described in polar coordinates and is shown in Equation (2.4). Then by rewriting and solving for "r" as shown in Equation (2.5).

$$y = - \left(\frac{\cos \theta}{\sin \theta} \right) \cdot x + \left(\frac{r}{\sin \theta} \right) \quad (2.4)$$

$$r = x \cdot \cos \theta + y \cdot \sin \theta \quad (2.5)$$

For example, for an arbitrary point P in an image, to get all possible lines through this point, then the line has to be rotated from 0 to 180 degrees. In Figure 2.18 an illustration is shown. In the left side of the Figure the line will be rotating around the point P . On the right side of the Figure the corresponding visualization of r and θ . The result will be a sinusoidal function. When the two points are on the same line the trajectories (Red and green sinusoidal) will cross each other. The r and θ in this intersection is the exact description of the line.

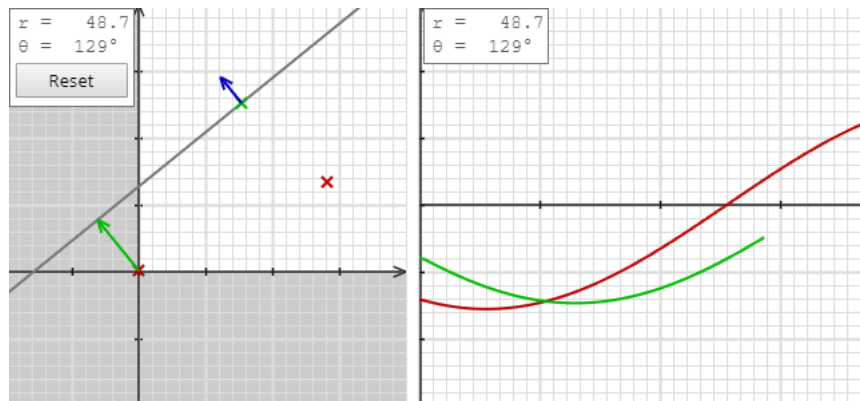


Figure 2.18: Illustration of the Function [8]

Hough transform summarized, draw sinusoidal trajectories for the points. Then find the highest number of crossed trajectories in a single point. This crossing describes the line.[8]

2.9 Template Matching

Template matching is an image processing technique which makes it possible to find objects or small parts in an image. It searches over the image to find similarities between the template and the input image. In other words, it slides the template over the source image and compares the pixels, this technique is called 2D convolution which is further explained in section 2.10. After choosing comparison methods and run the algorithm it returns a gray scale image where each pixel expresses how much the neighbor of the same pixel match the template. A certain threshold has to be set manually and if the result of template matching is greater than the threshold, it is a match [43]. In Figure 2.19 an example is illustrated where the template is located in the upper left corner. If it finds the area which is higher than a threshold, a rectangle is drawn around the area which matched.



Figure 2.19: Template Matching - Illustration

There are 3 main template matching algorithms. *Sum of squared difference*, *cross correlation* and *correlation coefficient matching*. These 3 methods can be normalized which means that in theory there are 6 different methods. By normalizing, the result will be more accurate but requires more computation. Below the 3 main techniques are further described. [44]

The sum of square difference. It squares the difference between the image point and the template point. The ideal result would be 0 which is a full match, and the bad matches will approach to 1. In Equation (2.6) the formula for this technique is shown.

$$R_{sq}(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2 \quad (2.6)$$

Where:

$R_{sq}(x, y)$: The result of the template match.

$T(x', y')$: The image pixel value in the template image.

$I(x, y)$: The image pixel value in the source image.

Cross correlation. This technique relies on multiplication. The template is multiplied with the image. Therefore, a perfect match would be 1, and 0 means no match. The template goes over all pixels position and multiply the pixel value at the template with the pixel value of the source image. This is done for all the pixels in the source image. Lastly it sums all the products to get a result between 0 and 1 for the template match. In Equation (2.7) the formula for this technique is shown.

$$R_{cc}(x, y) = \sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))^2 \quad (2.7)$$

Where:

$R_{cc}(x, y)$: The result of the template match.

$T(x', y')$: The image pixel value in the template image.

$I(x, y)$: The image pixel value in the source image.

Correlation coefficient matching. This is a technique where it check the match between the mean of the template and image relative to its own mean value. For this reason a perfect match would be 1 and a mismatch would be -1. In Equation (2.8) the formula for this technique is shown.

$$R_{coeff}(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))^2 \quad (2.8)$$

Where:

$R_{coeff}(x, y)$: The result of the template match.

$T'(x', y')$: The image pixel value in the template image.

$I'(x, y)$: The image pixel value in the source image.

The disadvantages of template matching is rotating and scaling. If the template matching is performed and the template has wrong image size it would be difficult to find the object. Therefore to perform image operations before applying template matching, it can improve the result.

2.10 2D Convolution

In image processing 2D convolution is used in the algorithms. For example, in blurring, sharpening and edge detection. It is a way of multiplying two arrays of numbers by help of neighborhood operation. For example, if you have a mask and an original image. The mask goes over the image and then calculates a new value. The mask depends of what operation that is executed. In Figure 2.20 an example of 2D convolution is illustrated.

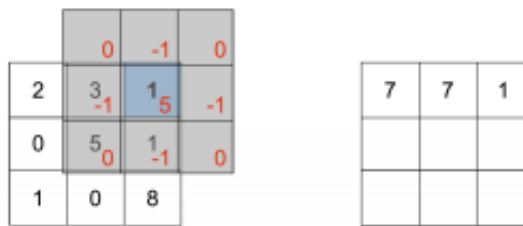


Figure 2.20: 2D Convolution [9]

2.11 Sensors

2.11.1 Inertial Measurement Unit

The inertial measurement unit, referred to as *IMU*, consists of a cluster of sensors. Accelerometer, gyroscope and magnetometer. These sensors are mounted to a common base to maintain the same relative orientations. The accelerometer measures the force it experiences in x , y and z directions due to gravity. The gyroscope measures the angular velocity along x , y and z axis. The magnetometer is measuring the magnetism. With help of the earth magnetic field it measures the orientation.[45]

2.11.2 Photoconductive Sensor

The photoconductive sensor is used in applications where light measurements are involved. Photoconductive sensors are made of semiconducting materials and have high resistance. The working principle is by photoconductivity, which is an optical phenomenon, where the materials conductivity is reduced when light is absorbed by the material. In short, when the light is absorbed by the photoconductive sensor the light energy forces the electrons to jump from the valence band to the conduction band as shown in Figure 2.21. This operation reduces the conductivity of the material and lowers the resistance in an electrical circuit. When identifying the resistance in the photoconductive sensor, the light energy can be determined. [10]

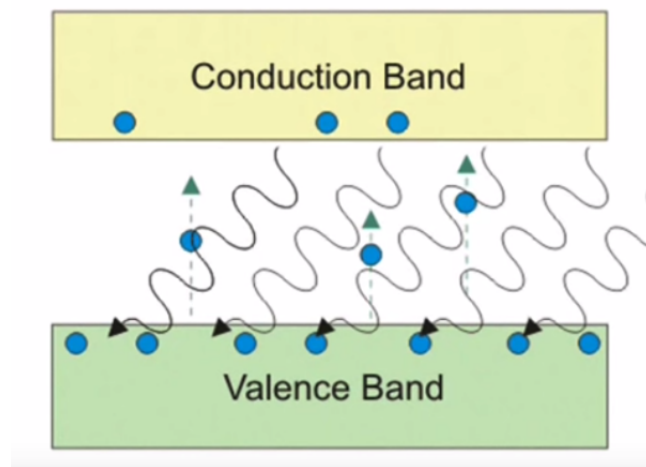


Figure 2.21: Valence and Conduction Band [10]

The most common photoconductive sensor is a *Light Dependant Resistor*. The change in the electrical energy is directly related to the change in the light intensity. The LDR is made of the semiconducting material *cadmium*. The electrical resistance will due to absorbed light change from over thousands of ohms in the dark to hundreds of ohms when the light occurs. In Figure 2.22 the change in resistance and the change in lux is illustrated for better understanding. [11]

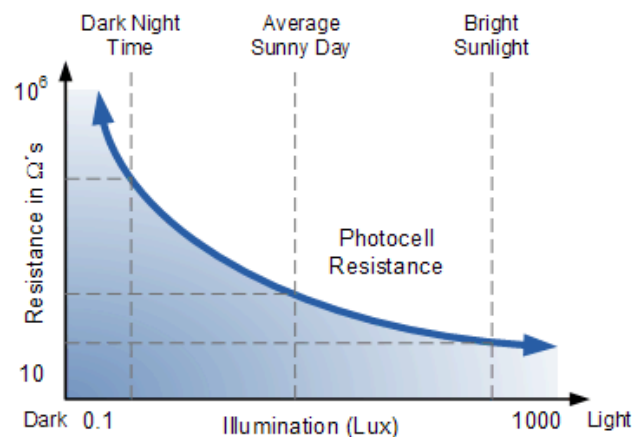


Figure 2.22: Change in Resistance Vs. Change in Lux [11]

There are different circuits including the LDR which can be used to measure the light intensity. One example is the *Voltage Divider*. The circuits are built up by a resistor with a permanent resistance, a variable resistor and a DC supply voltage. A big advantage of this is that different voltage appears at the intersection for different levels of light. An illustration of this circuit is shown in Figure 2.23. [11]

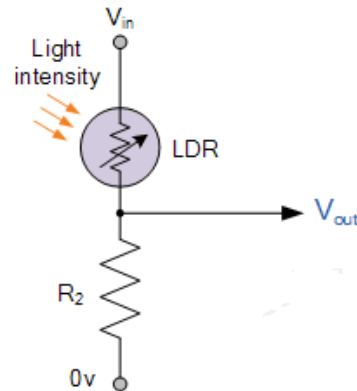


Figure 2.23: Circuit of the Voltage Divider [11]

In Equation (2.9) the formula for calculating the voltage out of the circuit is shown.

$$V_{Out} = V_{In} \cdot \frac{R_2}{R_{LDR} + R_2} \quad (2.9)$$

Where:

V_{Out} : Voltage output.

V_{In} : DC supply voltage.

R_2 : A resistor. [Ohm]

R_{LDR} : A Variable resistor / LDR. [Ohm]

2.11.3 GPS Receiver

GPS is short for *Global Positioning System*. GPS system is a satellite-based navigation system which consist of 24 satellites. At any location on the earth four satellites are detectable by the GPS receiver. By the *Time-of-Flight* principle it can calculate how far away the receiver is, but to measure this the GPS demands very accurate timing.

The last factor that must be compensated for is the delays the signal experiences as it travels through the atmosphere. The clock difference between the GPS receiver and the satellites must be compensated for. To determine the position of the GPS receiver it needs information from at least three satellites. Using trilateration, the position can be determined. By locating where the three circles intersect the position can be determined. The more detected satellites the more accurate the position will be. A graphical explanation of the trilateration is presented in Figure 2.24. [12]

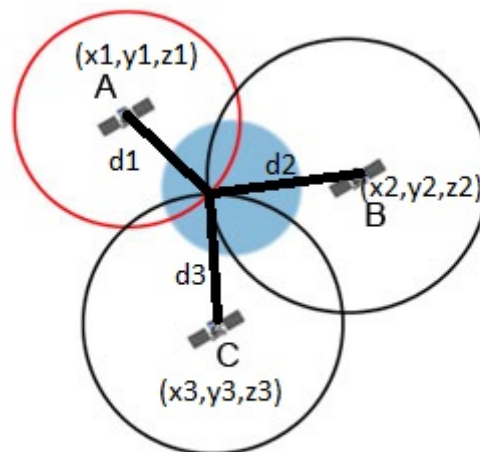


Figure 2.24: Trilateration - GPS Positioning [12]

2.11.4 Infrared Temperature Measurement

An infrared thermometer consists of a lens to gather the emitted energy and a detector that converts energy to an electrical signal. The heat is transferred between objects through convection, conduction or radiation. Most radiation is in the infrared spectrum of the electromagnetic specter. Infrared energy is emitted from object surfaces and the energy from the infrared takes part in the electromagnetic spectrum. Infrared radiations are represented as wavelengths and it is strongest between 0.7 and 14 microns. The infrared thermometer measures the wavelength emitted from the object surfaces and estimates the surface temperature. It is worth to mention that the red dot pointer from the infrared thermometer only indicates the center of measured circle on the object. The lens in the thermometer gather the radiation to the detector, which converts the radioactive power to an electrical signal. This can be displayed in temperature units. It has to be compensated for ambient temperature. The functionality of the IR temperature sensor is shown in Figure 2.25.[13]

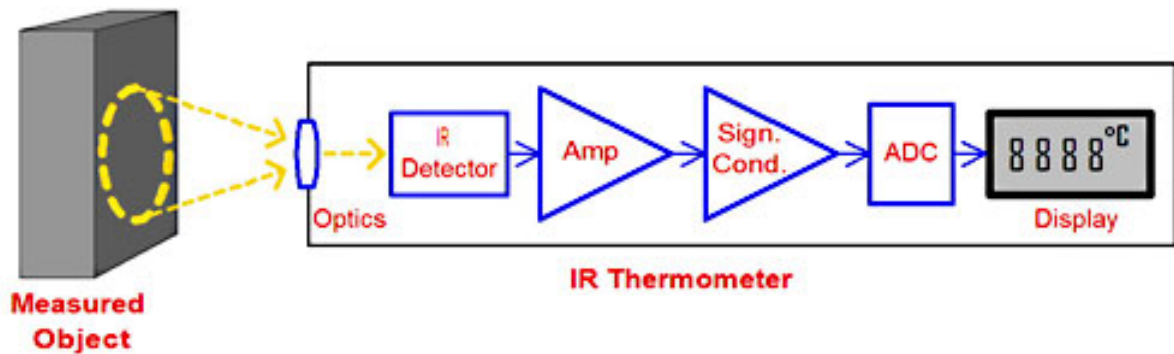


Figure 2.25: Infrared Thermometer [13]

2.12 Edge Computing Vs. Cloud Computing

With edge computing, it allows data to be processed where the data is created or collected, instead of sending the data across long routes to data centers or clouds. By computing the data at the edge, it allows data analysis real-time, a need for companies and organizations in the automotive and maintenance industry. Cloud computing makes computer system resources such as storage and computational power available in the cloud. The purpose of cloud computing is to send data to the cloud, where computation is performed in a data center with powerful computers. When the data is processed, it is pushed to a desired location. It is necessary with internet connection when performing cloud computing. In Figure 2.26 an illustration of edge and cloud computing are shown for easier understatement. In tables 2.2 and 2.3 the advantages and disadvantages of cloud and edge computing are listed.

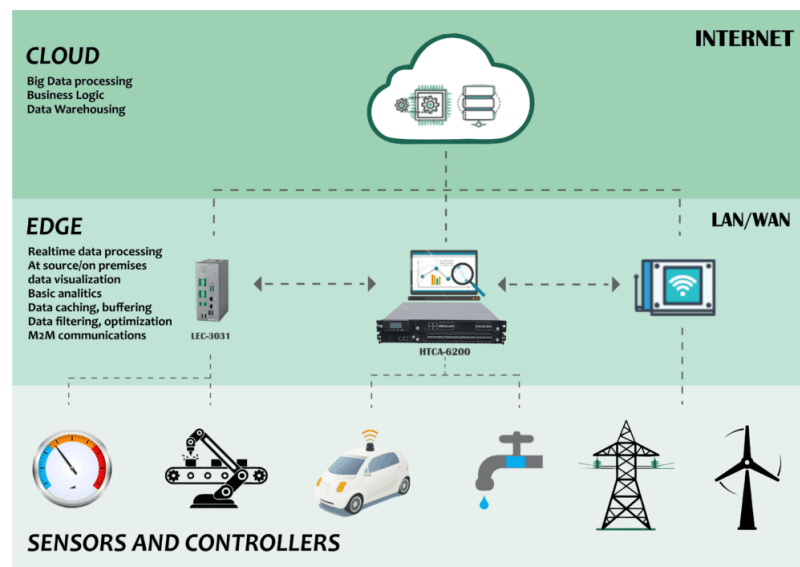


Figure 2.26: Edge and Cloud Computing Visualized [14]

Table 2.2: Cloud Computing

Advantages	Disadvantages
Cost Savings	Downtime
Reliability	Security
Manageability	Vendor Lock-In
Strategic Edge	Limited Control

Table 2.3: Edge Computing

Advantages	Disadvantages
Low Latency	Potential Loss or Corruption of data
Real Time Availability	Longer Outage Time
Real Time Data Transmission	Higher Risk
Increased Productivity	Hardware

2.13 GPU Vs. CPU Computing

Computers are equipped with microprocessors for handling different operations. The computers have central processing unit (CPU) and graphics processing unit (GPU). The two processors are similar in look and they are both made of silicone-based materials, thus they have different roles in a computer. A CPU is known as the brain of the computer, the CPU's task is to manage the running programs on the computer. A CPU can perform calculations, actions, and run programs. Examples where CPUs can perform without problems are, low resolution applications such as MS PowerPoint, Skype and Google Chrome.

The first CPUs came with a single core, this resulted in limited task operations and slow computing. Since the technology has improved, modern computers come with quad-cores, some are provided with octa-cores. Higher number of cores gives the computer increased computational power, and it can perform multiple tasks simultaneously. When there is assigned a higher number of tasks, the CPU reaches a limit, and the computer is occurring to lag, therefore the GPUs are invented.

The GPU is designed for complex mathematical and geometrical computations. A GPU can handle thousands of tasks simultaneously due to the thousands of cores that it contains. The GPUs are known for performing difficult computational tasks such as training models for machine learning, rendering images and computer games. In addition, the GPU works as an accelerator for the CPU. In this project, the amount of data processed is vast. For example, the camera is set to deliver 30 frames per second, this means that the processing unit receives 30 images per second, and perform image processing (mathematical operations) over 30 images in that second. Since the CPU is not designed for a high number of complex mathematical calculations simultaneously, it will not be able to process all the delivered data, and a GPU should be included in the computer that is used for this project. [46]

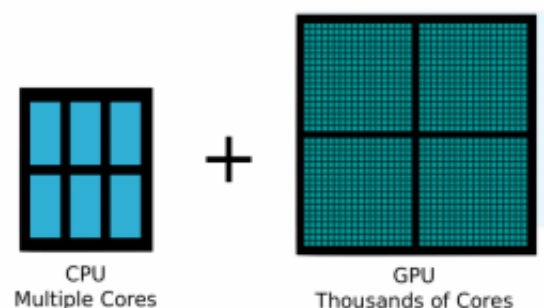


Figure 2.27: CPU Vs. GPU Cores [15]

2.14 Geospatial Information System

GIS is a system for capturing, storing and analyzing data. Geospatial information systems are softwares used for storing data and geographic data more effective. It usually consists of coordinates, longitude and altitude values, street addresses or zip codes. In other words, any information which ties it to a specific location at earth. [47]

2.15 Diffuser for Light Sensor

A diffuser is made of a material that diffuses the light, so the light sensor receives ambient light. By using a diffuser, a more realistic impression of the light surroundings is established. By not using a diffuser the light sensor is detecting concentrated light and the output of the sensor will vary and is not realistic. Therefore, when measuring the intensity of the light it is necessary to implement a diffuser on the light sensor.

2.16 Camera Calibration

To ensure that the camera accuracy is sufficient the camera needs to be calibrated. If the camera is *out of the box*, calibration should not be necessary, unless otherwise is mentioned.

When the camera is calibrated the intention is to estimate the intrinsic camera parameters. In Figure 2.28 the homography is illustrated, which shows the relationship of 3D world coordinates and 2D image pixel coordinates. As shown the intrinsic parameters is necessary and a poor calibration results in deviation in the world coordinates.[16]

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates
Intrinsic properties (Optical Centre, scaling)
Extrinsic properties (Camera Rotation and translation)
3D World Coordinates

Figure 2.28: From 3D World Coordinates to 2D Image Coordinates [16]

The intrinsic parameters describe the optical, geometric, and digital characteristics of the camera. The intrinsic parameters are defined by the transformation between the camera frame and the pixel coordinates. The intrinsic parameters are represented in a intrinsic matrix, K , which is shown in Figure 2.10. [17]

$$K = \begin{bmatrix} f_x & \gamma & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.10)$$

Where:

f_x : Focal length in x -direction.

f_y : Focal length in y -direction.

C_x : The horizontal displacement of the image from the optical axis in mm .

C_y : The vertical displacement of the image from the optical axis in mm .

γ : The skew angle.

The focal lengths are measured in pixel and is equal if the image pixels are squares. The formula for focal length in x and y directions is shown in Equation (2.11) and (2.12). [17]

$$f_x = \frac{F}{S_x} \quad (2.11)$$

$$f_y = \frac{F}{S_y} \quad (2.12)$$

Where:

F : Is the focal length in mm . The focal length is the distance between the center of the lens and to the surface of the imaging sensor.

S_x : Equals the horizontal size of a pixel in the camera sensor in pixels per mm .

S_y : Equals the vertical size of a pixel in the camera sensor in pixels per mm .

The skew angle is calculated with formula in Equation (2.13).

$$s = f_y \cdot \tan \alpha \quad (2.13)$$

Where:

s : The skew angle.

f_y : See formula in Equation 2.12.

α : The offset angle which is illustrated in Figure 2.29.

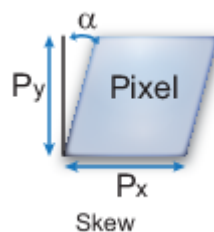


Figure 2.29: The Pixel Skew Illustration [17]

Most of the camera which is used in machine vision is a pin hole camera and an ideal pin hole camera model does not have a lens. Therefore, the camera matrix does not take distortion into account. To represent a camera as accurate as possible some small tangential and radial distortion can be compensated for. In Figure 2.30 is illustrated. The tangential distortion appears when the camera lens and the plane is not parallel.

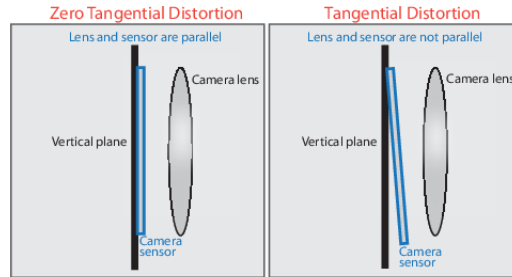


Figure 2.30: Tangential Distortion [17]

Tangential distortion can be compensated for by adding distortion coefficients to the pixel location (x, y) . The compensated point can therefore be expressed as (x_{td}, y_{td}) . The formula for compensation is shown in Equation (2.14) and (2.15). [17]

$$x_{td} = x + (2 \cdot p_1 \cdot x \cdot y + p_2(r^2 + 2 \cdot x^2)) \quad (2.14)$$

$$y_{td} = y + (p_1(r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y) \quad (2.15)$$

Where:

x and y : Undistorted pixel locations.

p_1 and p_2 : Distortion coefficients.

r^2 : $x^2 + y^2$.

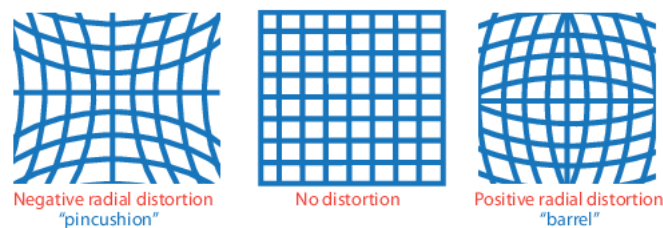


Figure 2.31: Radial Distortion [17]

Radial distortion is experienced when the light beams bends more at the edges of the lens than the center. This can be compensated for, the point that is compensated for the distortion can be expressed as (x_{rd}, y_{rd}) . [17]

$$x_{rd} = x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (2.16)$$

$$y_{rd} = y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (2.17)$$

Where:

x and y : Undistorted pixel locations.

k_1 , k_2 and k_3 : Distortion coefficients.

r^2 : $x^2 + y^2$.

When the camera is calibrated the accuracy can then be tested, for example by calculating the reprojection errors. Furthermore, to improve the intrinsic parameters, more images can be put into the calibration. Another improvement can be to exclude the image with high reprojection errors and then re-calibrate. If the calibration is done in *MATLAB*, the reprojection error of the images is illustrated in a graph which is shown in Figure 2.32. Then it can easily be seen which images that are above the mean value and can be replaced with new images to improve the result. [18]

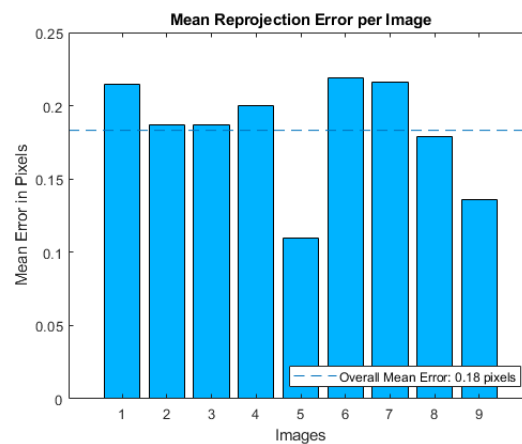


Figure 2.32: Reprojection Error of Images [18]

2.17 Thermal Camera Calibration

To calibrate an infrared thermometer a blackbody source can be used. This is a common method. A blackbody is in theory a perfect emitter, this means that it emits the maximum amount of energy no matter which temperature. In practice there is no perfect blackbody that exist, but the principle contributes a strong basis for the calibration. It radiates the same intensity of the radiations in all directions, which makes it a diffuse emitter.

Before the calibration a blackbody needs to be determined. It can be broken down into two infrared calibration sources, *Hot Plate* or *Cavity-Type* blackbody source. The *Cavity-Type* consist of a hole in a sphere. Further the temperature is controlled, and it is measured by a thermocouple probe. This type has a higher surface emissivity than the Hot Plate, with 0.98 or higher. The hot plate normally consists of an aluminum plate. Further the temperature of the plate is measured by thermocouple or an RTD probe.

Commonly the infrared camera is pointed into the source with a distance between $0.2m$ and $1m$. To keep in mind when choosing blackbody, the higher the target emissivity the higher accuracy. [48]

2.18 IMU Calibration - Intel Real Sense D435i

The intention to calibration the IMU is to get as correct measurements as possible. If the camera is been exposed for hard bumps it is recommended to calibrate. By calibrating the camera, the intrinsic parameters and extrinsic parameters are updated and wrote to the device. [49]

The intrinsic parameters include:

- The scale factor sensitivity accelerometer - This is a factor that is multiplied with the data to ensure a metric output
- Bias accelerometer - The bias will cancel any value so it would be zero, when the sensor should be reading so.
- Off axis terms accelerometer - This is a factor that is used to correct the axes if it is not orthogonal.
- Bias gyro - Cancel the value to be zero when the sensor should be reading so.

The extrinsic parameters include:

- Rotation - This is the rotation from the left camera to the IMU. 3x3 rotation matrix.
- Translation - Translation from the left camera and to the IMU.

3. Concept Development

The scope of this chapter is to create, detail and select a concept.

The concept phase investigates the different resources and details existing to establish and generate new concepts. These could be design specifications, cost, weight limits or application area. When a good overview is established, it is possible to generate a concept specification. The product in development must fulfill the criterion which is described in the section below. A total of 3 concepts was generated.

3.1 Product Specification

There are requirements that the product must fulfill. The product must be able to inspect the road and give information about the given conditions which are described in section 1.3. In addition, the product criterion is mentioned below:

- The sensor package must be able to be installed on a vehicular platform.
- The sensor package must be a low price product, with a compact design. The price for the product must not exceed 10 000 NOK. This price includes all components but not software development or working hours.
- The sensor package must be easy to install and mobile which means that it can easily put the sensor package into a new car without any competent staff.
- The system cannot consume more power than the 12V cigarette lighter can deliver.
- The data must be collected and be available for maintenance crew.

3.2 Concept Generation

In this section the general layout and logic of the concept phase is described. Every concept is equipped with a GPS sensor, hardware and a power supply. The product will not be able to detect the given condition during the wintertime or heavy rain. A graphical explanation of the concept phase is illustrated in Figure 3.1.

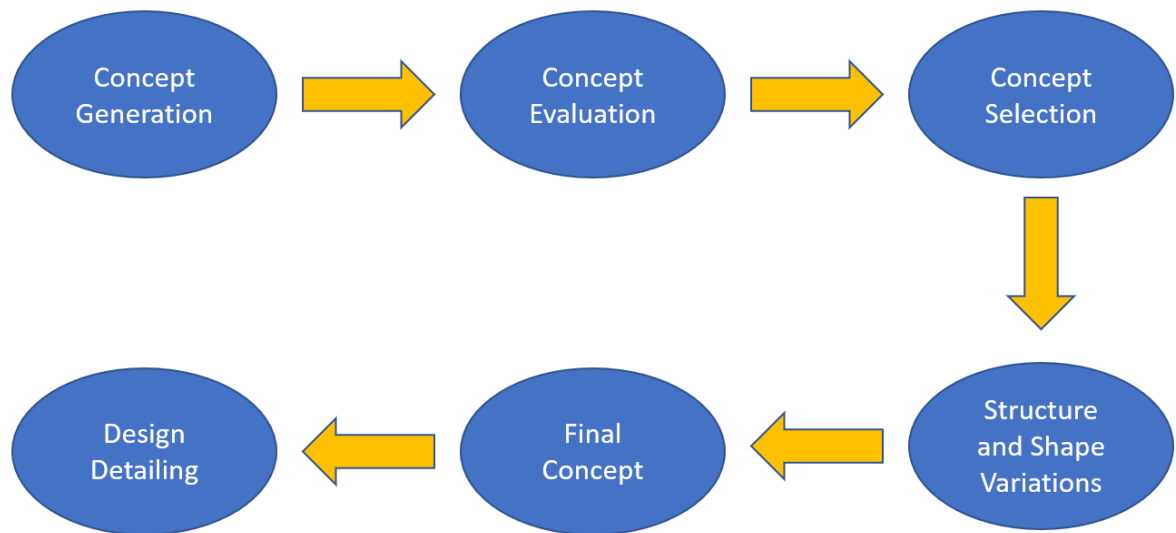


Figure 3.1: Concept Phase

3.2.1 Concept 1 - Standard Road Detection with Temperature Measuring

First concept includes one standard RGB camera. Concept 1 will not be able to perform edge computing, but will be able to:

- Detect speed limit signs
- Detect and evaluate the lane mark quality
- Vibration measurement
- Measure the surface temperature on the road

This concept will need an external IMU sensor for the vibration measurements.

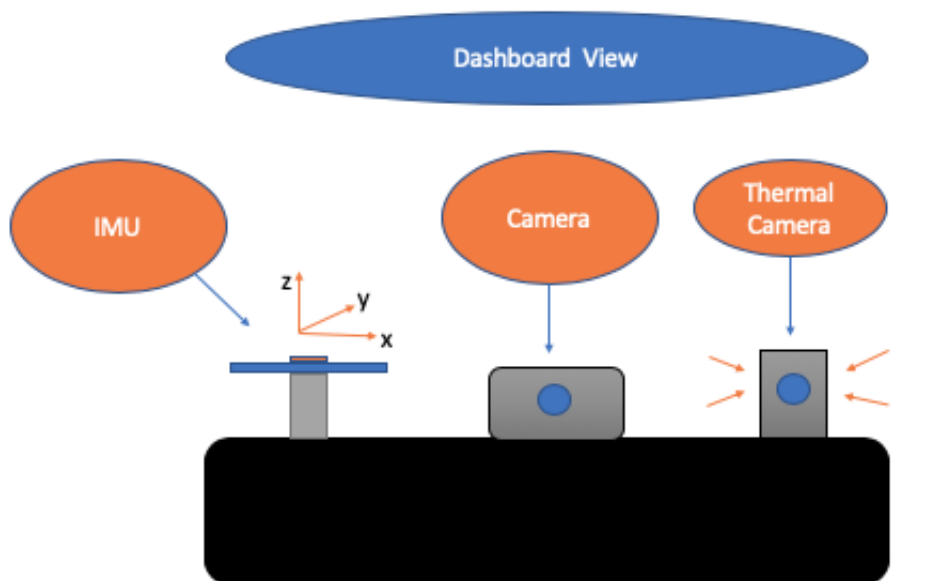


Figure 3.2: Concept 1 - Dashboard View

3.2.2 Concept 2 - Hybrid Camera with Light Intensity Measurements

The second concept includes a camera which contains one standard RGB camera and an integrated measurement unit. This sensor package performs edge computing, it is therefore necessary with an upgraded hardware performance. Concept 2 is capable to :

- Detect speed limit signs
- Detect and evaluate the lane mark quality
- Vibration Measurement
- Measure the light intensity

This concept does not need external IMU due to the hybrid camera.

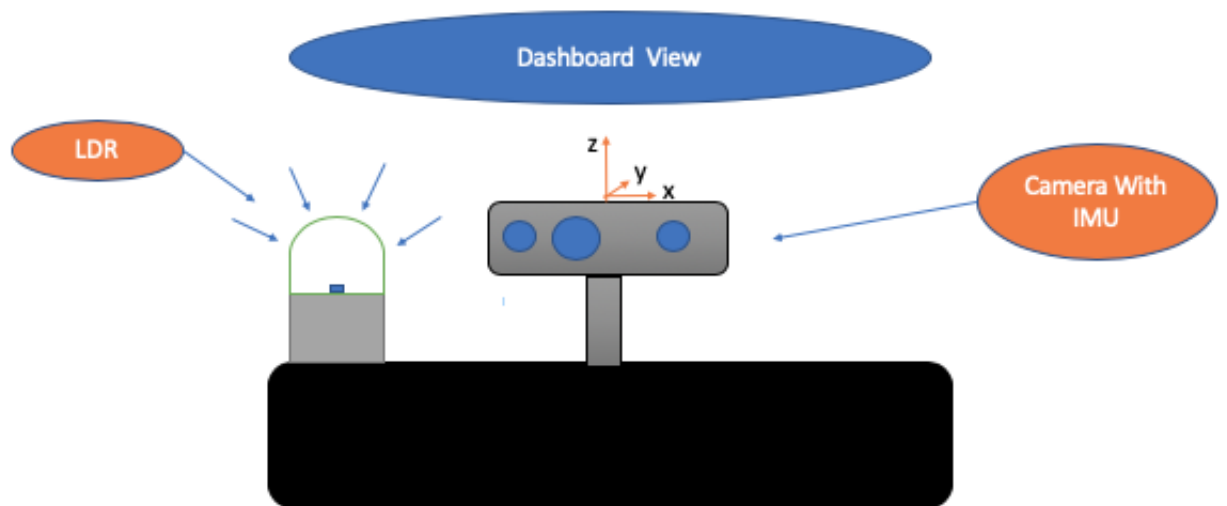


Figure 3.3: Concept 2 - Dashboard View

3.2.3 Concept 3 - Hybrid Camera with Light and Surface Temperature Measurements

The third concept includes one camera with integrated inertial measurement unit. This concept will be able to perform edge computing. This concept measures the road surface temperature and the light intensity from the dashboard. Concept 3 can provide information as following

- Detect speed limit signs
- Detect and evaluate the lane mark quality
- Vibration measurement
- Measure the surface temperature
- Measure the light intensity

This concept will detect all the necessary parameters as well as performing edge computing which makes this a complete package.

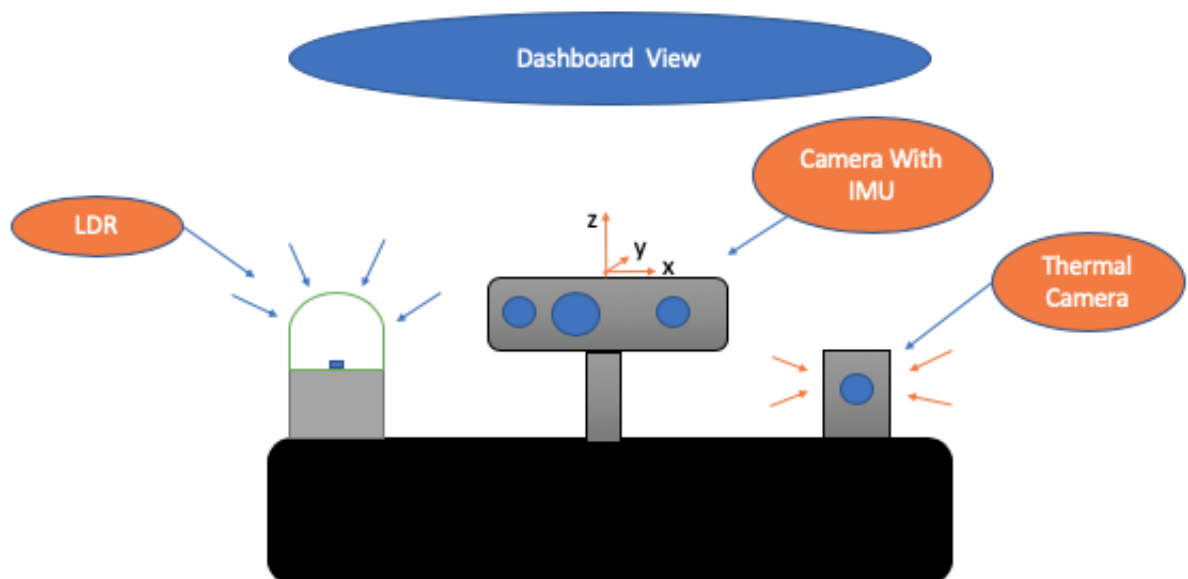


Figure 3.4: Concept 3 - Dashboard View

3.3 Concept Evaluation & Selection

Each concept was evaluated, and two tables were created. One for cost evaluation and the second for functionality. The evaluation topics are split into subcategories to estimate an overall result. This gives an indication for which concept is best suited for this application.

Concept 1 - Standard Road Detection with Temperature Measurements

Concept 1 is a road detection package with temperature and vibration measurements. This concept does not include edge computing which results in lower price due to hardware selection. Though, the concept includes temperature measurement of the surface which means that it can be challenging to measure it through the windshield. Alternatively, it can be mounted on a different location. In addition, this camera does not include IMU, therefore an external IMU is necessary.

Concept 2 - Hybrid Camera with Light Intensity Measurements

Concept 2 which contains the hybrid camera solution is a compact concept, where only one camera is used for RGB video and measurement from the integrated IMU. This setup does not require much space and electrical wiring, this makes the sensor package easy to mount and install in vehicles. The package can provide standard RGB video up to 30fps, vibrations from the road and the light intensity along the road and in tunnels. Another feature that can be provided is depth data, since the camera includes stereo camera.

Concept 3 - Hybrid Camera with Light and Surface Measurements

Concept 3 contains the same equipment as concept 2 plus the surface temperature sensor. This concept uses edge computing which leads to higher cost. Therefore, this concept is the most expensive concept, but will be the most complete sensor package with most detectable parameters.

Evaluation Tables

The concepts are evaluated in two tables, cost and functionality. The concepts are rated on a scale ranging from 1-10, where 10 is great functionality and low cost. Table 3.1 and 3.2 displays the evaluation table for functionality and cost.

Table 3.1: Evaluation - Functionality

Criterion	Concept 1	Concept 2	Concept 3	Ideal
Detectable Parameters	8	8	10	10
Complexity	9	8	7	10
Reliability	8	7	7	10
Physical size	9	9	9	10
Mobility	8	8	7	10
Sum	42	39	40	50
Relative Value	0.84	0.78	0.8	1

Table 3.2: Evaluation - Cost

Criterion	Concept 1	Concept 2	Concept 3	Ideal
Number of Components	8	8	6	10
Hardware	8	7	7	10
Installation	9	9	8	10
Sum	25	24	21	30
Relative Value	0.83	0.8	0.7	1

Concept Selection

Every concept has been evaluated based on the given criterion in the table 3.1 and 3.2. The table address is key properties of each concept and works as a guidance in the process of selecting a final concept.

The values in the table could give a false representation of the characteristics needed in the system. For instance, detectable parameters are more important than physical size. A lot of values are connected, like complexity and number of detectable parameters.

Concept 1 gets the highest overall score in the evaluation tables. This concept has scored better than the other 2 concepts, yet it cannot detect all parameters.

Concept 3 scored well at the functionality but not ideal for the cost. Since the cost is within the budget, concept 3 is chosen. In the next chapter the structure and shape variation of this concept is presented.

3.4 Structure & Shape Variations

In this section the structure and shape of concept 3 is considered.

3.4.1 Power Supply

The sensor package requires a power supply. It needs constant power and one of the options is the 12V contact in the car. By using the 12V contact from the car itself, an external battery should be used for a manual shutdown. This is to maintain a secure shut down for the device and to protect the data and the hardware.

The first concept is direct power from the car, an external battery is not required. The sensor package must therefore be shut down manually, before removing the key. The ignition can be still be off since the car is delivering power if the key is in the ignition.

The second concept is that the sensor package is directly coupled to an external battery. when the car runs, the battery is charged by the 12V contact in the car. This makes the sensor package independent and much more mobile.

3.4.2 Mounting Location Temperature Sensor

There are several locations where the temperature sensor can be mounted in the car. Simplicity of the installation and the mobility is the key requirement. It can be necessary to mount the sensor outside if the error by measuring through the windshield occur.

One possible mounting location is in the front of the car, behind the front bumper. The wiring can go through the firewall or directly from the fuse box in the engine compartment. Since the sensor is moved outside the cockpit it adds more complexity to the installation and relocation.

The second location could be at the left or right front fender. Fastened with a magnet which makes it easy to install and easy to clean the sensor. The wiring will have to go through the window or through the left mirror. The disadvantage is that the sensor is more exposed for the weather.

The third option is to mount the temperature sensor on the top of the sensor package frame. The sensor package is mounted at the dashboard. A challenge is to measure through the windshield due to heat vaporization from a warm engine.

3.4.3 Mounting Location Light Intensity Sensor

The light intensity sensor needs to be located where it easily can detect the light from the outside surroundings. As for the temperature sensor the installation and mobility are emphasized.

The first location is at the dashboard. This makes it mobile and does not require complex installation, since it is integrated at the sensor package frame.

The second location is outside, with a magnet on the hood. From that location it is highly exposed for the light which gives more accurate results. The disadvantage is that installation gets more advanced and the mobility is reduced.

3.4.4 Light Intensity Detection

A luxmeter uses a photodiode. A photodiode is a semiconductor that converts incoming light energy to electrical current. The sensor conducts current proportional to the amount of light that the sensor receives. Filters and build in lenses are providing the photodiode and outputs the light measurements in lux. [50]

Luxmeters comes in different shapes and sizes. Since our design must be compact, the group must provide a method for measuring the light intensity. Light intensity can be measured by using a light dependent resistor, as explained in chapter 2.11.2 under section "*Photoconductive Sensor*". A light dependent resistor is a photo conductive sensor that reduces its conductivity by absorbing light. This means if the sensor is placed in an electrical circuit, the resistance is high in dark and almost no current is flowing through the circuit. When the sensor is absorbing light, the resistance is reduced, and it allows current to flow through the circuit. The current can be measured in a micro controller and converted to a lux value. By performing this method, the light intensity can be detected and determined.

3.4.5 Hardware

The computer for the sensor package must be compact, cost effective and powerful enough to perform edge computing. There are 4 different hardware/microcontroller setups that are relevant for this project. Nvidia Jetson TX2 development kit, Intel UP AI Square kit, Raspberry pi and Arduino Uno. Either Jetson or Intel must be used to run the system, and Arduino or Raspberry Pi for operating the sensors.

Jetson TX2 is a powerful computer made for difficult tasks and application with required large amount of data processing. It is a AI computing device because of its power and efficiency. In addition, Nvidia has provided libraries for building AI applications with the Jetson hardware called JetPack. The disadvantage is that the original carrier board for the Jetson series requires relatively big space, which makes it difficult to make a compact solution for a reasonable price.

Intel UP Square Vision kit is a compact and powerful computer designed for computer vision and edge computing. The Intel UP shares similar performances as the Nvidia Jetson developer kit, but it has a more compact design and carrier board out the box. The AI computer from Intel has pre-installed computer vision network library, openVINO. The openVINO is a toolkit to make AI applications with the Intel and it has pre-trained models.

The Raspberry Pi could be an alternative to combine with the super computers Jetson or Intel up2. Arduino is a micro controller that can be used in the same way as the Raspberry pi. Both Arduino and the Raspberry Pi could run read and operate sensors like GPS, temperature and light intensity sensor. The larger computers Jetson TX2 or Intel UP square subscribe to the information from the micro controllers and use them in their favor.

3.4.6 Temperature Sensor

Surface road temperature describes the state condition of the road. When the surface temperature is known, the surface condition can be predicted. If the road temperature is decreasing towards 0 degree Celsius, ice is then formed, to avoid ice, de-icing trucks are informed, and they spread salt on the highways. There are different ways of measuring temperature. In this project, methods for non-contact surface measurement is considered.

Measuring temperature without contacting the surface can be performed with a thermal camera or an infrared temperature sensor. Both the thermal camera and the infrared sensor works on the same principle by reading the emitted infrared energy from the surroundings. The main difference is that the infrared temperature sensor measures one spot and outputs one number. The thermal camera in the other hand outputs a surface image and provides information over the capable measuring surface. By measuring spots, crucial information can be lost, and the true overall temperature of the road can be determined wrong. The accuracy of these two types of contact less measurements are similar due to the same working principle. Expected error is approximately 1 degree of Celsius of both thermal camera and the infrared sensor.

3.5 Final Concept

The final concept was generated after evaluating all the different layouts and structure variations. Concept that was chosen is concept 3 with further iterations. The mounting location for the light intensity sensor is in the dashboard integrated in the frame for the actual sensor package, this make the system less complicated and simplifies the installation. The temperature sensor will be mounted at the right or front fender for more accurate measurements. As shown in Figure 3.5 the location of the sensor package, camera, temperature- and light intensity sensor are illustrated.

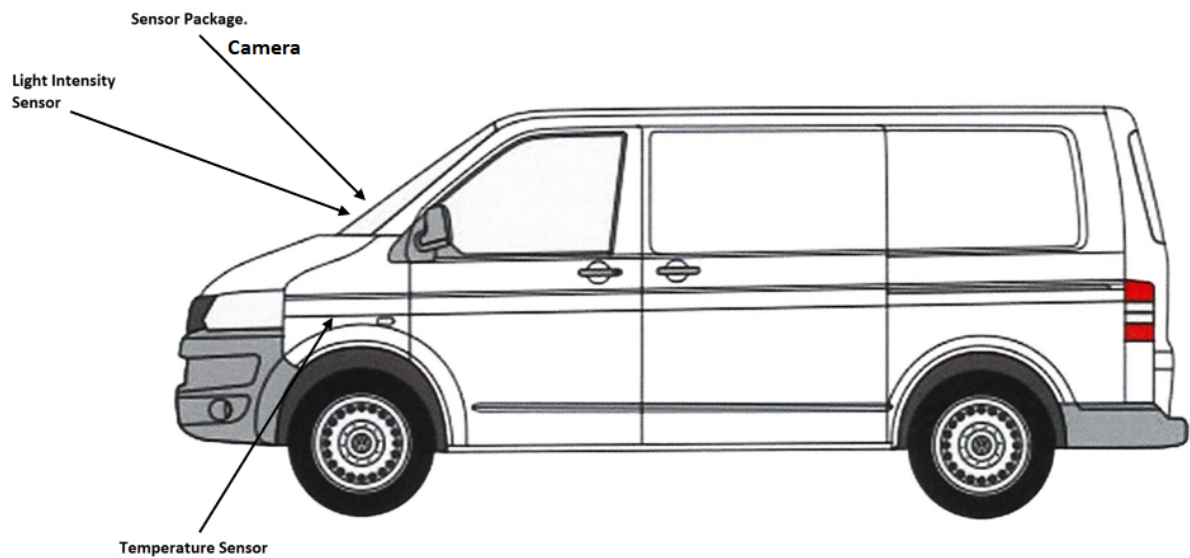


Figure 3.5: Final Concept [19]

The power supply which was chosen for the final concept is first concept which is direct power from the car but without external battery. Since the 12V contact still deliver power when the ignition is off it can still perform a safe shut down. The communication between the sensors and the sensor package is wired since most of the sensors are mounted at the actual sensor package. For detecting the light intensity, a photo conductive sensor is used, this is due to its simplicity and price. To measure the temperature a thermal camera was chosen.

Design detailing is presented in chapter 4.1.

4. Methods

In this chapter the methodology of the project is presented.

4.1 System Design

4.1.1 Design Detailing

The advantage of the chosen design is the simplicity and mobility as well as the number of detectable parameters. In this chapter the products used are shown, in addition the total cost for the final concept is presented. The computer is the main driver for the system, other sensor components used in this concept rely on the computational strength of the computer. If the computational power is low, all states of the road cannot be provided. Therefore, a powerful computer and a reliable power source is necessary.

Camera

The camera that was chosen is the *Intel Real Sense D435i*. The reason for selecting this camera was the price and various abilities and it is possible to implement with the computer from Intel. The camera offer resolution up to 1920p x 1080p and up to 30 frames per second. It has integrated IMU which suits the chosen concept. The field of view is 85.2° horizontal, 58° vertical and 94° diagonal.[20] The camera is shown in Figure 4.1.



Figure 4.1: Intel Real Sense D435i [20]

Thermal Camera

For road surface temperature, a thermal camera was chosen. The MLX90640, provided by melexis is an infrared and cost-effective thermal camera with high accuracy. The operational temperature is between -40 to 85 degree Celsius and it can measure objects from -40 to 300 degree Celsius. The MLX90640 can be provided with two different field of view options. For long range thermal measurements, the field of view is $55^\circ \times 35^\circ$. If an application for close range thermal measurements is applied, the MLX90640 is provided with a $110^\circ \times 75^\circ$ field of view. Therefore for this project the MLX90640 with $55^\circ \times 35^\circ$ field of view is chosen.[21] The thermal camera is shown in Figure 4.2.

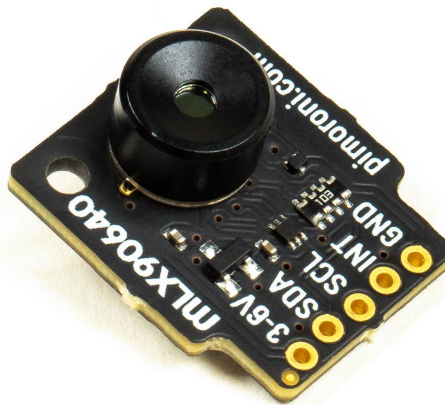


Figure 4.2: MLX90640 Thermal Camera [21]

Hardware

The hardware that was chosen is the Intel UP AI Square Vision Kit in combination with Arduino Uno. The reason for choosing the computer from Intel was the price, size and performance. The Arduino Uno R3 is chosen for operating the light intensity sensor, GPS sensor and the temperature sensor. The thermal camera for measuring the temperature require 20 000 or more bytes of ram which the Arduino can provide. By using several sensors at the same time, a more powerful micro controller is necessary, for example a Raspberry Pi.

Power Supply

To supply power to the sensor package in the car the original power supply could not be used since it is rated for 230V input power. Therefore, a power supply had to be bought or made. The power supply requires a cigarette lighter contact that can transform 12V to 5V and deliver 6A. The power supply is further explained in details in section 4.1.4.

Fixture for Fastening 3D Camera

To fasten the camera a suction cup was used. The suction cup was fastened on the dashboard beside the sensor package for better field of view.

To fasten it to the dashboard a suction cup from Joby was chosen. The suction cup is adjustable. In addition, the fastening bolt on the top fits the internal threads at the chosen camera. The Joby suction cup is shown in Figure 4.3.



Figure 4.3: Joby Suction Cup

GPS Receiver

The GPS receiver that was chosen was a GY-NEO6MV2, which is a module for Arduino. This makes it more compatible to the chosen microcontroller. In addition, it is small and the price is reasonable. The accuracy was precise. The GPS receiver is shown in Figure 4.4.

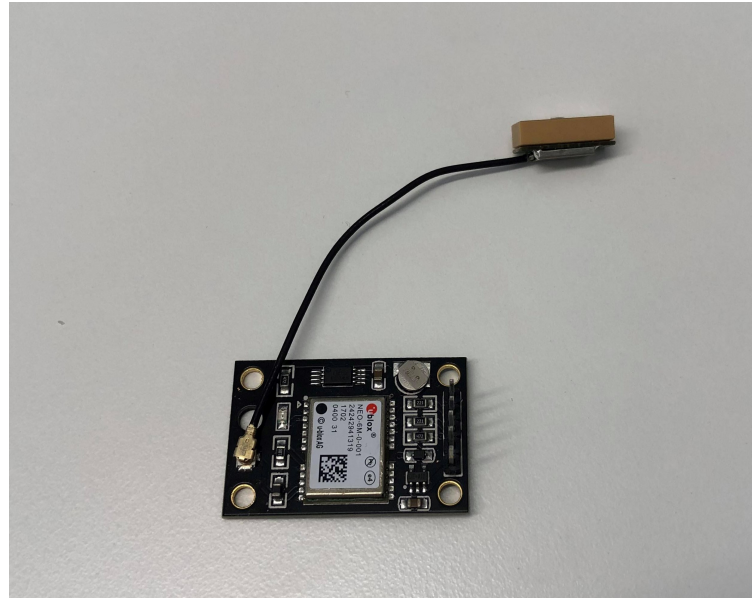


Figure 4.4: NEO 6M - GPS Receiver

Light Dependent Resistor

For measuring the light intensity, a photocell was chosen and wired as a light dependent resistor. This is a cheap solution but gives a reasonable result after calibration. The photocell is shown in Figure 4.5.



Figure 4.5: Photocell

Cost

In table 4.1 the estimated cost for the chosen concept is shown. This estimate is quite accurate but may vary due to variation in exchange rate since several components are ordered abroad. The shipping cost from the dealers are not included in this estimate. Material such as wiring and printing is not included as well.

Table 4.1: Cost for Concept 3

Component	Price in NOK
Intel UP Ai Square kit 3G antenna	4640
3G Adapter	660
Photo Cell and diffuser	100
Intel RealSense D435i	1535
Thermal Camera	458
Arduino Uno R3	200
GPS Sensor	30
Joby Suction Cup	299
Step Down Module	110
Sum	8032

4.1.2 Heat Dissipation Up2

To design the frame for the sensor package and whether the up2 needs to be placed inside the frame or if an external fan was necessary, the group needs to determine how much heat the up2 emits. To calculate the emitted heat, the dimensions of the up2 is required, it was assumed that it is a square. In Figure 4.6 the dimensions are shown.

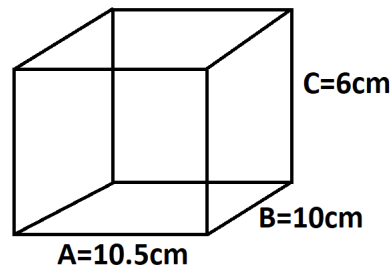


Figure 4.6: Up2 Dimensions

The surface area of up2 must be calculated, which is done in Equation (4.1). The variables are defined in Figure 4.6.

$$Area = 2 \cdot ((A \cdot B) + (A \cdot C) + (B \cdot C)) \quad (4.1)$$

$$Area = 2 \cdot ((10.5cm \cdot 10) + (10.5 \cdot 6) + (10 \cdot 6)) = 456cm^2 = 456000mm^2 = 4.9083ft^2$$

The up2 emits 30W, in Equations (4.2) the input power in watts per square foot is calculated. [22]

$$InputPower = \frac{30W}{4.9ft^2} = 6.11 \frac{W}{ft^2} \quad (4.2)$$

To determine the temperature rise, the graph in Figure 4.7 was used. With the input power at $6.11 \frac{W}{ft^2}$ and the material is *painted non-metallic*, the temperature rise is approximately 15 degrees or 30 fahrenheit.

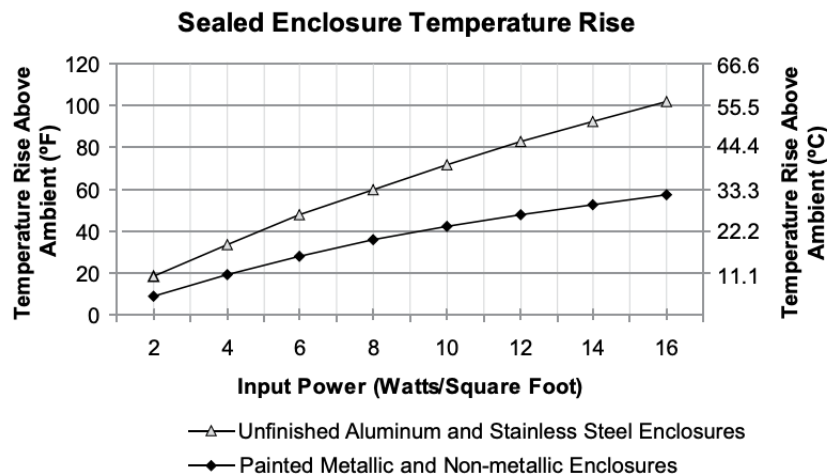


Figure 4.7: Temperature Rise Above Ambient [22]

The temperature rise is only an approximation and will vary due to internal fan use and air flow. Therefore, a safety factor of 25% is added.

Furthermore, it was necessary to calculate the volume airflow to see if an external fan was needed. The formula in Equation (4.3) was used. [22]

$$CFM = \frac{3.16 \cdot W}{\Delta T} = \frac{30 \cdot 3.16}{113 - 73.4} = 2.39 \quad (4.3)$$

Where:

CFM: Volume airflow.

W: Internal heat load in watt.

ΔT : Internal temperature minus ambient temperature in fahrenheit.

A normal 120mm fan which is commonly used in stationary computers provides volume airflow around 80CFM. The volume airflow in the up2 is calculated to be 2.39, therefore an external fan was not necessary in this case. Normal air circulation is sufficient for maintaining normal operating temperature. Since the computer was placed on the dashboard, the air circulation was sufficient, therefore the up2 is placed outside the frame.

4.1.3 Sensor Package Frame

A base frame for the sensor package was designed and produced. The design was done in SolidWorks. The purpose of this design was to get access to the components in the sensor package and to be compact. The frame consists of the base frame, top cover and bracket for light intensity sensor. The base frame is illustrated in Figure 4.8. As viewed the up2 will not have a top cover, this is due to the required airflow which is described in section 4.1.2 above.

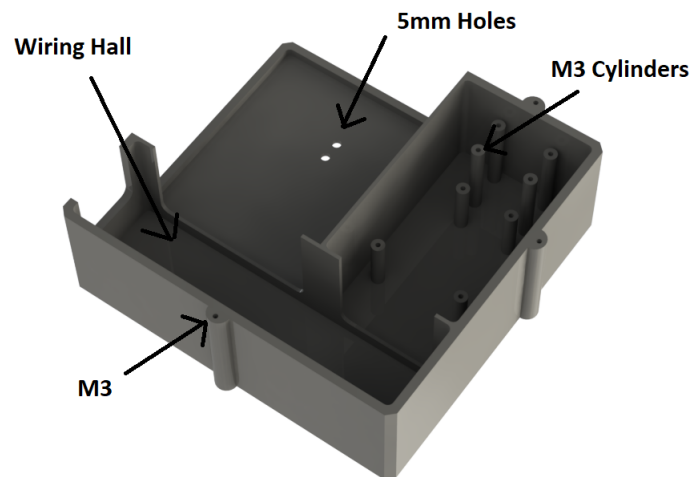


Figure 4.8: Base Frame

The base frame includes $5mm$ holes for fastening the up2, M3 threaded cylinders to fasten the Arduino and the GPS receiver, M3 threaded holes for fastening the top cover to the base frame and a spacious wiring hall. This is shown in Figure 4.8.

Furthermore, a top cover was designed, and 3D printed. The top cover was fastened to the base frame by using 3 x M3 bolts which makes it easy to disassemble and get access to the up2, sensors and Arduino. As seen in Figure 4.9 the top cover got openings for the wiring from the Arduino to the sensors.

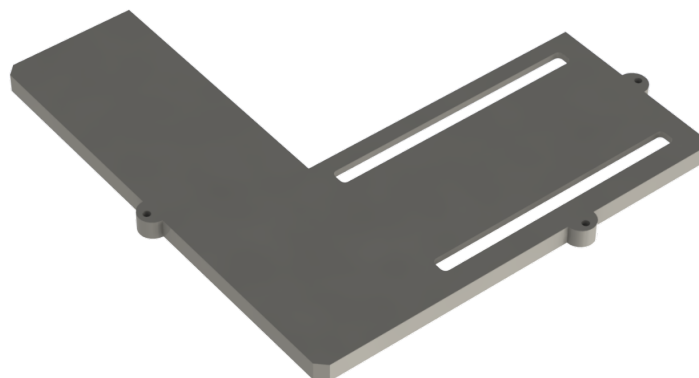


Figure 4.9: Top Cover

Additionally, a bracket for the light intensity sensor was designed. The bracket for light intensity was designed in such a way to receive as much light as possible. Therefore, the high height of the bracket and the 55 degree angle. In Figure 4.10 the bracket for light intensity sensor is shown.

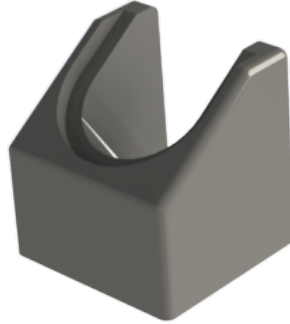


Figure 4.10: Bracket for Light Intensity Sensor

In Figure 4.11 the base frame is illustrated with the computer, Arduino and the GPS receiver mounted. Lastly in figure 4.12 the total assembly is shown.

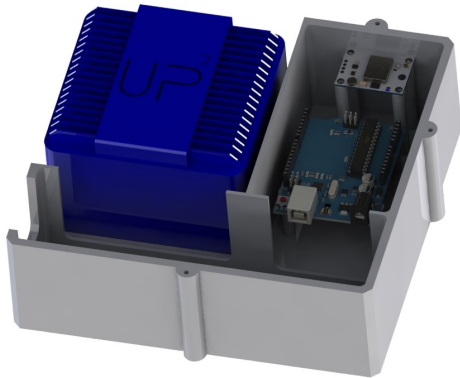


Figure 4.11: Base Frame with Hardware

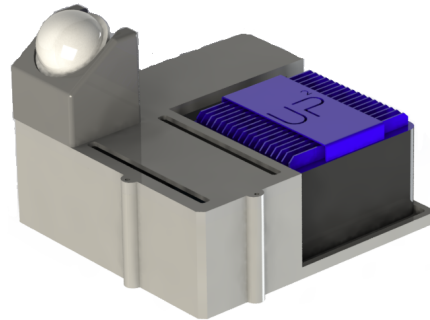


Figure 4.12: Sensor Package - Assembly

All parts are printed in *3D Ultimaker Printer* and the specifications are listed in table 4.2.

Table 4.2: 3D Printer Specifications

Material	PLA
Nozzle	0.4mm
Layer Height	0.15mm
Infill	20 %

4.1.4 Power Supply

The up2 is supplied with 5V and consumes 6A on maximum performance through a DC jack 5.5/2.1mm cable. The power supply must convert 12V from the cigarette lighter in the car to 5V DC jack 5.5/2.1mm. This is not a very common power supply, and the group had trouble getting hands on this type from suppliers without ordering a larger quantity.

Therefore, a power supply had to be made. A step-down module was obtained. The step-down module is shown in Figure 4.13 with the input and output specified. [51]

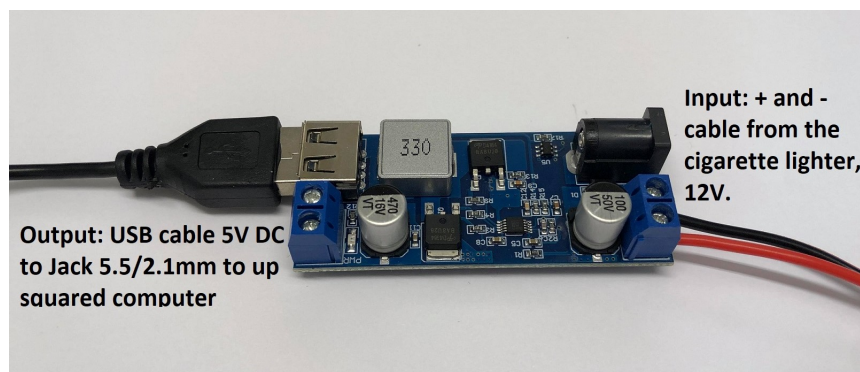


Figure 4.13: Step-Down Module - Inputs and Outputs

Furthermore, to implement this to the sensor package and for testing, a frame had to be designed to the step-down module. This frame was designed in SolidWorks and 3D printed in PLA plastic. This will ensure that the step-down module will not get in touch with anything that leads electricity. The frame consists of a base frame and a top cover with M3 threaded holes for fastening. The frame can be seen in Figure 4.14.

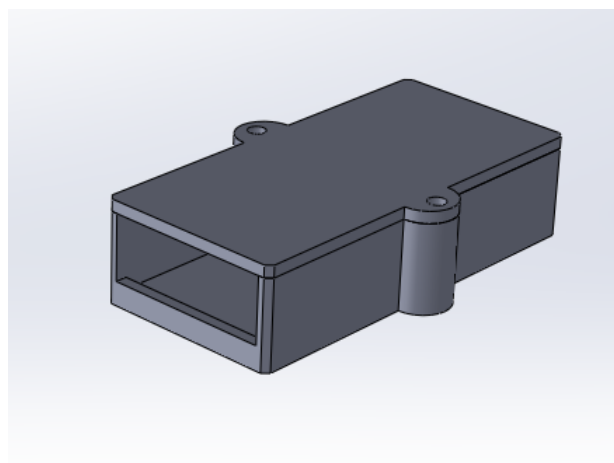


Figure 4.14: Frame for Power Supply

4.2 Data Flow

The sensor package is using applications and scripts to receive, store and illustrate data for the end user. A simple illustration of how the communication was determined to be for sensor package is shown in Figure 4.15.

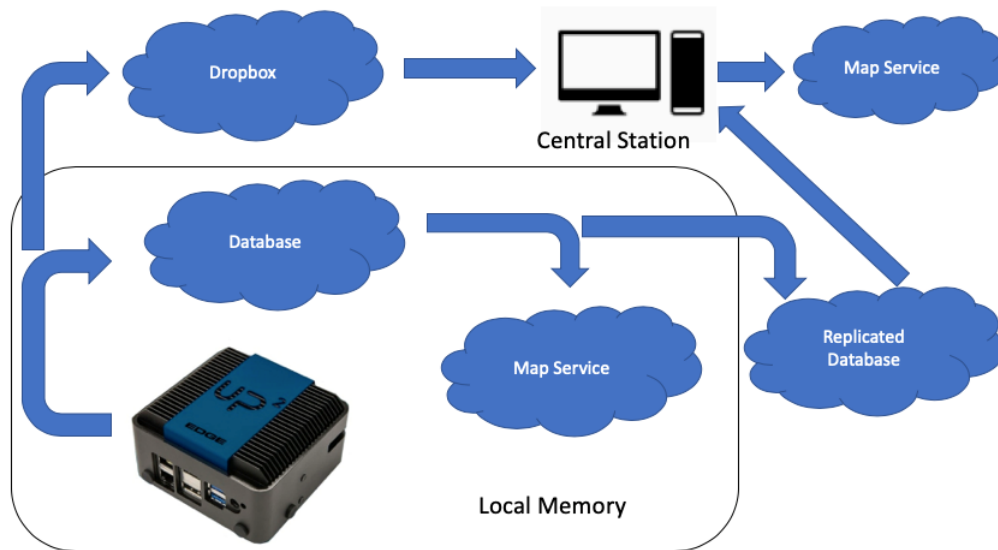


Figure 4.15: Illustration of the Data Flow

4.3 Sensor Package System

The system is communicating with Robotic Operating System, ROS. The programming languages used was Python and Arduino. The hardware which was operating and writing information to the scripts are Intel RealSense D435i, light intensity sensor, GPS receiver, up2 as the main computer and Arduino micro controller for sensor reading. All code cannot operate in a single script. If all code is running in one single script, image processing goes slow and computer lag is occurring. Therefore, the code is distributed into different scripts. ROS is an operating system that gives opportunities for communications between nodes and scripts. To access and communicate with nodes and usage in scripts are explained in the subsections below. All the communication happens in the terminal window.

4.3.1 Create Workspace

Since ROS was used, a workspace that fulfills ROS requirements must be provided. The tool required for a workspace that enables ROS to work properly is catkin. Catkin is a build system and is default when installing ROS, this structure simplifies the build process for the ROS packages. When catkin is installed it must be sourced to our environment as shown in the code below.

```
1 $ Source /opt/ros/kinetic/setup.bash
```

The commands to build the workspace using catkin is shown in the code below. The name of the folder can be replaced to a desired folder name. In the code below the folder name is set to *catkin_ws*, but this can be replaced.

```
1 $ mkdir -p ~/catkin_ws/src
2 $ cd ~/catkin_ws/
3 $ catkin_make
```

When the catkin tool has created the environment, a *CMakeLists.txt* file is created in the *src* folder. A *build* and a *devel* folder is added. In the *devel* folder an important file is stored, which is *setup.sh*. This is a sourcing file, that overlays the operating workspace at the top of the environment, and when running ROS, the sourced folder is used. The code for sourcing the workspace is shown below. It is important to mention when sourcing the workspace, it must be done from the workspace folder. [52]

```
1 $ source devel/setup.bash
```

4.3.2 Create Package

For ROS to find the scripts, it must be placed in a package that supports ROS communication. To create a package the catkin tool was used. The commands for creating a package is shown below and they must be executed from the *src* folder in the workspace by using the terminal from the computer. It is important to include the dependencies for the created package. It is normal to include *rospy*, *roscpp* and *std_msgs*. These dependencies are included in the *CMakeLists.txt* file inside the package folder and can be modified if other dependencies are needed. [53]

```
1 # This is an example, do not try to run this
2 # catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

4.3.3 Camera Node

For detection with a camera, a video frame was provided for the scripts. Since there is more than one script that uses video frame for detection, a camera node is provided. This is because there cannot be more than one script receiving information from a USB port, unless it is a node. Another problem is that the RealSense camera is not a normal USB camera, and video frames cannot be provided from the USB port as on normal cameras. This is due to the multiple functions the camera can provide. Developers have created a RealSense package for ROS implementation with the RealSense camera. When the package is installed in the workspace, the camera can be a node in ROS. The camera publishes all topics that the camera can provide. Some of the topics provided by the RealSense camera is shown below:

- RGB camera node is */camera/color/image_raw*
- Depth camera node is */camera/depth/image_rect_raw*
- infrared1 camera node is */camera/infra1/image_rect_raw*
- infrared2 camera node is */camera/infra2/image_rect_raw*

The camera has 60 topics available when it is running. The code for importing the RGB video frame from the RealSense is shown below, which is the subscriber to the camera node.

```
1 def camera_callback(msg):
2     global frame
3     global sharpened
4     frame = bridge.imgmsg_to_cv2(msg, "bgr8")
5
6
7 def main():
8     # Create Node
9     rospy.init_node('listener_90', anonymous=True)
10    # Define Image Topic
11    camera_topic = "/camera/color/image_raw"
12    # Set Subscriber and Define its Callback
13    rospy.Subscriber(camera_topic, Image, camera_callback)
14    # Spin until ctrl + c
15    rospy.spin()
16
17
18 if __name__ == '__main__':
19     main()
```

It is important to import libraries such as *cv2*, *cv_bridge* and *rospy* to the Python script for translating the frame message published by the camera node and to be able to process the video frame further.

4.3.4 Sensor Node

The sensors used in this sensor package except the camera are operated by the Arduino micro controller. The Python scripts running on the up2 are using ROS to communicate with the Arduino. The Arduino is a publisher and the Python scripts are subscribers to the Arduino node. There was experienced slow communication between the Arduino and ROS when using the package provided for ROS communication with Arduino. The provided package occupies memory on the Arduino, and it becomes slow. The solution was to enable communication between Arduino and a Python script using libraries made for Python and Arduino communication. When the communication was established, the Python script was created to be a publisher node. The communication was faster, and the subscribers got the information that was required to run. The code for reading the sensor values with python and creating a publisher is shown below, which is the sensor node publisher.

```

1 import serial
2 import pycpp2
3 import rospy
4 from std_msgs.msg import Float64
5
6 #Connect to Serial Port
7 try:
8     arduino = serial.Serial('/dev/ttyACM0', 115200)
9     ok=1
10 except:
11     print("Check port")
12
13 def talker():
14     #Read Serial and Create a Publisher
15     latitude=rospy.Publisher('lat', Float64, queue_size=10)
16     longitude=rospy.Publisher('lng', Float64, queue_size=10)
17     lux = rospy.Publisher('lx', Float64, queue_size=10)
18     rospy.init_node('talker',anonymous=True)
19     rate = rospy.Rate(10)
20     while not rospy.is_shutdown():
21         gps_point = str(arduino.readline())
22         if ok==1:
23             try:
24
25                 lat = float(gps_point[0:9])
26                 lng = float(gps_point[10:18])
27                 lx = float(gps_point[19:27])
28             except:
29                 lat=0.0;

```

```
30         lng=0.0;
31         lx=0.0;
32         #print (lat , lng , lx)
33         rospy . loginfo (lat)
34         rospy . loginfo (lng)
35         rospy . loginfo (lx)
36         longitude . publish (lng)
37         latitude . publish (lat)
38         lux . publish (lx)
39         rate . sleep ()
40 while True:
41
42     try:
43         talker ()
44
45     except rospy . ROSInterruptException:
46         pass
```

4.3.5 GPS Subscriber Node

When the sensor package detects states in the road, the GPS position for the detected state was required. The scripts are subscribers to the sensor node and the code for subscribing the GPS coordinates is shown below.

```
1 def lat_callback(msg):
2     global lat
3     lat=msg.data
4
5 def lng_callback(msg):
6     global lng
7     lng=msg.data
8
9 def main():
10    # Create Node
11    rospy.init_node('listener', anonymous=True)
12    # Define Subscriber and Define its Callback
13    rospy.Subscriber("lat", Float64, lat_callback)
14    rospy.Subscriber("lng", Float64, lng_callback)
15    # Spin until ctrl + c
16    rospy.spin()
17
18
19 if __name__ == '__main__':
20    main()
```

4.3.6 IMU Subscriber Node

To give an approximation of the road condition, an accelerometer was used. Since the camera used in this project is hybrid, it contains an IMU with an accelerometer. The camera node provides the topic for extracting information from the IMU. The acceleration in Z -direction can be extracted with a coded script. The code for acceleration in Z -direction is shown below.

```
1 import rospy
2 from sensor_msgs.msg import Imu
3
4 def accel_callback(msg):
5     global acc
6     acc=abs(msg.linear_acceleration.z-9.32)
7     print(acc)
8     if acc > 4.5:
9         #update_folder()
10        update_table()
11
12 while True:
13
14     try:
15         rospy.init_node('accel_list', anonymous=True)
16         rospy.Subscriber("/camera/accel/sample", Imu,
17                           accel_callback)
18         rospy.spin()
19
20     except rospy.ROSInterruptException:
21         pass
```

4.3.7 Database Communication

The observations made from the sensor package was stored in a database for illustrating the observations on a map application. Furthermore, the company that is responsible for repairing the road can determine whether the detected fault needs repair. The database used was PostgreSQL and Python to push data, the library psycopg2 was imported and a code was provided to ensure pushed data to the database. Since the server is password protected, the connection line contains *username* and *password* to gain server access. The information was stored in tables, one column that hold the information about the detected state and the image number, the second column contains the GPS information stored as a *point*. Map service application QGIS, cannot plot the GPS location of the state if it was not stored as a *point*. The code for pushing information to the database from the Python script is shown below.

```

1 import rospy
2 import psycopg2
3 import cv2
4 from std_msgs.msg import Float64
5 from sensor_msgs.msg import Imu
6
7
8 def globallyChange():
9     global i
10    i += 1
11
12 def lat_callback(msg):
13     global lat
14     lat=msg.data
15
16 def lng_callback(msg):
17     global lng
18     lng=msg.data
19
20 def update_table():
21     name= 'Bad_Road{:>03}'.format(i)
22     cursor.execute(''INSERT INTO roadcondition(geom,info)
23     VALUES(ST_GeomFromText('POINT(%s %s)',4326),%s)''',(lng,
24     lat ,name))
25     connection.commit()
26     count=cursor.rowcount
27
28 def update_folder():
29
30     cv2.imwrite('/home/upsquared/Desktop/detected_images/
31     bad_road/pic{:>03}.jpg'.format(i), frame)

```

```
28         globallyChange ()
29
30     while True:
31
32         try:
33             #Connect to Database and Create Node and Subscribers
34             connection= psycopg2.connect (user="postgres",
35             password="upsquared", host="localhost", port
36             =5432,database="mas500")
37             cursor = connection.cursor ()
38             i = 0
39             rospy.init_node ('accel_list', anonymous=True)
40             rospy.Subscriber ("/camera/accel/sample", Imu,
41             accel_callback)
42             rospy.Subscriber ("lat", Float64, lat_callback)
43             rospy.Subscriber ("lng", Float64, lng_callback)
44             rospy.spin ()
45
46     except rospy.ROSInterruptException:
47         pass
```

4.3.8 Dropbox Communication

The images from detected states are stored in Dropbox and to upload images to Dropbox with a Python script the library *dropbox* was imported. It is not possible to access folders with Python without creating an application folder in Dropbox. Dropbox has provided an application solution for developers when uploading files with scripts. When an application is made, a access key can be generated for the account. The access key allows scripts to access the desired Dropbox folders made for the project. The code for uploading files to a Dropbox folder is shown below.

```

1 import psycopg2
2 import rospy
3 from std_msgs.msg import Float64
4 import dropbox
5 import time
6 import os
7 import numpy as np
8
9 #Access Dropbox Account
10
11 access_token = '
    MpY68MMU6cAAAAAAAAAADyIpSgD1Dh6mJHqv4D5FnJm3ykA_rZW4iRatzyG_bW19'
12 dbx = dropbox.Dropbox(access_token)
13
14 #Direction to Detected Images
15 rootdir = '/home/upsquared/Desktop/detected_images/90_sign'
16
17 def accel_callback(msg):
18     global upl
19     upl(msg.data
20     upload_img())
21
22 def upload_img():
23 #Code for Uploading to Dropbox
24     for dir, dirs, files in os.walk(rootdir):
25         for file in files:
26             try:
27                 file_path= os.path.join(dir,
28                 file)
29                 #Name of the Folder in
30                 Dropbox Apps Folder
31                 dest_path=os.path.join('/90
32                 _sign/', file)

```



```
30         print('Uploading %s to %s' %
31               (file_path, dest_path))
32         with open(file_path, "rb") as
33             f:
34                 dbx.files_upload(f.
35                                 read(), dest_path
36                                 , mute=True)
37     except Exception as err:
38
39         print("failed to upload %s\n
40               %s" % (file, err))
41
42 def main():
43     upload_img()
44     # Spin until ctrl + c
45     rospy.spin()
```

4.3.9 System Launch

To operate all scripts simultaneously there were two possibilities. The first method was to manually start each node in the command prompt with ROS commands. The second method was to create a launch file that starts the system with one ROS command. When launching the system, ROS handles the scripts to start simultaneously in a single terminal tab. The code for the written launch file is shown below.

```
1 <launch>
2     <include file= "$(find realsense2_camera)/launch/rs_camera.
3         launch"/>
4     <node name= "lane_mark_listener" pkg="python_skripter" type="
5         line_detection.py" />
6     <node name= "listener_100" pkg="python_skripter" type="
7         camera_100.py" />
8     <node name= "listener_90" pkg="python_skripter" type="
9         camera_90.py" />
10    <node name= "talker" pkg="python_skripter" type="
11        ros_coord_node.py" />
12    <node name=" accel_list" pkg="python_skripter" type=" accel.py"
13        />
14 </launch>
```

4.4 Calibration

In this section the method for calibrating the sensors are presented.

4.4.1 Camera Calibration - Intel RealSense D435i

To perform the calibration of Intel RealSense D435i, software and hardware was necessary. Below the software and hardware that was used is listed :

- Intel Real Sense D435i with firmware 5.11.01. Connected with a USB C cable
- A tripod for supporting the camera
- A PC with Ubuntu 16.04
- Calibration grid on *Dynamic Target Tool* app at a smart phone. [54]
- Intel RealSense Dynamic Calibrator tool. Version: 2.6.8.0. [55]

The setup of the calibration is shown in Figure 4.16, while the calibration grid is shown in Figure 4.17. The device was placed approximately $700mm$ from the target. Intel recommends distance between $600mm$ and $850mm$.

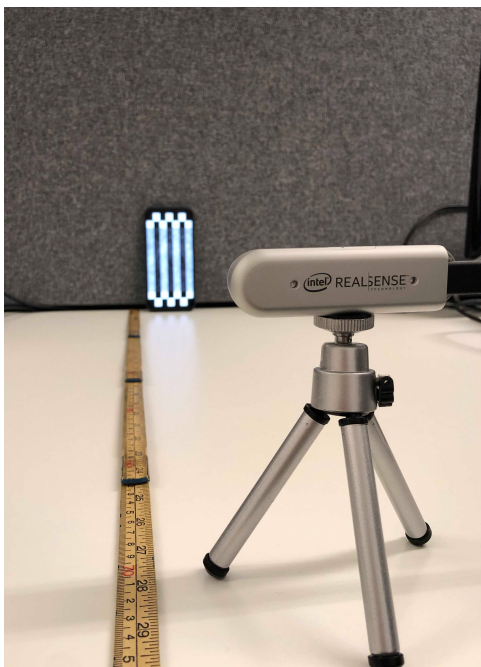


Figure 4.16: Calibration Setup

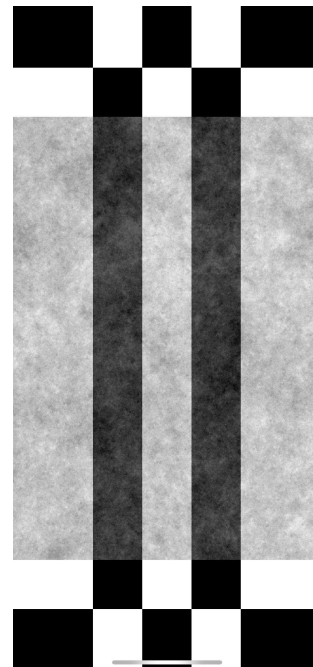


Figure 4.17: Calibration Grid

The program was launched and the main screen is shown in Figure 4.18. Both RGB and depth camera was calibrated during this process. The resolution on the camera throughout the calibration was $1280p \times 720p$ with 30 frames per second. By clicking on "*Show Demo*" it is possible with a simple review of how the calibration process work. The calibration grid consists of black and white bars, and height depends on which smart phone that i used.

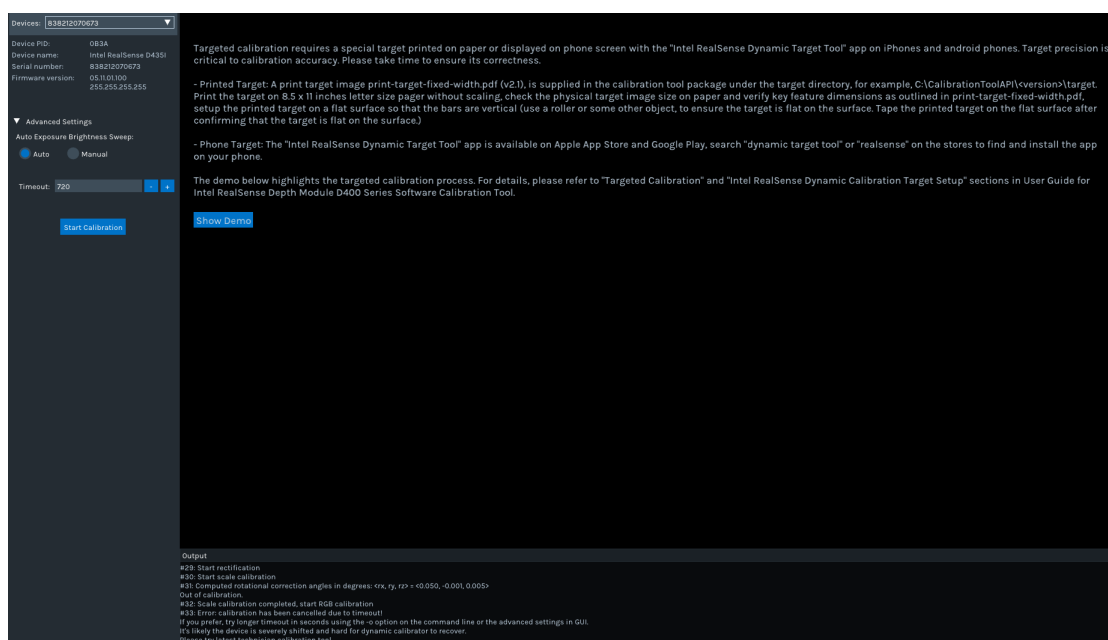


Figure 4.18: Intel RealSense Dynamic Calibrator - Start Up Screen

When the calibration process was started, the device was moved until the target device has *cleared* the blue area. This stage of the process is shown in Figure 4.19.

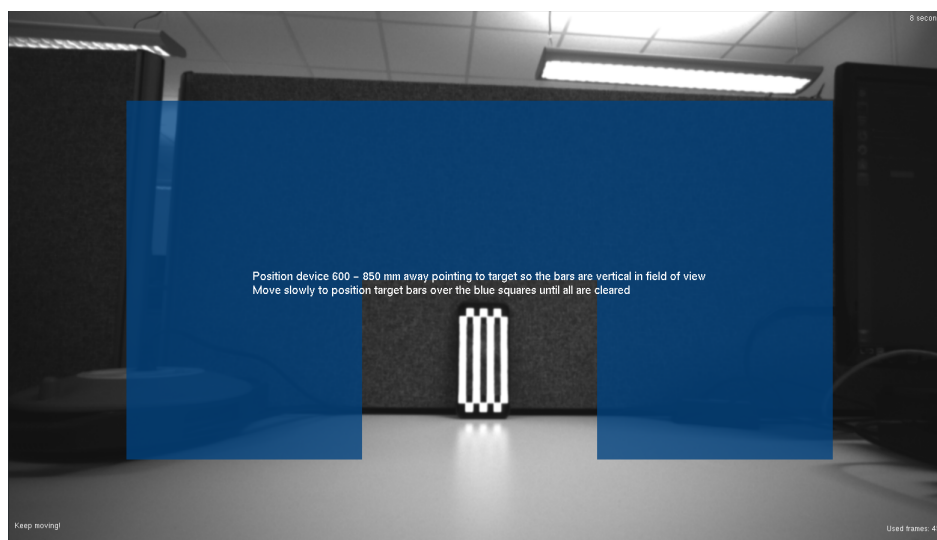


Figure 4.19: Intel RealSense Dynamic Calibrator - Initialization Step

When the RGB camera was calibrated. The device was slowly moved around until the green bar was at 100%. The blue frames are the approved positions. This is shown in Figure 4.20.

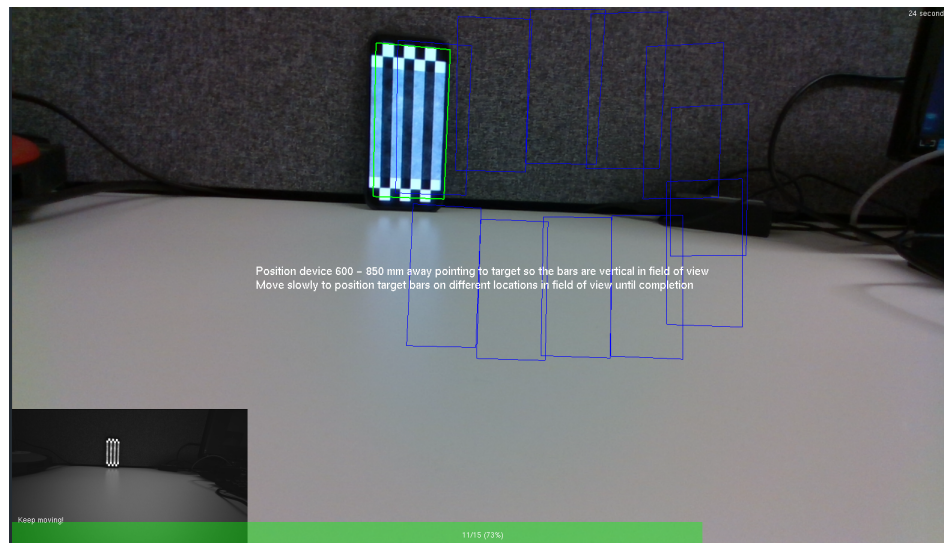


Figure 4.20: Intel RealSense Dynamic Calibrator - RGB Camera

Lastly the depth camera was calibrated. This was done in the same way as for the RGB. Slowly move the camera around until it is satisfied, which is shown in Figure 4.21.

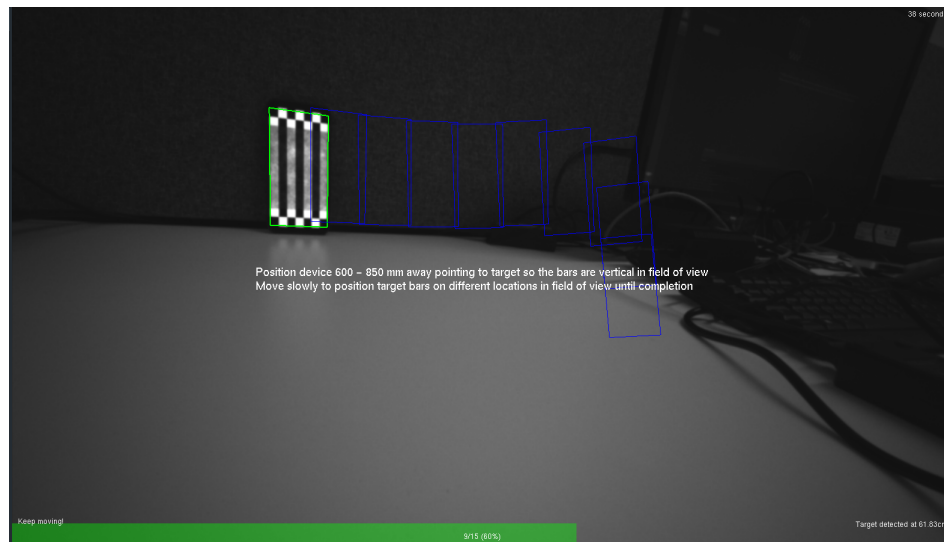


Figure 4.21: Intel RealSense Dynamic Calibrator - Depth Camera

When the steps are completed, it appears a *calibration is successful* message. The results are automatically updated to the device.

4.4.2 Light Intensity Sensor Calibration

Before calibration, it was important that the light meter and the photocell sensor was placed in the same surroundings and placed at the same angle. To calibrate the light intensity sensor a certified light meter was necessary. Therefore in this calibration a RS certified *RS PRO ILM01* light meter was used and it is shown in Figure 4.22.

The photocell and diffuser was wired to a multi-meter which measured the resistance of the photocell, in the respective light surroundings. This setup is shown in Figure 4.23.



Figure 4.22: Certified RS Light Meter

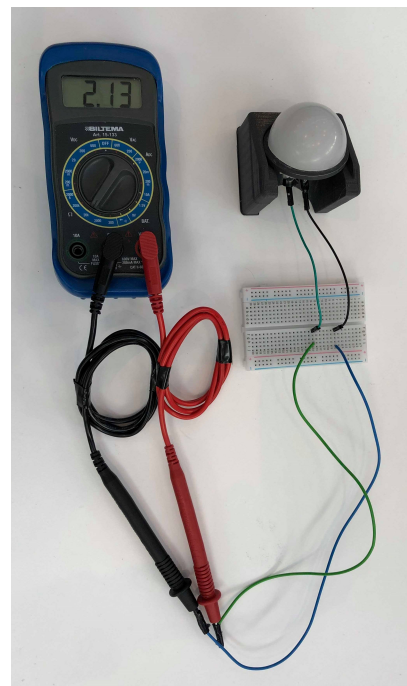


Figure 4.23: Setup - Light Intensity Calibration

In table 4.3 the measurements are shown. In Figure 4.24 the lux as a function of resistance is illustrated graphically. Furthermore, this is written into an excel sheet.

Table 4.3: Measurements - Numerical

RS Light Meter[LUX]	Photo Cell [Ohm]
1	180000
2	117000
20	25700
43	13800
100	8360
200	5570
250	4880
300	4380
350	4000
400	3660
500	3500
590	3110
640	2990
700	2820
930	2350
1270	1980
1976	1460
2460	1330
3340	1130

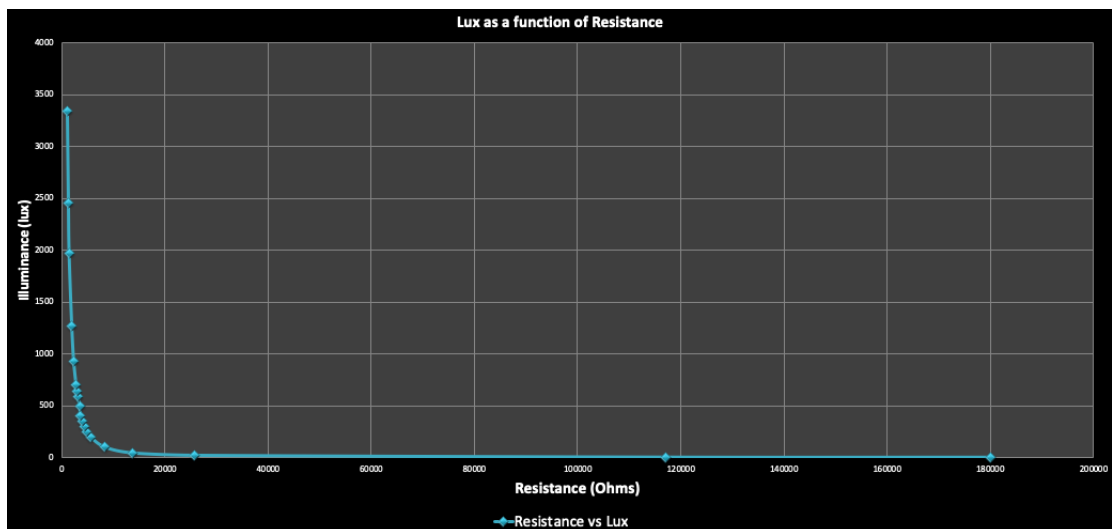


Figure 4.24: Measurements - Graphical Illustration

As seen from the graphical representation, the function was not linear. To linearize the measured values are converted to a logarithmic value. The converted values are shown in the table 4.4. The graph for logarithmic conversion is shown in Figure 4.25.

Table 4.4: Measurements - Numerical [log]

$\log(\text{RS Light Meter[LUX]})$	$\log(\text{Photo Cell [Ohm]})$
0.0	5.255272505
0.301029996	5.068185862
1.301029996	4.409933123
1.633468456	4.139879086
2.0	3.922206277
2.301029996	3.745855195
2.397940009	3.688419822
2.477121255	3.641474111
2.544068044	3.602059991
2.602059991	3.563481085
2.698970004	3.544068044
2.770852012	3.492760389
2.806179974	3.475671188
2.84509804	3.450249108
2.968482949	3.371067862
3.103803721	3.29666519
3.29578694	3.164352856
3.390935107	3.123851641
3.523746467	3.053078443

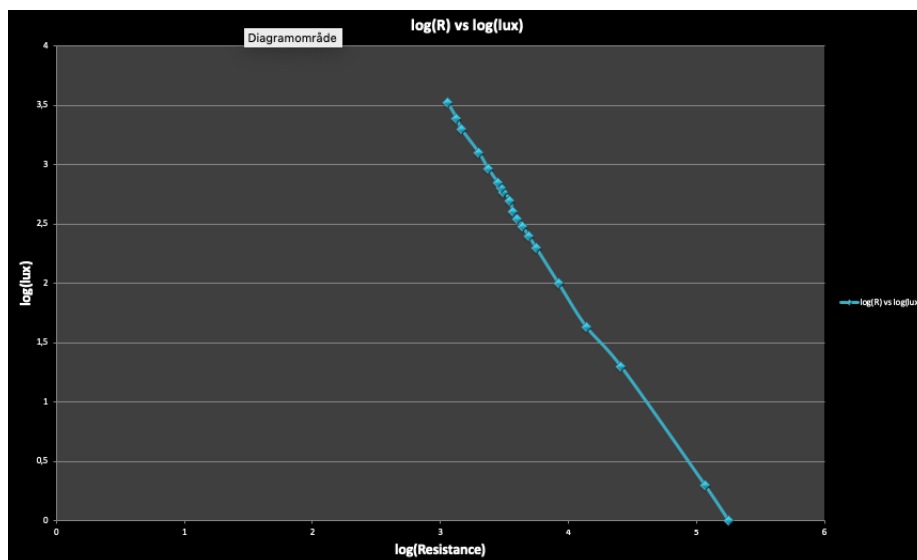


Figure 4.25: Measurements - Logarithmic Conversion

To estimate the light intensity of the surroundings with the photocell, mathematical operations was applied to the convert the numbers. From mathematics, a straight line can be represented as $y = m \cdot x + b$. Therefore, the formula for the straight line is shown in Equation (4.4).

$$\log Lux = m \cdot \log R + b \quad (4.4)$$

$$Lux = 10^{R^m + b} \quad (4.5)$$

$$Lux = 10^b \cdot R^m \quad (4.6)$$

Where:

R: Resistance of the LDR sensor

m: Constant

b: Constant

Lux: Light intensity measured in Lux

For our system the estimated constants m and b , are $-1,59366$ and 8.322372561 .

4.4.3 IMU - Intel Real Sense 435i

To calibrate the IMU of Intel RealSense D435i it required software and hardware. Below the software and hardware that were used is listed :

- A PC with Windows 10
- Intel RealSense D435i and a USB C cable.
- Python 2.7
- Python calibration script provided by Intel [56]
- Libraries in Python; Pip, Numpy, Enium, pyrealsense2

The calibration process contains 3 stages. First recording IMU data in 6 different positions, then computing the parameters. Lastly, write the parameters over to the camera.

First the calibration script was launched, and the first stage of the process started. In every position the camera was held steady for 3 seconds or longer until enough data was collected. Further the script requests the next position automatically when enough data from previous position is collected. From Figure 4.26 to 4.31 all positions are shown in the correct order.



Figure 4.26: Position 1 - Upright Facing Out



Figure 4.27: Position 2 - USB Cable Up and Facing Out



Figure 4.28: Position 3 - Upside Down Facing Out



Figure 4.29: Position 4 - USB Cable Down and Facing Out



Figure 4.30: Position 5 - Facing Down



Figure 4.31: Position 6 - Facing Up

In Figure 4.32 the command prompt from the script is shown. As seen, it requests the desired position and the lower right corner prints true in x , y and z if the camera is in the correct position.

```

Direction data collected.
Align to direction: [1. 0. 0.]   USB cable up facing out
Status.collect_data@[0;32m[.....]@[0;0mm

Direction data collected.
Align to direction: [0. 1. 0.]   Upside down facing out
Status.rotate:      [-0.4979  1.2794 -0.821 ]:           [False False False]

```

Figure 4.32: Calibration Script - Command Prompt

After the script has collected the necessary information and finished the calibration after 6000 measurements it provides an overall score for the calibration which can be compared with previous calibration that has been uploaded. Further it can be decided with a keystroke to overwrite the previous calibration file. Then the device is calibrated and ready for use. The full calibration script which is written in Python and provided by Intel is shown in Appendix F.2. [56]

4.5 Electrical Connections

The prototype was wired as shown in Figure 4.33. The camera and Arduino were directly wired and powered by the up2. The GPS receiver and the light intensity sensor is powered by the Arduino. The up2 is powered from the cigarette lighter in the cockpit of the car.

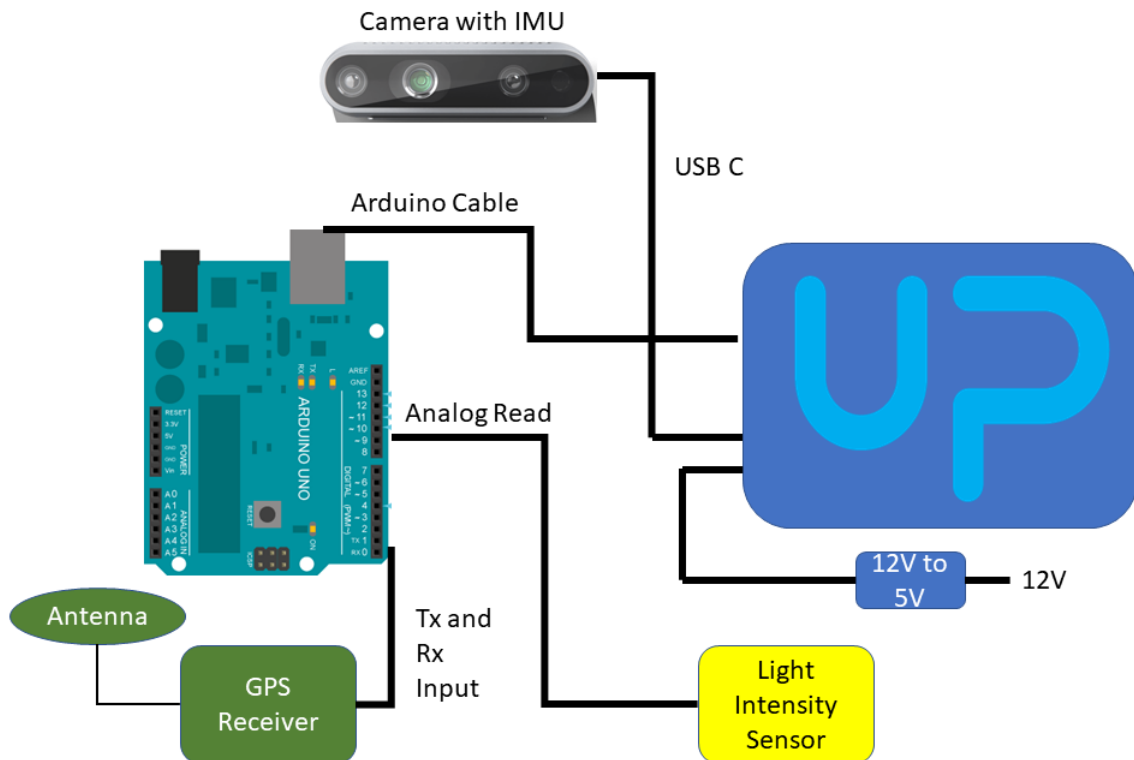


Figure 4.33: Circuit Diagram

4.6 Speed Limit Sign Recognition

This algorithm detects speed limit signs from dashboard position in the vehicle. The signs that was detected with this algorithm is 90 and 100 $\frac{km}{h}$ speed limit signs. The signs can be seen in Figure 4.34.



Figure 4.34: Speed Limit Signs

For detecting speed limit signs there were two methods that the group considered. Machine learning, and HSV thresholding with template matching. Machine learning is a more complex method than HSV thresholding, which means more powerful computer is required. The group first started training a model for traffic sign recognition, but the accuracy was not good enough and the number of outliers was not satisfactory. Therefore, the second method was used. The general overview of the algorithm is illustrated in the flowchart in Figure 4.35. The algorithm and the code is shown and explained in detail further below in this section.

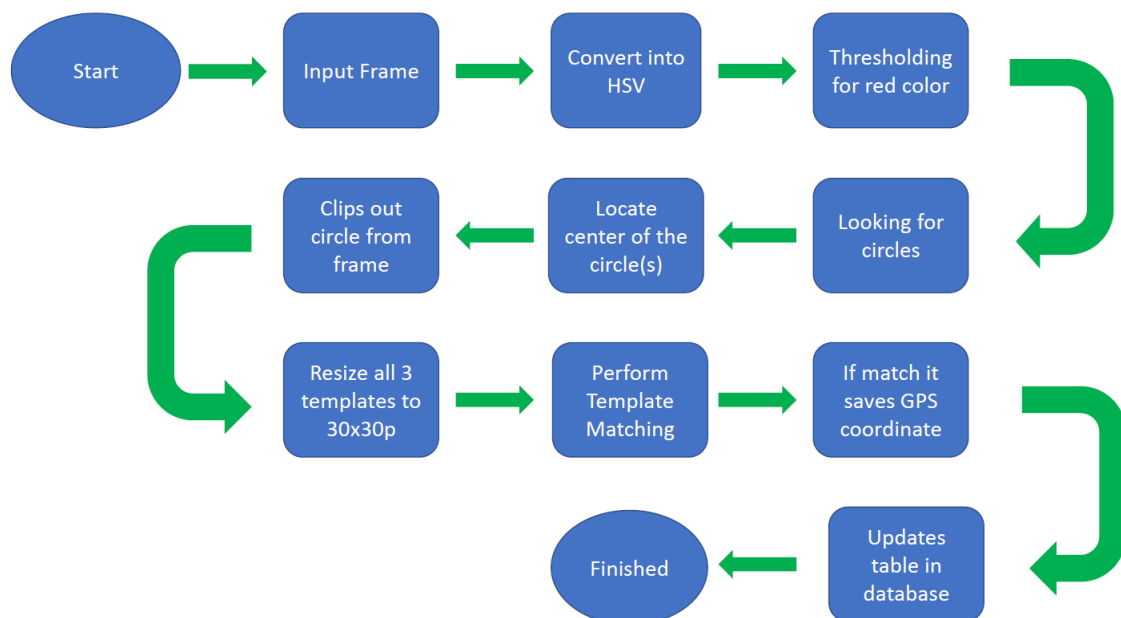


Figure 4.35: Flowchart - Traffic Sign Recognition

The image from the camera frame is converted from RGB to a HSV color model. HSV color model representation was used since HSV better represents how humans relate to colors than RGB color model does. Since the focus is on speed limit signs, one color is repeatable in all of them. The speed limit signs in Norway contains a red circle at the edge of the sign. Therefore, if only the red color is extracted, it is possible to detect circles with the *cv2.HoughCircles* function in OpenCV. For testing the concept, printed Norwegian speed signs was placed on a wall to indicate if the algorithm for detecting the signs with color threshold could work. In Figure 4.83 the testing facilities are shown.



Figure 4.36: Speed Limit Sign Recognition Test Setup

See code below for converting the image to HSV and extracting the red color is shown. The upper and lower threshold for the color red is chosen for red color.

```
1 #Convert Image to HSV
2 hsv = cv2.cvtColor(copy_frame , cv2.COLOR_BGR2HSV)
3 #Set Lower and Upper Boundaries for Red Color
4 lower_red = np.array([0,150,95])
5 upper_red = np.array([10,255,255])
6 mask1 = cv2.inRange(hsv, lower_red, upper_red)
7 lower_red = np.array([170,150,95])
8 upper_red = np.array([180,255,255])
9 mask2 = cv2.inRange(hsv, lower_red, upper_red)
10 mask = mask1 + mask2
```

The resulted frame after extracting the red colors is shown in Figure 4.37.

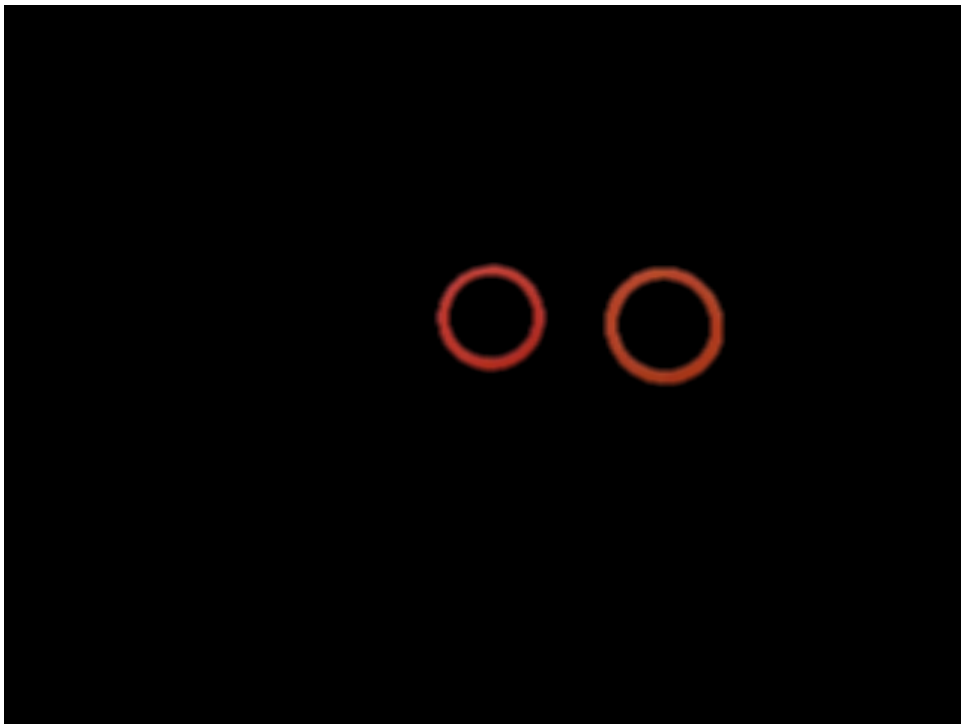


Figure 4.37: Threshold for Red Circles

To extract the red colors from the image, a reference was found on the internet. The reference image used for thresholding the red colors is shown in Figure 4.38.

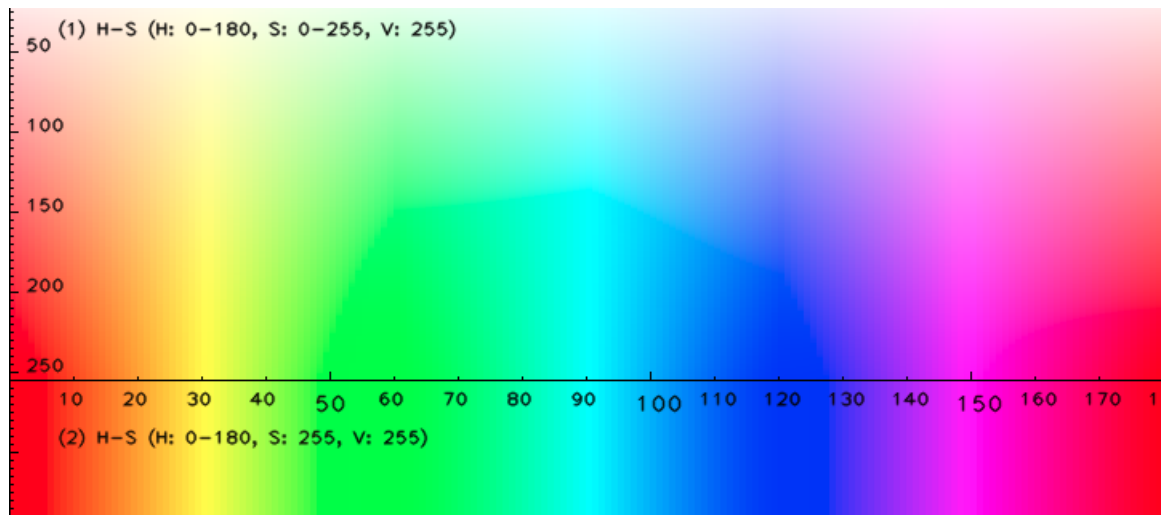


Figure 4.38: HSV Color Range [23]

As shown in Figure 4.37, the circles can be detected and *cv2.HoughCircles* was applied. When the circles were detected, a bounding box was drawn around the circle. The bounding box was printed on the frame and is shown in Figure 4.39.



Figure 4.39: Bounded Box around Circles

When the signs are detected, the script snips out the bounded area from the original frame and a template match method is executed to identify the sign. The template images are resized and has the same size as the input image. The code for template matching is shown below. In this example the sign identification and score is shown for the 90 sign in Figure 4.40.

```

1 def template():
2     #Snapshot the Rectangle from the Original Frame
3     h=z+5
4     detected_img = frame[y-h:y+h, x-h:x+h]
5     detected_img_gray = cv2.cvtColor(detected_img, cv2.
        COLOR_BGR2GRAY)
6     detected_img_gray = cv2.blur(detected_img_gray, (5,5),3)
7     #Resize Template to the Real Size
8     template_90 = cv2.imread('/home/upsquared/MAS500_ws/src/
        python_skripter/src/scripts/fartsgrense_90.png',0)
9     template_90 = cv2.resize(template_90, (2*h, 2*h))
10    template_90 = cv2.blur(template_90, (5,5),3)
11
12    global result
13    result = cv2.matchTemplate(detected_img_gray, template_90, cv2.
        TM_CCOEFF_NORMED)
14    w, h = template_90.shape[:-1]
15    threshold = 0.7
16    loc = np.where( result >= threshold)
17    for pt in zip(*loc[:-1]):
18        #res = "{}%".format(result, float(result))
19        font = cv2.FONT_HERSHEY_SIMPLEX
20        #cv2.putText(copy_frame, res, (x-h, y+h), font,
            1,(255,255,255),2,cv2.LINE_AA);
21        cv2.putText(copy_frame, '90_sign', (x-h-50, y-h), font,
            1,(255,255,255),2,cv2.LINE_AA);
22        update_folder()
23        update_table()

```

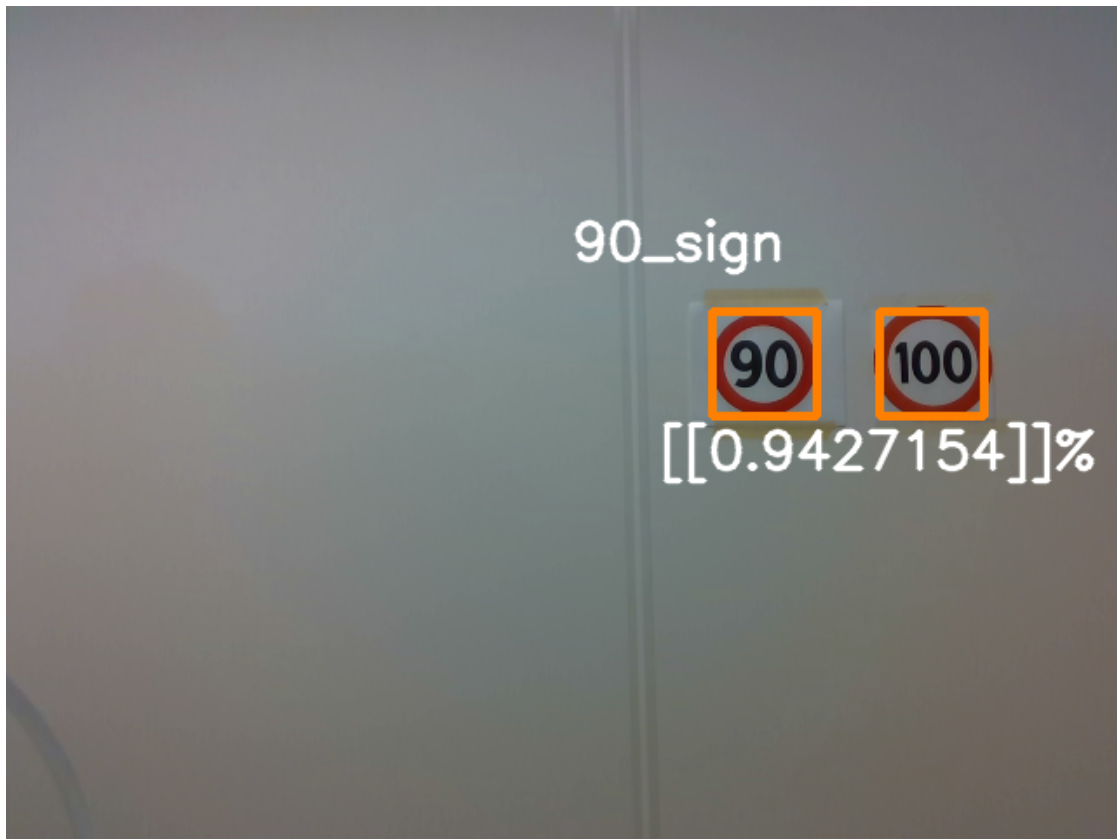


Figure 4.40: Template Score and Match

When the sign was identified, the script requested the GPS location of the sensor package and it updated a table in the database containing detected sign and its location.

4.7 Lane Mark Quality

This section explains how the quality of the lane marks were determined. The lane marks was categorized into *Bad Lane Mark* and *Good Lane Mark*. This was determined by thresholds which can be modified. The route that was tested was on E18 between Grimstad and Lillesand, change in lane mark pattern was not considered at exits and entrance ramps. In addition, this algorithm was developed for cars in the right lane, but with adjustments this algorithm can work in the left lane. In Figure 4.41 the general overview of the algorithm is shown in a flow chart.

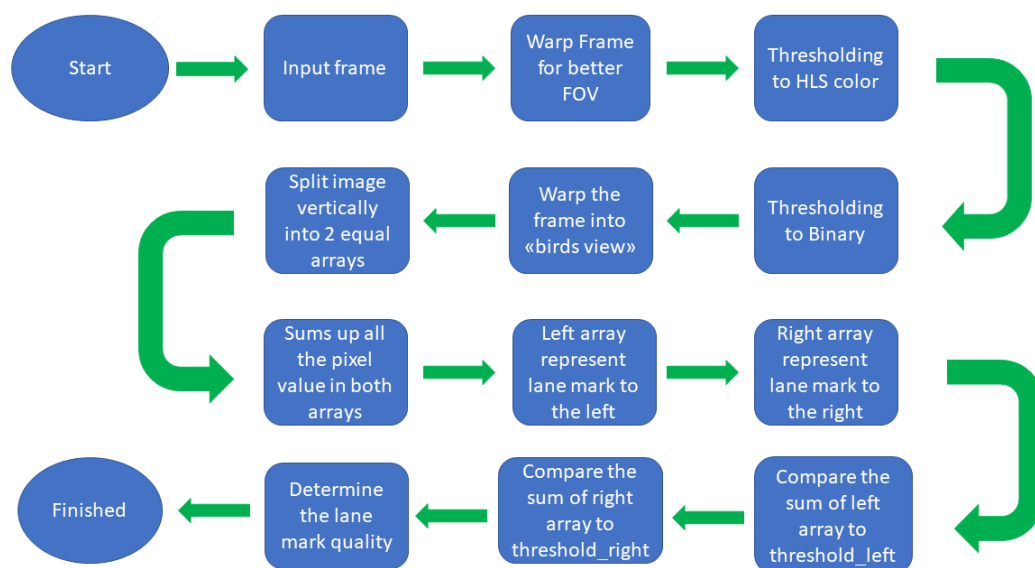


Figure 4.41: Flowchart - Lane Mark Quality

As seen in the flowchart the algorithm sums all the pixels vertically which results in an indication of how worn the lane marking is. When the car was driving in the right lane the field of view is shown in Figure 4.42.



Figure 4.42: Field Of View - From E18

There is two kinds of lane markings, the solid lane to the right and the dashed marking to the left which separate the lanes. In Figure 4.43 the dimensions of the lane marking is shown.

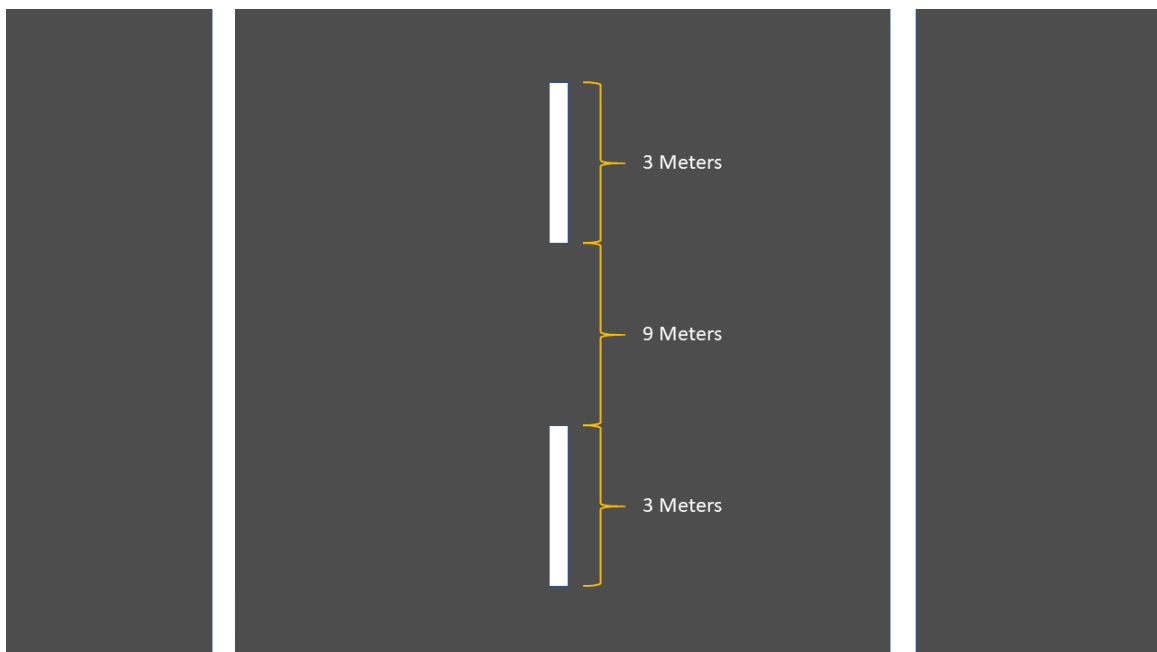


Figure 4.43: Lane Mark Dimension - E18 with Speed Limit Over $90 \frac{km}{h}$

Below the code which was written in Python is partially described. The full script can be seen in Appendix F.4.

First the input frame which is shown in Figure 4.42 was warped for a better view, the code is shown below. The *src* points is the array that determine the warping points. The output frame is shown in Figure 4.44.

```
1 src = np.float32([[0, 600], [1010, 600], [450, 400], [700, 400]])
2 bottom_left = src[0][0] + 0, src[0][1]
3 bottom_right = src[1][0] - 0, src[1][1]
4 top_left = src[3][0] - 0, 1
5 top_right = src[2][0] + 0, 1
6 dst = np.float32([bottom_left, bottom_right, top_right, top_left])
7 img_warped = Transform_Camera_View(copy_frame, src, dst)[0]
```



Figure 4.44: Warped Frame

Then the next step was to threshold the warped frame into HLS color space which helped highlighting the lane markings. Furthermore, it was thresholded into a binary frame. The code is shown below and the binary thresholded frame is shown in Figure 4.45.

```
1 def HLS_L_Threshold(img, thresh=(195, 255)):  
2     img = img[:, :, 1]  
3     img = img * (255 / np.max(img))  
4     binary_output = np.zeros_like(img)  
5     binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1  
6     return binary_output
```

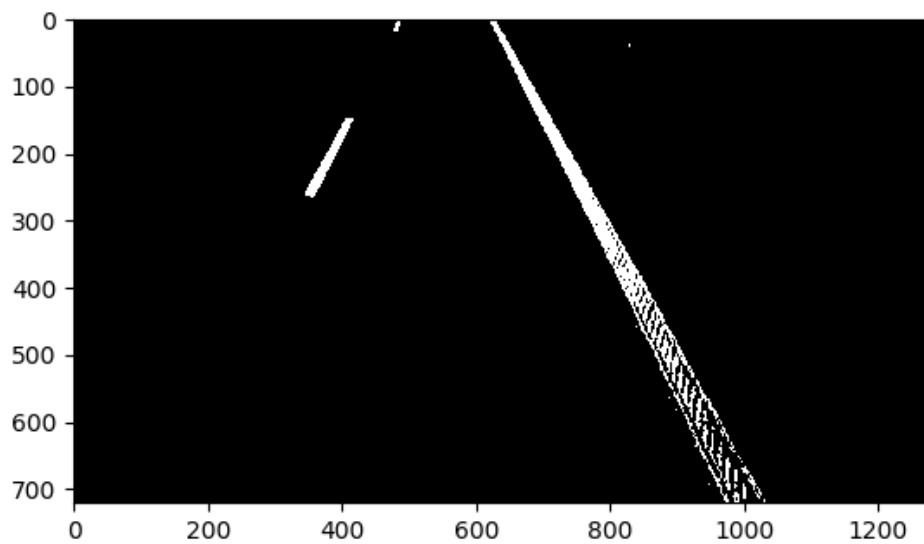


Figure 4.45: Binary Output

Next, the frame was further warped into *Birds-Eye View*, the reason for this was to sum up the pixel values for determining the lane mark quality. The code is shown below and the warped image is shown in Figure 4.46.

```
1 pts1 = np.float32([[456, 0], [665, 0], [0, 720], [1280, 720]]) #  
   Old points  
2 pts2 = np.float32([[0, 0], [1280, 0], [0, 720], [1280, 720]]) # New  
   points  
3 matrix = cv2.getPerspectiveTransform(pts1, pts2) # Transformation  
   matrix  
4 result = cv2.warpPerspective(thresh_HLS, matrix, (1280, 720)) # The  
   transformed image
```

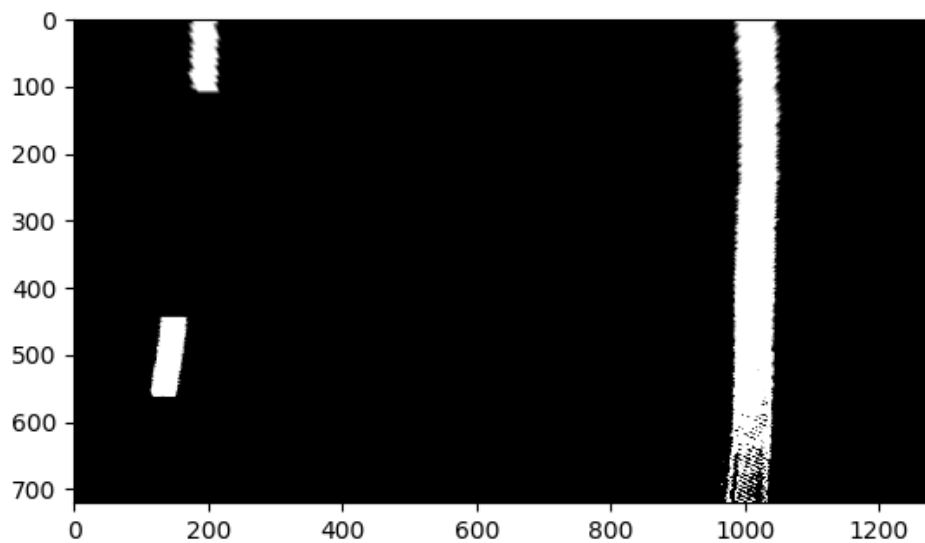


Figure 4.46: Warped Output

Lastly the image pixels are divided into two equal arrays vertically of the image. The white pixel in the array has a value of 255. Which means that when summing up all the pixel value in left side of the image and in the right side it will indicate if the lane marking is good or bad. Since the lane marking to the left has a different pattern than the lane marking to the right, different thresholds are required for correct results. The code for splitting the image into two arrays and sum the values is shown below.

```
1 left , right = np.hsplit(thresh_HLS , 2)
2 counts_left = np.sum(left == 1, axis=0)
3 counts_right = np.sum(right == 1, axis=0)
4 total_left = math.fsum(counts_left)
5 total_right = math.fsum(counts_right)
```


4.7.1 Determine Lane Mark Length

To determine the lane mark length or give an indication of the lane mark quality a conversion between image pixels to world coordinates can be calculated. For further work on lane mark quality the area of the lane mark can be calculated. The camera used in this method was the camera included with the up2. The camera was calibrated in MATLAB with the calibration tool for 2D cameras. A checkerboard with $20 \times 20 \text{ mm}$ squares was used to perform the calibration. It was recommended to use 10 - 20 images from different locations pointing on the checkerboard. The group provided 25 images to improve the result. The calibration file for the camera was saved and included to the script for measuring the lines. The camera model was assumed to be a pinhole camera model. To calculate the length of the line, focal length and the camera lens center from the camera is required. Focal length and camera lens center are provided in the calibration file. The test setup is shown in Figure 4.47. The geometry that can determine line length is shown in Figure 4.48.



Figure 4.47: Test Setup Line Length

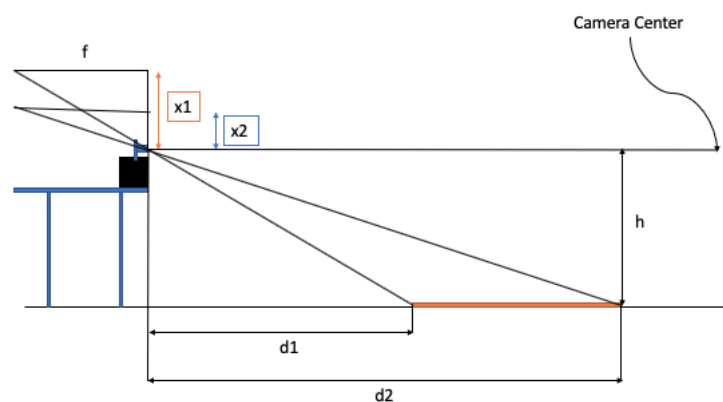


Figure 4.48: Geometrical Model to Determine Line Length

To ensure better results, the angle between camera and floor should equal zero. Therefore, a leveler was used to make the angle difference as small as possible. In Figure 4.49, 4.50, 4.51 and 4.52 the floor and camera level is shown.



Figure 4.49: Floor Level 1



Figure 4.50: Floor Level 2



Figure 4.51: Camera Level 1



Figure 4.52: Camera Level 2

The equations for calculating the length of the line are shown in Equation (4.7), (4.8) and (4.9).

$$d_1 = \frac{f}{x_1} \cdot h \quad (4.7)$$

$$d_2 = \frac{f}{x_2} \cdot h \quad (4.8)$$

$$L = d_2 - d_1 \quad (4.9)$$

Where:

f: Focal Length Camera

x_1 : Distance in Pixel From Camera Pixel Center

x_2 : Distance in Pixel From Camera Pixel Center

L: Line Length

h: Height From Floor to Camera Lens Center

d_1 : Horizontal Distance from Camera Lens to First Point on Line

d_2 : Horizontal Distance from Camera Lens to Second Point on Line

The MATLAB script for calculating the length is shown below. The line was moved further away to express the accuracy of the test. The Figures 4.53 and 4.54 shows the position of the line during the test.

```
1 I = imread('measure8.png');
2 undistorted = undistortImage(I, calibrationSession.CameraParameters);
3 marker = insertMarker(undistorted,[945.06325899310
   587.529936931676]);
4 marker = rgb2gray(marker);
5 bw = imbinarize(marker, graythresh(marker));
6 bw = bwareaopen(bw, 7000);
7 imshow(bw)
8 f=1378;
9 l0=1181;
10 h=817;
11 d1 = (f/(898-587.529936931676))*h;
12 d2 = (f/(812-587.529936931676))*h;
```

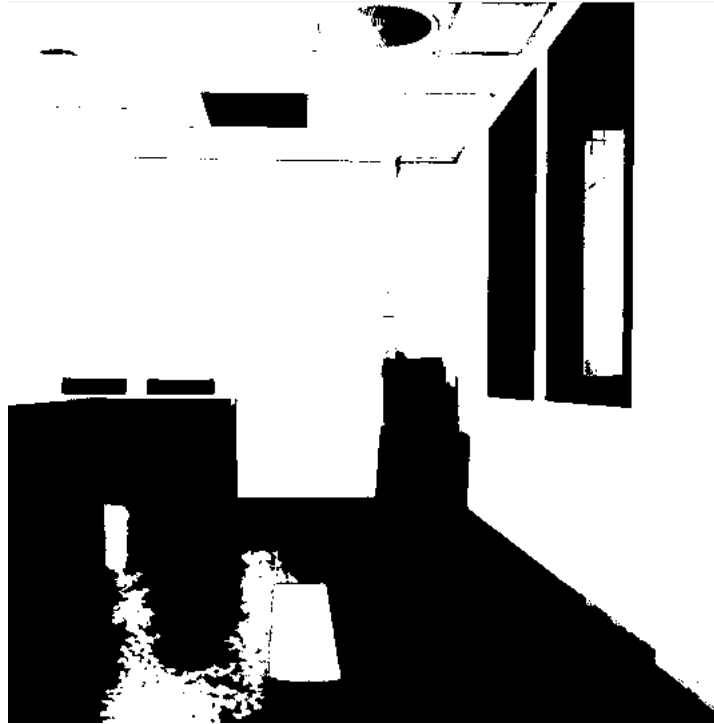


Figure 4.53: Binary Image - Line Length Test 1



Figure 4.54: Binary Image - Line Length Test 2

The coordinates x_1 and x_2 was measured manually with a cursor and the result of Test 1 and Test 2 is shown in Figure 4.55 and 4.56.

```
L =  
1.3544e+03  
  
p =  
1.1468
```

Figure 4.55: Result - Length Test 1

```
L =  
1.3893e+03  
  
p =  
1.1764
```

Figure 4.56: Result - Length Test 2

The results are in *mm* and the measurements has an offset of approximately 15%. There is provided an measured error due to angle offset in subsection 4.7.2. The measurements for the height are shown in Figure D.1, D.2 and D.3 in Appendix D.

4.7.2 Alignment Error on Line Length Measurement

The purpose of this section was to show how the measurement of a line is affected due to camera alignment relative to the line surface.

To show the measured error, the camera was placed in different angles. The first angle was 0 degrees relative to the surface where the lines are measured. A white sheet indicating the 0 degree line is shown in Figure 4.57 and the green line represents camera center in the image.

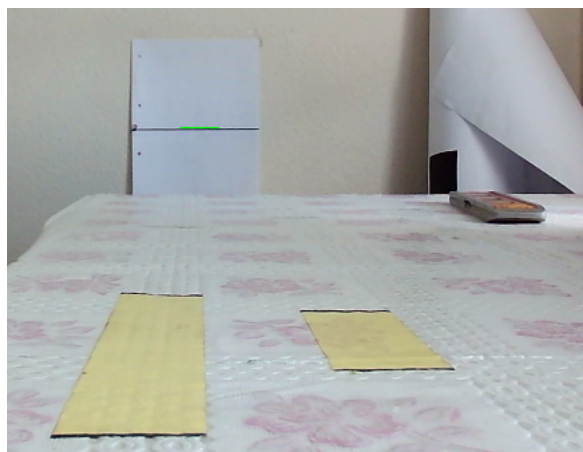


Figure 4.57: Camera Angle at 0 Degrees

There were two yellow line samples on the table and the size was different. The longest line was 200mm and the short line was 100mm long. The converted measurements are executed as in chapter 4.7.1, and the result of measuring the lines is provided in Figure 4.58.

Actual Short Line [mm]	Actual Long Line[mm]	Measured Short Line[mm]	Measured Long Line[mm]	Angle	Accuracy Short	Accuracy Long
100	200	82.63	165.1	0 degree	0.8263	0.8255
100	200	56.5566	114.3411	2.5 degree	0.5656	0.5717
100	200	39.19	78.3841	5 degree	0.3686	0.3919
100	200	103.4928	204.8678	Removed Spacer	1.0349	1.0243

Figure 4.58: Result with Different Angles

The camera angles are shown in Figure 4.59, 4.60 and 4.61.

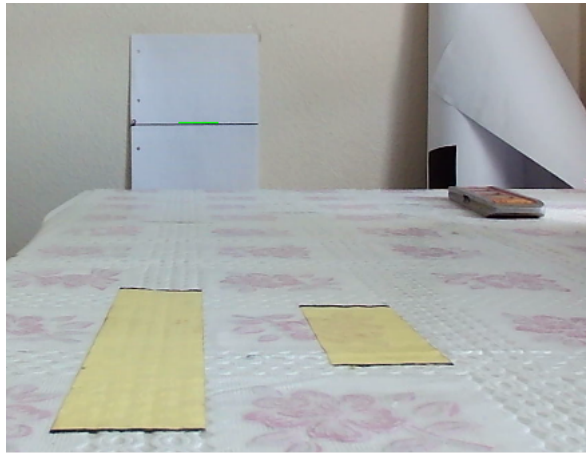


Figure 4.59: 0 Degree Angle

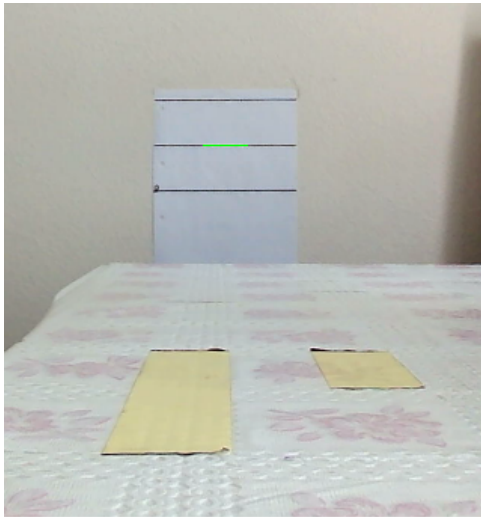


Figure 4.60: 2.5 Degree Angle



Figure 4.61: 5 Degree Angle

From the results, bad measurements occur if the camera angle was not 0 degrees relative to the measuring surface. The sensitivity was high. By having a 5 degree misalignment the accuracy was cut in half. There was used spacers to create different angles with the camera. At the end, the group did a test without the spacer and calculated the line length. The result was 98 % accurate, which means using this method for indicating if a line is in good condition is reliable if the camera angle is set correct relative to the measuring surface. The actual measurements of the lines and distance from the wall can be found in Figures D.4 and D.5 in Appendix D.

4.8 Lane Curvature, Curve Radius and Vehicle Position

The group decided to determine the curvature of the lane, the radius and the vehicle position. This was performed for continuation of this thesis. For example, detecting unexpected objects in the current lane. This thesis is a beginning towards a finished product which can be used for road monitoring, with several detectable parameters.

Furthermore, in this section the code for finding lane curvature, curve radius and vehicle position is explained. The code was fetched from Mohamed Ameen via *GitHub* and provided fully from there [57]. The group had to do adjustments to make it work. Although the group did not write this code, it was time consuming to make it work, since the author used different setup than the group. The script required time to understand. Below the code is partially described. The whole script can be seen in Appendix F.1.

In Figure 4.62 the function *display_Images* allows to show images. This function was used during the script and allows two images to be shown beside each other.

```
def display_Images(img1, img2, lbl1, lbl2, x, y, img3=[], lbl3=[], cmap=None, n=2):  
  
    plt.figure(figsize=(x, y))  
    plt.subplot(1, n, 1)  
    plt.imshow(img1, cmap=cmap)  
    plt.xlabel(lbl1, fontsize=15)  
    plt.xticks([])  
    plt.yticks([])  
    plt.subplot(1, n, 2)  
    plt.imshow(img2, cmap=cmap)  
    plt.xlabel(lbl2, fontsize=15)  
    plt.xticks([])  
    plt.yticks([])  
    if n == 3:  
        plt.subplot(1, n, 3)  
        plt.imshow(img3, cmap=cmap)  
        plt.xlabel(lbl3, fontsize=15)  
        plt.xticks([])  
        plt.yticks([])  
    plt.show()
```

Figure 4.62: Library Function - Display Images

Further the frames were warped from the dashboard view to the *Birds Eye View*. This was executed in function *Transform_Camera_View* which is shown in Figure 4.63. As seen the input was the frame, source image coordinates and the destination images coordinates. This function returns the warped image, the transformation matrix, and inverse transformation matrix. In Figure 4.64 the input image which was from E18 Grimstad is shown and the Figure 4.65 illustrates the warped image.


```
def Transform_Camera_View(img, src, dst):

    image_shape = img.shape
    img_size = (image_shape[1], image_shape[0])

    M = cv2.getPerspectiveTransform(src, dst)
    Minv = cv2.getPerspectiveTransform(dst, src)

    warped = cv2.warpPerspective(img, M, img_size)
    return warped, M, Minv
```

Figure 4.63: Library Function - Image Warping

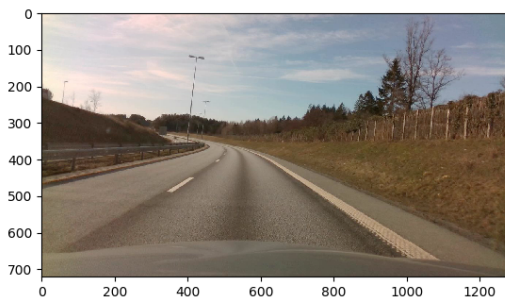


Figure 4.64: Input Image



Figure 4.65: Warped Image

Further the warped image was thresholded to the L-channel of the HLS color, which is the brightness. This function highlights white lane markings and then threshold it to binary. The inputs in this function were the warped image and low / high threshold. The function is shown in Figure 4.66. In Figure 4.67 the HLS image and the binary HLS image is shown.

```
def HLS_L_Threshold(img, thresh=(195, 255)):

    img = img[:, :, 1]
    img = img * (255 / np.max(img))
    binary_output = np.zeros_like(img)
    binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1
    return binary_output
```

Figure 4.66: Library Function - HLS Thresholding

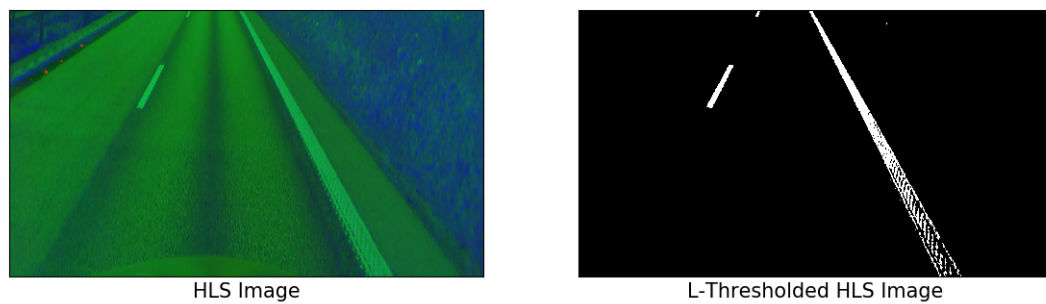


Figure 4.67: HLS Image and the Binary HLS Image

Then the warped image was again put into *LAB_B_Threshold* function. This function was thresholding the input image to the B-channel of the LAB color space. The B-channel is a color range scale which goes from -128(blue) to 128(yellow). This functions extract yellow lane markings. In Figure 4.68 the script is shown. Further the LAB image and the binary thresholded LAB image is shown in Figure 4.69. As seen, the image is black, this is due to no yellow lanes in input image.

```
def LAB_B_Threshold(img, thresh=(230, 255)):
    img = img[:, :, 2]
    if np.max(img) > 175:
        img = img * (255 / np.max(img))
    binary_output = np.zeros_like(img)
    binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1
    return binary_output
```

Figure 4.68: Library Function - LAB Thresholding

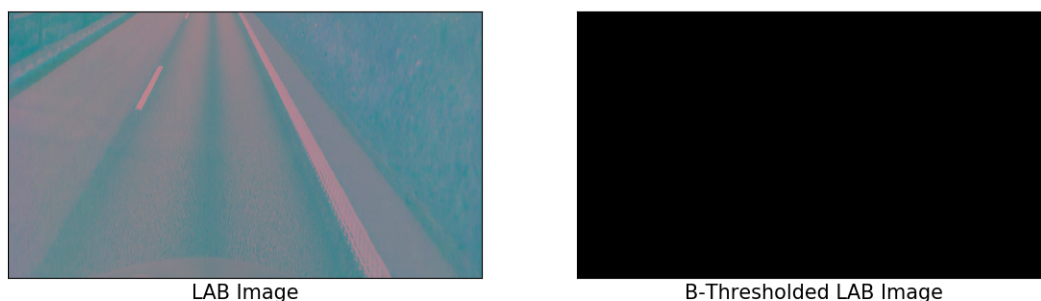


Figure 4.69: LAB Image and the Binary LAB Image

Further a function was made for combining the HLS threshold and the LAB threshold. This function is called *Combined_HLS_LAB_Threshold*, this basically combine both the binary thresholded image from HLS and LAB. The function can be viewed in Figure 4.70. In Figure 4.71 the warped RGB image is shown as well as the combined binary thresholded image.

```
def Combined_HLS_LAB_Threshold(img):  
  
    img_HLS = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)  
    img_LAB = cv2.cvtColor(img, cv2.COLOR_RGB2Lab)  
    img_thresh_HLS = HLS_L_Threshold(img_HLS)  
    img_thresh_LAB = LAB_B_Threshold(img_LAB)  
    combined_img = np.zeros_like(img_thresh_HLS)  
    combined_img[((img_thresh_HLS == 1) | (img_thresh_LAB == 1))] = 1  
    return combined_img
```

Figure 4.70: Library Function - LAB Combined with HLS Thresholding

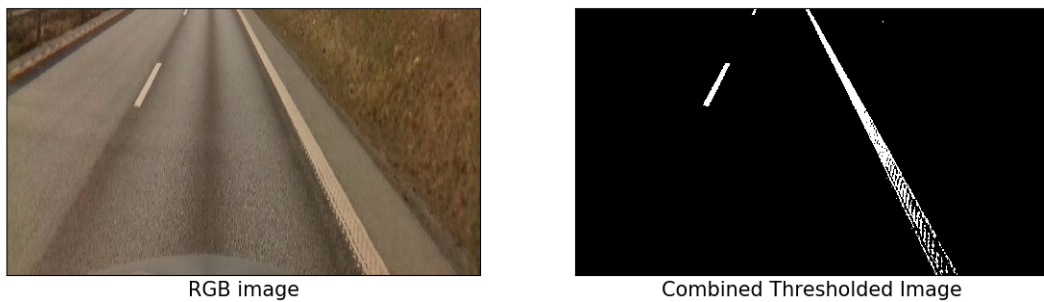


Figure 4.71: Warped RGB Image and the Combined Thresholded Image

Furthermore, to find the lanes a function called sliding window method was used. The script is shown in Figure 4.72 and 4.73. In the Figure it is described how the sliding window method works.

```

def Sliding_Window_Method(img):

    histogram = np.sum(img[img.shape[0] // 2:, :], axis=0)

    midpoint = np.int(histogram.shape[0] // 2)
    quarter_point = np.int(midpoint // 2)
    # Previously the left/right base was the max of the left/right half of the histogram
    # this changes it so that only a quarter of the histogram (directly to the left/right) is considered
    leftx_base = np.argmax(histogram[quarter_point:midpoint]) + quarter_point
    rightx_base = np.argmax(histogram[midpoint:(midpoint + quarter_point)]) + midpoint
    # Choose the number of sliding windows
    nwindows = 70
    # Set height of windows
    window_height = np.int(img.shape[0] / nwindows)
    # Identify the x and y positions of all nonzero pixels in the image
    nonzero = img.nonzero()
    nonzero_y = np.array(nonzero[0])
    nonzero_x = np.array(nonzero[1])
    # Current positions to be updated for each window
    leftx_current = leftx_base
    rightx_current = rightx_base
    # Set the width of the windows +/- margin
    margin = 80
    # Set minimum number of pixels found to recenter window
    minpix = 40
    # Create empty lists to receive left and right lane pixel indices
    left_lane_inds = []
    right_lane_inds = []
    # Rectangle data for visualization
    rectangle_data = []

```

Figure 4.72: Library Function - Sliding Window Method Part 1

```

# Step through the windows one by one
for window in range(nwindows):
    # Identify window boundaries in x and y (and right and left)
    win_y_low = img.shape[0] - (window + 1) * window_height
    win_y_high = img.shape[0] - window * window_height
    win_xleft_low = leftx_current - margin
    win_xleft_high = leftx_current + margin
    win_xright_low = rightx_current - margin
    win_xright_high = rightx_current + margin
    rectangle_data.append((win_y_low, win_y_high, win_xleft_low, win_xleft_high, win_xright_low, win_xright_high))
    # Identify the nonzero pixels in x and y within the window
    good_left_inds = ((nonzero_y >= win_y_low) & (nonzero_y < win_y_high) & (nonzero_x >= win_xleft_low) &
                     (nonzero_x < win_xleft_high)).nonzero()[0]
    good_right_inds = ((nonzero_y >= win_y_low) & (nonzero_y < win_y_high) & (nonzero_x >= win_xright_low) &
                      (nonzero_x < win_xright_high)).nonzero()[0]
    # Append these indices to the lists
    left_lane_inds.append(good_left_inds)
    right_lane_inds.append(good_right_inds)
    # If you found > minpix pixels, recenter next window on their mean position
    if len(good_left_inds) > minpix:
        leftx_current = np.int(np.mean(nonzero_x[good_left_inds]))
    if len(good_right_inds) > minpix:
        rightx_current = np.int(np.mean(nonzero_x[good_right_inds]))
    # Concatenate the arrays of indices
    left_lane_inds = np.concatenate(left_lane_inds)
    right_lane_inds = np.concatenate(right_lane_inds)
    # Extract left and right line pixel positions
    leftx = nonzero_x[left_lane_inds]
    lefty = nonzero_y[left_lane_inds]
    rightx = nonzero_x[right_lane_inds]
    righty = nonzero_y[right_lane_inds]
    left_fit, right_fit = (None, None)
    # Fit a second order polynomial to each
    if len(leftx) != 0:
        left_fit = np.polyfit(lefty, leftx, 2)
    if len(rightx) != 0:
        right_fit = np.polyfit(righty, rightx, 2)

    visualization_data = (rectangle_data, histogram)

return left_fit, right_fit, left_lane_inds, right_lane_inds, visualization_data

```

Figure 4.73: Library Function - Sliding Window Method Part 2

In Figure 4.74 the output of the sliding window function is shown.

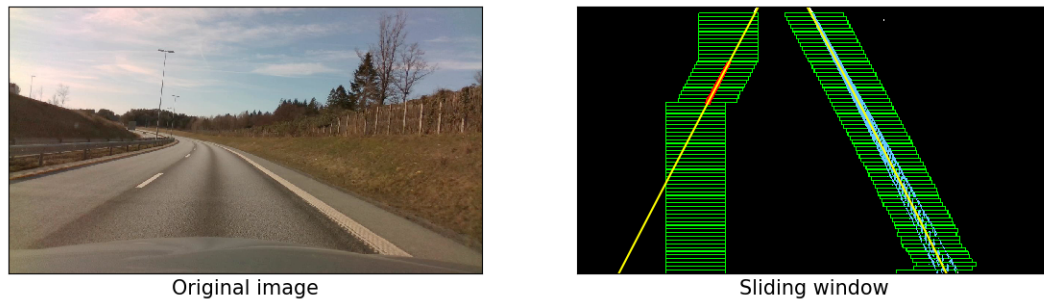


Figure 4.74: Sliding Window Method - Output

Additionally, a polynomial fit function was used to determine the curvature. This was done to the input binary image based on the previous fit, which makes an indication of the position of the lane markings in the frame. This function assumes that there is no significant change from one frame to another. The script can be seen in Figure 4.75, and the output from the polynomial fit function is showed in Figure 4.76.

```
def Polynomialfit_Previous_Fit(img, left_fit_prev, right_fit_prev):
    nonzero = img.nonzero()
    nonzeroy = np.array(nonzero[0])
    nonzerox = np.array(nonzero[1])
    margin = 80
    left_lane_inds = ((nonzerox > (left_fit_prev[0]*(nonzeroy**2) + left_fit_prev[1]*nonzeroy + left_fit_prev[2] - margin))
                    & (nonzerox < (left_fit_prev[0]*(nonzeroy**2) + left_fit_prev[1]*nonzeroy + left_fit_prev[2] + margin)))
    right_lane_inds = ((nonzerox > (right_fit_prev[0]*(nonzeroy**2) + right_fit_prev[1]*nonzeroy + right_fit_prev[2] - margin))
                    & (nonzerox < (right_fit_prev[0]*(nonzeroy**2) + right_fit_prev[1]*nonzeroy + right_fit_prev[2] + margin)))
    leftx = nonzerox[left_lane_inds]
    lefty = nonzeroy[left_lane_inds]
    rightx = nonzerox[right_lane_inds]
    righty = nonzeroy[right_lane_inds]
    left_fit_new, right_fit_new = (None, None)
    if len(leftx) != 0:
        left_fit_new = np.polyfit(lefty, leftx, 2)
    if len(rightx) != 0:
        right_fit_new = np.polyfit(righty, rightx, 2)
    return left_fit_new, right_fit_new, left_lane_inds, right_lane_inds
```

Figure 4.75: Library Function - Polynomial Previous Fit



Figure 4.76: Polynomial Previous Fit - Output

When the polynomial approach for the lane markings was found, the next step was to find the lane curvature and the position of the vehicle. In Figure 4.77 the script for the function *Curve_Position* can be seen.

```
def Curve_Position(img, l_fit, r_fit, l_lane_inds, r_lane_inds):
    """
    Calculating the lane curvature and the vehicle position on the lane.
    Parameters:
        img: Input image.
        l_fit, r_fit, l_lane_inds, r_lane_inds: Detected lane lines.
    """
    # Define conversions in x and y from pixels space to meters
    ym_per_pix = 3.048/100 # meters per pixel in y dimension, lane line is 10 ft = 3.048 meters
    xm_per_pix = 3.5/378 # meters per pixel in x dimension, lane width is 12 ft = 3.7 meters
    left_curverad, right_curverad, center_dist = (0, 0, 0)
    # Define y-value where we want radius of curvature
    # I'll choose the maximum y-value, corresponding to the bottom of the image
    h = img.shape[0]
    ploty = np.linspace(0, h-1, h)
    y_eval = np.max(ploty)

    # Identify the x and y positions of all nonzero pixels in the image
    nonzero = img.nonzero()
    nonzeroy = np.array(nonzero[0])
    nonzerox = np.array(nonzero[1])
    # Again, extract left and right line pixel positions
    leftx = nonzerox[l_lane_inds]
    lefty = nonzeroy[l_lane_inds]
    rightx = nonzerox[r_lane_inds]
    righty = nonzeroy[r_lane_inds]

    if len(leftx) != 0 and len(rightx) != 0:
        # Fit new polynomials to x,y in world space
        left_fit_cr = np.polyfit(lefty*ym_per_pix, leftx*xm_per_pix, 2)
        right_fit_cr = np.polyfit(righty*ym_per_pix, rightx*xm_per_pix, 2)
        # Calculate the new radii of curvature
        left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])
        right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])
        # Now our radius of curvature is in meters

    # Distance from center is image x midpoint - mean of l_fit and r_fit intercepts
    if r_fit is not None and l_fit is not None:
        car_position = img.shape[1]/2
        l_fit_x_int = l_fit[0]*h**2 + l_fit[1]*h + l_fit[2]
        r_fit_x_int = r_fit[0]*h**2 + r_fit[1]*h + r_fit[2]
        lane_center_position = (r_fit_x_int + l_fit_x_int) / 2
        center_dist = (car_position - lane_center_position) * xm_per_pix
    return left_curverad, right_curverad, center_dist
```

Figure 4.77: Curve Position - Function

The script for writing data and drawing the line indication can be seen in the full script in Appendix F.1.

4.9 Light Intensity Detection

To measure the light intensity at the roads, a photocell was used and wired as a light dependent resistor and placed at the dashboard. This means that the change in lux is directly related to the change in resistance in the photocell. The wiring diagram is shown in Figure 4.78.

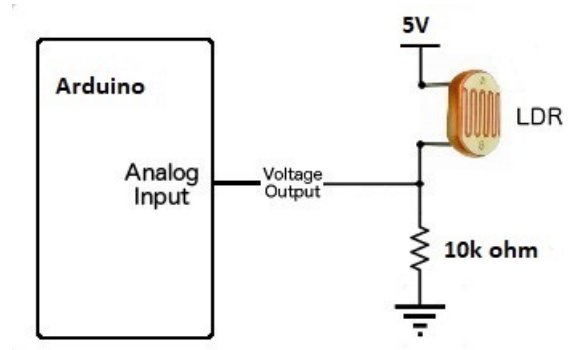


Figure 4.78: LDR Circuit Wiring

To get a correct measurement of the light surroundings a diffuser was made, since the light intensity gives a wrong indication of the lux without a diffuser. With a diffuser it creates a more flattering light and the lux measurement is more accurate.

A diffuser dome was used of a LED bulb which is shown in Figure 4.79. The LED bulb consist of hard plastic. By using this diffuser dome it allows more even measurements from the light surroundings results more realistic measurements.



Figure 4.79: Making Diffuser

To assemble the photocell and the diffusor, a frame had to be designed and 3D-printed. The 3D model of the diffuser frame is shown in Figure 4.80.

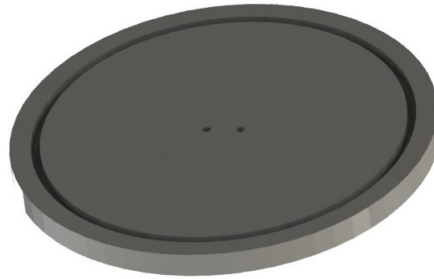


Figure 4.80: 3D Model of Diffuser Frame

In Figure 4.81 all the parts of the diffuser is shown before assembling. The diffuser consists of the diffuser dome, diffuser frame, photocell and wirings. The finished and assembled diffuser ready for use is viewed in Figure 4.82.



Figure 4.81: Diffuser Frame, Diffuser Dome and Photo Cell



Figure 4.82: Assembled Diffuser

As described in chapter 2.11.2, to approximate the light intensity of the surroundings, an equation describing the light intensity has to be determined. When the constants were identified, the equation was implemented to the Arduino controller and the lux was measured. The code for identifying the light intensity is shown below, which was written in Arduino.

```
1 #include <SoftwareSerial.h>
2
3 float v_in=5.0;
4 float v_out;
5 float ldr_voltage;
6 float r=10000.0;
7 float ldr_input = A0;
8 float ldr_value;
9 float rldr;
```

```

10 float lux_scalar = 210074123.7;
11 float exponent = -1.59366;
12 float ldrlux;
13 float i;
14 float res_voltage;
15
16 static const int RXPin = 4, TXPin = 3;
17 static const uint32_t GPSBaud = 9600;
18
19 TinyGPSPlus gps;
20 SoftwareSerial ss(RXPin, TXPin);
21
22 void setup()
23 {
24   Serial.begin(115200);
25   ss.begin(GPSBaud);
26 }
27
28 void loop()
29 {
30   ldr_value = analogRead(ldr_input);
31   res_voltage = ldr_value*v_in/1023;
32   ldr_voltage = v_in - res_voltage;
33   rldr = ldr_voltage/res_voltage * r;
34   ldrlux = (lux_scalar)*pow(rldr ,exponent)
35 }

```

The script reads an analog signal from the LDR circuit and converts the analog signal to identify the voltage over the 10k resistor. When the voltage over the 10k resistor is known, the voltage over the LDR sensor can be identified. From Equation (4.10) all parameters were known except the LDR resistor. By modifying the equation, the LDR resistor can be determined. Since the equation for the light intensity is based on the LDR resistance and constants, the determined resistance is further used to output the light intensity.

$$V_{Out} = V_{In} \cdot \frac{R_2}{R_{LDR} + R_2} \quad (4.10)$$

Where:

V_{Out} : Voltage output.

V_{In} : DC supply voltage.

R_2 : A resistor.[Ohm]

R_{LDR} : A Variable resistor / LDR. [Ohm]

4.10 Prototype Testing

The chosen area of focus is speed limit sign detection, lane mark detection, vibration measurement and light intensity including uploading information to a database with geospatial information. In Figure 4.83 the setup is shown. The vehicle that was used was a Volkswagen Transporter T7. The test route was along E18 from Grimstad to Nedenes.

To execute and start the program a monitor, keyboard and a mouse was necessary for prototype testing.



Figure 4.83: Test Setup

4.11 Vibration Measurements

When measuring the quality of the road, the IMU from the camera was used. To determine whether the road was in good or bad condition, the acceleration from the y -axis was recorded. The script is limited for the specific test vehicle. Different suspension gives different IMU results. Therefore, if the car is changed, thresholds must be updated. In Figure 4.84 the setup that was used is shown, including the cameras coordinate system.



Figure 4.84: Cameras Coordinate System

It was measured along the same route with the same velocity. The group recorded 7564 samples of the IMU which is approximately 120 seconds with a sample rate at 63. As seen from the Figures there was a difference in the output, since the suspension on the cars were different. In Figure 4.85 the car has a more soft suspension and in Figure 4.86 the suspension is more stiff, due to a relatively new car. As seen in the figures an older car with more soft suspension produces more noise. After approximately 95 seconds both cars detect the same unevenness at the road.

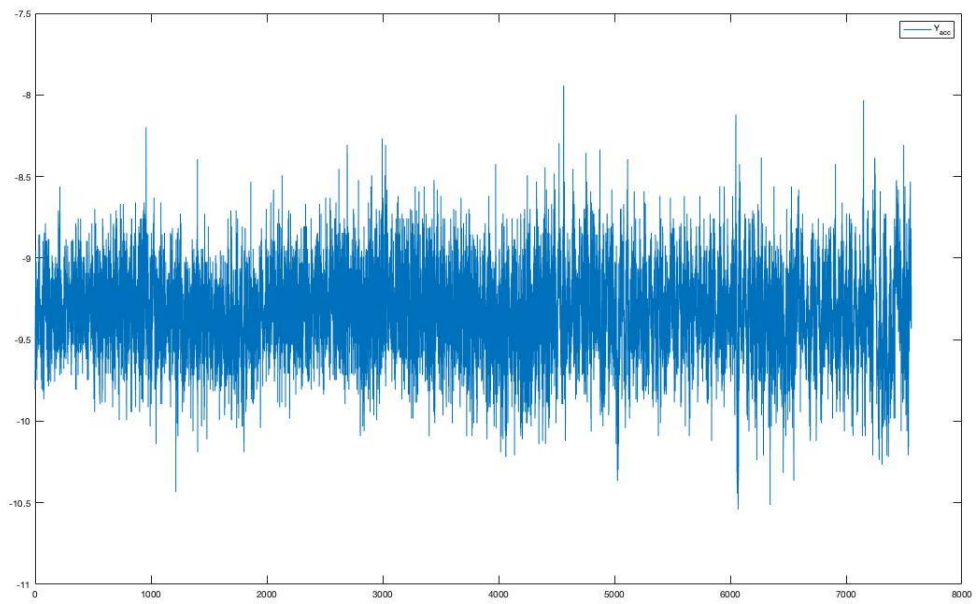


Figure 4.85: Measurement from Car with Soft Suspension

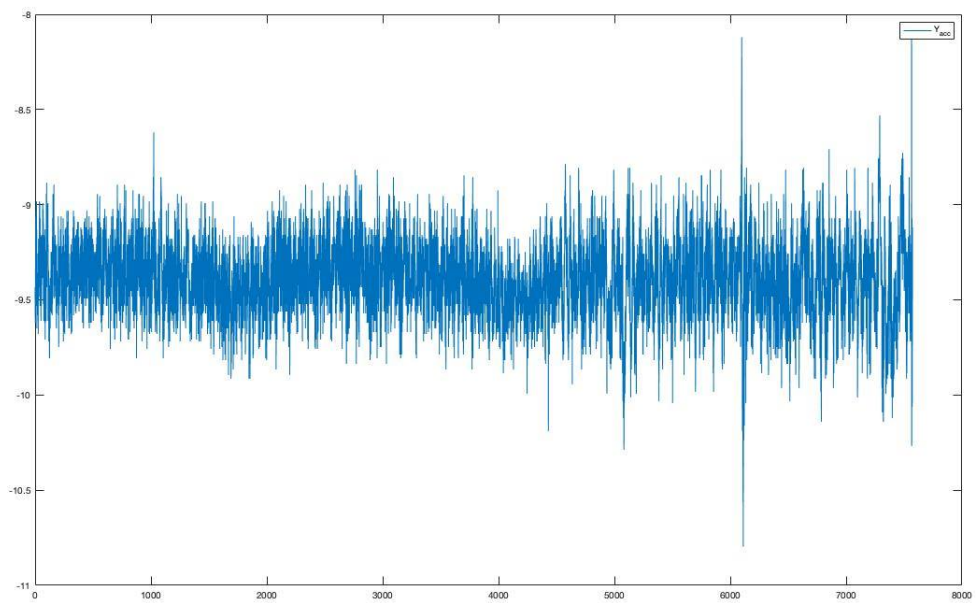


Figure 4.86: Measurement from Car with Stiff Suspension

4.12 Communication in ROS

To ensure communication between sensors and scripts, ROS was used. To be able to communicate, there must be provided subscribers and publishers. The subscribers are the scripts that receive the sensor data. The scripts that provide sensor data are the publishers. In this project Arduino was used to provide sensor data to the scripts. ROS has provided libraries to make sure that Arduino can be set as a node and a publisher or a subscriber. In this case the Arduino is a publisher. It was experienced that the Arduino runs slow when the libraries for GPS and ROS was included, the Arduino updated the location slow and it was not sufficient to run the GPS signal as a publisher from Arduino itself. The solution was to use a Python script to read the sensor values from the Arduino and to create a publisher from the Python script. The result was improved, and the publisher was running faster.

The script for reading the values from the Arduino, and creating a publisher from the Python script is shown in the code below.

```
1 import serial
2 import pycopg2
3 import rospy
4 from std_msgs.msg import Float64
5
6 #Connect to Serial Port
7 try:
8     arduino = serial.Serial('/dev/ttyACM0', 115200)
9     ok=1
10 except:
11     print("Check port")
12
13 def talker():
14     #Read Serial and Create a Publisher
15     latitude=rospy.Publisher('lat', Float64, queue_size=10)
16     longitude=rospy.Publisher('lng', Float64, queue_size=10)
17     lux = rospy.Publisher('lx', Float64, queue_size=10)
18     rospy.init_node('talker',anonymous=True)
19     rate = rospy.Rate(10)
20     while not rospy.is_shutdown():
21         gps_point = str(arduino.readline())
22         if ok==1:
23             try:
24
25                 lat = float(gps_point[0:9])
26                 lng = float(gps_point[10:18])
```

```

27         lx = float(gps_point[19:27])
28     except:
29         lat=0.0;
30         lng=0.0;
31         lx=0.0;
32         #print(lat , lng , lx)
33         rospy.loginfo(lat)
34         rospy.loginfo(lng)
35         rospy.loginfo(lx)
36         longitude.publish(lng)
37         latitude.publish(lat)
38         lux.publish(lx)
39         rate.sleep()
40 while True:
41
42     try:
43         talker()
44
45     except rospy.ROSInterruptException:
46         pass

```

For another script to access the publisher data, the script subscribes to the publisher node. The code below illustrates how the subscriber receives data from the publisher. The functions from the code below are implemented in the scripts that requests sensor data.

```

1 import rospy
2 from std_msgs.msg import Float64
3
4 def lat_callback(msg):
5     global lat
6     lat=msg.data
7
8 def lng_callback(msg):
9     global lng
10    lng=msg.data
11
12 while True:
13
14     try:
15         rospy.init_node('listener', anonymous=True)
16         rospy.Subscriber("lat", Float64, lat_callback)
17         rospy.Subscriber("lng", Float64, lng_callback)

```

```
18         rospy.spin()
19
20     except rospy.ROSInterruptException:
21         pass
```

The system was launched by a launch file. When the system was launched, it started a ROS environment. The environment starts a master node that registers all the created nodes. The master node controls which node subscribes, and which node publishes. Figure 4.87 shows the layout of the ROS environment.

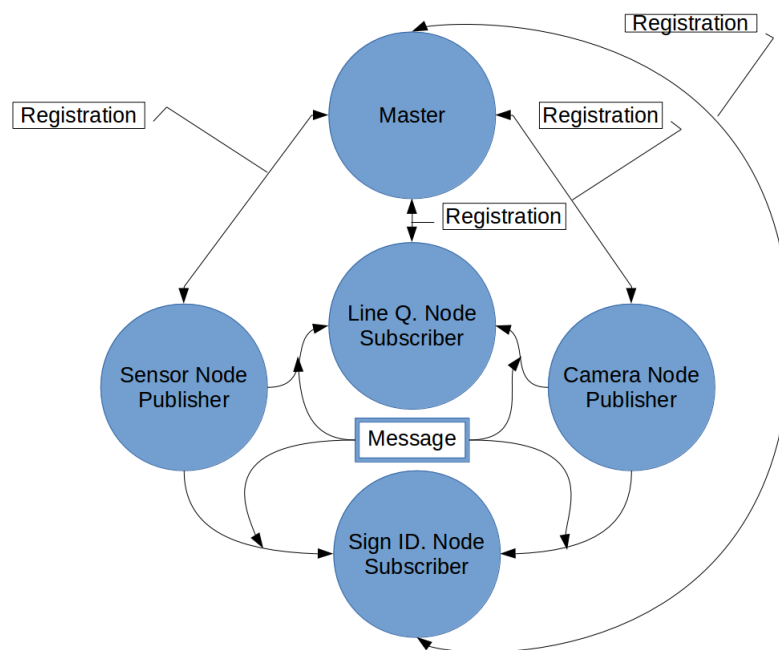


Figure 4.87: ROS Environment

5. Results

5.1 Traffic Sign Recognition

The sign recognition test was performed on E18, from Grimstad to Nedenes. The goal was to detect 90 signs on the highway. The sensor package detected 90 signs that occurred along the road and pushed geospatial information to the database. The sensor package saved the detected signs in a folder for later uploading to Dropbox. One of the detected 90 signs that were captured from Nedenes to Grimstad is shown in Figure 5.1, another 90 sign that was detected from Grimstad to Nedenes is shown in Figure 5.2



Figure 5.1: Detected 90 Sign Nedenes-Grimstad



Figure 5.2: Detected 90 Sign Grimstad-Nedenes

The sensor package detects other signs during driving. The detected signs are not pushed to the database since the template does not match a 90 sign. One example of a detected 50 sign during testing is shown in Figure 5.3. The package detected a 100 sign that is shown in Figure 5.4. Signs that are shown in Figure 5.3 and 5.4 was detected outside the testing range.



Figure 5.3: Detected 50 Sign After Testing



Figure 5.4: Detected 100 Sign After Testing

5.2 Vibration Measurements

The test for vibration measurements was performed on E18 from Grimstad to Nedenes. The script was set to detect high peaks during driving. The average value when the car was standing still was measured to be $-9.32 \frac{m}{s^2}$ in z -direction. The value was subtracted from the measured value. A peak was set to be ± 4.5 . The result is illustrated as a layer in QGIS and shown in Figure 5.5. One of the detected peaks is shown in Figure 5.6.



Figure 5.5: Detected Peaks in Z-direction



Figure 5.6: Detected Peak

5.3 Database

The detected states were GPS tagged and saved to the database. The information can later be illustrated as a layer in QGIS. The vibration measurements stored in the database are described as "BadRoad0000" because no frames are captured. The locations where the road was bad is shown in Figure 5.7. The information for the 90 sign detection is shown in Figure 5.8. The image info is set in a way to be recognized in Dropbox when inspecting the detected state.

	info character varying(255)	geom geometry(Point,4326)
1	Bad_Road000	0101000020E6100000CF126404540021401975ADBD4F274D40
2	Bad_Road000	0101000020E610000070951BEA0052140384D9F1D70274D40
3	Bad_Road000	0101000020E610000092CEC0C8CB0221401E1B81785D274D40
4	Bad_Road000	0101000020E610000092CEC0C8CB0221401E1B81785D274D40
5	Bad_Road000	0101000020E610000096986725AD00214071581AF851274D40
6	Bad_Road000	0101000020E610000096986725AD00214071581AF851274D40
7	Bad_Road000	0101000020E610000096986725AD00214071581AF851274D40
8	Bad_Road000	0101000020E6100000D8B79388F0FF2040EA245B5D4E274D40
9	Bad_Road000	0101000020E61000008E06F0164800214003B5183C4C274D40
10	Bad_Road000	0101000020E610000032C7F2AE7A002140AF264F594D274D40
11	Bad_Road000	0101000020E6100000F6D214014E47214022516859F72F4D40
12	Bad_Road000	0101000020E6100000F6D214014E47214022516859F72F4D40
13	Bad_Road000	0101000020E6100000F6D214014E47214022516859F72F4D40
14	Bad_Road000	0101000020E6100000F6D214014E47214022516859F72F4D40
15	Bad_Road000	0101000020E6100000F6D214014E47214022516859F72F4D40
16	Bad_Road000	0101000020E6100000B3295778973B2140D3C1FA3F872F4D40
17	Bad_Road000	0101000020E610000007EBFF1CE63B21406EDFA3FE7A2F4D40
18	Bad_Road000	0101000020E61000002383DC4598422140304AD05FE82F4D40
19	Bad_Road000	0101000020E610000092770E65A842214097AB1F9BE42F4D40
20	Bad_Road000	0101000020E610000092770E65A842214097AB1F9BE42F4D40
21	Bad_Road000	0101000020E610000081E9B46E8342214014CD0358E42F4D40
22	Bad_Road000	0101000020E610000081E9B46E8342214014CD0358E42F4D40

Figure 5.7: Pushed Road Condition Data to PostgreSQL

13	0101000020E6100000CF488446B0592140909F8D5C37334D40	90pic001.jpg
14	0101000020E6100000CF488446B0592140909F8D5C37334D40	90pic002.jpg
15	0101000020E61000005EBEF561BD5921400BF148BC3C334D40	90pic003.jpg
16	0101000020E61000005EBEF561BD5921400BF148BC3C334D40	90pic004.jpg
17	0101000020E61000005EBEF561BD5921400BF148BC3C334D40	90pic005.jpg
18	0101000020E61000007B32FFE89B64214077A04E7974354D40	90pic006.jpg
19	0101000020E6100000C861307F856421401900AAB871354D40	90pic007.jpg
20	0101000020E6100000C861307F856421401900AAB871354D40	90pic008.jpg

Figure 5.8: Pushed Sign Data to PostgreSQL

5.4 QGIS

QGIS was used to show where the detected states occur. The map was imported as a WMS-service from kartkatalog.geonorge.no. The map used was *Topografisk Norgeskart graatone*. This is a topological map of Norway in grayscale. Detected states are dragged over the topological map and the different colors represent different states. The result of detected 90 sign has green points and shown in Figure 5.9. The detected peaks by the IMU has purple points and shown in Figure 5.10.



Figure 5.9: 90 Sign Illustrated in QGIS



Figure 5.10: IMU Peaks Illustrated in QGIS

By enlarging the image in QGIS, and clicking on the GPS tagged point, the information about the state can be shown. In Figure 5.11, the enlarged probe of the image with point information is shown.

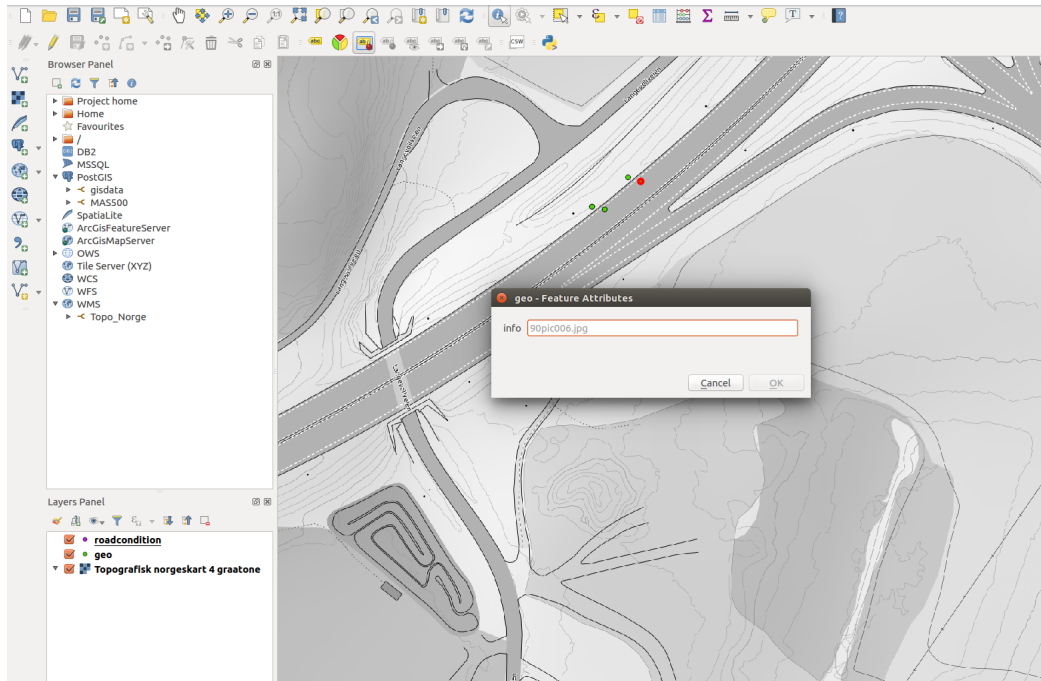


Figure 5.11: Information about GPS Point

5.5 Dropbox

Images stored in Dropbox was uploaded with ethernet internet connection after the testing. 4G module on the up2 does not work and therefore the upload to Dropbox was shown after the test run. The script for uploading images to Dropbox was run from the command prompt with ROS commands. The commands and printed uploading confirmation is shown in Figure 5.12.

```

upsquared@localhost:~/MAS500_ws$ roslaunch python skriptor uploader.py
/usr/local/lib/python2.7/dist-packages/requests/_internal.py:83: RequestsDependencyWarning: Old version of cryptography ([1, 2, 3]) may cause slowdown.
  warnings.warn(warning, RequestsDependencyWarning)
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic006.jpg to /90_sign/pic006.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic001.jpg to /90_sign/pic001.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic003.jpg to /90_sign/pic003.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic002.jpg to /90_sign/pic002.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic004.jpg to /90_sign/pic004.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic005.jpg to /90_sign/pic005.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic000.jpg to /90_sign/pic000.jpg
Uploading /home/upsquared/Desktop/detected_images/90_sign/pic007.jpg to /90_sign/pic007.jpg
upsquared@localhost:~/MAS500_ws$

```

Figure 5.12: Uploading Detected Images to Dropbox

By viewing the "Apps" folder on the local computer, the uploaded images are stored in the folder for the detected states. In this case the images were uploaded to the folder for 90 sign recognition. In Figure 5.13 the Dropbox folder for 90 sign detection is shown.

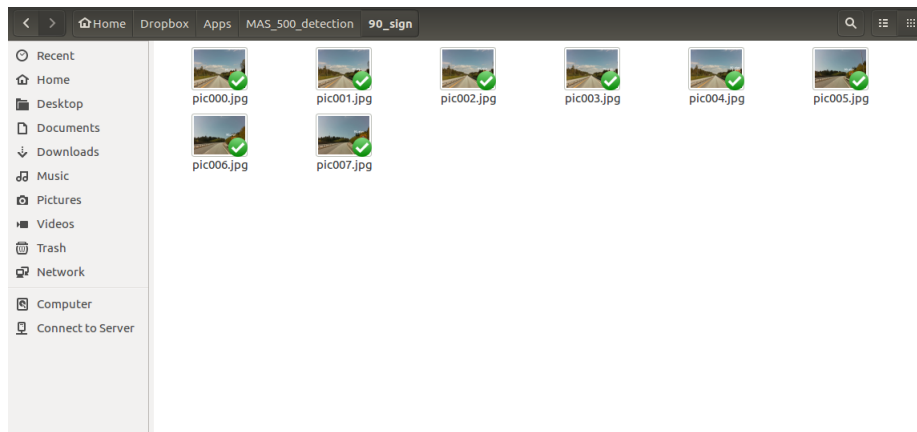


Figure 5.13: Uploaded Detected Images to Dropbox Folder

To access the images from another computer, the group logged into the Dropbox user account that was made for the up2. The images was found in the 90 sign folder and the confirmation is shown in Figure 5.14. There is important to notice that the image found in Dropbox folder and the information in describing the GPS tagged point has the same containment in the name. See the browser tab in Figure 5.14.

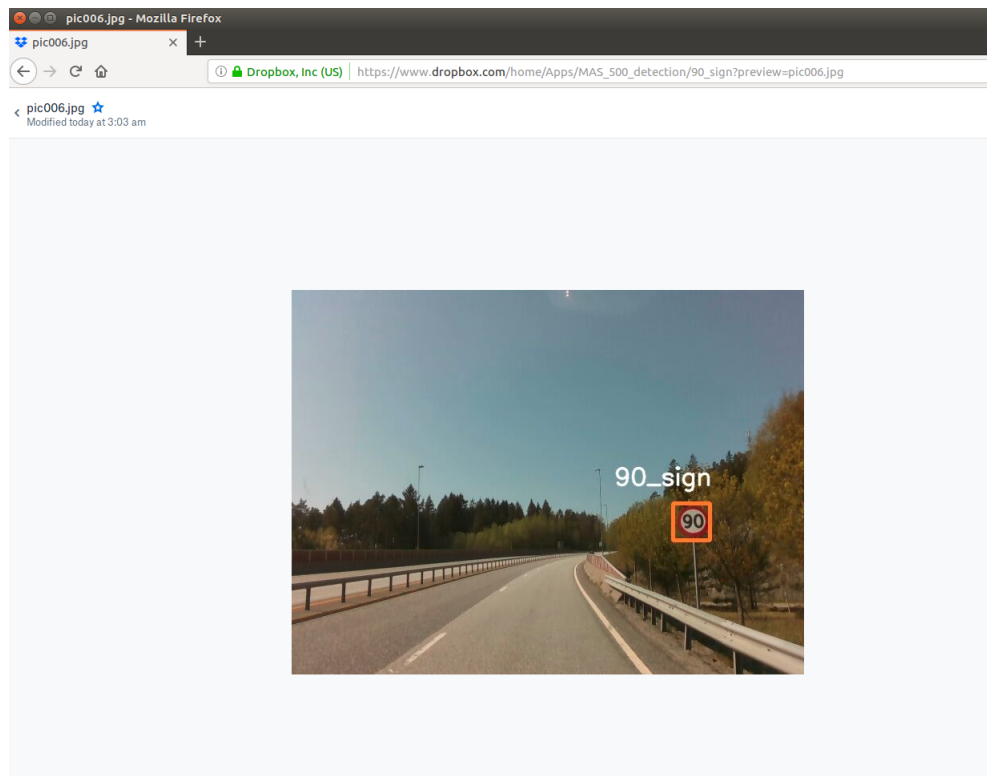


Figure 5.14: Detected Images to Dropbox From another Computer

5.6 Lane Mark Quality

The tests was along E18 from Grimstad to Kristiansand. Test setup was the same as for testing for speed limit sign recognition. The group captured screenshots during the testing of the frame, warped frame and the binary warped frame. Since the GPS sensor stopped working the group did not manage to test this algorithm properly.

As seen in Figure 5.15 the algorithm did manage to detect the lane marks. With a proper working GPS sensor and correct thresholds, the sensor package sends an image of the frame including GPS coordinates to the database when the sum of pixels is below the threshold.

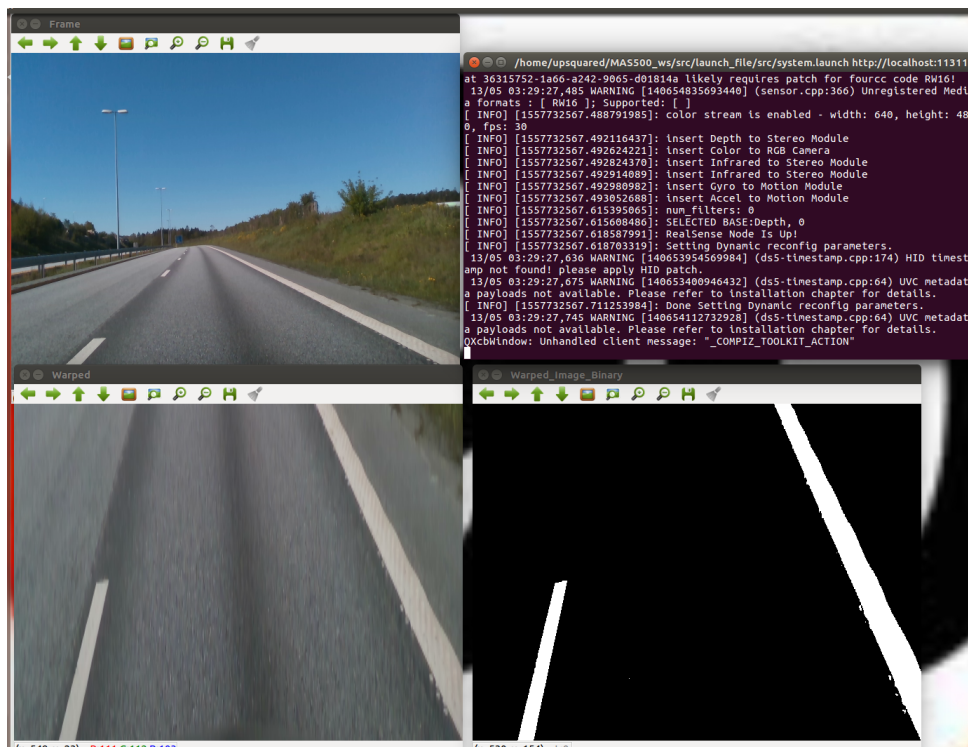


Figure 5.15: Lane Mark Detection Between Grimstad and Kristiansand

Bright daylight resulted in shadows from trees and lamp post, which affected the result. In Figure 5.16 the shadow from a lamp post crossed the lane mark, resulting that the sensor package did not manage to detect this area as a part of the line. This is shown in figure 5.16

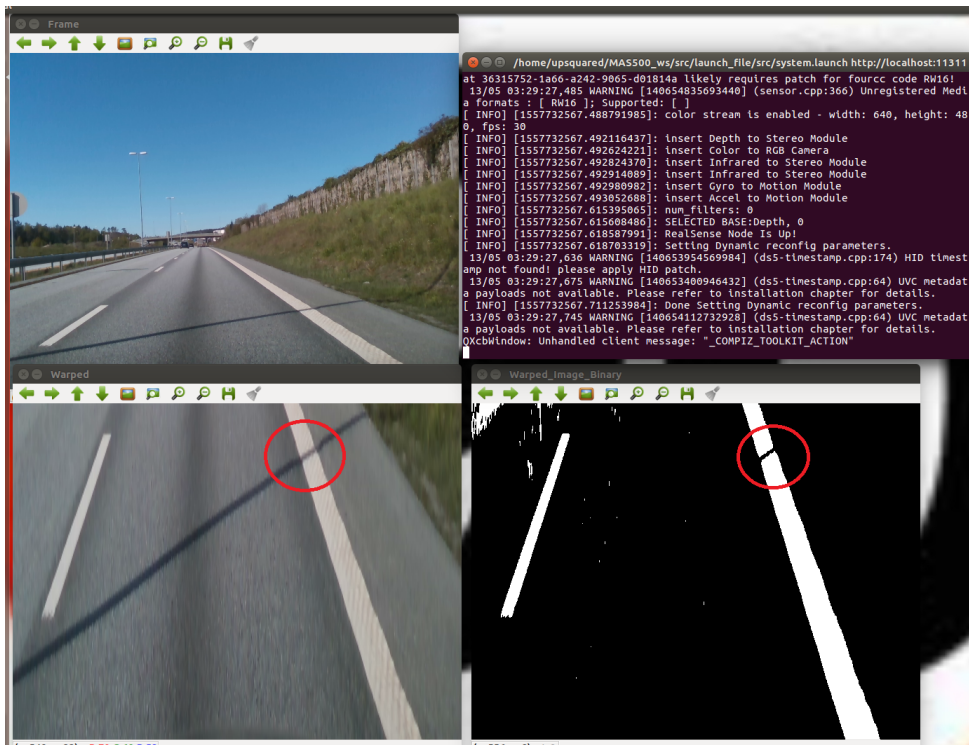


Figure 5.16: Shadow from Lamp Post Crossing the Lane Mark

Additionally when the vehicles from the left lane drove by and the shadow occurred the sensor package was not able to detect the lane marks, as seen in Figure 5.17.

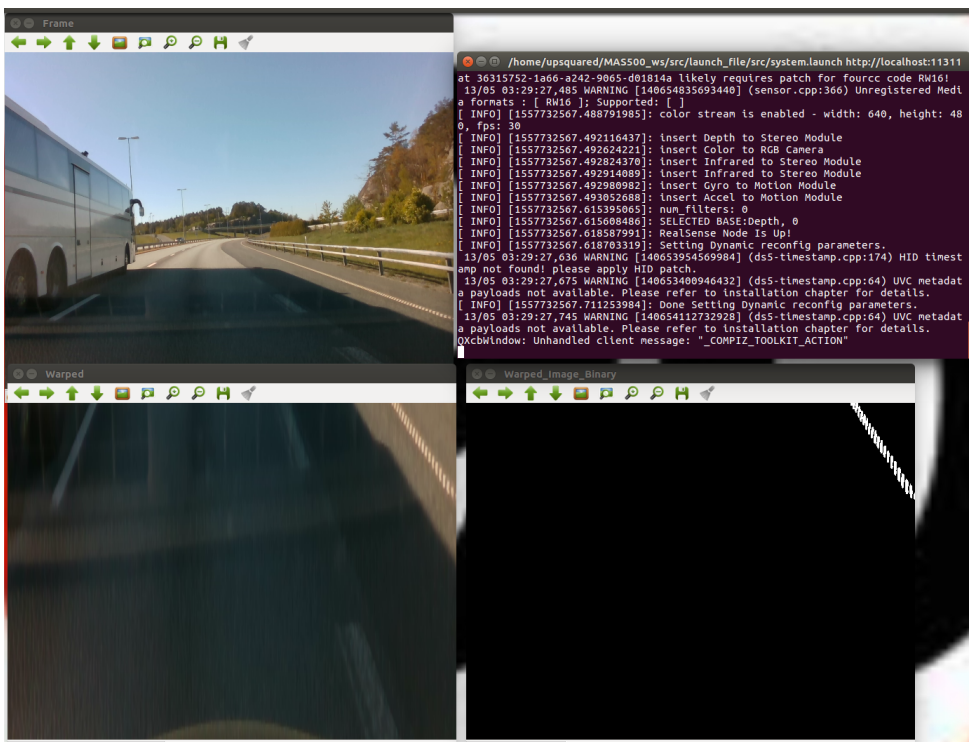


Figure 5.17: Lane Mark Detection - Shadows from Vehicles

In Figure 5.18 the lane is worn which make it brighter, resulting the pixel value to be below the threshold. This causes disturbance in the frame and difficult to determine the lane mark quality.

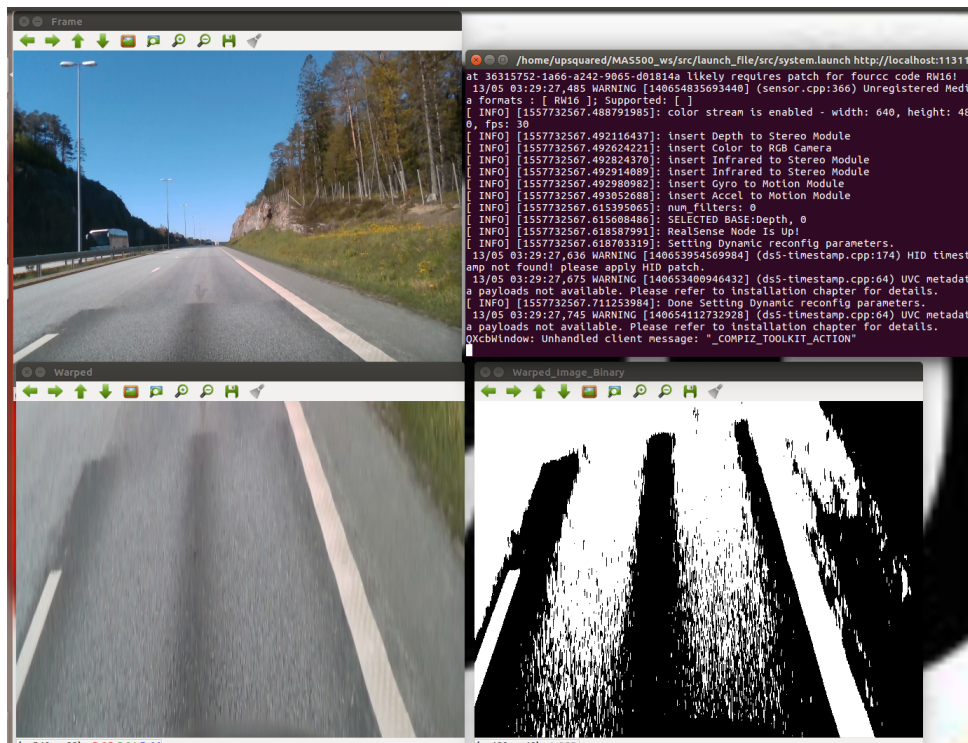


Figure 5.18: Lane Mark Detection - Worn Lane

5.7 Light Intensity Detection

The light intensity sensor was calibrated with a certified lux sensor as described earlier in the report, which means that the value the light intensity sensor can be considered correct. The sensor is sensitive, and the sensor varies several hundreds of lux over a short amount of time. During heavy raining it varies more than for example in sunny weather. In Figure 5.19 the test setup during heavy rain is shown.



Figure 5.19: Test Setup - Heavy Rain

The group did the testing at E18 near Grimstad on a route of 7.6 km which is illustrated on a map in Figure 5.20.

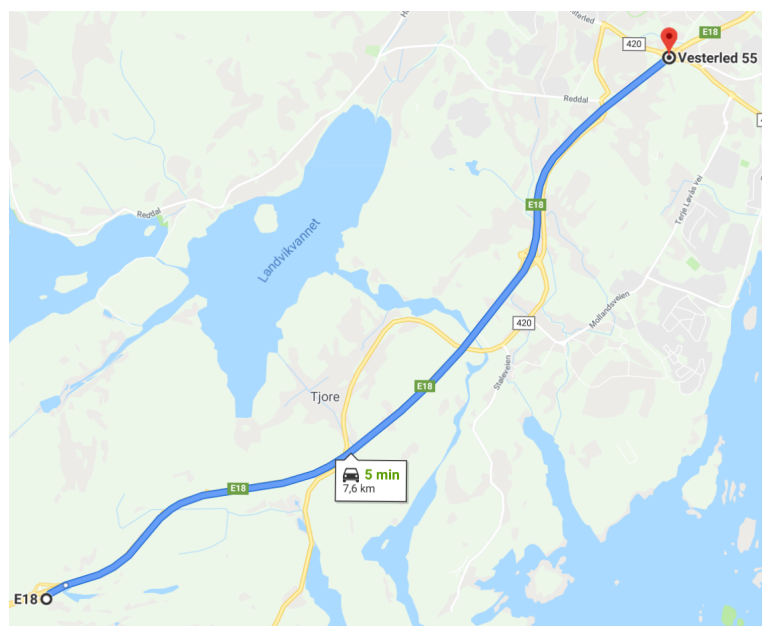


Figure 5.20: Test Route

The sensor package was outputting lux value every second. In Figure 5.21 a graph of the values is illustrated. As seen the lux varies due to the weather and the long distance traveled. In addition, the lux measurements can vary in terms of how the vegetation next to the highway.

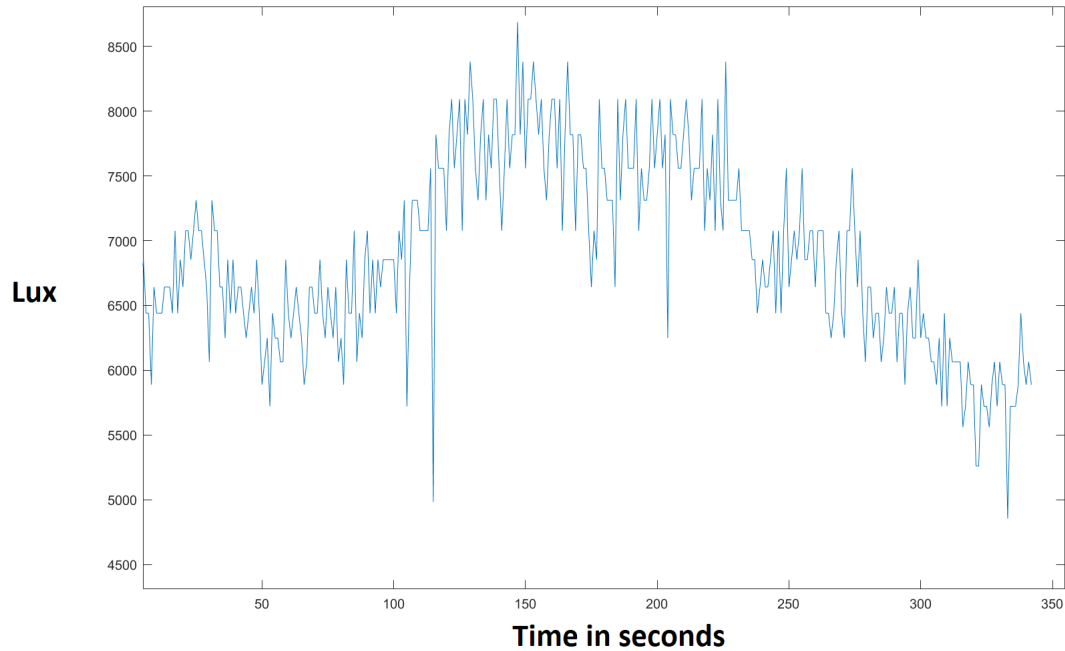


Figure 5.21: Light Intensity Measurements - Rainy Weather

Furthermore, the group tested when the weather was more stable. The test setup is shown in Figure 5.22 which also shows the sunny weather condition.

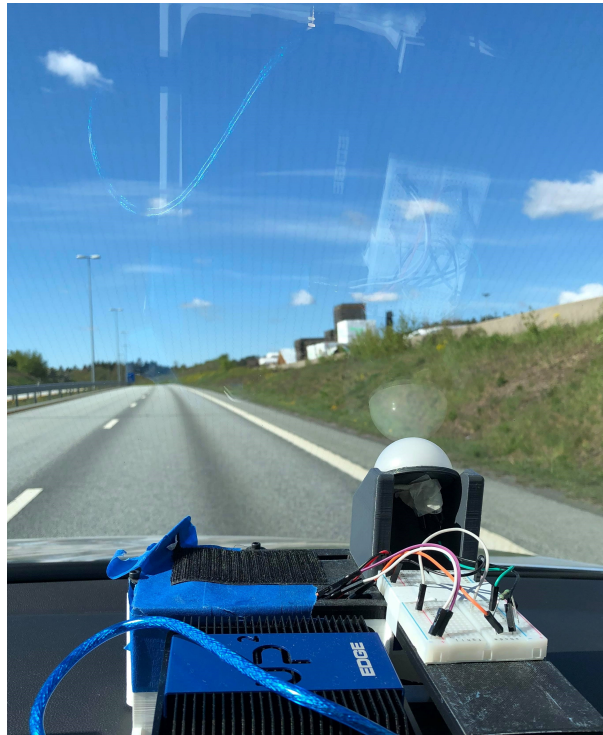


Figure 5.22: Test Setup - Sunny Weather

In Figure 5.23 the graph of the measurement in sunny condition is shown. The same route was driven as for measuring in rainy conditions. As seen in the figure the measurements are more stable. The lux is approximately around 20 000 lux on average. The drop at the graph around 270 seconds and at 320 seconds is caused by the vegetation that leaves shadow at the road.

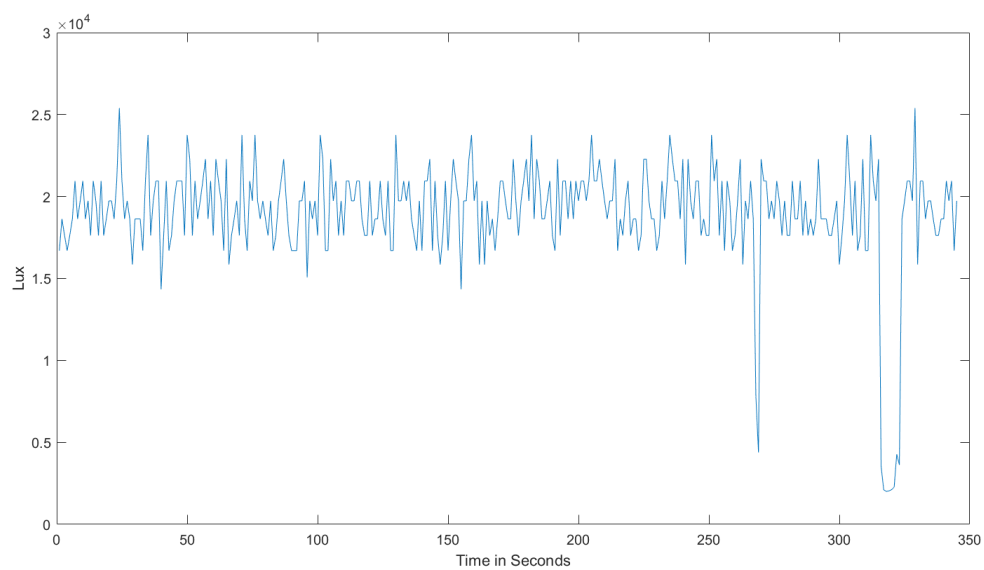


Figure 5.23: Light Intensity Measurements - Sunny Weather

5.8 Lane Curvature

The curvature of the lane, vehicle position and curve radius are written to the final output. In addition, the lanes are highlighted. In Figure 5.24 the final output is shown. *on-site* tests were not performed of the lane curvature algorithm. All though images were imported from the dashboard cam to run the algorithm which is illustrated in the figure below.

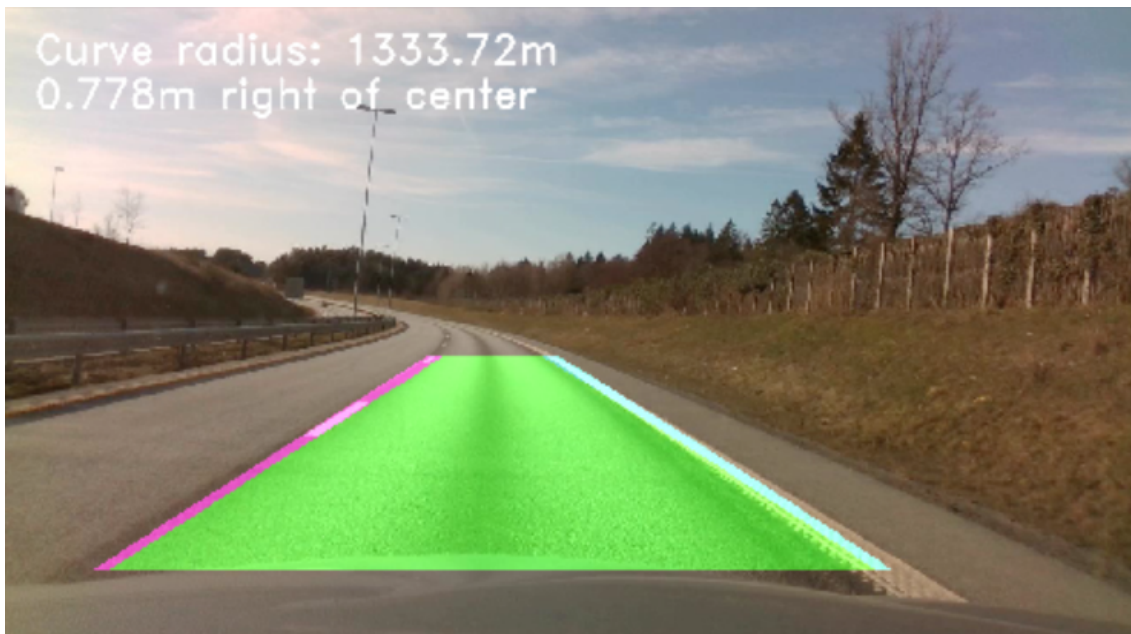


Figure 5.24: Final Output - Lane Detection

6. Discussion

6.1 Bright Conditions

During the testing different lightening conditions were an issue. During bright conditions, the light was intense and when light hits the camera lens, the camera experienced difficulties with detecting the speed limit signs. The red color becomes too dark and the color threshold in the script does not look for this type of color. In Figure 6.1 a state where the camera does not detect the sign is shown. Figure 6.1 can be compared with Figure 6.2. It is worth to mention that both figures are from the same test run.

The lane mark detection is affected by the lighting condition. When the lanes become old the asphalt gets brighter. The bright asphalt can result as noise when thresholding for bright objects in the lane, but it can give an indication for worn lanes. In Figure 6.3 the worn lanes are shown. When there was rain, the asphalt was darker than the lines and the problem is eliminated.



Figure 6.1: Dark Frame - Not Detected Sign



Figure 6.2: Detected Sign

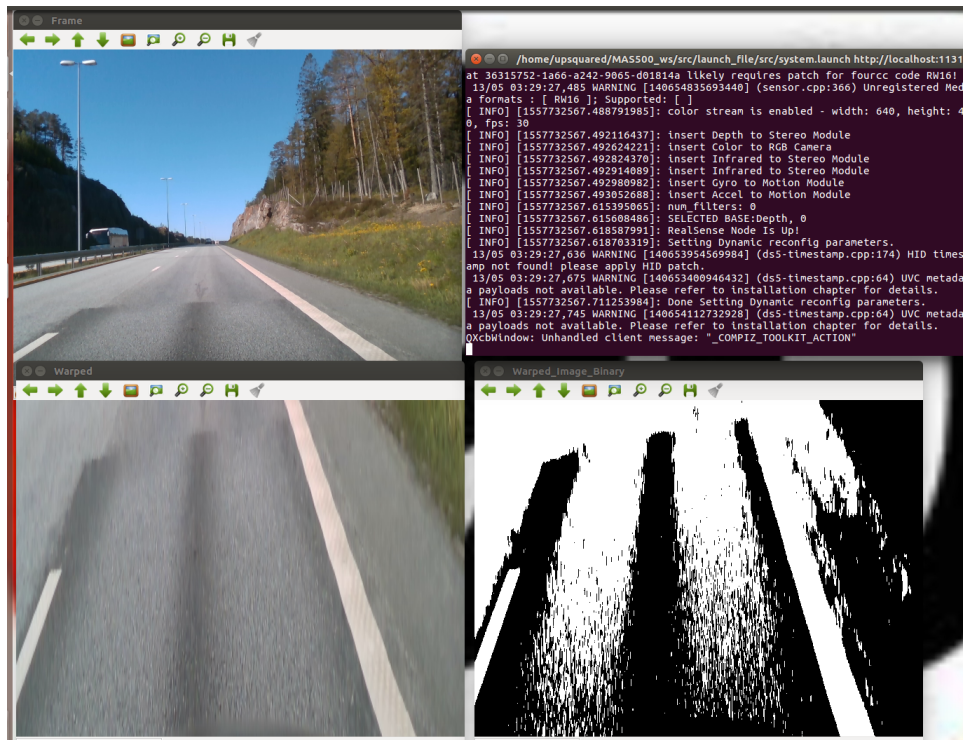


Figure 6.3: Lane Mark Detection - Worn Lane

One solution to the problem is to test with different filter for the camera lens. Polarizing filter is a good solution, since this filter is eliminating unwanted reflections during very light conditions. Additionally, it saturates the colors in a more normal way.

Since the camera has options for modifying the setup, a solution could be to modify the camera parameters for bright and dark lightning. The light intensity sensor can be used to distinguish between bright and dark surroundings. By determining the light condition with the light intensity sensor, the camera setup can be modified for correct surroundings.

6.2 Light Intensity Sensor

In the current design of the light intensity sensor, the diffuser is not placed inside a conical form, this makes the diffuser exposed for unwanted shadows from the surroundings. During testing the group experienced variations in the measurements if there was movement in the room or near the sensor package in the car. This is due to the shadows the diffuser experienced from a long distance. To make it less sensitive for the movement from the surroundings and make it more responsive to the actual light, the diffuser must be placed inside a conical form. An example of this is shown in Figure 6.4.



Figure 6.4: Diffuser with Conical Shape

6.3 Lane Curvature

The group did not perform tests of the algorithm with video for finding the lane curvature, only images taken from dashboard. Though, the group found it interesting due to further work. This algorithm requires more testing and tuning. When the lane curvature is found and the lane area is determined, the next step is to look for unwanted objects in this area. The lanes are the region of interest.

If the algorithm is looking for unwanted objects in the whole frame, the algorithm can detect objects outside the road shoulders. This is not relevant. By defining the region of interest, it can improve results and computational power is reduced.

The region of interest limits the algorithm to only search for unwanted objects in the lane the car is driving. Additionally, the region of interest can be adjusted to include the road shoulder. Due to safety the road shoulder should be free for objects that may cause danger for the traffic.

6.4 Camera Position Error

When the group measured the lane mark, both the camera and the object had to be aligned correctly to achieve great results. Optionally both cannot have the different offset. This affects the result in a negative way. It means that the rotation of the x -axis and y -axis of the camera needs to be the same as for the measured object. The setup and the coordinate systems are shown in Figure 6.5.



Figure 6.5: Coordinate System for Object and Camera

Therefore, when measuring in practice it is important to mount the camera in level with the road at the dashboard. The road surface can be assumed to be in level, except uphill and downhill. Some error in the measurements is to be expected when going uphill or downhill. To remove human measuring error and to improve the result, the mounting of the camera should be measured with a laser tracker. By using a laser tracker, the height is more accurately measured, but there is always a measurement error, but lower than measuring with handheld measuring equipment.

6.5 Power Supply

During testing a voltage converter from *Biltema* was used. This is not an ideal solution for the end product, since it converts 12V up to 230V and then the power supply for the up2 converts the 230V to 5V. All these conversions may cause disturbance and the system would not work properly. All tough, the group find this solution sufficient for the testing purposes. The voltage converter from *Biltema* is shown in Figure 6.6.



Figure 6.6: Voltage Converter from Biltema [24]

For completing the product, the group strongly recommended to find a better solution to a power supply. As mentioned earlier in the report, an appropriate power supply which directly converts 12V to 5V with enough power was found. Unfortunately, it was not possible to order one piece, the company required orders of several pieces. The Transporter that was used during testing have power supply in the cigarette contact when the car is turned off. In Figure 6.7 the *Biltema* converter is supplied with power while the key is not in contact. If the car does not have this option, an external power supply is recommended to have a safe system shutdown.

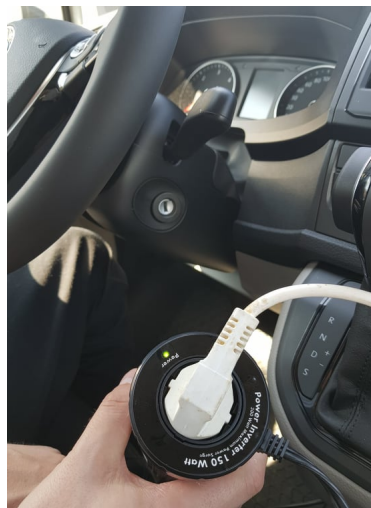


Figure 6.7: Power Supply with Ignition Off

6.6 IMU Coordinate System

When extracting information from the camera IMU, the group faced problems with the coordinate system. The coordinate system in the RealSense-Viewer application that is provided from Intel is not the same as in the Realsense package provided for the ROS environment. When measuring acceleration with Realsense-Viewer, the vertical axis is Y-axis. When extracting acceleration data using ROS, the vertical axis is Z-axis. In Figure 6.9 and 6.8 the difference in coordinate system is shown.

```
upsquared@localhost:~$ rostopic echo /camera/accel/sample
header:
  seq: 3414
  stamp:
    secs: 1556985529
    nsecs: 290985584
  frame_id: "camera_accel_optical_frame"
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 0.0
orientation_covariance: [-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.0
  y: 0.0
  z: 0.0
angular_velocity_covariance: [0.01, 0.0, 0.0, 0.0, 0.01, 0.0, 0.0, 0.0, 0.01]
linear_acceleration:
  x: -0.107873149216
  y: 0.245166242123
  z: 9.40457725525
linear_acceleration_covariance: [0.01, 0.0, 0.0, 0.0, 0.01, 0.0, 0.0, 0.0, 0.01]
---
```

Figure 6.8: Z-Axis for Vertical Measurement

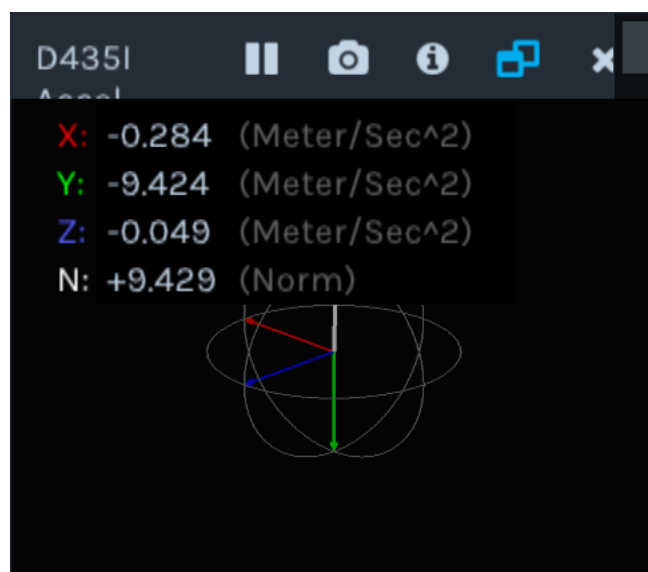


Figure 6.9: Y-Axis for Vertical Measurement

6.7 Machine Learning Vs. Template Matching

In the beginning of the project the group spent time on machine learning, training datasets and trying different machine learning techniques. Afterwards the group concluded that performing edge computing and training datasets required a powerful computer that exceeds the budget. Therefore, the options was to exceed the budget and buy a powerful computer, or buy a less powerful computer and rethink the process. The group approached the task in a more simplified direction and tried to find a way to use template matching to recognize the traffic signs instead of neural network.

6.8 Light Intensity Sensor - GPS Problem

During testing, when the vehicle entered tunnels, the GPS signal was lost. Since the light intensity meter runs on the Arduino with the GPS sensor, the light measuring stops.

One solution is to run the GPS sensor individually from other sensors, meaning that there can be used two Arduino micro controllers where the sensors are independent of each other. In that way the problem can be solved and prevent issues from the GPS signal.

6.9 ROS, QGIS and PostgreSQL

For the group, ROS, QGIS, Python and PostgreSQL was unknown topics before the thesis. The group lost time due to understanding the softwares and to get enough knowledge for project implementation. All though, the group find these softwares relevant for the project. If the group had basic knowledge about the softwares before thesis start up, better results could have been achieved.

6.10 Lane Mark Detection

The lane mark quality is determined by summing white pixels in the left and right side with a threshold. This method is not sufficient due to noise and car position. The estimation with pixel summation does not give an explanation about the actual line. The car can turn, and the line is out of the frame, the script would push wrong information to the database. By measuring the line with camera intrinsic parameters, a better condition of the line is provided. The line measurements can be performed in vertical and horizontal directions. Real measurement estimation would give a better understanding of the line quality.

6.11 Dropbox

Dropbox is only used as a simple solution during prototype testing. Dropbox is not an ideal solution for storing images. Services such as private storage facilities, MS Azure, Amazon Web Services etc are better suited to an actual implementation intended for production.

6.12 GPS Antenna

The GPS antenna had issues from the start. It used 10 to 15 minutes to receive signal from the satellites. When the connection was established, the GPS worked without faults. At the end of the project, the GPS module stopped receiving signals from the satellites. The power indication for the module is on, but not blinking. It is recommended to replace the antenna or buy a better GPS module.

7. Conclusion

Digitized road maintenance is an important topic due to increased efficiency of road maintenance. If the digitized road monitoring system can reduce manual road inspection, it may result in significantly lower maintenance cost.

Difficulties with present road maintenance is that most of the inspection is provided manually by humans and in some cases driving special made vehicles. Since the number of highways increases, more maintenance and inspection is required. As a result of increased number of highways with manual inspection, the cost increases. Therefore, by producing a compact and cost effective sensor package and by implementation in trafficking vehicles it can replace the special made vehicles for road inspection and monitoring.

By detecting states as traffic signs, lane mark quality and light condition with a sensor package, it results in a more effective road inspection. The stored data is pushed to a database where it is illustrated on a map service to show where the detected states are located. The sensor package updates the detected faults until the fault is repaired. For the tested conditions in Chapter 5 the sensor package gave positive results on sign detection, vibration measurements, storing and plotting data.

Some new vehicles are equipped with autonomous driving on the highways, where navigation using the lane marks is a key factor. By implementing the line length measuring to the sensor package, it can give an indication of the lane mark quality. The owner of the highway can predict when it is necessary to renew the lane marks.

The sensor package is not complete and requires more testing and implementation for perfection. This first version shows that it is possible to create a low cost and compact sensor package for implementation in a vehicular platform. Because the sensor package is compact, it simplifies the installation process and can be implemented in several vehicles. By increasing the computational power and utilizing the sensors maximum performance, the result can further be improved.

8. Suggestions for Further Work

8.1 Speed Limit Sign Recognition

To improve the algorithm and to make it more automatic, improvements can be done. At "Kartverket", the GPS locations for the speed limit signs are available. When the car is at the area where it is a speed limit sign, the sensor package runs the algorithm to detect if a sign is recognizable at this location. If the sensor package detects the signs no states are provided to the database. If the sign is not detected at the desired location, a probe of the location is pushed to the database to show the fault. This could improve the result instead of just letting the sensor package search for speed limit signs continuously.

8.2 Camera

The RealSense camera has several tuning parameters for the RGB camera. During this thesis the camera was set on "Auto". To manipulate the tuning parameters, the camera can get improved during bright conditions and the issue with dark signs would be solved.

The camera is a depth camera which has a camera depth topic available for further work. Since the camera is already integrated in ROS, the depth information can be provided from the camera node. The depth information can be used for scanning the road rails, detect worn tracks etc.

8.3 Hardware

To implement trained models for detecting different states, a more powerful computer is recommended to use. When using the up2 with the highest camera resolution activated, lag occurred. An example of a more powerful supercomputer is the Jetson Xavier Developer Kit provided by NVIDIA.

8.4 Light Intensity Algorithm

As described in the introduction chapter there is lightning requirements at the road and in the tunnels at different times of the day. An improvement could be to make an algorithm which determines the average lightning in tunnels and road. They could be set accordingly to the requirements and whether it is measuring along the road or in tunnels.

8.5 Touch Monitor for Up2

To have a complete product the sensor package should include a monitor to give an overview for the operator. The monitor should show that the sensor package is up and running without errors. The monitor should also be able to have a "Start" and "Stop" function which allows the driver to start the sensor package and shut it down in a secure way before the key is removed from the ignition. Additionally, the monitor should be able to abort and launch each of the algorithm, lane mark quality, vibration measurements, speed limit sign recognition and light intensity sensor.

An example could be a 7-inch touch monitor, which is possible to get hands on for a low cost.

8.6 Lane Mark Quality

For further work on this algorithm, the area of the lane marks can be determined and verified against standards. Since the length of lane marks now can be identified, next step is to find the lines in the frame with for example Canny Edge detection. Furthermore, when the length and width of the lane marks are known the area can be calculated.

8.7 Improve Algorithm for Bright Conditions

To improve the algorithm for bright conditions histogram normalization can be used. This algorithm normalize the pixel value from a desired image. The desired image has to be an image with ideal lightening conditions, since the algorithm uses this as reference.

Bibliography

- [1] Python Calibration Script - IMU Intel Real Sense 435i
<https://github.com/IntelRealSense/librealsense> [12.03.19].
- [2] Håndbok N500 Vegtunneler
https://www.vegvesen.no/_attachment/61913/binary/1143816?fast_title=H%C3%A5ndbok+N500+Vegtunneler.pdf [11.03.19].
- [3] ROS Overview
<https://robohub.org/ros-101-intro-to-the-robot-operating-system/> [03.04.19].
- [4] Image Acquisition
http://www1.idc.ac.il/toky/imageProc-08/lectures/02_acquisitionx4.pdf
[01.02.19].
- [5] Hue Color Circle - Image
https://www.quackit.com/css/color/values/css_hsl_function.cfm [08.04.19].
- [6] Canny Edge Detection - OpenCV
https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html [14.02.19].
- [7] Track Edges by Hysteresis - Image
<http://www.meccanismocomplesso.org/opencv-python-canny-edge-detection/>
[14.02.19].
- [8] Hough Transform
<http://matlabtricks.com/post-39/understanding-the-hough-transform> [14.02.19].
- [9] 2D Convolution
http://graphics.stanford.edu/courses/cs148-10-summer/docs/04_imgproc.pdf
[16.05.19].
- [10] Photo Conductive Sensor
<https://www.youtube.com/watch?v=ilN8XIK77dc&t=136s> [28.01.19].
- [11] Photo Conductive Sensor - Light Dependant Resistor
https://www.electronics-tutorials.ws/io/io_4.html [22.02.19].
- [12] GPS Receiver Explained
<http://www.physics.org/article-questions.asp?id=55> [28.01.19].
- [13] Infrared Temperature Sensor
<https://www.sensortips.com/temperature/infrared-temperature-sensor/>
[29.01.19].

- [14] Visualisation of Edge and Cloud Computing
<https://www.lanner-america.com/blog/4-edge-computing-technologies-enabling-iot-ready-network-infrastructure/> [21.02.19].
- [15] CPU Vs. GPU Cores
<http://www.adaltas.com/en/2018/07/24/deep-learning-tenserflow-yarn-hadoop/> [06.03.19].
- [16] From 3D World Coordinates to 2D Image Coordinates
https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/agartia3/index.html [21.02.19].
- [17] What is Camera Calibration?
<https://uk.mathworks.com/help/vision/ug/camera-calibration.html> [21.02.19].
- [18] Evaluating the accuracy of camera calibration
<https://www.mathworks.com/help/vision/examples/evaluating-the-accuracy-of-single-camera-calibration.html> [21.02.19].
- [19] VW Transporter Drawing
<http://www.just-tow.co.uk/towbars/volkswagen-tow-bars/volkswagen-transporter-towbars.html> [16.05.19].
- [20] Intel Real Sense 435i
<https://click.intel.com/intelr-realsensetm-depth-camera-d435.html> [07.02.19].
- [21] MLX90640 Thermal Camera
<https://shop.pimoroni.com/products/mlx90640-thermal-camera-breakout?variant=12549161746515> [07.02.19].
- [22] Heat Dissipation in Electrical Enclosures
http://www.hoffmanonline.com/stream_document.aspx?rRID=233309&pRID=162533&fbclid=IwAR1pmN1M505Iyw2uDN37wjpouXogYo1HUE5jpf_OxaZox97m7FB56VTkmgk [30.04.19].
- [23] HSV Color Range
<https://stackoverflow.com/questions/10948589/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-withcv> [20.03.19].
- [24] Voltage Converter from Biltema
<https://www.biltema.no/bil---mc/elektrisk-anlegg/kontakter-og-uttak-12-volt/spenningsomformer-2000030753> [10.05.19].
- [25] Håndbok V124 Teknisk planlegging av veg- og tunnelbelysning
https://www.vegvesen.no/_attachment/61499/binary/963994 [08.03.19].
- [26] Intel UP Squared AI Vision kit

- <https://up-shop.org/home/285-up-squared-ai-vision-x-developer-kit.html>
[29.01.19].
- [27] Sensor vehicular platform for road maintenance
<https://ieeexplore.ieee.org/document/7577114> [15.02.19].
- [28] An Automatic Road Distress Visual Inspection System Using an Onboard In-Car Camera
<https://www.hindawi.com/journals/am/2018/2561953/> [15.02.19].
- [29] ROADWATCH ROAD SURFACE TEMPERATURE SENSORS FROM M.S. FOSTER
<https://msfoster.com/products/winter-products/roadwatch-temperature-measuring-system/roadwatch-road-surface-temperature-sensors/> [15.02.19].
- [30] Measuring salt on road surfaces - A discussion of salt concentration versus salt amount
https://wiki.math.ntnu.no/_media/tma4850/2009v/feste_og_saltmengde_vs_saltskonsentrasjon.pdf [15.02.19].
- [31] Yutaka Suya. New road surface maintenance of expressway using an On-Vehicle salinity Sensor System which measures the salinity continuously. *Yamada Giken Co LTD*, 2014.
- [32] Adaption Luminance
<https://encyclopedia2.thefreedictionary.com/adaptation+luminance> [11.03.19].
- [33] Python Explained
<https://www.python.org/doc/essays/blurb/> [10.02.19].
- [34] QGIS - Software
<https://no.wikipedia.org/wiki/QGIS> [15.04.19].
- [35] PostgreSQL - Software
https://en.wikipedia.org/wiki/PostgreSQL#Storage_and_replication [15.04.19].
- [36] What is OpenCV?
<https://opencv.org/about.html> [28.01.19].
- [37] Arduino Uno R3
https://www.elfadistrelec.no/no/mikrokontrollerkort-uno-arduino-a000066/p/11038919?channel=b2c&price_gs=235&source=googleps&ext_cid=shg00aqnono-na&pup_e=1&pup_cid=35879&pup_id=11038919&gclid=Cj0KQCQiAzKnjBRDPARIsAKxfTRBJam0KwAwcwp_ou7rQU07Fu7TY4wDvrngW1HJal7hqkf5vKFkz_lsaAqXVEALw_wcB [18.02.19].
- [38] Image Processing
<https://www.engineersgarage.com/articles/image-processing-tutorial-applications> [04.02.19].
- [39] Image Segmentation

- https://en.wikipedia.org/wiki/Image_segmentation [04.02.19].
- [40] HSV and HSL Color Spaces
<https://codeitdown.com/hsl-hsb-hsv-color/> [08.04.19].
- [41] LAB - Color Space
<https://hidefcolor.com/blog/color-management/what-is-lab-color-space/>
[08.04.19].
- [42] Canny Edge Detector Explained
https://en.wikipedia.org/wiki/Canny_edge_detector [28.01.19].
- [43] Template Matching Basics
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html [06.03.19].
- [44] Gary Bradski & Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'REILLY Media Inc, 2008.
- [45] What Are Inertial Measurement unit(IMU)?
<http://students.iitk.ac.in/roboclub/2017/12/21/Beginners-Guide-to-IMU.html>
[28.01.19].
- [46] The Difference Between CPU and GPU
<https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/> [06.03.19].
- [47] Geospatial Information System
https://disruptivetechasean.com/big_news/geospatial-data-explained/
[20.02.19].
- [48] Calibration of Infrared Thermometer
<https://www.rdmag.com/article/2004/07/how-choose-ir-blackbody-calibration-source> [21.02.19].
- [49] IMU Intel Real Sense 435i Calibration
https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/RealSense_Depth_D435i_IMU_Calib.pdf [06.03.19].
- [50] Light Intensity Detection
<https://www.instrumentchoice.com.au/emails/Monthly%20Newsletter/10-10-17/how-does-a-lumen-lux-meter-work> [04.02.19].
- [51] Step Down Module
https://www.banggood.com/XH-M249-DC5V-6A-Step-Down-Module-12V24V-to-5V-Power-Supply-USB-Charging-5A-30W-p-1264861.html?rmmds=myorder&cur_

- warehouse=CN [16.05.19].
- [52] Create Workspace - ROS
http://wiki.ros.org/catkin/Tutorials/create_a_workspace [28.04.19].
- [53] Create Package - ROS
<http://wiki.ros.org/catkin/Tutorials/CreatingPackage> [28.04.19].
- [54] Dynamic Target Tool App - Calibration Grid
https://play.google.com/store/apps/details?id=com.intel.realsenseviewer17613&hl=en_US [13.03.19].
- [55] Intel RealSense Dynamic Calibrator Program - Download
<https://downloadcenter.intel.com/download/28517/Intel-RealSense-D400-Series-Calibration-Tools-and-API?product=128255> [13.03.19].
- [56] IMU Calibration Intel RealSense 435i
https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/RealSense_Depth_D435i_IMU_Calib.pdf [12.03.19].
- [57] Advanced Lane Finding Using OpenCV
<https://github.com/mohamedameen93/Advanced-Lane-Finding-Using-OpenCV> [26.03.19].
- [58] Intel RealSense D400 Series - Datasheet
https://www.mouser.com/pdfdocs/Intel_D400_Series_Datasheet.pdf [16.05.19].
- [59] MLX 90640 Thermal Camera - Datasheet
<https://cdn.sparkfun.com/assets/7/b/f/2/d/MLX90640-Datasheet-Melexis.pdf> [16.05.19].

Appendices



A. Cost

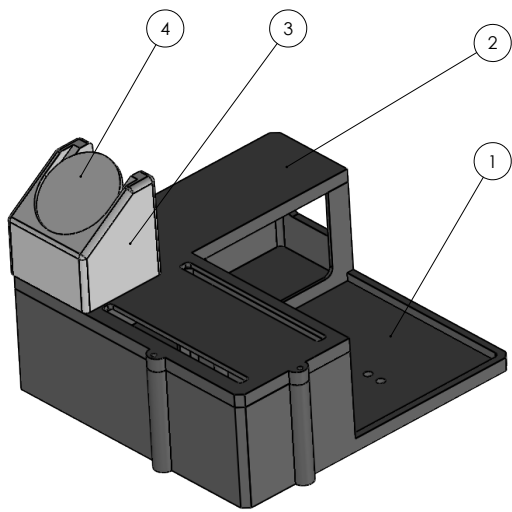
This estimate is accurate but may vary due to variation in exchange rate since several components are ordered abroad. The shipping cost from the dealers are not included in this estimate. Material such as wiring, and printing is not included as well.

Table A.1: Cost


Component	Price in NOK
Intel UP Ai Square kit 3G antenna	4640
3G Adapter	660
Photo Cell and diffuser	100
Intel RealSense D435i	1535
Arduino Uno R3	200
GPS Sensor	30
Joby Suction Cup	299
Step Down Module	110
Power Supply for Testing	299
Sum	7873

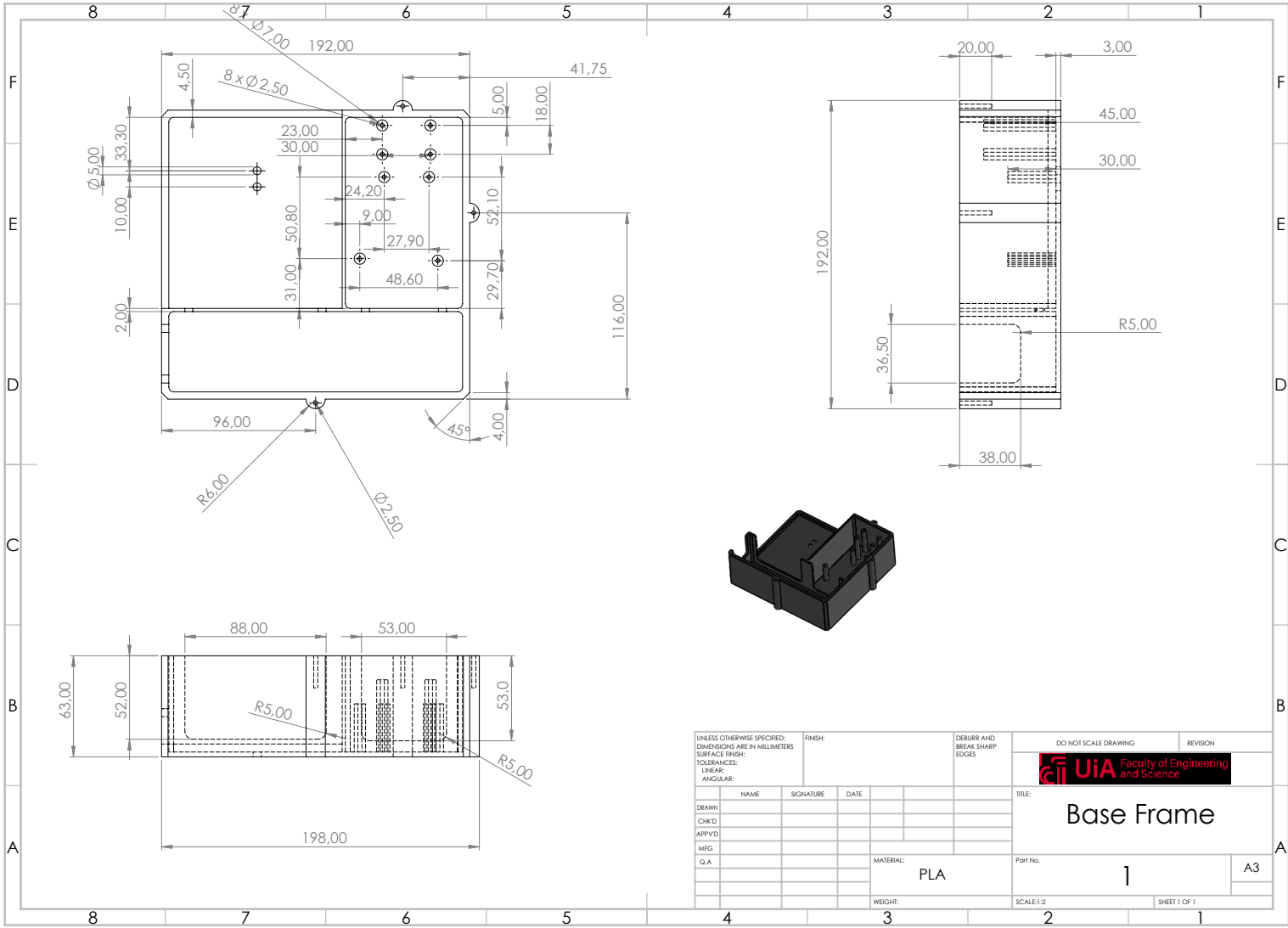
B. Solidworks Drawings


B.1 Sensor Package

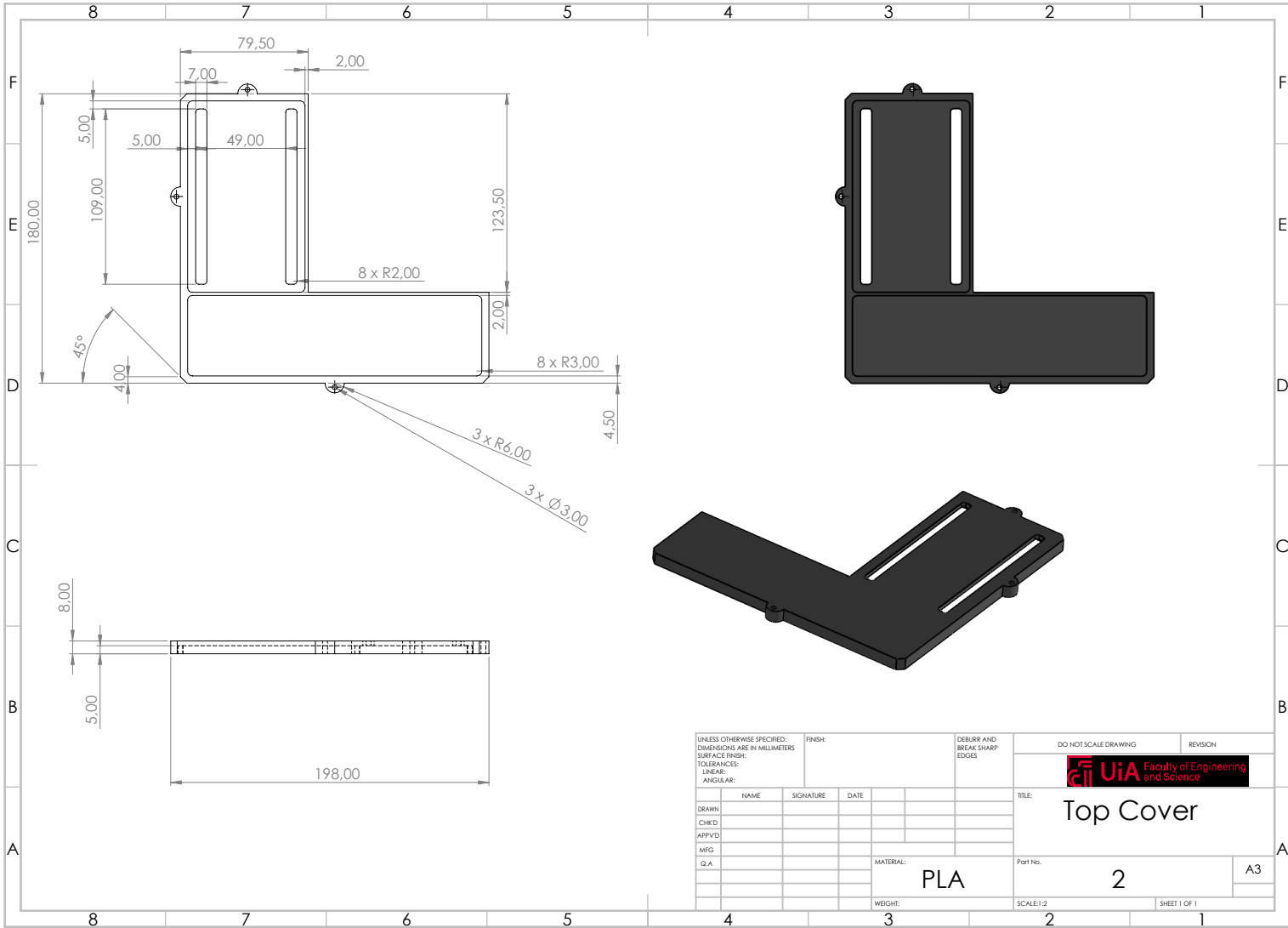


ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1		Base Frame	1
2		Top Cover	1
3		Bracket for Light Intensity Sensor	1
4		Diffuser Frame	1

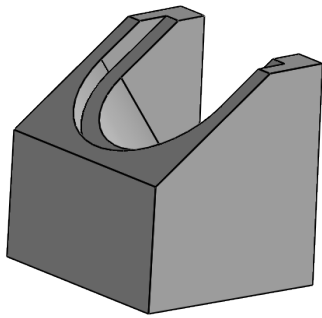
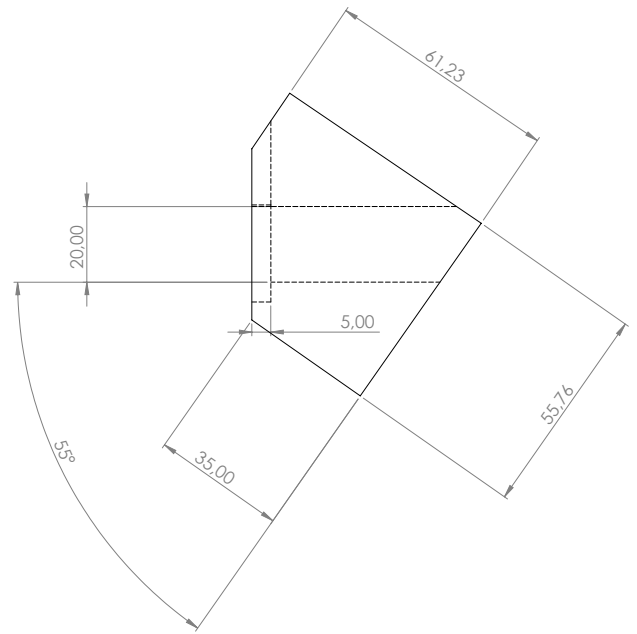
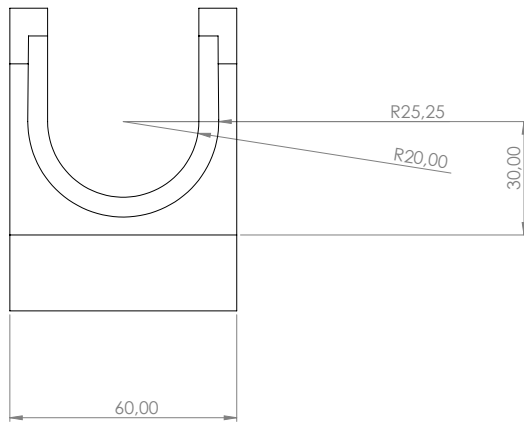
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:						 UiA Faculty of Engineering and Science			
TOLERANCES:								TITLE:	
LINEAR:						Sensor Package			
ANGULAR:						DWG NO.		A3	
DRAWN:	NAME	SIGNATURE	DATE			MATERIAL:		PLA	
CHKD:						WEIGHT:		3	
APPVD:						SCALE: 1:1		SHEET 1 OF 1	
MEG:									
Q.A:									




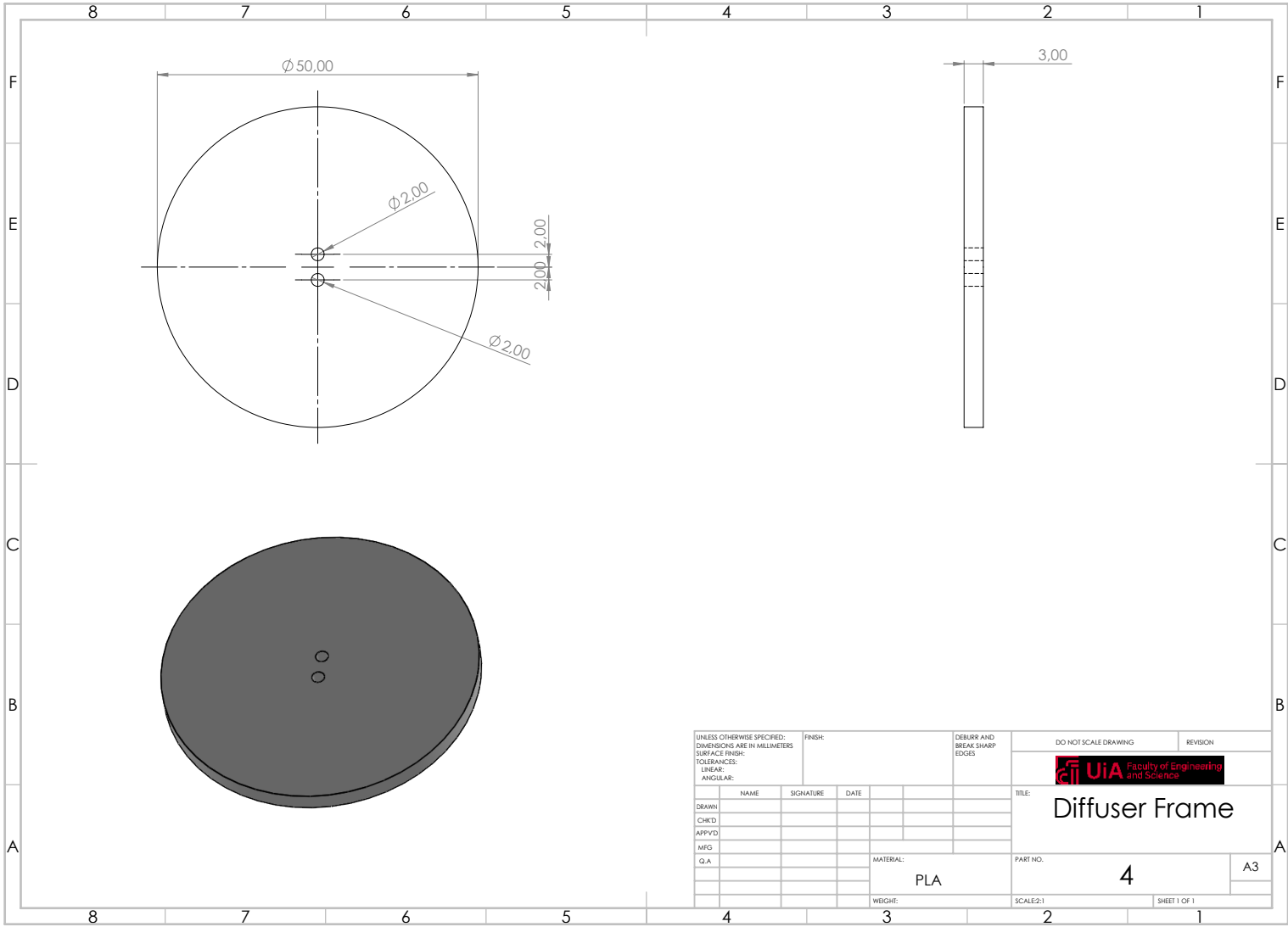
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:						 UiA Faculty of Engineering and Science			
TOLERANCES:								TITLE:	
LINEAR:								Part No. 1	
ANGULAR:								A3	
DRAWN:	NAME	SIGNATURE	DATE			MATERIAL:		PLA	
CHKD:						WEIGHT:		3	
APPVD:						SCALE:1:2		SHEET 1 OF 1	
MEG:									
Q.A:									




UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:									
TOLERANCES:									
LINEAR:									
ANGULAR:									
DRAWN:		NAME	SIGNATURE	DATE		TITLE:			
CHKD:						Top Cover			
APPVD:									
MFG:									
Q.A:									
		MATERIAL:		PLA		Part No.:		2	
		WEIGHT:		3		SCALE:1:2		A3	
						SHEET 1 OF 1			

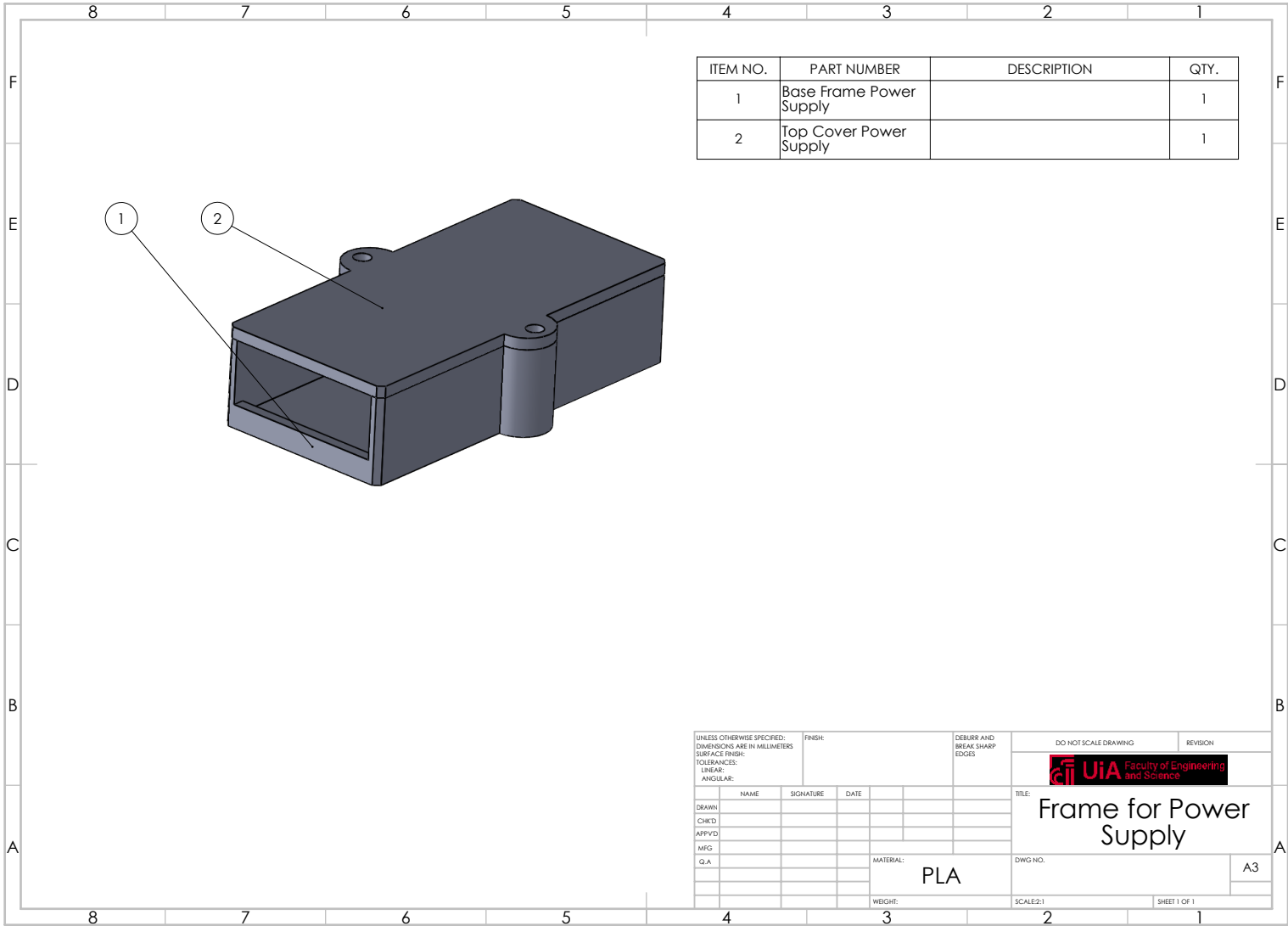


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:									
TOLERANCES:						TITLE:		Bracket for Light Intensity Sensor	
LINEAR:						PART NO.:		3	
ANGULAR:						MATERIAL:		PLA	
DRAWN:		NAME		SIGNATURE		DATE		SCALE: 1:1	
CHKD:								SHEET 1 OF 1	
APPVD:								A3	
MFG:									
Q.A.:									
						WEIGHT:		3	




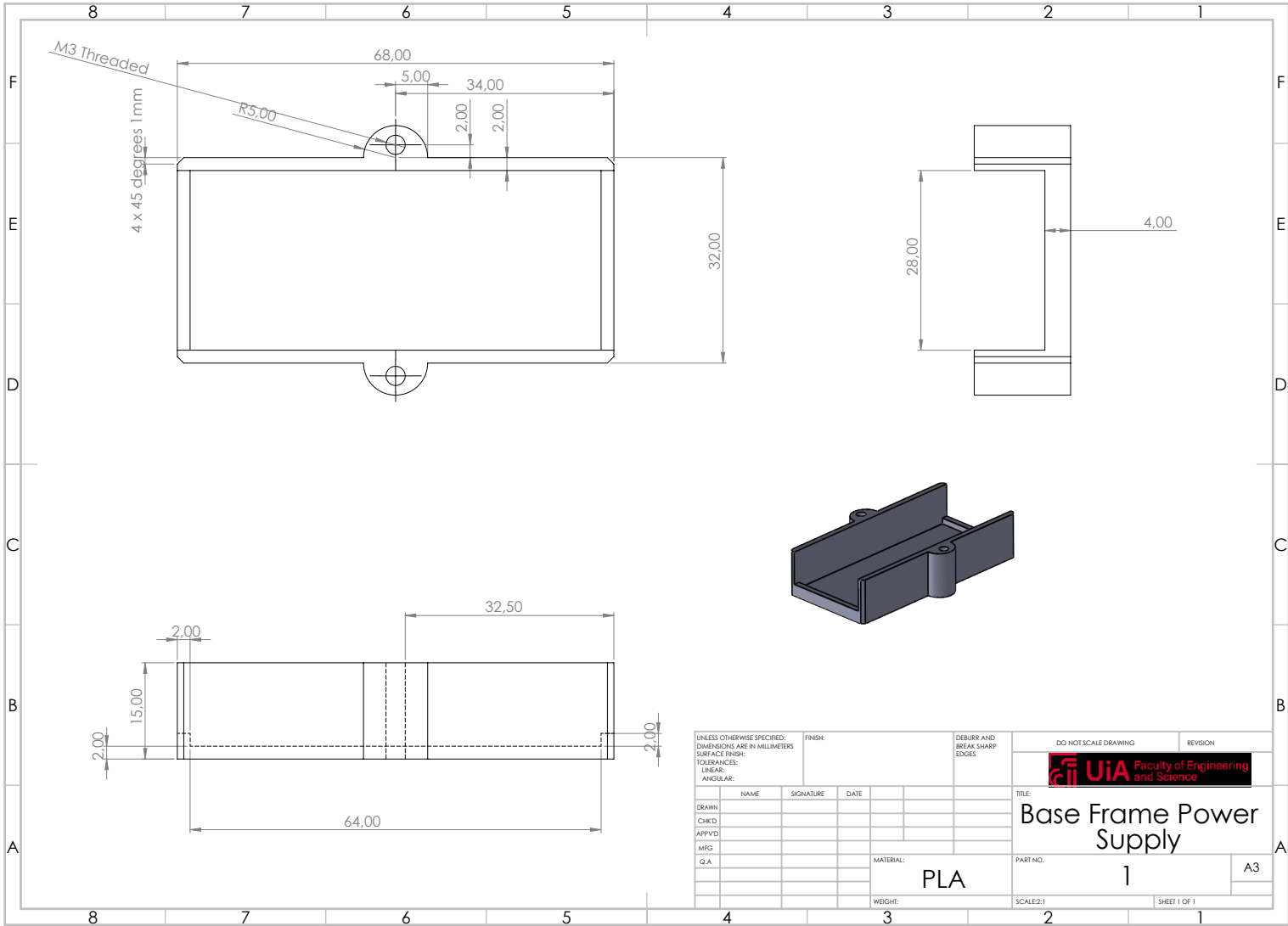
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:						 UiA Faculty of Engineering and Science			
TOLERANCES:								TITLE:	
LINEAR:									
ANGULAR:									
DRAWN	NAME	SIGNATURE	DATE						
CHK'D									
APP'VD									
MEG									
Q.A.									
				MATERIAL:		PART NO.			
				PLA		4		A3	
				WEIGHT:		SCALE:1:1		SHEET 1 OF 1	
				3		2		1	


B.2 Frame for Power Supply

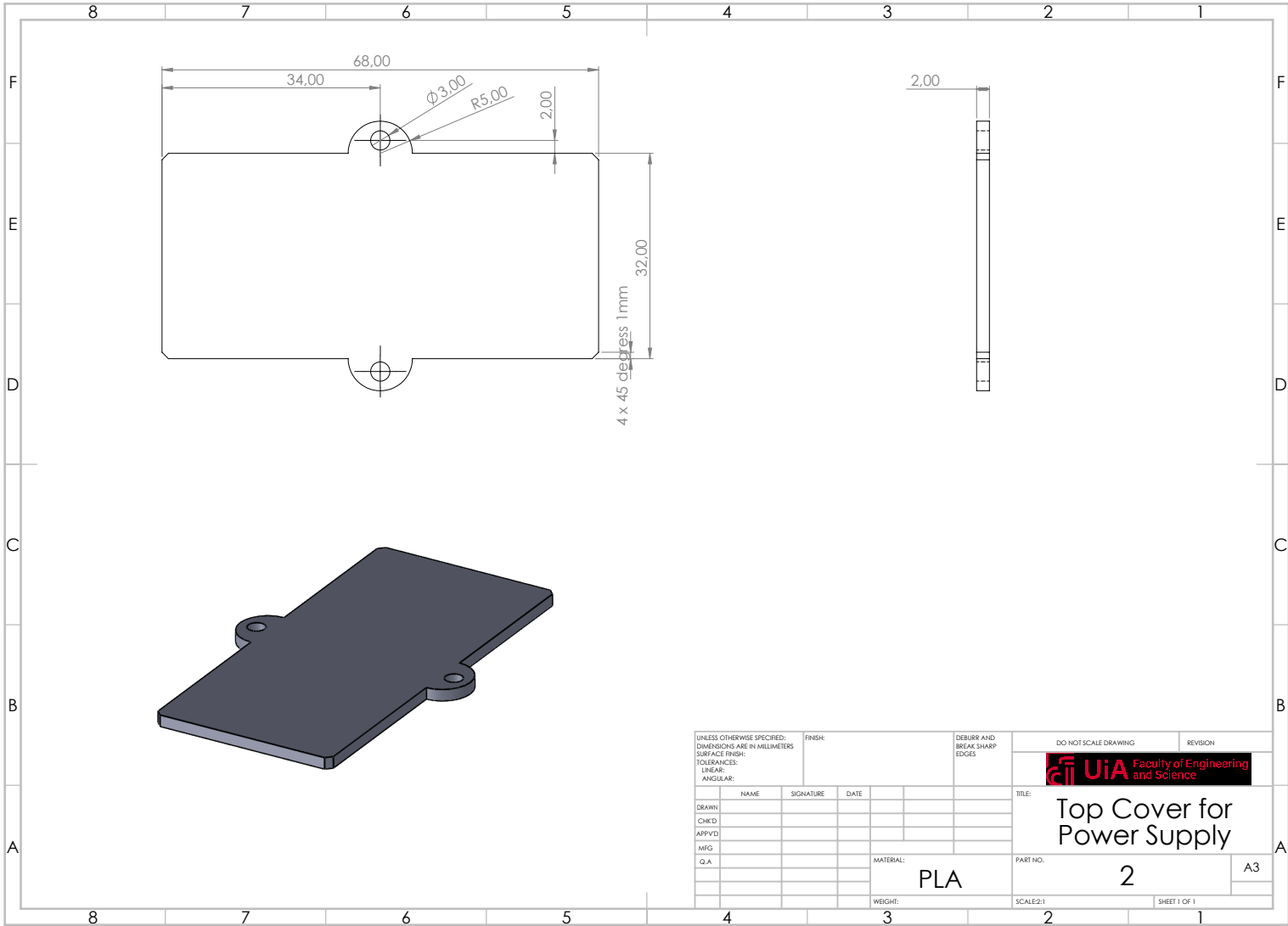


ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Base Frame Power Supply		1
2	Top Cover Power Supply		1

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:									
TOLERANCES:								TITLE: Frame for Power Supply	
LINEAR:						DWG NO.		A3	
ANGULAR:						MATERIAL: PLA			
DRAWN:	NAME	SIGNATURE	DATE			WEIGHT:		SCALE: 1:1	
CHKD:								SHEET 1 OF 1	
APPVD:									
MEG:									
G.A:									



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:						 UiA Faculty of Engineering and Science			
TOLERANCES:								TITLE:	
LINEAR:								Base Frame Power Supply	
ANGULAR:								PART NO.	
								1	
DRAWN:	NAME	SIGNATURE	DATE	MATERIAL:		SCALE: 1:1		SHEET 1 OF 1	
CHKD:				PLA				A3	
APPVD:									
MEG:									
Q.A:									
				WEIGHT:					
				3					



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION	
SURFACE FINISH:									
TOLERANCES:						TITLE:		Top Cover for Power Supply	
LINEAR:						PART NO.		2	
ANGULAR:						MATERIAL:		PLA	
DRAWN:		NAME	SIGNATURE	DATE		WEIGHT:		3	
CHKD:						SCALE: 1:1		SHEET 1 OF 1	
APPVD:						A3			
MEG:									
Q.A:									

C. Gantt Chart

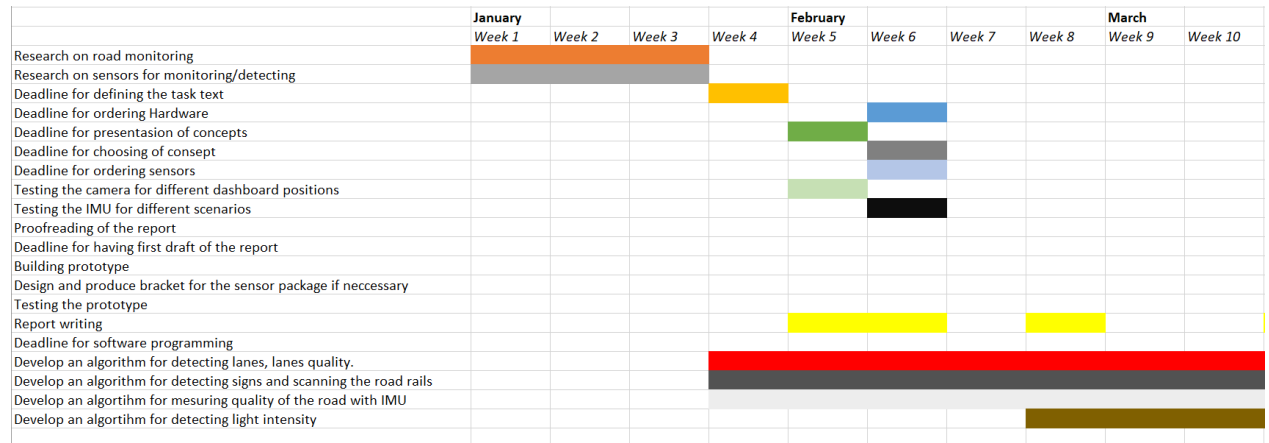


Figure C.1: Gantt Chart - Part 1

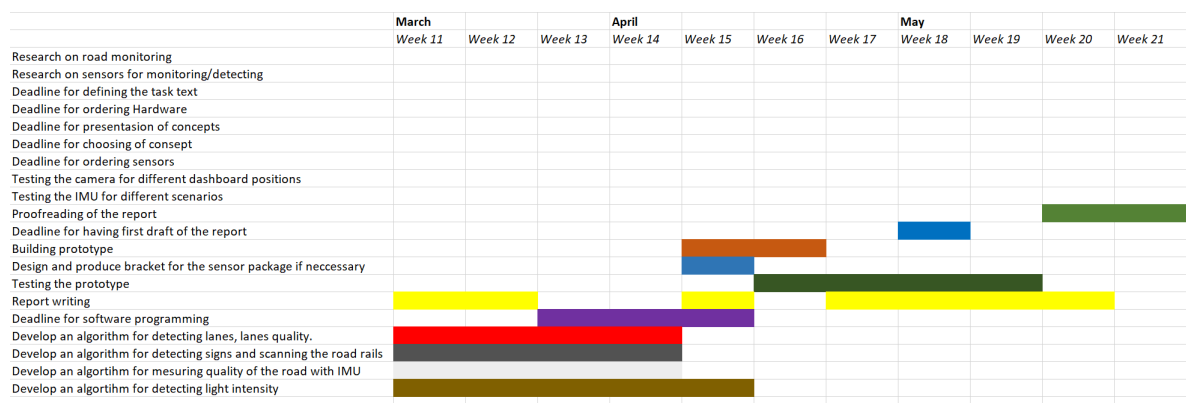


Figure C.2: Gantt Chart - Part 2

D. Verification of Test Setup During Length Measurement



Figure D.1: Height Table



Figure D.2: Height Box

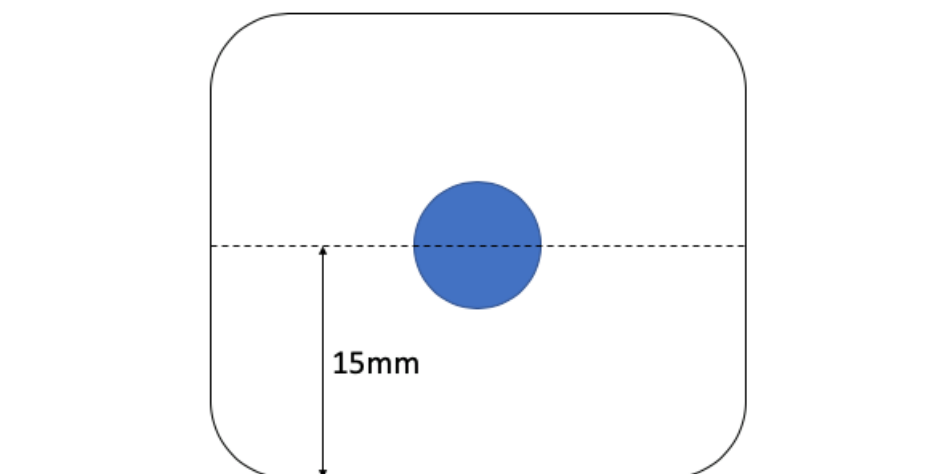


Figure D.3: Height Camera



Figure D.4: Length Long Line



Figure D.5: Length Short Line

E. Data Sheets

E.1 Intel RealSense 435i

Due to the size of the datasheet of the Intel RealSense 435i, the datasheet is listed as a reference in the bibliography. [58]

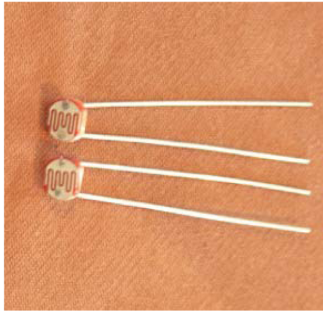
E.2 MLX 90640 - Thermal Camera

Due to the size of the datasheet of the thermal camera MLX 90640, the datasheet is listed as a reference in the bibliography. [59]

E.3 Photocell

CdS PHOTOCONDUCTIVE CELLS

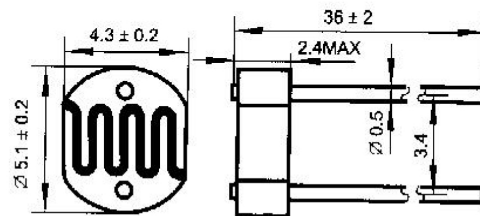
GL5528



- ▲ Epoxy encapsulated
- ▲ Quick response
- ▲ Small size
- ▲ High sensitivity
- ▲ Reliable performance
- ▲ Good characteristic of spectrum

Light Resistance at 10Lux (at 25°C)	8~20KΩ
Dark Resistance at 0 Lux	1.0MΩ(min)
Gamma value at 100-10Lux	0.7
Power Dissipation(at 25°C)	100mW
Max Voltage (at 25°C)	150V
Spectral Response peak (at 25°C)	540nm
Ambient Temperature Range:	- 30~+70°C

Outline

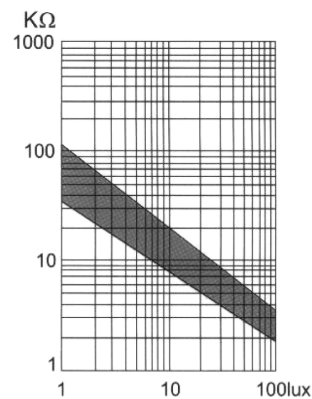


Measuring Conditions

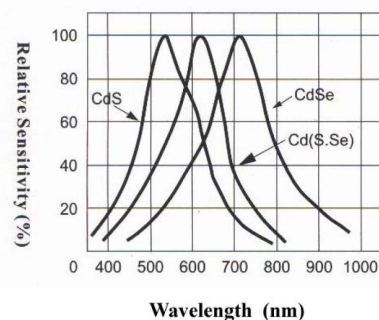
1. Light Resistance: measured at 10 lux with standard light A (2854k color temperature) and 2h pre-illumination at 400-600 lux prior to testing.
2. Dark Resistance: measured 10 seconds after pulsed 10 lux.
3. Gamma Characteristic: between 10 lux and 100 lux and given by

$$T = \frac{\log(R_{10}/R_{100})}{\log(100/10)} = \log(R_{10}/R_{100})$$
 R10, R100 cell resistance at 10 lux and 100 lux.
 The error of T is +0.1.
4. Pmax: Max. power dissipation at ambient temperature of 25°C.
5. Vmax: Max. voltage in darkness that may be applied to the cell continuously.

Illuminance Vs. Photo Resistance



Spectral Response



E.4 GPS Sensor

NEO-6 series

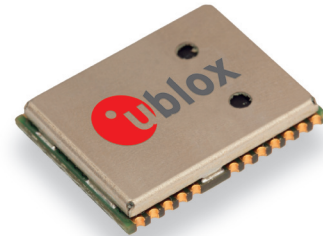
Versatile u-blox 6 GPS modules

Highlights

- UART, USB, DDC (I²C compliant) and SPI interfaces
- Available in Crystal and TCXO versions
- Onboard RTC crystal for faster warm and hot starts
- 1.8 V and 3.0 V variants

Features

- u-blox 6 position engine:
 - Navigate down to -162 dBm and -148 dBm coldstart
 - Faster acquisition with AssistNow Autonomous
 - Configurable power management
 - Hybrid GPS/SBAS engine (WAAS, EGNOS, MSAS)
 - Anti-jamming technology
- Simple integration with u-blox wireless modules
- A-GPS: AssistNow Online and AssistNow Offline services, OMA SUPL compliant
- Backward compatible (hardware and firmware); easy migration from NEO-5 family or NEO-4S
- LCC package for reliable and cost effective manufacturing
- Compatible with u-blox GPS Solution for Android
- Based on GPS chips qualified according to AEC-Q100
- Manufactured in ISO/TS 16949 certified production sites
- Qualified according to ISO 16750



NEO-6:
12.2 x 16.0 x 2.4 mm

Product description

The NEO-6 module series brings the high performance of the u-blox6 position engine to the miniature NEO form factor. u-blox6 has been designed with low power consumption and low costs in mind. Intelligent power management is a breakthrough for low-power applications. These receivers combine a high level of integration capability with flexible connectivity options in a small package. This makes them perfectly suited for mass-market end products with strict size and cost requirements. The DDC interface provides connectivity and enables synergies with u-blox LEON and LISA wireless modules.

All NEO-6 modules are manufactured in ISO/TS 16949 certified sites. Each module is tested and inspected during production. The modules are qualified according to ISO 16750 - Environmental conditions and electrical testing for electrical and electronic equipment for road vehicles.

Product selector

Model	Type	Supply	Interfaces	Features
	Standalone GPS Standalone GLONASS Timing & Raw Data Dead Reckoning	1.75 V – 2.0 V 2.7 V – 3.6 V	UART USB SPI DDC (I ² C compliant)	Programmable (Flash) FW update Oscillator RTC crystal Antenna supply and supervisor Configuration pins Timepulse External interrupt / Wakeup
NEO-6G	•	•	• • • •	T • ○ 3 1 •
NEO-6Q	•	•	• • • •	T • ○ 3 1 •
NEO-6M	•	•	• • • •	C • ○ 3 1 •

○ = requires external components and integration on application processor

C = Crystal / T = TCXO

Receiver performance data

Receiver type	50-channel u-blox 6 engine GPS L1 C/A code SBAS: WAAS, EGNOS, MSAS	
Navigation update rate	up to 5 Hz	
Accuracy ¹	Position	2.5 m CEP
	SBAS	2.0 m CEP
Acquisition ¹	NEO-6G/Q	NEO-6M
	Cold starts:	26 s 27 s
	Aided starts ² :	1 s < 3 s
	Hot starts:	1 s 1 s
Sensitivity ³	NEO-6G/Q	NEO-6M
	Tracking:	-162 dBm -161 dBm
	Cold starts:	-148 dBm -147 dBm
	Hot starts:	-157 dBm -156 dBm

¹ All SV @ -130 dBm

² Dependent on aiding data connection speed and latency

³ Demonstrated with a good active antenna

Electrical data

Power supply	2.7 V – 3.6 V (NEO-6Q/6M) 1.75 V – 2.0 V (NEO-6G)	
Power consumption	111 mW @ 3.0V (continuous) 33 mW @ 3.0V Power Save Mode (1 Hz) 68 mW @ 1.8V (continuous) 22 mW @ 1.8V Power Save Mode (1 Hz)	
Backup power	1.4 V – 3.6V, 22 µA	
Supported antennas	Active and passive	

Interfaces

Serial interfaces	1 UART 1 USB V2.0 full speed 12 Mbit/s 1 DDC (I ² C compliant) 1 SPI	
Digital I/O	Configurable timepulse 1 EXTINT input for Wakeup	
Serial and I/O	Voltages	2.7 – 3.6 V (NEO-6Q/6M) 1.75 – 2.0 V (NEO-6G)
Timepulse	Configurable	0.25 Hz to 1 kHz
Protocols	NMEA, UBX binary, RTCM	

Legal Notice

u-blox reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. Reproduction, use, modification or disclosure to third parties of this document or any part thereof without the express permission of u-blox is strictly prohibited.

The information contained herein is provided "as is". No warranty of any kind, either express or implied, is made in relation to the accuracy, reliability, fitness for a particular purpose or content of this document. This document may be revised by u-blox at any time. For most recent documents, please visit www.u-blox.com.

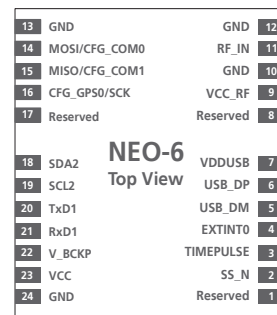
Copyright © 2011, u-blox AG

Specification applies to FW 7

Package

24 pin LCC (Leadless Chip Carrier): 12.2 x 16.0 x 2.4 mm, 1.6 g

Pinout



Environmental data, quality & reliability

Operating temp. -40° C to 85° C

Storage temp. -40° C to 85° C

RoHS compliant (lead-free)

Qualification according to ISO 16750

Manufactured in ISO/TS 16949 certified production sites

Support products

u-blox 6 Evaluation Kits:

Easy-to-use kits to get familiar with u-blox 6 positioning technology, evaluate functionality, and visualize GPS performance.

EVK-6H: u-blox 6 Evaluation Kit with TCXO, suitable for NEO-6G, NEO-6Q

EVK-6P: u-blox 6 Evaluation Kit with crystal, suitable for NEO-6M

Ordering information

NEO-6G-0 u-blox 6 GPS Module, 1.8V, TCXO, 12x16mm, 250 pcs/reel

NEO-6M-0 u-blox 6 GPS Module, 12x16mm, 250 pcs/reel

NEO-6Q-0 u-blox 6 GPS Module, TCXO, 12x16mm, 250 pcs/reel

Available as samples and tape on reel (250 pieces)

Contact us

HQ Switzerland
+41 44 722 7444
info@u-blox.com

China
+86 10 68 133 545
info_cn@u-blox.com

EMEA
+41 44 722 7444
info@u-blox.com

Japan
+81 3 5775 3850
info_jp@u-blox.com

Americas
+1 703 483 3180
info_us@u-blox.com

Korea
+82 2 542 0861
info_kr@u-blox.com

APAC – Singapore
+65 6734 3811
info_ap@u-blox.com

Taiwan
+886 2 2657 1090
info_tw@u-blox.com

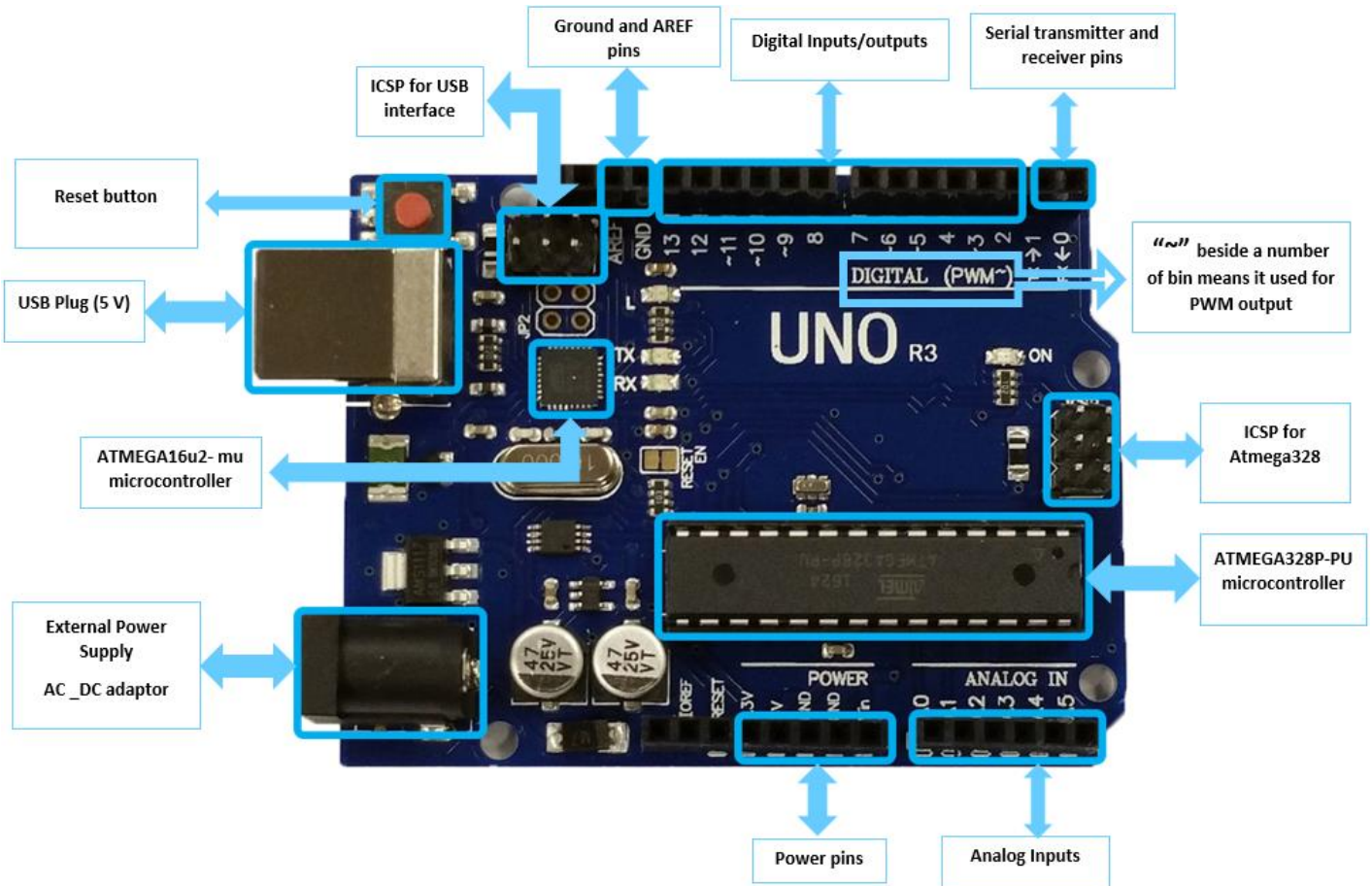
E.5 Intel Up Ai Squared Computer

DATA SHEET INTEL UP SQUARED AI VISION X DEVELOPER KIT

UP board version	UP Squared
Compatible with UP Squared	UP Squared
SoC	Intel ATOM x7-E3950
VPU	Intel Movidius Myriad X (Version B)
Graphics	Intel HD Graphic 505
System memory	8GB LPDDR4
Storage capacity	64GB eMMC
USB2.0 external connector	x4
USB 3.0 port	1x USB 3.0 Host , 1x USB 3.0 OTG
Ethernet	2x Gb Ethernet (full speed) RJ-45
Connectivity	WiFi (option), LTE (optional)
WOL	YES
Video output	HDMI+DP
Power input	5V@6A with DC jack 5.5/2.1mm
Operating tempature	0-40 °C
RTC	YES
PXE	YES
Expansion	40 pin General Purpose bus, supported by Altera Max V. ADC 8-bit@188ksos
M.2 2230 E-key	x1
SATA	x1
Operating humidity	10%~80%RH non-condensing
Certificate	CE/FCC Class A, RoHS complaint Microsoft Azure certified
Country of Origin	Netherlands
Vertical Market	Vision

E.6 Arduino Uno R3

Arduino Uno R3



INTRODUCTION

Arduino is used for building different types of electronic circuits easily using of both a physical programmable circuit board usually microcontroller and piece of code running on computer with USB connection between the computer and Arduino.

Programming language used in Arduino is just a simplified version of C++ that can easily replace thousands of wires with words.

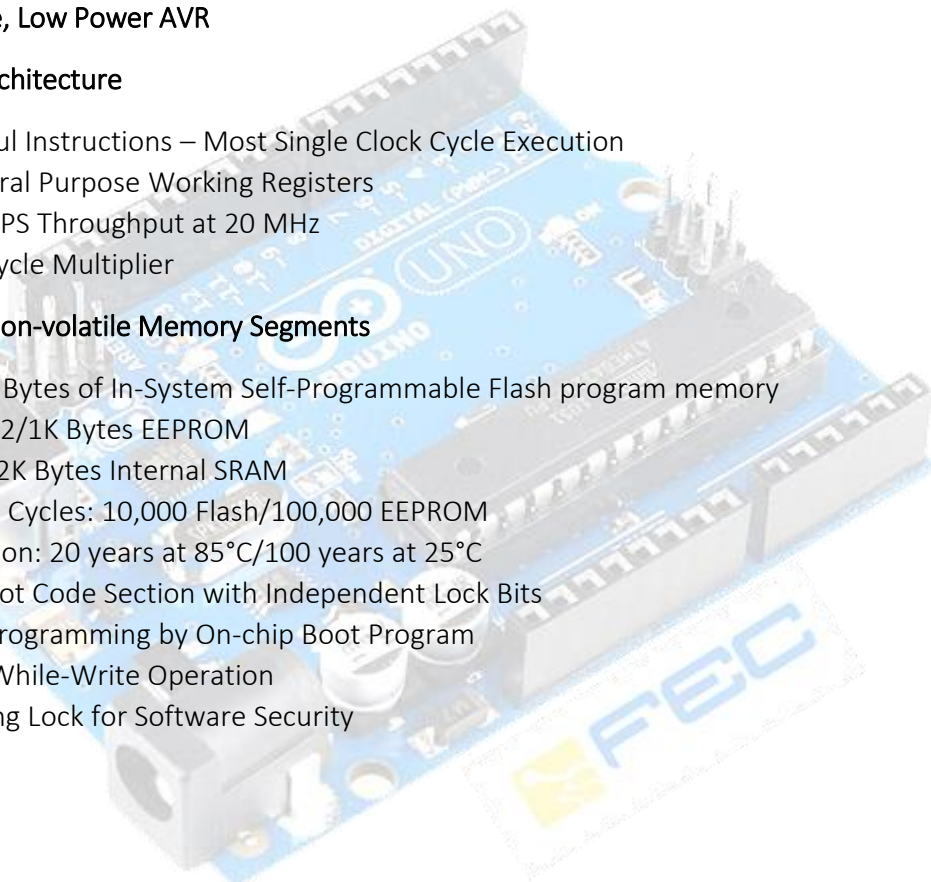


ARDUINO UNO-R3 PHYSICAL COMPONENTS

ATMEGA328P-PU microcontroller

The most important element in Arduino Uno R3 is ATMEGA328P-PU is an 8-bit Microcontroller with flash memory reach to 32k bytes. It's features as follow:

- High Performance, Low Power AVR
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Up to 20 MIPS Throughput at 20 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory
 - 256/512/512/1K Bytes EEPROM
 - 512/1K/1K/2K Bytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Six PWM Channels
 - 8-channel 10-bit ADC in TQFP and QFN/MLF package
 - Temperature Measurement
 - 6-channel 10-bit ADC in PDIP Package
 - Temperature Measurement
 - Programmable Serial USART





- Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

• **Special Microcontroller Features**

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

• **I/O and Packages**

- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

• **Operating Voltage:**

- 1.8 - 5.5V

• **Temperature Range:**

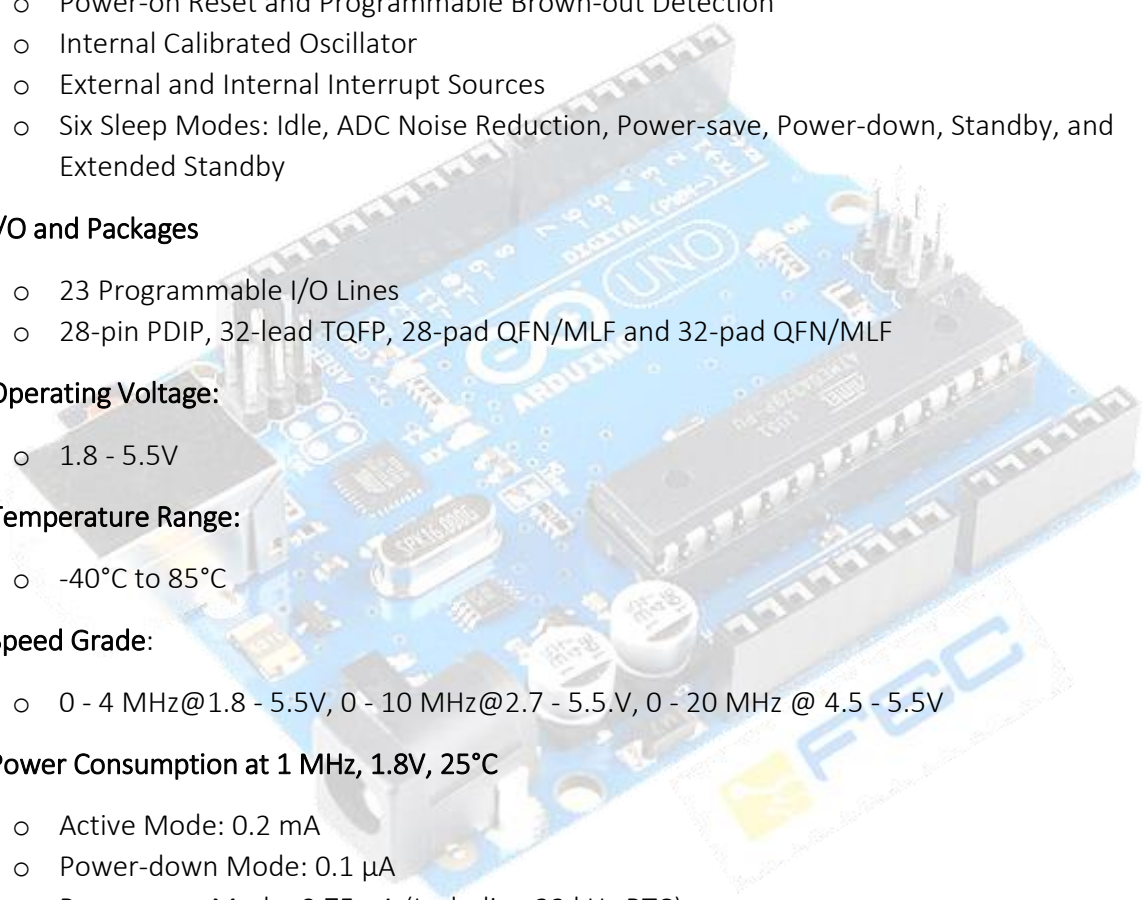
- -40°C to 85°C

• **Speed Grade:**

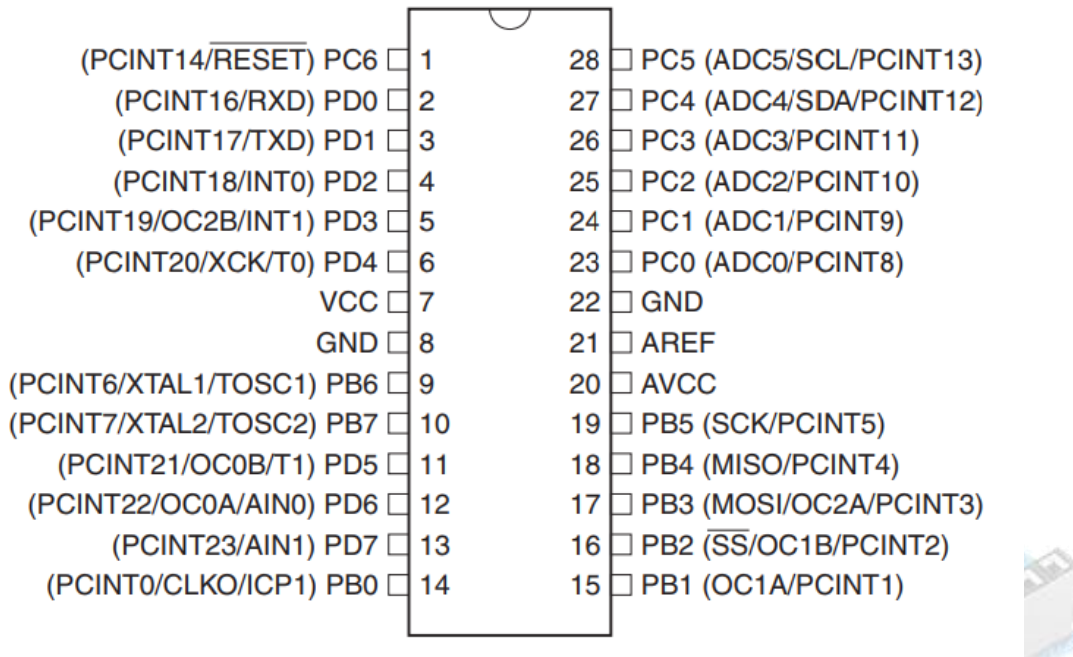
- 0 - 4 MHz@1.8 - 5.5V, 0 - 10 MHz@2.7 - 5.5.V, 0 - 20 MHz @ 4.5 - 5.5V

• **Power Consumption at 1 MHz, 1.8V, 25°C**

- Active Mode: 0.2 mA
- Power-down Mode: 0.1 μ A
- Power-save Mode: 0.75 μ A (Including 32 kHz RTC)



- Pin configuration



ATMEGA16u2- mu microcontroller

Is a 8-bit microcontroller used as USB driver in Arduino uno R3 it's features as follow:

- High Performance, Low Power AVR
- Advanced RISC Architecture
 - 125 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
- Non-volatile Program and Data Memories
 - 8K/16K/32K Bytes of In-System Self-Programmable Flash
 - 512/512/1024 EEPROM
 - 512/512/1024 Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
 - Data retention: 20 years at 85°C/ 100 years at 25°C



- Optional Boot Code Section with Independent Lock Bits
- In-System Programming by on-chip Boot Program hardware-activated after reset
- Programming Lock for Software Security

• USB 2.0 Full-speed Device Module with Interrupt on Transfer Completion

- Complies fully with Universal Serial Bus Specification REV 2.0
- 48 MHz PLL for Full-speed Bus Operation: data transfer rates at 12 Mbit/s
- Fully independent 176 bytes USB DPRAM for endpoint memory allocation
- Endpoint 0 for Control Transfers: from 8 up to 64-bytes
- 4 Programmable Endpoints:
 - IN or Out Directions
 - Bulk, Interrupt and Isochronous Transfers
 - Programmable maximum packet size from 8 to 64 bytes
 - Programmable single or double buffer
- Suspend/Resume Interrupts
- Microcontroller reset on USB Bus Reset without detach
- USB Bus Disconnection on Microcontroller Request

• Peripheral Features

- One 8-bit Timer/Counters with Separate Prescaler and Compare Mode (two 8-bit PWM channels)
- One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Mode (three 8-bit PWM channels)
- USART with SPI master only mode and hardware flow control (RTS/CTS)
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

• On Chip Debug Interface (debug WIRE)

• Special Microcontroller Features

- Power-On Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Five Sleep Modes: Idle, Power-save, Power-down, Standby, and Extended Standby

• I/O and Packages

- 22 Programmable I/O Lines
- QFN32 (5x5mm) / TQFP32 packages

- Operating Voltages

- 2.7 - 5.5V

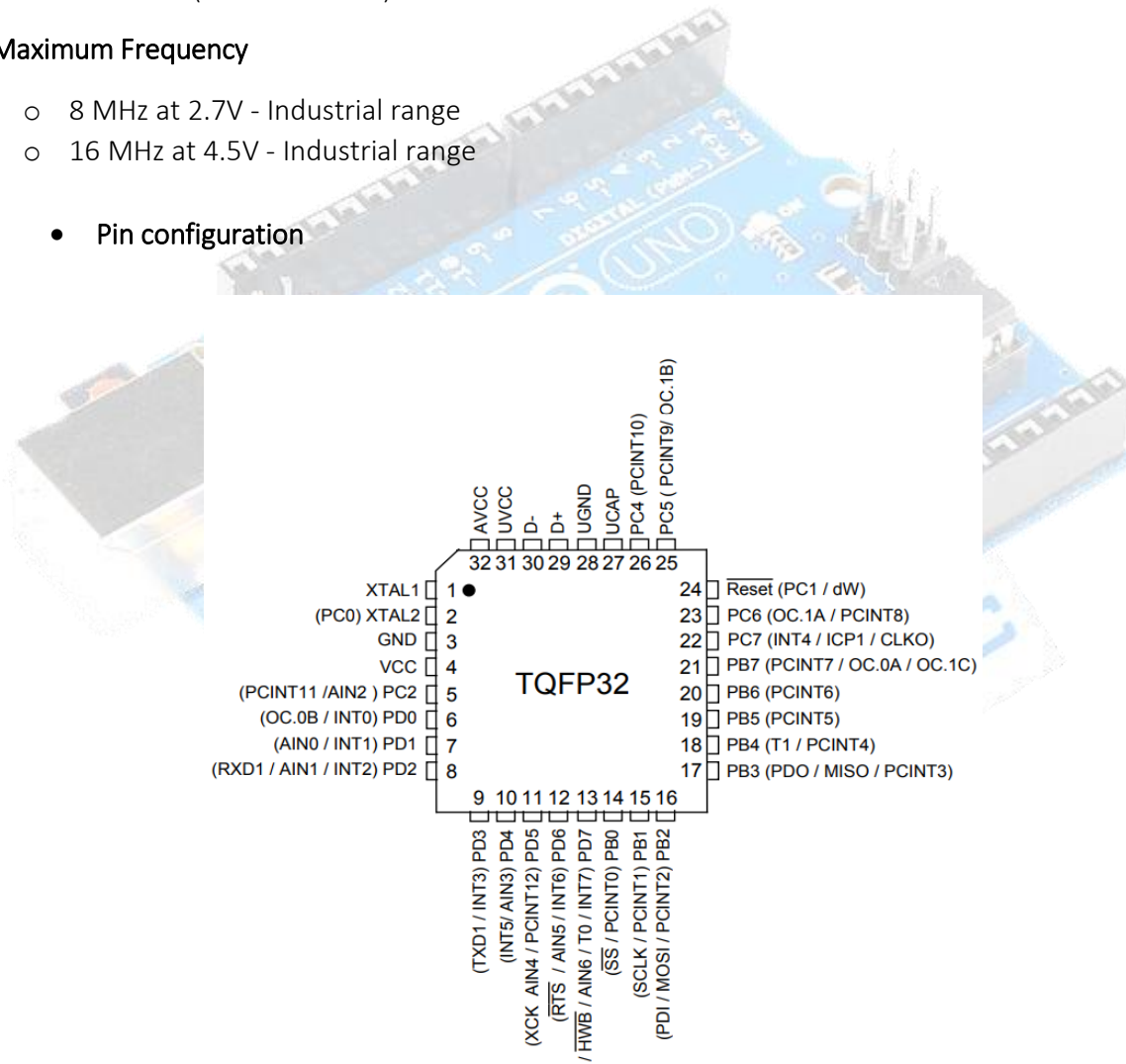
- Operating temperature

- Industrial (-40°C to +85°C)

- Maximum Frequency

- 8 MHz at 2.7V - Industrial range
- 16 MHz at 4.5V - Industrial range

- Pin configuration





OTHER ARDUINO UNO R3 PARTS

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k Ohms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the `analogWrite()` function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the `analogReference()` function. Additionally, some pins have specialized functionality:

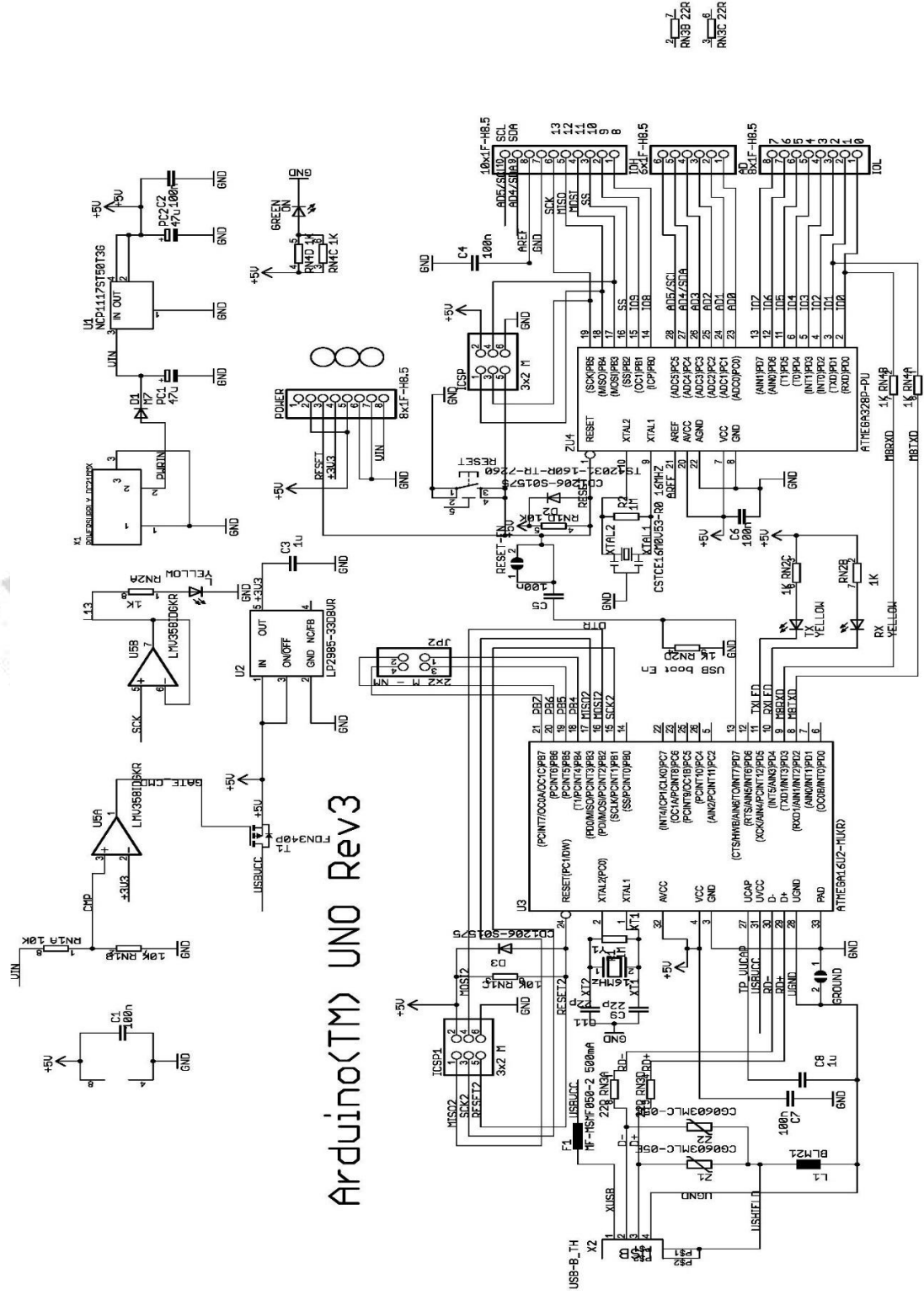
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- AREF: Reference voltage for the analog inputs. Used with `analogReference()`.
- Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.



ARDUINO UNO R3 SCHEMATIC DIAGRAM



Arduino™ UNO Rev3

2 7
RN33 22R
3 8
RN3C 22R

E.7 Biltema Power Supply

SPÄNNINGSMVANDLARE SPENNINGSMFORMER JÄNNITTEENMUUNNIN SPÆENDINGSMFORMER



SPÄNNINGSOMVANDLARE

SÄKERHETSINSTRUKTIONER

- Läs bruksanvisningen noga innan produkten används.
- Använd alltid spänningsomvandlaren i ett välventilerat utrymme.
- Spänningsomvandlaren ska endast användas i dammfria och torra utrymmen. Får inte användas i våta eller fuktiga utrymmen.
- Placera spänningsomvandlaren oåtkomligt för barn.
- Undvik att placera spänningsomvandlaren i direkt solljus eller nära annan värmekälla.
- Spänningsomvandlaren kan bli mycket varm. Se till att det är 5 cm fritt utrymme runt enheten.
- Använd inte spänningsomvandlaren i närheten av brandfarliga vätskor eller gaser.
- Använd inte spänningsomvandlaren i omgivningstemperatur över 40°C.
- Spänningsomvandlaren får inte demonteras eller modifieras på något sätt.
- Överbelasta inte spänningsomvandlaren.
- Om spänningsomvandlaren blir för varm; stäng av den inkopplade apparaten för att minska effektuttaget. Skulle inte det räcka; stäng även av spänningsomvandlaren. Starta sedan spänningsomvandlaren utan någon apparat inkopplad.
- Om batteriet verkar dåligt laddat, ladda det innan spänningsomvandlaren används igen.
- Ta bort spänningsomvandlaren från spänningsmatning när den inte används.

SPÄNNINGSMATNING

Enheten måste ha en kontinuerlig spänningsförsörjning på min. 10.5 V och max. 15 V DC från batteri eller motsvarande DC-spänningskälla.

ANSLUTNING TILL SPÄNNINGSMATNING

- Sätt ON/OFF-knappen i läge OFF.
- Anslut cigarettändaruttaget pluggen i uttaget.

VARNING: Omvandlaren kan endast anslutas till 12 V DC spänningskälla.

Anslut apparat till spänningsomvandlaren

- Se till att ansluten apparat är inom angivna specifikationer.
- Anslut stickkontakten till uttaget på spänningsomvandlaren.
- Sätt ON/OFF-knappen i läge ON. Grön lysdiod tänds.

Lysdioden slocknar om spänningen sjunker under 10 V och omvandlaren stängs av. Om detta sker – stäng av ansluten apparat och dra ur nätkabeln.

UPPLADDNINGSBARA ENHETER

Uppladdningsbara enheter som kan anslutas direkt till standard sockel kan skada spänningsomvandlaren.

VID ANSLUTNING TILL BILBATTERI TÄNK PÅ ATT:

- Kör bilmotorn ungefär 15 minuter varje timme för att undvika att batteriet laddas ur.
- Spänningsomvandlaren kan användas både då motorn är igång eller är stoppad.
- Spänningsomvandlaren kan sluta fungera för ett kort ögonblick då bilen startas.

SÄKERHETSFUNCTIONER I SPÄNNINGSOMVANDLAREN

- Om spänningsmatningen sjunker under 10 V stängs omvandlaren av.
- Överstiger spänningsmatningen 15 V stängs omvandlaren av.
- Kortslutningsskydd – 15 A säkring
- Överhettningsskydd stänger av omvandlaren om den interna temperaturen blir 65°C. Låt svalna i ca. 15 minuter.



DETALJBESKRIVNING

1. ON/OFF-knapp
2. USB-anslutning, 5DC V, 500 mA
3. Anslutningskabel till cigarettuttag
4. AC uttag
5. Överlastindikering

GENERELLA PROBLEM

- Apparaten startar men stannar igen. Slå på och av omvandlaren snabbt några gånger.
- Brus i musiksystemet. Vissa högtalare kan inte filtrera de modifierade sinusvågorna som omvandlaren genererar.
- TV fungerar dåligt. Omvandlaren är skärmad men kan ändå ge störningar, särskilt om TV-signalen är svag. Placera TV:n så långt från spänningsomvandlaren som möjligt och använd antennkabel av god kvalitet.

TEKNISKA DATA

Märkeffekt	.150 W
Max effekt	.300 W
Frekvens	.50~60 Hz
Utgångsspänning	.100~240 V AC
USB-anslutning	.5 V DC, 500 mA
Strömförbrukning	.>0,35 A
Ingångsspänning	.12 V
Spänningsområde	.10~15 V
Säkring	.20 A
Omgivningstemperatur	.10°C–27°C

Felsökning

ORSAK	REKOMMENDERAD ÅTGÄRD
Omvandlaren är överbelastad	Minska belastningen
Spänningsmatningen är under 10,6 V	Se till att spänningsmatningen är över 10,6 V
Dåligt batteri	Byt batteri
Glappkontakt	Kontrollera cigarettuttaget, rengör eller byt.
Omvandlaren uppnår inte arbetstemperatur	Slå på och av omvandlaren några gånger.
Cigarettändaren behöver spänning	Starta motorn
Batterispänningen är under 10 V	Ladda eller byt batteri
Omvandlaren har stängts av p.g.a överhettning	Låt omvandlaren svalna
Säkringen har löst ut	Byt säkring. Kontrollera anslutningar till batteri. Rätt polaritet.

SPENNINGSSOMFORMER

SIKKERHETSINSTRUKSJONER

- Les bruksanvisningen nøye før du tar produktet i bruk.
- Bruk alltid spenningsomformeren i et godt ventilert rom.
- Spenningsomformeren skal brukes kun i støvfrie og tørre rom, og må ikke brukes i våte eller fuktige rom.
- Plasser spenningsomformeren utilgjengelig for barn.
- Unngå å plassere spenningsomformeren i direkte sollys eller nær en annen varmekilde.
- Spenningsomformeren kan bli svært varm. Sørg for at det er 5 cm fritt rom rundt enheten.
- Bruk ikke spenningsomformeren i nærheten av brannfarlige væsker eller gasser.
- Bruk ikke spenningsomformeren i omgivelsestemperatur på over 40 °C.
- Spenningsomformeren må ikke tas fra hverandre eller modifiseres på noen som helst måte.
- Overbelast ikke spenningsomformeren.
- Om spenningsomformeren blir for varm, slå av det tilkoblede apparatet for å minske effektuttaket. Skulle dette ikke være tilstrekkelig, slå også av spenningsomformeren. Start så spenningsomformeren igjen uten noe apparatet tilkoblet.
- Om batteriet virker dårlig ladet, lad det før spenningsomformeren brukes igjen.
- Koble spenningsomformeren fra strømtilførsel når den ikke er i bruk.

SPENNINGSTILFØRSEL

Enheden må ha en kontinuerlig spenningstilførsel på min. 10,5 V og maks. 15 V DC fra batteri eller tilsvarende DC spenningskilde.

TILKOBLING TIL SPENNINGSTILFØRSEL

- Sett ON/OFF-knappen i posisjon OFF.
- Koble sigarettenerpluggen i uttaket.

ADVARSEL: Omformeren kan kun kobles til 12 V DC spenningskilde.

KOBLE APPARAT TIL SPENNINGSSOMFORMEREN

- Sjekk at tilkoblet apparat ligger innenfor angitte spesifikasjoner.
- Koble støpselet til uttaket på spenningsomformeren.
- Sett ON/OFF-knappen i posisjon ON. Grønn lysdiode tennes.

Lysdioden slukkes dersom spenningen synker under 10 V, og dersom omformeren slås av. Om dette skjer – slå av tilkoblet apparat og trekk ut nettkabelen.

OPPLADBARE ENHETER

Oppladbare enheter som kan kobles direkte til standard sokkel, kan skade spenningsomformeren.

VED TILKOBLING TIL BILBATTERI HUSK PÅ FØLGENDE:

- Kjør bilmotoren ca. 15 minutter hver time slik at batteriet ikke lades ut.
- Spenningsomformeren kan brukes både når motoren er i gang og når den er stanset.
- Spenningsomformeren kan slutte å fungere et kort øyeblikk når bilen startes.

SIKKERHETSFUNKSJONER I SPENNINGSSOMFORMEREN

- Om spenningstilførselen synker under 10 V, slås omformeren av.
- Dersom spenningstilførselen overstiger 15 V, slås omformeren av.
- Kortslutningsvern – 15 A sikring.
- Overopphetingsvern slår av omformeren dersom den innvendige temperaturen blir 65 °C. La den avkjøles i ca. 15 minutter.



DELEBESKRIVELSE

1. ON/OFF-knapp (på/av)
2. USB-tilkobling, 5 DC V, 500 mA
3. Tilkoblingskabel til sigarettuttak
4. AC-uttak
5. Overbelastningsindikering

GENERELLE PROBLEMER

- Apparatet starter, men stanser igjen. Slå omformeren raskt på og av noen ganger.
- Støy i musikkanlegget. Noen høyttalere kan ikke filtrere de modifiserte sinusbølgene som omformeren genererer.
- TV fungerer dårlig. Omformeren er skjermet men kan likevel gi forstyrrelser, spesielt dersom TV-signalet er svakt. Plasser TV-en så langt fra spenningsomformeren som mulig, og bruk antennekabel av god kvalitet.

TEKNISKE DATA

Merkeeffekt	150 W
Maks. effekt	300 W
Frekvens	50~60 Hz
Utgangsspenning	100~240 V AC
USB-tilkobling	5 V DC, 500 mA
Strømforbruk	>0,35 A
Inngangsspenning	12 V
Spenningsområde	10~15 V
Sikring	20 A
Omgivelsestemperatur	10 °C–27 °C

FEILSØKING

ÅRSAK	ANBEFALT TILTAK
Omformeren er overbelastet	Reduser belastningen
Spenningsstilførsel er under 10,6 V	Se til at spenningstilførsel er over 10,6 V
Dårlig batteri	Skift batteri
Løs kontakt	Kontroller sigarettuttaket, rengjør eller skift
Omformeren kommer ikke opp i arbeidstemperatur	Slå omformeren på og av noen ganger
Sigarettenneren trenger spenning	Start motoren
Batterispenningen er under 10 V	Lad eller skift batteri
Omformeren er slått av pga. overoppheting	La omformeren avkjøles
Sikringen har gått	Skift sikring. Kontroller tilkoblinger til batteri Riktig polaritet.

JÄNNITTEENMUUNNIN

TURVAOHJEET

- Lue käyttöohje huolellisesti ennen tuotteen käyttämistä.
- Käytä invertteriä vain hyvin tuuletetussa tilassa.
- Invertteriä saa käyttää vain pölyttömissä ja kuivissa tiloissa. Sitä ei saa käyttää märissä tai kosteissa tiloissa.
- Säilytä invertteriä poissa lasten ulottuvilta.
- Vältä altistamatta invertteriä suoralle auringonpaisteelle tai kuumuudelle.
- Invertteri voi kuumentua voimakkaasti. Varmista, että sen ympärillä on 5 cm vapaata tilaa.
- Älä käytä invertteriä helposti syttyvien nesteiden tai kaasujen lähellä.
- Älä käytä invertteriä, jos ympäristön lämpötila ylittää 40 °C.
- Invertteriä ei saa purkaa, eikä siihen saa tehdä mitään muutoksia.
- Älä ylikuormita invertteriä.
- Jos invertteri lämpenee liikaa, katkaise siihen yhdistetty laite kuormituksen vähentämiseksi. Jos tämä ei riitä, katkaise virta myös invertteristä. Käynnistä invertteri tämän jälkeen uudelleen ilman että siihen on yhdistetty mitään laitetta.
- Jos akku on ladattu huonosti, lataa se täyteen ennen invertterin käyttämistä.
- Irrota invertteri virransyötöstä, kun sitä ei käytetä.

VIRRANSYÖTTÖ

Laitteeseen on syötettävä jatkuvasti 10,5–15 voltin tasavirtaa esimerkiksi akusta.

YHDISTÄMINEN VIRRANSYÖTTÖÖN

- Varmista, että ON/OFF-painike on OFF-asennossa.
- Yhdistä laite savukkeensytytinliitäntään.

VAROITUS: Invertterin saa yhdistää vain 12 voltin tasavirtaan.

SÄHKÖLAITTEEN YHDISTÄMINEN INVERTTERIIN

- Varmista, että yhdistettävän laitteen tekniset tiedot vastaavat invertterin asettamia vaatimuksia.
- Yhdistä laitteen pistoke invertterin sähköpistorasiaan.
- Aseta ON/OFF-painike ON-asentoon. Vihreä merkkivalo syttyy.

Merkkivalo sammuu, jos jännite alittaa 10 voltia. Tällöin invertteristä katkaistaan virta. Jos näin käy, katkaise yhdistetystä laitteesta virta ja irrota pistoke.

LADATTAVAT LAITTEET

Ladattavat suoraan tavalliseen pistorasiaan yhdistettävät laitteet voivat vaurioittaa invertteriä.

OTA HUOMIOON YHDISTETTÄESSÄ AUTON AKKUUN

- Käytä auton moottoria noin 15 minuuttia kerran tunnissa, jotta akku ei tyhjene.
- Invertteriä voi käyttää moottorin ollessa käynnissä tai pysähdyksissä.
- Kun auto käynnistetään, invertteri voi lakata toimimasta hetkeksi.

INVERTTERIN TURVALLISUUSTOIMINNOT

- Jos jännite alittaa 10 voltia, invertteristä katkaistaan virta.
- Jos jännite ylittää 15 voltia, invertteristä katkaistaan virta.
- Invertterissä on 15 A:n oikosulkusulake.
- Invertteristä katkaistaan virta ylikuumenemissuojauksen avulla, jos sen lämpötila ylittää 65 °C. Anna jäähtyä noin 15 minuuttia.



OSIEN KUVAUS

1. ON/OFF-painike
2. USB-liitäntä: 5 V DC, 500 mA
3. Savukkeensytytinliitäntäjohto
4. Virtapistoke
5. Ylikuormituksen ilmaisin

YLEISET ONGELMAT

- Laite käynnistyy ja sammuu. Käynnistä ja sammuta invertteri nopeasti muutaman kerran.
- Musiikkijärjestelmästä kuuluu kohinaa. Tietyt kaiuttimet eivät pysty suodattamaan pois invertterin tuottamia sinusaaltoja.
- Televisio toimii huonosti. Invertterissä on suojaus, mutta se voi silti aiheuttaa häiriötä varsinkin jos tv-signaali on heikko. Aseta televisio mahdollisimman kauas invertteristä. Jos mahdollista, käytä hyvälaatuista antennijohtoa.

TEKNISET TIEDOT

Nimellisteho	150 W
Enimmäisteho	300 W
Taajuus	50–60 Hz
Lähtöjännite	100–240 V AC
USB-liitäntä	5 V DC, 500 mA
Virrankulutus	> 0,35 A
Tulojännite	12 V
Jännitealue	10–15 V
Sulake	20 A
Ympäristön lämpötila	10–27 °C

VIANETSINTÄ

SYY	SUOSITELTU TOIMENPIDE
Invertteri ylikuormittuu.	Vähennä kuormitusta.
Jännitteensyöttö alittaa 10,6 voltia.	Varmista, että jännitteensyöttö ylittää 10,6 voltia.
Huono akku.	Vaihda akku.
Huono liitos.	Tarkista savukkeensytytinliitäntä. Puhdista tai vaihda se.
Invertteri ei saavuta työskentelylämpötilaa.	Käynnistä ja sammuta invertteri nopeasti muutaman kerran.
Savukkeensytytinliitäntään ei syötetä virtaa.	Käynnistä moottori.
Akun jännite alittaa 10 voltia.	Lataa tai vaihda akku.
Invertteristä on katkaistu virta esimerkiksi ylikuormituksen vuoksi.	Anna invertterin jäähtyä.
Sulake on lauennut.	Vaihda sulake. Tarkista akkuliitännät. Tarkista napaisuus.

SPÆNDINGSOMFORMER

SIKKERHEDSINSTRUKTIONER

- Læs omhyggeligt brugsanvisningen, inden produktet bruges.
- Anvend altid spændingsomformeren i et godt ventileret rum.
- Spændingsomformeren må kun anvendes i støvfrie og tørre omgivelser. Må ikke anvendes i fugtige eller våde omgivelser.
- Placer spændingsomformeren utilgængeligt for børn.
- Placer ikke spændingsomformeren i nærheden af en varmekilde eller i direkte sollys.
- Spændingsomformeren kan blive meget varm. Sørg for, at der er 5 cm frit rum omkring enheden.
- Brug ikke spændingsomformeren i nærheden af brandfarlige luftarter eller væsker.
- Anvend aldrig spændingsomformeren i temperaturer over 40° C.
- Spændingsomformeren må ikke adskilles eller ændres på nogen måde.
- Overbelast ikke spændingsomformeren.
- Hvis spændingsomformeren bliver for varm skal du slukke det tilkoblede apparat for at mindske effektudgangen. Hvis det ikke er nok, skal du slukke for spændingsomformeren. Start derefter spændingsomformeren uden apparater tilkoblet.
- Hvis batteriet forekommer at være dårligt opladet, skal det oplades, inden spændingsomformeren anvendes igen.
- Fjern spændingsomformeren fra cigartænderstikket, når den ikke er i brug.

SPÆNDINGSFORSYNING

Enheden skal tilføres en kontinuerlig spænding på min. 10,5 V og maks. 15 V DC fra batteri eller tilsvarende DC-spændingskilde.

TILSLUTNING TIL SPÆNDINGSFORSYNINGEN

- Sæt ON/OFF-knappen i OFF-position.
- Slut stikket til cigartænderstikket.

ADVARSEL: Omformeren må kun tilsluttes en 12 V DC spændingskilde.

Sæt et apparat til spændingsomformeren

- Sørg for, at det tilsluttede apparat holder sig inden for de angivne specifikationer.
- Slut stikket til udtaget på spændingsomformeren.
- Sæt ON/OFF-knappen i ON-position. Grøn lysdiode tændes.

Hvis spændingen synker til under 10 V, slukker lysdioden, og omformeren slukkes. Hvis dette sker – sluk det tilsluttede apparat og tag netstikket ud.

OPLADELIGE ENHEDER

Opladelige enheder, som kan sluttes direkte til en standardsokkel, kan beskadige spændingsomformeren.

VED TILSLUTNING TIL BILBATTERIET, HUSK:

- Kør med bilmotoren cirka 15 minutter hver time, så batteriet ikke aflades.
- Spændingsomformeren kan anvendes, både når motoren kører, og når den er standset.
- Spændingsomformeren kan holde op med at fungere et kort øjeblik, når motoren startes.

SIKKERHEDSFUNKTIONER I SPÆNDINGSOMFORMEREN

- Hvis spændingen falder til under 10 V, slukkes omformeren.
- Hvis spændingen overstiger 15 V, slukkes omformeren.
- Kortslutningssikring – 15 A sikring
- Overophedningssikringen slukker automatisk spændingsomformeren, når den indvendige temperatur er 65° C. Lad den køle af i ca. 15 minutter.



OVERSIGT

1. ON/OFF-knap
2. USB-tilslutning, 5DC V, 500 mA
3. Tilslutningskabel til cigartænderudtag
4. AC-udtag
5. Overbelastningsindikering

Generelle problemer

- Apparatet starter, men stopper igen. Tænd og sluk for omformeren nogle gange.
- Sus i musiksystemet. Visse højtalere kan ikke filtrere de modificerede sinusbølger, som omformeren genererer.
- TV fungerer dårligt. Omformeren er skærmet, men kan alligevel give forstyrrelser, især hvis TV-signalet er svagt. Placer fjernsynet så langt fra spændingsomformeren som muligt og et antennekabel af god kvalitet.

TEKNISKE DATA

Mærkeeffekt:	150 W
Maks. effekt	300 W
Frekvens	50~60 Hz
Udgangsspænding	100~240 V AC
USB-tilslutning	5 V DC, 500 mA
Strømforbrug	>0,35 A
Indgangsspænding	12 V
Spændingsområde	10~15 V
Sikring	20 A
Temperatur	10° C–27° C

FEJLSØGNING

ÅRSAG	ANBEFALET FORANSTALTNING
Omformeren er overbelastet.	Formindsk belastningen
Spændingen er under 10,6 V	Sørg for, at spændingstilførslen er over 10,6 V
Dårligt batteri	Udskift batteri
Dårlig kontakt	Kontroller cigartænderstikket, rengør eller udskift
Omformeren opnår ikke arbejdstemperatur	Tænd og sluk for omformeren nogle gange
Cigartænderen skal have spænding på	Start motoren
Batterispændingen er under 10 V	Oplad eller skift batteri
Omformeren er slukket pga. overophedning	Lad omformeren køle af
Sikringen er udløst	Skift sikring. Kontroller batteriets tilslutninger. Rigtig polaritet.

F. Python Scripts

F.1 Finding Lane Curvature, Curve Radius and Vehicle Position

The source code is fetched from GitHub, and the credit goes to the owner of the script. [57]

```
1 import numpy as np
2 import cv2
3 import pickle
4 import glob
5 import matplotlib.pyplot as plt
6 import matplotlib.image as mpimg
7 import os
8
9
10
11 def display_Images(img1, img2, lbl1, lbl2, x, y, img3=[], lbl3=[],
12                  cmap=None, n=2):
13     plt.figure(figsize=(x, y))
14     plt.subplot(1, n, 1)
15     plt.imshow(img1, cmap=cmap)
16     plt.xlabel(lbl1, fontsize=15)
17     plt.xticks([])
18     plt.yticks([])
19     plt.subplot(1, n, 2)
20     plt.imshow(img2, cmap=cmap)
21     plt.xlabel(lbl2, fontsize=15)
22     plt.xticks([])
23     plt.yticks([])
24     if n == 3:
25         plt.subplot(1, n, 3)
26         plt.imshow(img3, cmap=cmap)
27         plt.xlabel(lbl3, fontsize=15)
28         plt.xticks([])
29         plt.yticks([])
30     plt.show()
31
```

```
32 def Transform_Camera_View(img, src, dst):
33     """
34     Convert the vehicles camera into a Birds eye View
35     """
36     image_shape = img.shape
37     img_size = (image_shape[1], image_shape[0])
38     # Given src and dst points, calculate the perspective transform
        matrix
39     M = cv2.getPerspectiveTransform(src, dst)
40     Minv = cv2.getPerspectiveTransform(dst, src)
41
42     # Warp the image using OpenCV warpPerspective()
43     warped = cv2.warpPerspective(img, M, img_size)
44     # Return the resulting image and matrix
45     return warped, M, Minv
46
47
48
49 def HLS_L_Threshold(img, thresh=(195, 255)):
50     """
51     Threshold the input image to the L-channel of the HLS color
        space. Which is the lightness
52
53     """
54     img = img[:, :, 1]
55     img = img * (255 / np.max(img))
56     binary_output = np.zeros_like(img)
57     binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1
58     return binary_output
59
60
61
62 def LAB_B_Threshold(img, thresh=(230, 255)):
63     """
64     Threshold the input image to the B-channel of the LAB color
        space.
65
66     """
66     img = img[:, :, 2]
67     if np.max(img) > 175:
68         img = img * (255 / np.max(img))
69     binary_output = np.zeros_like(img)
70     binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1
```

```

71     return binary_output
72
73
74
75 def Combined_HLS_LAB_Threshold(img):
76     """
77     Threshold the input image to the L-channel of the HLS color
78     space and the B-channel of the LAB color space.
79     """
80     img_HLS = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
81     img_LAB = cv2.cvtColor(img, cv2.COLOR_RGB2Lab)
82     img_thresh_HLS = HLS_L_Threshold(img_HLS)
83     img_thresh_LAB = LAB_B_Threshold(img_LAB)
84     combined_img = np.zeros_like(img_thresh_HLS)
85     combined_img[((img_thresh_HLS == 1) | (img_thresh_LAB == 1))] =
86         1
87     return combined_img
88
89 def Sliding_Window_Method(img):
90     """
91     Fit a polynomial to the input binary image.
92     """
93     # Take a histogram of the bottom half of the image
94     histogram = np.sum(img[img.shape[0] // 2:, :], axis=0)
95     # Find the peak of the left and right halves of the histogram
96     # These will be the starting point for the left and right lines
97     midpoint = np.int(histogram.shape[0] // 2)
98     quarter_point = np.int(midpoint // 2)
99     # Previously the left/right base was the max of the left/right
100     # half of the histogram
101     # this changes it so that only a quarter of the histogram (
102     # directly to the left/right) is considered
103     leftx_base = np.argmax(histogram[quarter_point:midpoint]) +
104         quarter_point
105     rightx_base = np.argmax(histogram[midpoint:(midpoint +
106         quarter_point)]) + midpoint
107
108     # Choose the number of sliding windows
109     nwindows = 70
110     # Set height of windows

```

```

107     window_height = np.int(img.shape[0] / nwindows)
108     # Identify the x and y positions of all nonzero pixels in the
        image
109     nonzero = img.nonzero()
110     nonzeroy = np.array(nonzero[0])
111     nonzerox = np.array(nonzero[1])
112     # Current positions to be updated for each window
113     leftx_current = leftx_base
114     rightx_current = rightx_base
115     # Set the width of the windows +/- margin
116     margin = 80
117     # Set minimum number of pixels found to recenter window
118     minpix = 40
119     # Create empty lists to receive left and right lane pixel
        indices
120     left_lane_inds = []
121     right_lane_inds = []
122     # Rectangle data for visualization
123     rectangle_data = []
124
125     # Step through the windows one by one
126     for window in range(nwindows):
127         # Identify window boundaries in x and y (and right and left)
128         win_y_low = img.shape[0] - (window + 1) * window_height
129         win_y_high = img.shape[0] - window * window_height
130         win_xleft_low = leftx_current - margin
131         win_xleft_high = leftx_current + margin
132         win_xright_low = rightx_current - margin
133         win_xright_high = rightx_current + margin
134         rectangle_data.append((win_y_low, win_y_high, win_xleft_low,
            win_xleft_high, win_xright_low, win_xright_high))
135         # Identify the nonzero pixels in x and y within the window
136         good_left_inds = ((nonzeroy >= win_y_low) & (nonzeroy <
            win_y_high) & (nonzerox >= win_xleft_low) &
137             (nonzerox < win_xleft_high)).nonzero()[0]
138         good_right_inds = ((nonzeroy >= win_y_low) & (nonzeroy <
            win_y_high) & (nonzerox >= win_xright_low) &
139             (nonzerox < win_xright_high)).nonzero()
            [0]
140         # Append these indices to the lists
141         left_lane_inds.append(good_left_inds)
142         right_lane_inds.append(good_right_inds)

```

```

143     # If you found > minpix pixels , recenter next window on
        their mean position
144     if len(good_left_inds) > minpix:
145         leftx_current = np.int(np.mean(nonzerox[good_left_inds])
            )
146     if len(good_right_inds) > minpix:
147         rightx_current = np.int(np.mean(nonzerox[good_right_inds
            ]))
148
149     # Concatenate the arrays of indices
150     left_lane_inds = np.concatenate(left_lane_inds)
151     right_lane_inds = np.concatenate(right_lane_inds)
152
153     # Extract left and right line pixel positions
154     leftx = nonzerox[left_lane_inds]
155     lefty = nonzeroy[left_lane_inds]
156     rightx = nonzerox[right_lane_inds]
157     righty = nonzeroy[right_lane_inds]
158
159     left_fit , right_fit = (None, None)
160     # Fit a second order polynomial to each
161     if len(leftx) != 0:
162         left_fit = np.polyfit(lefty , leftx , 2)
163     if len(rightx) != 0:
164         right_fit = np.polyfit(righty , rightx , 2)
165
166     visualization_data = (rectangle_data , histogram)
167
168     return left_fit , right_fit , left_lane_inds , right_lane_inds ,
        visualization_data
169
170 def PolynomialFit_Previous_Fit(img, left_fit_prev , right_fit_prev):
171     """
172     Fit a polynomial to the input binary image based upon a previous
        fit .
173     This assumes that the fit will not change significantly from one
        video frame to the next .
174     Parameters:
175         img: Input image .
176         left_fit_prev :
177         right_fit_prev :
178     """

```

```

179     nonzero = img.nonzero()
180     nonzero_y = np.array(nonzero[0])
181     nonzero_x = np.array(nonzero[1])
182     margin = 80
183     left_lane_inds = ((nonzero_x > (left_fit_prev[0]*(nonzero_y**2) +
184         left_fit_prev[1]*nonzero_y + left_fit_prev[2] - margin))
185         & (nonzero_x < (left_fit_prev[0]*(nonzero_y**2) +
186         left_fit_prev[1]*nonzero_y + left_fit_prev[2] +
187         margin)))
188     right_lane_inds = ((nonzero_x > (right_fit_prev[0]*(nonzero_y**2)
189         + right_fit_prev[1]*nonzero_y + right_fit_prev[2] - margin))
190         & (nonzero_x < (right_fit_prev[0]*(nonzero_y**2) +
191         right_fit_prev[1]*nonzero_y + right_fit_prev[2] +
192         margin)))
193     leftx = nonzero_x[left_lane_inds]
194     lefty = nonzero_y[left_lane_inds]
195     rightx = nonzero_x[right_lane_inds]
196     righty = nonzero_y[right_lane_inds]
197     left_fit_new, right_fit_new = (None, None)
198     if len(leftx) != 0:
199         left_fit_new = np.polyfit(lefty, leftx, 2)
200     if len(rightx) != 0:
201         right_fit_new = np.polyfit(righty, rightx, 2)
202     return left_fit_new, right_fit_new, left_lane_inds,
203         right_lane_inds
204
205
206 def Image_Process(img):
207     """
208     Apply undistortion, perspective transform, and color space
209     thresholding to the input image.
210     Parameters:
211         img: Input image.
212     """
213     # Perspective Transform
214     img, M, Minv = Transform_Camera_View(img, src, dst)
215
216     # Create a thresholded binary image
217     img = Combined_HLS_LAB_Threshold(img)
218
219     return img, Minv

```

```

213 def Curve_Position(img, l_fit , r_fit , l_lane_inds , r_lane_inds):
214     """
215     Calculating the lane curvature and the vehicle position on the
        lane.
216     Parameters:
217         img: Input image.
218         l_fit , r_fit , l_lane_inds , r_lane_inds: Detected lane
        lines.
219     """
220     # Define conversions in x and y from pixels space to meters
221     ym_per_pix = 3.048/100 # meters per pixel in y dimension, lane
        line is 10 ft = 3.048 meters
222     xm_per_pix = 3.7/378 # meters per pixel in x dimension, lane
        width is 12 ft = 3.7 meters
223     left_curverad , right_curverad , center_dist = (0, 0, 0)
224     # Define y-value where we want radius of curvature
225     # I'll choose the maximum y-value, corresponding to the bottom
        of the image
226     h = img.shape[0]
227     ploty = np.linspace(0, h-1, h)
228     y_eval = np.max(ploty)
229
230     # Identify the x and y positions of all nonzero pixels in the
        image
231     nonzero = img.nonzero()
232     nonzeroy = np.array(nonzero[0])
233     nonzerox = np.array(nonzero[1])
234     # Again, extract left and right line pixel positions
235     leftx = nonzerox[l_lane_inds]
236     lefty = nonzeroy[l_lane_inds]
237     rightx = nonzerox[r_lane_inds]
238     righty = nonzeroy[r_lane_inds]
239
240     if len(leftx) != 0 and len(rightx) != 0:
241         # Fit new polynomials to x,y in world space
242         left_fit_cr = np.polyfit(lefty*ym_per_pix , leftx*xm_per_pix ,
            2)
243         right_fit_cr = np.polyfit(righty*ym_per_pix , rightx*
            xm_per_pix , 2)
244         # Calculate the new radii of curvature
245         left_curverad = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix +
            left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])

```

```

246     right_curverad = ((1 + (2*right_fit_cr[0]*y_eval*ym_per_pix
      + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr
      [0])
247     # Now our radius of curvature is in meters
248
249     # Distance from center is image x midpoint - mean of l_fit and
      r_fit intercepts
250     if r_fit is not None and l_fit is not None:
251         car_position = img.shape[1]/2
252         l_fit_x_int = l_fit[0]*h**2 + l_fit[1]*h + l_fit[2]
253         r_fit_x_int = r_fit[0]*h**2 + r_fit[1]*h + r_fit[2]
254         lane_center_position = (r_fit_x_int + l_fit_x_int) /2
255         center_dist = (car_position - lane_center_position) *
      xm_per_pix
256     return left_curverad , right_curverad , center_dist
257
258
259 def DrawLane(original_img , binary_img , l_fit , r_fit , Minv):
260     """
261     Draw the detected lane over the input image.
262     Parameters:
263         original_img: Input frame.
264         binary_img: Preprocessed image.
265         l_fit , r_fit: Detected lanes.
266         Minv: Calibration matrix.
267     """
268     new_img = np.copy(original_img)
269     if l_fit is None or r_fit is None:
270         return original_img
271     warp_zero = np.zeros_like(binary_img).astype(np.uint8)
272     color_warp = np.dstack((warp_zero , warp_zero , warp_zero))
273     h,w = binary_img.shape
274     ploty = np.linspace(0, h-1, num=h)
275     left_fitx = l_fit[0]*ploty**2 + l_fit[1]*ploty + l_fit[2]
276     right_fitx = r_fit[0]*ploty**2 + r_fit[1]*ploty + r_fit[2]
277     pts_left = np.array([np.transpose(np.vstack([left_fitx , ploty]))
      ])
278     pts_right = np.array([np.flipud(np.transpose(np.vstack([
      right_fitx , ploty])))])
279     pts = np.hstack((pts_left , pts_right))
280     cv2.fillPoly(color_warp , np.int_([pts]) , (0,255, 0))

```

```

281     cv2.polylines(color_warp, np.int32([pts_left]), isClosed=False,
282                 color=(255,0,255), thickness=15)
283     cv2.polylines(color_warp, np.int32([pts_right]), isClosed=False,
284                 color=(0,255,255), thickness=15)
285     newwarp = cv2.warpPerspective(color_warp, Minv, (w, h))
286     result = cv2.addWeighted(new_img, 1, newwarp, 0.5, 0)
287     return result
288
289 def Write_Data(original_img, curv_rad, center_dist):
290     """
291     Write the lane curvature and vehicle position over the input
292     image.
293     Parameters:
294         original_img: Input frame.
295         curv_rad: Lane curvature.
296         center_dist: Vehicle position.
297     """
298     new_img = np.copy(original_img)
299     h = new_img.shape[0]
300     font = cv2.FONT_HERSHEY_DUPLEX
301     text = 'Curve radius: ' + '{:04.2f}'.format(curv_rad) + 'm'
302     cv2.putText(new_img, text, (40,70), font, 1.5, (255,255,255), 2,
303                 cv2.LINE_AA)
304     direction = ''
305     if center_dist > 0:
306         direction = 'right'
307     elif center_dist < 0:
308         direction = 'left'
309     abs_center_dist = abs(center_dist)
310     text = '{:04.3f}'.format(abs_center_dist) + 'm' + direction + '
311         of center'
312     cv2.putText(new_img, text, (40,120), font, 1.5, (255,255,255),
313                 2, cv2.LINE_AA)
314     return new_img
315
316 class Line():
317     def __init__(self):
318         # was the line detected in the last iteration?
319         self.detected = False

```

```

317     # x values of the last n fits of the line
318     self.recent_xfitted = []
319     #average x values of the fitted line over the last n
        iterations
320     self.bestx = None
321     #polynomial coefficients averaged over the last n iterations
322     self.best_fit = None
323     #polynomial coefficients for the most recent fit
324     self.current_fit = []
325     #radius of curvature of the line in some units
326     self.radius_of_curvature = None
327     #distance in meters of vehicle center from the line
328     self.line_base_pos = None
329     #difference in fit coefficients between last and new fits
330     self.diffs = np.array([0,0,0], dtype='float')
331     #number of detected pixels
332     self.px_count = None
333 def add_fit(self, fit, inds):
334     if fit is not None:
335         self.detected = True
336         self.px_count = np.count_nonzero(inds)
337         self.current_fit.append(fit)
338         if len(self.current_fit) > 5:
339             self.current_fit = self.current_fit[len(self.
                current_fit)-5:]
340         self.best_fit = np.average(self.current_fit, axis=0)
341     else:
342         self.detected = False
343         if len(self.current_fit) > 0:
344             self.current_fit = self.current_fit[:len(self.
                current_fit)-1]
345         if len(self.current_fit) > 0:
346             self.best_fit = np.average(self.current_fit, axis=0)
347
348
349 def Frame_Processor(img):
350     """
351     Process the input frame and return the frame with detected lane
        and curvature and vehicle position information.
352     Parameters:
353         img: Input frame.
354     """

```

```
355     new_img = np.copy(img)
356     img_bin, Minv = Image_Process(img)
357     if not l_line.detected or not r_line.detected:
358         l_fit, r_fit, l_lane_inds, r_lane_inds, _ =
359             Sliding_Window_Method(img_bin)
360     else:
361         l_fit, r_fit, l_lane_inds, r_lane_inds =
362             PolynomialFit_Previous_Fit(img_bin, l_line.best_fit,
363                                     r_line.best_fit)
364
365     l_line.add_fit(l_fit, l_lane_inds)
366     r_line.add_fit(r_fit, r_lane_inds)
367     img_out1 = Draw_Lane(new_img, img_bin, l_fit, r_fit, Minv)
368     rad_l, rad_r, d_center = Curve_Position(img_bin, l_fit, r_fit,
369                                           l_lane_inds, r_lane_inds)
370     img_out = Write_Data(img_out1, (rad_l+rad_r)/2, d_center)
371     return img_out
372
373
374
375
376
377
378 img = cv2.imread('Test_E18_Grimstad.PNG')
379 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
380 image_shape = img.shape
381 print("Image shape:", image_shape)
382 plt.imshow(img)
383 plt.show()
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```

393
394 img_HLS = cv2.cvtColor(img_RGB, cv2.COLOR_RGB2HLS)
395 img_HLS_L = img_HLS[:, :, 1]
396
397 thresh_HLS = HLS_L_Threshold(img_HLS)
398 display_Images(img_HLS, thresh_HLS, 'HLS Image', 'L-Thresholded HLS
    Image', 14, 7, cmap='gray')
399
400 thresh_LAB = LAB_B_Threshold(img_LAB)
401 display_Images(img_LAB, thresh_LAB, 'LAB Image', 'B-Thresholded LAB
    Image', 14, 7, cmap='gray')
402
403 threshold_color_img = Combined_HLS_LAB_Threshold(img_warped)
404 display_Images(img_warped, threshold_color_img, 'RGB image', '
    Combined Thresholded Image', 14, 7, cmap='gray')
405
406
407 #####NEW_START#####
408
409 image_org = cv2.imread('Test_E18_Grimstad.PNG')
410 image_org = cv2.cvtColor(image_org, cv2.COLOR_BGR2RGB)
411
412 image_processed, Minv = Image_Process(image_org)
413
414
415 display_Images(image_org, image_processed, 'Original test image', '
    Processed test image', 14, 7, cmap='gray')
416
417
418 img = image_processed
419 left_fit, right_fit, left_lane_inds, right_lane_inds,
    visualization_data = Sliding_Window_Method(img)
420 h = img.shape[0]
421 left_fit_x_int = left_fit[0]*h**2 + left_fit[1]*h + left_fit[2]
422 right_fit_x_int = right_fit[0]*h**2 + right_fit[1]*h + right_fit[2]
423 rectangles = visualization_data[0]
424 histogram = visualization_data[1]
425 # Create an output image to draw on and visualize the result
426 out_img = np.uint8(np.dstack((img, img, img))*255)
427 # Generate x and y values for plotting
428 ploty = np.linspace(0, img.shape[0]-1, img.shape[0])
429 left_fitx = left_fit[0]*ploty**2 + left_fit[1]*ploty + left_fit[2]

```

```

430 right_fitx = right_fit[0]*ploty**2 + right_fit[1]*ploty + right_fit
    [2]
431 for rect in rectangles:
432     # Draw the windows on the visualization image
433     cv2.rectangle(out_img,(rect[2],rect[0]),(rect[3],rect[1])
        ,(0,255,0), 2)
434     cv2.rectangle(out_img,(rect[4],rect[0]),(rect[5],rect[1])
        ,(0,255,0), 2)
435 # Identify the x and y positions of all nonzero pixels in the image
436 nonzero = img.nonzero()
437 nonzeroy = np.array(nonzero[0])
438 nonzeroy = np.array(nonzero[1])
439 out_img[nonzeroy[left_lane_inds], nonzeroy[right_lane_inds]] = [255,
    0, 0]
440 out_img[nonzeroy[right_lane_inds], nonzeroy[right_lane_inds]] =
    [100, 200, 255]
441 plt.figure(figsize=(14, 7))
442 plt.subplot(1, 2, 1)
443 plt.imshow(image_org)
444 plt.xlabel('Original image', fontsize=15)
445 plt.xticks([])
446 plt.yticks([])
447 plt.subplot(1, 2, 2)
448 plt.imshow(out_img)
449 plt.xlabel('Sliding window', fontsize=15)
450 plt.plot(left_fitx, ploty, color='yellow')
451 plt.plot(right_fitx, ploty, color='yellow')
452 plt.xlim(0, 1280)
453 plt.ylim(720, 0)
454 plt.xticks([])
455 plt.yticks([])
456 plt.show()
457
458
459 margin = 50
460 left_fit, right_fit, left_lane_inds, right_lane_inds,
    visualization_data = Sliding_Window_Method(img)
461 left_fit2, right_fit2, left_lane_inds2, right_lane_inds2 =
    PolynomialFit_Previous_Fit(img, left_fit, right_fit)
462 ploty = np.linspace(0, img.shape[0]-1, img.shape[0])
463 left_fitx = left_fit[0]*ploty**2 + left_fit[1]*ploty + left_fit[2]

```

```

464 right_fitx = right_fit[0]*ploty**2 + right_fit[1]*ploty + right_fit
    [2]
465 left_fitx2 = left_fit2[0]*ploty**2 + left_fit2[1]*ploty + left_fit2
    [2]
466 right_fitx2 = right_fit2[0]*ploty**2 + right_fit2[1]*ploty +
    right_fit2[2]
467 out_img = np.uint8(np.dstack((img, img, img))*255)
468 window_img = np.zeros_like(out_img)
469 nonzero = img.nonzero()
470 nonzeroy = np.array(nonzero[0])
471 nonzeroy = np.array(nonzero[1])
472 out_img[nonzeroy[left_lane_inds2], nonzeroy[right_lane_inds2]] =
    [255, 0, 0]
473 out_img[nonzeroy[right_lane_inds2], nonzeroy[left_lane_inds2]] =
    [0, 0, 255]
474 left_line_window1 = np.array([np.transpose(np.vstack([left_fitx -
    margin, ploty]))])
475 left_line_window2 = np.array([np.flipud(np.transpose(np.vstack([
    left_fitx+margin, ploty])))])
476 left_line_pts = np.hstack((left_line_window1, left_line_window2))
477 right_line_window1 = np.array([np.transpose(np.vstack([right_fitx -
    margin, ploty]))])
478 right_line_window2 = np.array([np.flipud(np.transpose(np.vstack([
    right_fitx+margin, ploty])))])
479 right_line_pts = np.hstack((right_line_window1, right_line_window2))
480 cv2.fillPoly(window_img, np.int_([left_line_pts]), (0,255, 0))
481 cv2.fillPoly(window_img, np.int_([right_line_pts]), (0,255, 0))
482 result = cv2.addWeighted(out_img, 1, window_img, 0.3, 0)
483 plt.figure(figsize=(14, 7))
484 plt.subplot(1, 2, 1)
485 plt.imshow(image_org)
486 plt.xlabel('Original image', fontsize=15)
487 plt.xticks([])
488 plt.yticks([])
489 plt.subplot(1, 2, 2)
490 plt.imshow(result)
491 plt.xlabel('Polyfit using previous fit', fontsize=15)
492 plt.plot(left_fitx2, ploty, color='yellow')
493 plt.plot(right_fitx2, ploty, color='yellow')
494 plt.xlim(0, 1280)
495 plt.ylim(720, 0)
496 plt.xticks([])

```

```
497 plt.yticks([])
498 plt.show()
499
500
501 rad_l, rad_r, d_center = Curve_Position(image_processed, left_fit,
    right_fit, left_lane_inds2, right_lane_inds2)
502 print('Radius of curvature for example:', rad_l, 'm', rad_r, 'm')
503 print('Distance from lane center for example:', d_center, 'm')
504 plt.figure(figsize=(14, 7))
505 plt.subplot(1, 2, 1)
506 plt.imshow(image_org)
507 plt.xlabel('Original image', fontsize=15)
508 plt.xticks([])
509 plt.yticks([])
510 plt.subplot(1, 2, 2)
511 plt.imshow(result)
512 plt.xlabel('Polyfit using previous fit', fontsize=15)
513 plt.plot(left_fitx2, ploty, color='yellow')
514 plt.plot(right_fitx2, ploty, color='yellow')
515 plt.xlim(0, 1280)
516 plt.ylim(720, 0)
517 plt.xticks([])
518 plt.yticks([])
519 plt.show()
520
521
522
523 left_fit, right_fit, left_lane_inds, right_lane_inds,
    visualization_data = Sliding_Window_Method(image_processed)
524 left_fit2, right_fit2, left_lane_inds2, right_lane_inds2 =
    PolynomialFit_Previous_Fit(image_processed, left_fit, right_fit)
525 rad_l, rad_r, d_center = Curve_Position(image_processed, left_fit,
    right_fit, left_lane_inds2, right_lane_inds2)
526 result = Draw_Lane(image_org, image_processed, left_fit, right_fit,
    Minv)
527 result = Write_Data(result, (rad_l+rad_r)/2, d_center)
528 plt.figure(figsize=(14, 7))
529 plt.subplot(1, 2, 1)
530 plt.imshow(image_org)
531 plt.xlabel('Original image', fontsize=15)
532 plt.xticks([])
533 plt.yticks([])
```

```
534 plt.subplot(1, 2, 2)
535 plt.imshow(result)
536 plt.xlabel('Detected Lane', fontsize=15)
537 plt.xlim(0, 1280)
538 plt.ylim(720, 0)
539 plt.xticks([])
540 plt.yticks([])
541 plt.show()
```

F.2 IMU Calibration Script - Intel RealSense 435i [1]

```
1 #!/usr/bin/python
2 from __future__ import print_function
3 import numpy as np
4 import sys
5 import json
6 import ctypes
7 import os
8 import binascii
9 import struct
10 import pyrealsense2 as rs
11 import ctypes
12 import time
13 import enum
14 import threading
15
16 is_data = None
17 get_key = None
18 if os.name == 'posix':
19     import select
20     import tty
21     import termios
22
23     is_data = lambda : select.select([sys.stdin], [], [], 0) == ([
24         sys.stdin], [], [])
25     get_key = lambda : sys.stdin.read(1)
26 elif os.name == 'nt':
27     import msvcrt
28     is_data = msvcrt.kbhit
29     get_key = lambda : msvcrt.getch()
30
```



```
31 else:
32     raise Exception('Unsupported OS: %s' % os.name)
33
34 if sys.version_info[0] < 3:
35     input = raw_input
36
37 max_float = struct.unpack('f',b'\xff\xff\xff\xff')[0]
38 max_int = struct.unpack('i',b'\xff\xff\xff\xff')[0]
39 max_uint8 = struct.unpack('B', b'\xff')[0]
40
41 g = 9.80665
42
43 COLOR_RED    = "\033[1;31m"
44 COLOR_BLUE   = "\033[1;34m"
45 COLOR_CYAN   = "\033[1;36m"
46 COLOR_GREEN  = "\033[0;32m"
47 COLOR_RESET  = "\033[0;0m"
48 COLOR_BOLD   = "\033[;1m"
49 COLOR_REVERSE = "\033[;7m"
50
51 class imu_wrapper:
52     class Status(enum.Enum):
53         idle = 0,
54         rotate = 1,
55         wait_to_stable = 2,
56         collect_data = 3
57
58     def __init__(self):
59         self.pipeline = None
60         self.imu_sensor = None
61         self.status = self.Status(self.Status.idle)# 0 - idle, 1 -
62             rotate to position, 2 - wait to stable, 3 - pick data
63         self.thread = threading.Condition()
64         self.step_start_time = time.time()
65         self.time_to_stable = 3
66         self.time_to_collect = 2
67         self.samples_to_collect = 1000
68         self.rotating_threshold = 0.1
69         self.moving_threshold_factor = 0.1
70         self.collected_data_gyro = []
71         self.collected_data_accel = []
72         self.callback_lock = threading.Lock()
```

```

72     self.max_norm = np.linalg.norm(np.array([0.5, 0.5, 0.5]))
73     self.line_length = 20
74     self.is_done = False
75     self.is_data = False
76
77     def escape_handler(self):
78         self.thread.acquire()
79         self.status = self.Status.idle
80         self.is_done = True
81         self.thread.notify()
82         self.thread.release()
83         sys.exit(-1)
84
85     def imu_callback(self, frame):
86         if not self.is_data:
87             self.is_data = True
88
89         with self.callback_lock:
90             try:
91                 if is_data():
92                     c = get_key()
93                     if c == '\x1b':          # x1b is ESC
94                         self.escape_handler()
95
96                 if self.status == self.Status.idle:
97                     return
98                 pr = frame.get_profile()
99                 data = frame.as_motion_frame().get_motion_data()
100                data_np = np.array([data.x, data.y, data.z])
101                elapsed_time = time.time() - self.step_start_time
102
103                ## Status.collect_data
104                if self.status == self.Status.collect_data:
105                    sys.stdout.write('\r %15s' % self.status)
106                    part_done = len(self.collected_data_accel) /
107                                float(self.samples_to_collect)
108                    # sys.stdout.write(': %-3.1f (secs)' % (self.
109                                time_to_collect - elapsed_time))
110
111                    color = COLOR.GREEN
112                    if pr.stream_type() == rs.stream.gyro:
113                        self.collected_data_gyro.append(np.append(

```

```

        frame.get_timestamp(), data_np))
112     is_moving = any(abs(data_np) > self.
        rotating_threshold)
113     else:
114         is_in_norm = np.linalg.norm(data_np - self.
        crnt_bucket) < self.max_norm
115         if is_in_norm:
116             self.collected_data_accel.append(np.
                append(frame.get_timestamp(), data_np
                ))
117         else:
118             color = COLOR_RED
119             is_moving = abs(np.linalg.norm(data_np) - g)
                / g > self.moving_threshold_factor
120
121             sys.stdout.write(color)
122             sys.stdout.write('[ '+'. '*int(part_done*self.
                line_length)+' ' *int((1-part_done)*self.
                line_length) + ']')
123             sys.stdout.write(COLOR_RESET)
124
125         if is_moving:
126             print('WARNING: MOVING')
127             self.status = self.Status.rotate
128             return
129
130         # if elapsed_time > self.time_to_collect:
131         if part_done >= 1:
132             self.status = self.Status.collect_data
133             sys.stdout.write('\n\nDirection data
                collected.')
134             self.thread.acquire()
135             self.status = self.Status.idle
136             self.thread.notify()
137             self.thread.release()
138             return
139
140         if pr.stream_type() == rs.stream.gyro:
141             return
142         sys.stdout.write('\r %15s' % self.status)
143         crnt_dir = np.array(data_np) / np.linalg.norm(
            data_np)

```

```

144         crnt_diff = self.crnt_direction - crnt_dir
145         is_in_norm = np.linalg.norm(data_np - self.
           crnt_bucket) < self.max_norm
146
147     ## Status.rotate
148     if self.status == self.Status.rotate:
149         sys.stdout.write(': %35s' % (np.array2string(
           crnt_diff, precision=4, suppress_small=True)
           ))
150         sys.stdout.write(': %35s' % (np.array2string(abs
           (crnt_diff) < 0.1)))
151         if is_in_norm:
152             self.status = self.Status.wait_to_stable
153             sys.stdout.write('\r'+ ' '*90)
154             self.step_start_time = time.time()
155             return
156
157     ## Status.wait_to_stable
158     if self.status == self.Status.wait_to_stable:
159         sys.stdout.write(': %-3.1f (secs)' % (self.
           time_to_stable - elapsed_time))
160         if not is_in_norm:
161             self.status = self.Status.rotate
162             return
163         if elapsed_time > self.time_to_stable:
164             self.collected_data_gyro = []
165             self.collected_data_accel = []
166             self.status = self.Status.collect_data
167             self.step_start_time = time.time()
168             return
169         return
170     except Exception as e:
171         print('ERROR?' + str(e))
172         self.thread.acquire()
173         self.status = self.Status.idle
174         self.thread.notify()
175         self.thread.release()
176
177     def get_measurements(self, buckets, bucket_labels):
178         measurements = []
179         print('_____')
180         print('*** Press ESC to Quit ***')
```

```

181     print( '—————' )
182     for bucket , bucket_label in zip( buckets , bucket_labels ):
183         self.crnt_bucket = np.array( bucket )
184         self.crnt_direction = np.array( bucket ) / np.linalg.norm(
185             np.array( bucket )
186         )
187         print( '\nAlign to direction: ', self.crnt_direction , ' ' ,
188             , bucket_label )
189         self.status = self.Status.rotate
190         self.thread.acquire()
191         while ( not self.is_done and self.status != self.Status.
192             idle ):
193             self.thread.wait( 3 )
194             if not self.is_data :
195                 raise Exception( 'No IMU data. Check connectivity
196                     .' )
197             if self.is_done :
198                 raise Exception( 'User Abort.' )
199             measurements.append( np.array( self.collected_data_accel ) )
200         return np.array( measurements ) , np.array( self.
201             collected_data_gyro )
202
203 def enable_imu_device( self , serial_no ):
204     self.pipeline = rs.pipeline()
205     cfg = rs.config()
206     cfg.enable_device( serial_no )
207     try :
208         self.pipeline.start( cfg )
209     except Exception as e :
210         print( 'ERROR: ', str( e ) )
211         return False
212
213 # self.sync_imu_by_this_stream = rs.stream.any
214 active_imu_profiles = []
215
216 active_profiles = dict()
217 self.imu_sensor = None
218 for sensor in self.pipeline.get_active_profile().get_device
219     ().sensors :
220     for pr in sensor.get_stream_profiles() :
221         if pr.stream_type() == rs.stream.gyro and pr.format
222             () == rs.format.motion_xyz32f :
223             active_profiles[ pr.stream_type() ] = pr

```

```

216         self.imu_sensor = sensor
217         if pr.stream_type() == rs.stream.accel and pr.format
218            () == rs.format.motion_xyz32f:
219             active_profiles[pr.stream_type()] = pr
220             self.imu_sensor = sensor
221         if self.imu_sensor:
222             break
223     if not self.imu_sensor:
224         print('No IMU sensor found.')
225         return False
226     print('\n'.join(['FOUND %s with fps=%s' % (str(ap[0]).split
227                    ('.')[1].upper(), ap[1].fps()) for ap in active_profiles.
228                    items()])))
229     active_imu_profiles = list(active_profiles.values())
230     if len(active_imu_profiles) < 2:
231         print('Not all IMU streams found.')
232         return False
233     self.imu_sensor.stop()
234     self.imu_sensor.close()
235     self.imu_sensor.open(active_imu_profiles)
236     self.imu_start_loop_time = time.time()
237     self.imu_sensor.start(self.imu_callback)
238
239     # Make the device use the original IMU values and not
240     # already calibrated:
241     if self.imu_sensor.supports(rs.option.
242                               enable_motion_correction):
243         self.imu_sensor.set_option(rs.option.
244                                   enable_motion_correction, 0)
245     return True
246
247 class CHeader:
248     def __init__(self, version, table_type):
249         self.buffer = np.ones(16, dtype=np.uint8) * 255
250         self.buffer[0] = int(version[0], 16)
251         self.buffer[1] = int(version[1], 16)
252         self.buffer.dtype=np.uint16
253         self.buffer[1] = int(table_type, 16)
254
255     def size(self):
256         return 16

```

```

252     def set_data_size(self, size):
253         self.buffer.dtype=np.uint32
254         self.buffer[1] = size
255
256     def set_crc32(self, crc32):
257         self.buffer.dtype=np.uint32
258         self.buffer[3] = crc32 % (1<<32)    # convert from signed to
                unsigned 32 bit
259
260     def get_buffer(self):
261         self.buffer.dtype=np.uint8
262         return self.buffer
263
264
265     def bitwise_int_to_float(ival):
266         return struct.unpack('f', struct.pack('i', ival))[0]
267
268     def bitwise_float_to_int(fval):
269         return struct.unpack('i', struct.pack('f', fval))[0]
270
271     def parse_buffer(buffer):
272         cmd_size = 24
273         header_size = 16
274
275         buffer.dtype=np.uint32
276         tab1_size = buffer[3]
277         buffer.dtype=np.uint8
278         print('tab1_size (all_data): ', tab1_size)
279
280         tab1 = buffer[cmd_size:cmd_size+tab1_size] # 520 == epprom++
281         tab1.dtype=np.uint32
282         tab2_size = tab1[1]
283         tab1.dtype=np.uint8
284         print('tab2_size (calibration-table): ', tab2_size)
285
286         tab2 = tab1[header_size:header_size+tab2_size] # calibration
                table
287         tab2.dtype=np.uint32
288         tab3_size = tab2[1]
289         tab2.dtype=np.uint8
290         print('tab3_size (calibration-table): ', tab3_size)
291

```

```

292     tab3 = tab2[header_size:header_size+tab3_size] # D435 IMU Calib
           Table
293     tab3.dtype=np.uint32
294     tab4_size = tab3[1]
295     tab3.dtype=np.uint8
296     print('tab4_size (D435_IMU_Calib_Table): ', tab4_size)
297
298     tab4 = tab3[header_size:header_size+tab4_size] # calibration
           data
299     return tab1, tab2, tab3, tab4
300
301 def get_D435_IMU_Calib_Table(X):
302     version = ['0x02', '0x01']
303     table_type = '0x20'
304     header = CHeader(version, table_type)
305
306     header_size = header.size()
307     data_size = 37*4 + 96
308     size_of_buffer = header_size + data_size # according to table
           "D435 IMU Calib Table" here: https://user-images.
           githubusercontent.com/6958867/50902974-20507500-1425-11e9-8
           ca5-8bd2ac2d0ea1.png
309     assert(size_of_buffer % 4 == 0)
310     buffer = np.ones(size_of_buffer, dtype=np.uint8) * 255
311
312     use_extrinsics = False
313     use_intrinsics = True
314
315     data_buffer = np.ones(data_size, dtype=np.uint8) * 255
316     data_buffer.dtype = np.float32
317
318     data_buffer[0] = bitwise_int_to_float(np.int32(int(
           use_intrinsics)) << 8 |
319                                           np.int32(int(
           use_extrinsics)))
320
321     intrinsic_vector = np.zeros(24, dtype=np.float32)
322     intrinsic_vector[:9] = X[:3, :3].T.flatten()
323     intrinsic_vector[9:12] = X[:3, 3]
324     intrinsic_vector[12:21] = X[3:, :3].flatten()
325     intrinsic_vector[21:24] = X[3:, 3]
326

```

```

327     data_buffer[13:13+X.size] = intrinsic_vector
328     data_buffer.dtype = np.uint8
329
330     header.set_data_size(data_size)
331
332     header.set_crc32(binascii.crc32(data_buffer))
333     buffer[:header_size] = header.get_buffer()
334     buffer[header_size:] = data_buffer
335     return buffer
336
337
338 def get_calibration_table(d435_imu_calib_table):
339     version = ['0x02', '0x00']
340     table_type = '0x20'
341
342     header = CHeader(version, table_type)
343
344     d435_imu_calib_table_size = d435_imu_calib_table.size
345     sn_table_size = 32
346     data_size = d435_imu_calib_table_size + sn_table_size
347
348     header_size = header.size()
349     size_of_buffer = header_size + data_size # according to table
        "D435 IMU Calib Table" in "https://sharepoint ger.ith.intel.
        com/sites/3D_project/Shared%20Documents/Arch/D400/FW/
        D435i-IMU-Calibration-EEPROM-0-52.xlsx"
350     assert(size_of_buffer % 4 == 0)
351     buffer = np.ones(size_of_buffer, dtype=np.uint8) * 255
352
353     data_buffer = np.ones(data_size, dtype=np.uint8) * 255
354     data_buffer[:d435_imu_calib_table_size] = d435_imu_calib_table
355
356     header.set_data_size(data_size)
357     header.set_crc32(binascii.crc32(data_buffer))
358
359     buffer[:header_size] = header.get_buffer()
360     buffer[header_size:header_size+data_size] = data_buffer
361     return buffer
362
363 def get_eeeprom(calibration_table):
364     version = ['0x01', '0x01']
365     table_type = '0x09'

```

```

366
367     header = CHeader(version, table_type)
368
369     DC_MM_EEPROM_SIZE = 520
370     # data_size = calibration_table.size
371
372     header_size = header.size()
373     size_of_buffer = DC_MM_EEPROM_SIZE
374     data_size = size_of_buffer - header_size
375     # size_of_buffer = header_size + data_size
376
377     assert(size_of_buffer % 4 == 0)
378     buffer = np.ones(size_of_buffer, dtype=np.uint8) * 255
379
380     header.set_data_size(data_size)
381     buffer[header_size:header_size+calibration_table.size] =
           calibration_table
382     header.set_crc32(binascii.crc32(buffer[header_size:]))
383
384     buffer[:header_size] = header.get_buffer()
385
386     return buffer
387
388 def write_eeprom_to_camera(eeprom, serial_no=''):
389     # DC_MM_EEPROM_SIZE = 520
390     DC_MM_EEPROM_SIZE = eeprom.size
391     DS5_CMD_LENGTH = 24
392
393     MMEW_Cmd_bytes = b'\x14\x00\xab\xcd\x50\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00'
394
395
396     buffer = np.ones([DC_MM_EEPROM_SIZE + DS5_CMD_LENGTH, ], dtype =
           np.uint8) * 255
397     cmd = np.array(struct.unpack('I'*6, MMEW_Cmd_bytes), dtype=np.
           uint32)
398     cmd.dtype = np.uint16
399     cmd[0] += DC_MM_EEPROM_SIZE
400     cmd.dtype = np.uint32
401     cmd[3] = DC_MM_EEPROM_SIZE # command 1 = 0x50
402                               # command 2 = 0
403                               # command 3 = size

```

```

404 cmd.dtype = np.uint8
405 buffer[:len(cmd)] = cmd
406 buffer[len(cmd):len(cmd)+eeprom.size] = eeprom
407
408 debug = get_debug_device(serial_no)
409 if not debug:
410     print('Error getting RealSense Device.')
411     return
412 # tab1, tab2, tab3, tab4 = parse_buffer(buffer)
413
414 rcvBuf = debug.send_and_receive_raw_data(bytearray(buffer))
415 if rcvBuf[0] == buffer[4]:
416     print('SUCCESS: saved calibration to camera.')
417 else:
418     print('FAILED: failed to save calibration to camera.')
419     print(rcvBuf)
420
421
422 def get_debug_device(serial_no):
423     ctx = rs.context()
424     devices = ctx.query_devices()
425     found_dev = False
426     for dev in devices:
427         if len(serial_no) == 0 or serial_no == dev.get_info(rs.
428             camera_info.serial_number):
429             found_dev = True
430             break
431     if not found_dev:
432         print('No RealSense device found' + str('.') if len(serial_no)
433             == 0 else ' with serial number: '+serial_no))
434         return 0
435
436 # set to advance mode:
437 advanced = rs.rs400_advanced_mode(dev)
438 if not advanced.is_enabled():
439     advanced.toggle_advanced_mode(True)
440
441 # print(a few basic information about the device)
442 print(' Device PID: ', dev.get_info(rs.camera_info.product_id)
443     )
444 print(' Device name: ', dev.get_info(rs.camera_info.name))
445 print(' Serial number: ', dev.get_info(rs.camera_info.

```

```

        serial_number))
443     print(' Firmware version: ', dev.get_info(rs.camera_info.
           firmware_version))
444     debug = rs.debug_protocol(dev)
445     return debug
446
447 def check_X(X, accel, show_graph):
448     fdata = np.apply_along_axis(np.dot, 1, accel, X[:3, :3]) - X[3, :]
449     norm_data = (accel**2).sum(axis=1)**(1./2)
450     norm_fdata = (fdata**2).sum(axis=1)**(1./2)
451     if show_graph:
452         import pylab
453         pylab.plot(norm_data, '.b')
454         #pylab.hold(True)
455         pylab.plot(norm_fdata, '.g')
456         pylab.show()
457     print('norm (raw data ): %f' % np.mean(norm_data))
458     print('norm (fixed data): %f' % np.mean(norm_fdata), "A good
           calibration will be near %f" % g)
459
460
461 def main():
462     if any([help_str in sys.argv for help_str in ['-h', '--help',
           '/?']]):
463         print("Usage:", sys.argv[0], "[Options]")
464         print
465         print('[Options]:')
466         print('-i : /path/to/accel.txt [/path/to/gyro.txt]')
467         print('-s : serial number of device to calibrate.')
468         print('-g : show graph of norm values - original values in
           blue and corrected in green.')
469         print
470         print('If -i option is given, calibration is done using
           previously saved files')
471         print('Otherwise, an interactive process is followed.')
472         sys.exit(1)
473
474     try:
475         accel_file = None
476         gyro_file = None
477         serial_no = ''
478         show_graph = '-g' in sys.argv

```

```

479     for idx in range(len(sys.argv)):
480         if sys.argv[idx] == '-i':
481             accel_file = sys.argv[idx+1]
482             if len(sys.argv) > idx+2 and not sys.argv[idx+2].
                startswith('-'):
483                 gyro_file = sys.argv[idx+2]
484         if sys.argv[idx] == '-s':
485             serial_no = sys.argv[idx+1]
486
487     buckets = [[0, -g, 0], [ g, 0, 0],
488                [0,  g, 0], [-g, 0, 0],
489                [0, 0, -g], [ 0, 0, g]]
490
491     buckets_labels = ["Upright facing out", "USB cable up facing
                out", "Upside down facing out", "USB cable pointed down
                ", "Viewing direction facing down", "Viewing direction
                facing up"]
492
493     gyro_bais = np.zeros(3, np.float32)
494     old_settings = None
495     if accel_file:
496         if gyro_file:
497             #compute gyro bais
498
499             #assume the first 4 seconds the device is still
500             gyro = np.loadtxt(gyro_file, delimiter=",")
501             gyro = gyro[gyro[:, 0] < gyro[0, 0]+4000, :]
502
503             gyro_bais = np.mean(gyro[:, 1:], axis=0)
504             print(gyro_bais)
505
506             #compute accel intrinsic parameters
507             max_norm = np.linalg.norm(np.array([0.5, 0.5, 0.5]))
508
509             measurements = [[], [], [], [], [], []]
510             import csv
511             with open(accel_file, 'r') as csvfile:
512                 reader = csv.reader(csvfile)
513                 rnum = 0
514                 for row in reader:
515                     M = np.array([float(row[1]), float(row[2]),
                                float(row[3])])

```

```

516         is_ok = False
517         for i in range(0, len(buckets)):
518             if np.linalg.norm(M - buckets[i]) < max_norm
                    :
519                 is_ok = True
520                 measurements[i].append(M)
521             rnum += 1
522         print('read %d rows.' % rnum)
523     else:
524         print('Start interactive mode:')
525         if os.name == 'posix':
526             old_settings = termios.tcgetattr(sys.stdin)
527             tty.setcbreak(sys.stdin.fileno())
528
529         imu = imu_wrapper()
530         if not imu.enable_imu_device(serial_no):
531             print('Failed to enable device.')
532             return -1
533         measurements, gyro = imu.get_measurements(buckets,
                    buckets_labels)
534         con_mm = np.concatenate(measurements)
535         if os.name == 'posix':
536             termios.tcsetattr(sys.stdin, termios.TCSADRAIN,
                    old_settings)
537
538         header = input('\nWould you like to save the raw data?
                    Enter footer for saving files (accel-<footer>.txt and
                    gyro-<footer>.txt)\nEnter nothing to not save raw
                    data to disk. >')
539         print('\n')
540         if header:
541             accel_file = 'accel-%s.txt' % header
542             gyro_file = 'gyro-%s.txt' % header
543             print('Writing files:\n%s\n%s' % (accel_file,
                    gyro_file))
544             np.savetxt(accel_file, con_mm, delimiter=',', fmt='%
                    s')
545             np.savetxt(gyro_file, gyro, delimiter=',', fmt='%s')
546         else:
547             print('Not writing to files.')
548         # remove times from measurements:
549         measurements = [mm[:,1:] for mm in measurements]

```

```

550
551         gyro_bais = np.mean(gyro[:, 1:], axis=0)
552         print(gyro_bais)
553
554     mlen = np.array([len(meas) for meas in measurements])
555     print(mlen)
556     print('using %d measurements.' % mlen.sum())
557
558     nrows = mlen.sum()
559     w = np.zeros([nrows, 4])
560     Y = np.zeros([nrows, 3])
561     row = 0
562     for i in range(0, len(buckets)):
563         for m in measurements[i]:
564             w[row, 0] = m[0]
565             w[row, 1] = m[1]
566             w[row, 2] = m[2]
567             w[row, 3] = -1
568             Y[row, 0] = buckets[i][0]
569             Y[row, 1] = buckets[i][1]
570             Y[row, 2] = buckets[i][2]
571             row += 1
572     np_version = [int(x) for x in np.version.version.split('.')]
573     rcond_val = None if (np_version[1] >= 14 or np_version[0] >
574                       1) else -1
575
576     X, residuals, rank, singular = np.linalg.lstsq(w, Y, rcond=
577           rcond_val)
578
579     print(X)
580     print("residuals:", residuals)
581     print("rank:", rank)
582     print("singular:", singular)
583     check_X(X, w[:, :3], show_graph)
584
585     calibration = {}
586     calibration["device_type"] = "D435i"
587     calibration["imus"] = list()
588     calibration["imus"].append({})
589     calibration["imus"][0]["accelerometer"] = {}
590     calibration["imus"][0]["accelerometer"]["scale_and_alignment
591           "] = X.flatten()[ :9].tolist()
592     calibration["imus"][0]["accelerometer"]["bias"] = X.flatten

```

```

    () [9:].tolist()
589 calibration["imus"][0]["gyroscope"] = {}
590 calibration["imus"][0]["gyroscope"]["scale_and_alignment"] =
    np.eye(3).flatten().tolist()
591 calibration["imus"][0]["gyroscope"]["bias"] = gyro_bais.
    tolist()
592 json_data = json.dumps(calibration, indent=4, sort_keys=True
    )
593
594 directory = os.path.dirname(accel_file) if accel_file else '.'
595
596 with open(os.path.join(directory, "calibration.json"), 'w')
    as outfile:
597     outfile.write(json_data)
598
599 #concatinate the two 12 element arrays and save
600 intrinsic_buffer = np.zeros([6,4])
601
602 intrinsic_buffer[:3, :4] = X.T
603 intrinsic_buffer[3:, :3] = np.eye(3)
604 intrinsic_buffer[3:, 3] = gyro_bais
605
606 # intrinsic_buffer = ((np.array(range(24), np.float32)+1)/10)
    .reshape([6,4])
607
608 d435_imu_calib_table = get_D435_IMU_Calib_Table(
    intrinsic_buffer)
609 calibration_table = get_calibration_table(
    d435_imu_calib_table)
610 eeprom = get_eeprom(calibration_table)
611
612 with open(os.path.join(directory, "calibration.bin"), 'wb')
    as outfile:
613     outfile.write(eeprom.astype('f').tostring())
614
615 is_write = input('Would you like to write the results to the
    camera\'s eeprom? (Y/N) ')
616 is_write = 'Y' in is_write.upper()
617 if is_write:
618     print('Writing calibration to device.')
619     write_eeprom_to_camera(eeprom, serial_no)

```



```
620         print('Done.')
```

```
621     else:
```

```
622         print('Abort writing to device')
```

```
623 except Exception as e:
```

```
624     print ('\nDone. %s' % e)
```

```
625 finally:
```

```
626     if os.name == 'posix' and old_settings is not None:
```

```
627         termios.tcsetattr(sys.stdin, termios.TCSADRAIN,
```

```
        old_settings)
```

```
628
```

```
629     """
```

```
630     wtw = dot(transpose(w),w)
```

```
631     wtwi = np.linalg.inv(wtw)
```

```
632     print(wtwi)
```

```
633     X = dot(wtwi, Y)
```

```
634     print(X)
```

```
635     """
```

```
636 if __name__ == '__main__':
```

```
637     main()
```

F.3 Traffic Sign Detection Script

F.3.1 Speed Limit 100

```
1  #! /usr/bin/python
```

```
2
```

```
3  # Import Libraries
```

```
4  import rospy
```

```
5  from sensor_msgs.msg import Image
```

```
6  import cv2
```

```
7  import numpy as np
```

```
8  import psychopg2
```

```
9  from std_msgs.msg import Float64
```

```
10 from cv_bridge import CvBridge, CvBridgeError
```

```
11 import dropbox
```

```
12 import time
```

```
13 import os
```

```
14 import numpy as np
```

```
15
```

```
16 #Connect to Database
```

```
17
```

```
18 bridge = CvBridge()
```

```
19 connection= psychopg2.connect(user="postgres", password="upsquared",
```

```

        host="localhost", port=5432,database="mas500")
20 cursor = connection.cursor()
21 i = 0
22 def globallyChange():
23     global i
24     i += 1
25
26 def lat_callback(msg):
27     global lat
28     lat=msg.data
29
30 def lng_callback(msg):
31     global lng
32     lng=msg.data
33
34 def update_table():
35     name= '100pic{:>03}.jpg'.format(i)
36     cursor.execute(''INSERT INTO geo100(geom,info) VALUES(
        ST_GeomFromText('POINT(%s %s)',4326),%s)''',(lng , lat , name
        ))
37     connection.commit()
38     count=cursor.rowcount
39 def update_folder():
40
41     cv2.imwrite('/home/upsquared/Desktop/detected_images/100
        _sign/pic{:>03}.jpg'.format(i), copy_frame)
42     globallyChange()
43
44 def camera_callback(msg):
45     global frame
46     frame = bridge.imgmsg_to_cv2(msg, "bgr8")
47     global copy_frame
48     copy_frame= frame.copy()
49     #Convert Image to HSV
50     hsv = cv2.cvtColor(copy_frame , cv2.COLOR_BGR2HSV)
51     #Set Lower and Upper Boundaries for Red Color
52     lower_red = np.array([0,150,95])
53     upper_red = np.array([10,255,255])
54     mask1 = cv2.inRange(hsv, lower_red , upper_red)
55     lower_red = np.array([170,150,95])
56     upper_red = np.array([180,255,255])
57     mask2 = cv2.inRange(hsv, lower_red , upper_red)

```

```

58     mask = mask1 + mask2
59     #Sharpen the Image and Strengthen the Red Color and Convert to
        GrayScale
60     kernel = np.ones((5,5), np.uint8)
61     erosion = cv2.erode(mask,kernel,iterations=1)
62     red_circles = cv2.bitwise_and(copy_frame, copy_frame, mask =
        mask)
63     dilate = cv2.dilate(red_circles, kernel, iterations=1)
64     kernel_1 = np.array([[ -1, -1, -1],[ -1, 9, -1],[ -1, -1, -1]])
65     sharpened = cv2.filter2D(dilate, -1, kernel_1)
66     gray = cv2.cvtColor(sharpened, cv2.COLOR_BGR2GRAY)
67     #Find Circles and Set Global Variables x,y,z where z is the
        Radius
68     global x
69     global y
70     global z
71     circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 2, 20, param1
        =100, param2=70, minRadius=15, maxRadius=70)
72     if circles is not None:
73         circles = np.round(circles[0, :]).astype("int")
74         for (x,y,z) in circles:
75             if x>0:
76                 #Mark Sign With Rectangle
77                 h=z+5
78                 cv2.rectangle(copy_frame, (x-h, y-h), (x+h, y+h)
                    ,(0,128,255), 3)
79                 template()
80
81     #cv2.imshow(" Stream_100: ", copy_frame)
82
83     if cv2.waitKey(1) & 0xFF == ord('q'):
84         cv2.destroyAllWindows()
85 def template():
86     #Snapshot the Rectangle from the Original Frame
87     h=z+5
88     detected_img = frame[y-h:y+h, x-h:x+h]
89     detected_img_gray = cv2.cvtColor(detected_img, cv2.
        COLOR_BGR2GRAY)
90     detected_img_gray = cv2.blur(detected_img_gray, (5,5), 3)
91     #Resize Template to the Real Size
92     template_100 = cv2.imread('/home/upsquared/MAS500_ws/src /
        python_skripter/src/scripts/fartsgrense_100.png', 0)

```

```

93     template_100 = cv2.resize(template_100, (2*h, 2*h))
94     template_100 = cv2.blur(template_100, (5,5),3)
95
96     global result
97     result = cv2.matchTemplate(detected_img_gray, template_100,
98                               cv2.TMCCOEFF_NORMED)
99     w, h = template_100.shape[::-1]
100    threshold = 0.7
101    loc = np.where( result >= threshold)
102    for pt in zip(*loc[::-1]):
103        #res = "{}%".format(result, float(result))
104        font = cv2.FONT_HERSHEY_SIMPLEX
105        #cv2.putText(copy_frame, res, (x-h, y+h), font,
106                    1,(255,255,255),2,cv2.LINE_AA);
107        cv2.putText(copy_frame, '100_sign', (x-h-50, y-h),
108                    font, 1,(255,255,255),2,cv2.LINE_AA);
109        update_folder()
110        update_table()
111
112 def main():
113     # Create Node
114     rospy.init_node('listener_100', anonymous=True)
115     # Define Subscriber and Define its Callback
116     rospy.Subscriber("/camera/color/image_raw", Image,
117                     camera_callback)
118     rospy.Subscriber("lat", Float64, lat_callback)
119     rospy.Subscriber("lng", Float64, lng_callback)
120     # Spin until ctrl + c
121     rospy.spin()
122
123 if __name__ == '__main__':
124     main()

```

F.3.2 Speed Limit 90

```

1 #! /usr/bin/python
2
3 #Import Libraries
4 import rospy
5 from sensor_msgs.msg import Image
6 import cv2

```

```
7 import numpy as np
8 import psycopg2
9 from std_msgs.msg import Float64
10 from cv_bridge import CvBridge, CvBridgeError
11 import time
12 import numpy as np
13
14 #Connect to Database
15
16 bridge = CvBridge()
17 connection= psycopg2.connect(user="postgres", password="upsquared",
18     host="localhost", port=5432,database="mas500")
19 cursor = connection.cursor()
20 i = 0
21
22 def globallyChange():
23     global i
24     i += 1
25
26 def lat_callback(msg):
27     global lat
28     lat=msg.data
29
30 def lng_callback(msg):
31     global lng
32     lng=msg.data
33
34 def update_table():
35     name= '90pic{:>03}.jpg'.format(i)
36     cursor.execute(''INSERT INTO geo(geom,info) VALUES(
37         ST_GeomFromText('POINT(%s %s)',4326),%s)''',(lng,lat,name
38     ))
39     connection.commit()
40     count=cursor.rowcount
41
42 def update_folder():
43
44     cv2.imwrite('/home/upsquared/Desktop/detected_images/90_sign
45         /pic{:>03}.jpg'.format(i), copy_frame)
46     globallyChange()
47
48 def camera_calback(msg):
```

```

45     global frame
46     global sharpened
47     frame = bridge.imgmsg_to_cv2(msg, "bgr8")
48     global copy_frame
49     copy_frame= frame.copy()
50
51     #Convert Image to HSV
52     hsv = cv2.cvtColor(copy_frame, cv2.COLOR_BGR2HSV)
53     #Set Lower and Upper Boundaries for Red Color
54     lower_red = np.array([0,150,95])
55     upper_red = np.array([10,255,255])
56     mask1 = cv2.inRange(hsv, lower_red, upper_red)
57     lower_red = np.array([170,150,95])
58     upper_red = np.array([180,255,255])
59     mask2 = cv2.inRange(hsv, lower_red, upper_red)
60     mask = mask1 + mask2
61     #Sharpen the Image and Strengthen the Red Color and Convert to
        GrayScale
62     kernel = np.ones((5,5), np.uint8)
63     erosion = cv2.erode(mask,kernel,iterations=1)
64     red_circles = cv2.bitwise_and(copy_frame, copy_frame, mask =
        mask)
65     dilate = cv2.dilate(red_circles, kernel, iterations=1)
66     kernel_1 = np.array([[ -1, -1, -1],[ -1, 9, -1],[ -1, -1, -1]])
67     sharpened = cv2.filter2D(dilate, -1, kernel_1)
68     gray = cv2.cvtColor(sharpened, cv2.COLOR_BGR2GRAY)
69     #Find Circles and Set Global Variables x,y,z where z is the
        Radius
70     global x
71     global y
72     global z
73     circles = cv2.HoughCircles(gray, cv2.HOUGHGRADIENT, 2, 20, param1
        =100, param2=70, minRadius=10, maxRadius=70)
74     if circles is not None:
75         circles = np.round(circles[0, :]).astype("int")
76         for (x,y,z) in circles:
77             if x>0:
78                 #Mark Sign With Rectangle
79                 h=z+5
80                 cv2.rectangle(copy_frame, (x-h, y-h), (x+h, y+h)
                    ,(0,128,255), 3)
81                 template()

```

```

82     #Show the Frame
83     cv2.imshow("sone_90", copy_frame)
84     if cv2.waitKey(1) & 0xFF == ord('q'):
85         cv2.destroyAllWindows()
86
87 def template():
88     #Snapshot the Rectangle from the Original Frame
89     h=z+5
90     detected_img = frame[y-h:y+h, x-h:x+h]
91     detected_img_gray = cv2.cvtColor(detected_img, cv2.
92         COLOR_BGR2GRAY)
93     detected_img_gray = cv2.blur(detected_img_gray, (5,5),3)
94     #Resize Template to the Real Size
95     template_90 = cv2.imread('/home/upsquared/MAS500_ws/src/
96         python_skriptier/src/scripts/fartsgrense_90.png',0)
97     template_90 = cv2.resize(template_90, (2*h, 2*h))
98     template_90 = cv2.blur(template_90, (5,5),3)
99
100     global result
101     result = cv2.matchTemplate(detected_img_gray, template_90, cv2.
102         TMCCOEFF_NORMED)
103     w, h = template_90.shape[::-1]
104     threshold = 0.7
105     loc = np.where( result >= threshold)
106     for pt in zip(*loc[::-1]):
107         #res = "{}%".format(result, float(result))
108         font = cv2.FONT_HERSHEY_SIMPLEX
109         #cv2.putText(copy_frame, res, (x-h, y+h), font,
110             1,(255,255,255),2,cv2.LINE_AA);
111         cv2.putText(copy_frame, '90_sign', (x-h-50, y-h), font,
112             1,(255,255,255),2,cv2.LINE_AA);
113         update_folder()
114         update_table()
115
116
117 def main():
118     # Create Node
119     rospy.init_node('listener_90', anonymous=True)
120     # Define Image Topic
121     camera_topic = "/camera/color/image_raw"
122     # Set Subscriber and Define its Callback

```

```
119     rospy.Subscriber(camera_topic , Image, camera_callback)
120     rospy.Subscriber("lat", Float64, lat_callback)
121     rospy.Subscriber("lng", Float64, lng_callback)
122     # Spin until ctrl + c
123     rospy.spin()
124
125
126 if __name__ == '__main__':
127     main()
```

F.4 Lane Mark Detection Script

```
1  #! /usr/bin/python
2
3  # Import Libraries
4  import rospy
5  from sensor_msgs.msg import Image
6  import cv2
7  import numpy as np
8  import psycopg2
9  from std_msgs.msg import Float64
10 from cv_bridge import CvBridge, CvBridgeError
11 import time
12 import numpy as np
13 import math
14 from matplotlib import pyplot as plt
15
16 #Connect to Database
17 bridge = CvBridge()
18 connection= psycopg2.connect(user="postgres", password="upsquared",
19                               host="localhost", port=5432,database="mas500")
20 cursor = connection.cursor()
21 i = 0
22
23 def globallyChange():
24     global i
25     i += 1
26
27 def lat_callback(msg):
28     global lat
29     lat=msg.data
30
31 def lng_callback(msg):
```

```

31     global lng
32     lng=msg.data
33
34 def update_table():
35     name= 'pic{:>03}.jpg'.format(i)
36     cursor.execute(''INSERT INTO line(geom,info) VALUES(
37         ST_GeomFromText('POINT(%s %s)',4326),%s)''',(lng,lat,name
38         ))
39     connection.commit()
40     count=cursor.rowcount
41
42 def update_folder():
43     cv2.imwrite('/home/upsquared/Desktop/detected_images/
44         bad_lane/pic{:>03}.jpg'.format(i), frame)
45     globallyChange()
46
47 def Transform_Camera_View(img, src, dst):
48
49     image_shape = img.shape
50     img_size = (image_shape[1], image_shape[0])
51     # Given src and dst points, calculate the perspective transform
52     matrix
53     M = cv2.getPerspectiveTransform(src, dst)
54     Minv = cv2.getPerspectiveTransform(dst, src)
55     # Warp the image using warpPerspective()
56     warped = cv2.warpPerspective(img, M, img_size)
57     return warped, M, Minv
58
59 def HLS_L_Threshold(img, thresh=(135, 255)):
60     img = img[:, :, 1]
61     img = img * (255 / np.max(img))
62     binary_output = np.zeros_like(img)
63     binary_output[(img > thresh[0]) & (img <= thresh[1])] = 1
64     return binary_output
65
66 def camera_callback(msg):
67
68     global frame
69     frame = bridge.imgmsg_to_cv2(msg, "bgr8")
70     global copy_frame

```

```

69         copy_frame= frame.copy()
70         threshold_middle_lane = 120
71         threshold_right_lane = 300
72         threshold_plot = threshold_right_lane
73
74         src = np.float32([[50, 450], [520, 450], [250, 350],
75                          [360, 350]])
76         bottom_left = src[0][0] + 0, src[0][1]
77         bottom_right = src[1][0] - 0, src[1][1]
78         top_left = src[3][0] - 0, 1
79         top_right = src[2][0] + 0, 1
80         dst = np.float32([bottom_left, bottom_right,
81                          top_right, top_left])
82
83         img_warped = Transform_Camera_View(copy_frame, src,
84                                           dst)[0]
85
86         img_HLS = cv2.cvtColor(img_warped, cv2.COLOR_RGB2HLS
87                                )
88         img_HLS_L = img_HLS[:, :, 1]
89
90         thresh_HLS = HLS_L_Threshold(img_HLS)
91
92         pts1 = np.float32([[256, 39], [359, 39], [90, 480],
93                          [500, 480]]) # Old points
94         pts2 = np.float32([[0, 0], [640, 0], [0, 480], [640,
95                          480]]) # New points
96         matrix = cv2.getPerspectiveTransform(pts1, pts2) #
97         Transformation matrix
98
99         result = cv2.warpPerspective(thresh_HLS, matrix,
100                                     (640, 480)) # The transformed image
101         cv2.imshow("Frame", frame)
102         cv2.imshow("Warped", img_warped)
103         cv2.imshow("Warped_Image_Binary", thresh_HLS*255)
104         left, right = np.hsplit(thresh_HLS, 2)
105         counts_left = np.sum(left == 1, axis=0)
106         counts_right = np.sum(right == 1, axis=0)
107
108         total_left = math.fsum(counts_left)
109         total_right = math.fsum(counts_right)

```

```

103
104         threshold_left = 6000
105         threshold_left_lower = 4000
106         threshold_right = 35000
107         threshold_right_lower = 3000
108
109         if total_left < threshold_left:
110             if total_left < threshold_left_lower:
111                 print('Bad Lane left side')
112                 #update_folder()
113                 #update_table()
114             if total_right < threshold_right:
115                 if total_right < threshold_right_lower:
116                     print('Bad Lane right side')
117                     #update_folder()
118                     #update_table()
119
120         if cv2.waitKey(1) & 0xFF == ord('q'):
121             cv2.destroyAllWindows()
122
123
124 def main():
125     # Create Node
126     rospy.init_node('lane_mark_listener', anonymous=True)
127     # Define Subscriber and Define its Callback
128     rospy.Subscriber("/camera/color/image_raw", Image,
129                     camera_callback)
129     rospy.Subscriber("lat", Float64, lat_callback)
130     rospy.Subscriber("lng", Float64, lng_callback)
131     # Spin until ctrl + c
132     rospy.spin()
133
134
135 if __name__ == '__main__':
136     main()

```

F.5 Vibration Measurement Script

```

1 #!/usr/bin/env python
2 import rospy
3 import psycopg2
4 import cv2
5 from std_msgs.msg import Float64

```

```

6 from sensor_msgs.msg import Imu
7
8
9 def globallyChange():
10     global i
11     i += 1
12
13 def lat_callback(msg):
14     global lat
15     lat=msg.data
16
17 def lng_callback(msg):
18     global lng
19     lng=msg.data
20
21 def update_table():
22     name= 'Bad_Road{:>03}'.format(i)
23     cursor.execute(''INSERT INTO roadcondition(geom,info)
24                     VALUES(ST_GeomFromText('POINT(%s %s)',4326),%s)''',(lng ,
25                               lat ,name))
26     connection.commit()
27     count=cursor.rowcount
28
29 def update_folder():
30
31     cv2.imwrite('/home/upsquared/Desktop/detected_images/
32                bad_road/pic{:>03}.jpg'.format(i) , frame)
33     globallyChange()
34
35 def accel_callback(msg):
36     global acc
37     acc=abs(msg.linear_acceleration.z-9.32)
38     print(acc)
39     if acc > 4.5:
40         #update_folder()
41         update_table()
42
43 while True:
44
45     try:
46         #Connect to Database and Create Node and Subscribers
47         connection= psycopg2.connect(user="postgres",
48                                     password="upsquared", host="localhost", port

```

```

        =5432,database="mas500")
44     cursor = connection.cursor()
45     i = 0
46     rospy.init_node('accel_list', anonymous=True)
47     rospy.Subscriber("/camera/accel/sample", Imu,
        accel_callback)
48     rospy.Subscriber("lat", Float64, lat_callback)
49     rospy.Subscriber("lng", Float64, lng_callback)
50     rospy.spin()
51
52     except rospy.ROSInterruptException:
53         pass

```

F.6 Upload to Dropbox Script

```

1  #! /usr/bin/python
2  #Import Libraries
3  import psycopg2
4  import rospy
5  from std_msgs.msg import Float64
6  import dropbox
7  import time
8  import os
9  import numpy as np
10
11 #Access Dropbox Account
12
13 access_token = '
    MpY68MMU6cAAAAAAAAAADyIpsGd1Dh6mJHqv4D5FnJm3ykA_rZW4iRatzyG_bW19'
14 dbx = dropbox.Dropbox(access_token)
15
16 #Direction to Detected Images
17 rootdir = '/home/upsquared/Desktop/detected_images/90_sign'
18
19 def upload_img():
20 #Code for Uploading to Dropbox
21     for dir, dirs, files in os.walk(rootdir):
22         for file in files:
23             try:
24                 file_path= os.path.join(dir,
                file)
25                 #Name of the Folder in
                Dropbox Apps Folder

```

```

26         dest_path=os.path.join('/90
           _sign/', file)
27     print('Uploading %s to %s' %
           (file_path, dest_path))
28     with open(file_path,"rb") as
           f:
29         dbx.files_upload(f.
           read(), dest_path
           , mute=True)
30     except Exception as err:
31
32     print("failed to upload %s\n
           %s" % (file ,err))
33
34 def main():
35     upload_img()
36     # Spin until ctrl + c
37     rospy.spin()
38
39
40 if __name__ == '__main__':
41     main()

```

F.7 Communication Arduino and ROS

```

1  #! /usr/bin/python
2
3  #Import Libraries
4  import serial
5  import psychopg2
6  import rospy
7  from std_msgs.msg import Float64
8
9  #Connect to Serial Port
10 try:
11     arduino = serial.Serial('/dev/ttyACM0', 115200)
12     ok=1
13 except:
14     print("Check port")
15
16 def talker():
17     #Read Serial and Create a Publisher
18     latitude=rospy.Publisher('lat', Float64, queue_size=10)

```

```
19 longitude=rospy.Publisher('lng', Float64, queue_size=10)
20 lux = rospy.Publisher('lx', Float64, queue_size=10)
21 rospy.init_node('talker', anonymous=True)
22 rate = rospy.Rate(10)
23 while not rospy.is_shutdown():
24     gps_point = str(arduino.readline())
25     if ok==1:
26         try:
27
28             lat = float(gps_point[0:9])
29             lng = float(gps_point[10:18])
30             lx = float(gps_point[19:27])
31         except:
32             lat=0.0;
33             lng=0.0;
34             lx=0.0;
35         #print(lat, lng, lx)
36         rospy.loginfo(lat)
37         rospy.loginfo(lng)
38         rospy.loginfo(lx)
39         longitude.publish(lng)
40         latitude.publish(lat)
41         lux.publish(lx)
42         rate.sleep()
43 while True:
44
45     try:
46         talker()
47
48     except rospy.ROSInterruptException:
49         pass
```

G. ROS - Launch File

```
1 <launch>
2
3     <include file= "$(find realsense2_camera)/launch/rs_camera.
4         launch"/>
5
6     <node name= "lane_mark_listener" pkg="python_skripter" type="
7         line_detection.py" />
8
9     <node name= "listener_100" pkg="python_skripter" type="
10        camera_100.py" />
11
12    <node name= "listener_90" pkg="python_skripter" type="
13        camera_90.py" />
14
15    <node name= "talker" pkg="python_skripter" type="
16        ros_coord_node.py" />
17
18    <node name= "accel_list" pkg="python_skripter" type="accel.py
19        " />
20
21 </launch>
```


H. MatLab Scripts

H.1 Script for Reading ROS bag files

For testing the algorithms before car implementation, the group recorded certain routes along E18 with Intel RealSense D435i. The files from the Intel camera was stored as rosbag files. To access the stored information in the rosbag file, images was extracted and into .png or and .jpg files and then merged to .avi files.

```
1 close all
2 clear
3 clc
4
5 bagMsgs = robotics.ros.Bag.parse('\Users\benja\Desktop\MAS500\
   Bag_files\2019225_141440.bag');
6 %Extracts information of the
7 %bag file , by clicking on the bagMsgs you can locate the available
   topics
8 %From there you can choose what kind of topic you want to look into
9
10 bagMsgs2 = select(bagMsgs, 'Time', ...
11 [bagMsgs.StartTime bagMsgs.StartTime + 10], 'Topic', '/device_0/
   sensor_1/Color_0/image/data');
12
13 %Here we choose the topic of the bagfile we want to extract , here we
   just
14 %basically one
15 %second of the file. sat the interval to be from start bagfile to
   one second ,
16
17 msgs = readMessages(bagMsgs2);
18
19 %The readMessages reads the messages stored in the topic that we
   chose.
20 %from here we can loop all the messages and convert them to images
21
22 for ii = 1:length(msgs)
23
24     img = msgs{ii};
25
```

```

26     %image = uint8(readImage(img)/256);
27     image = (readImage(img));
28     baseFileName = sprintf('%d.png', ii);
29     fullFileName = fullfile('\', 'Users', 'benja', 'Desktop', 'MAS500', '
        bag_to_video', 'img_RGB', baseFileName);
30     imwrite(image, fullFileName);
31
32     %imwrite(image, sprintf('%d.png', ii))
33 end
34
35 % Make an avi movie from a collection of PNG images in a folder.
36 % Specify the folder.
37 myFolder = '\Users\benja\Desktop\MAS500\bag_to_video\img_RGB';
38 if ~isdir(myFolder)
39     errorMessage = sprintf('Error: The following folder does not
        exist:\n%s', myFolder);
40     uiwait(warndlg(errorMessage));
41     return;
42 end
43 % Get a directory listing.
44 filePattern = fullfile(myFolder, '*.PNG');
45 pngFiles = dir(filePattern);
46 % Open the video writer object.
47 writerObj = VideoWriter('\Users\benja\Desktop\MAS500\bag_to_video\
        Ferdig_Videoer\test123.avi');
48 open(writerObj);
49 % Go through image by image writing it out to the AVI file.
50 for ii = 1 : length(pngFiles)
51     % Construct the full filename.
52     baseFileName = sprintf('%d.png', ii);
53     fullFileName = fullfile('\', 'Users', 'benja', 'Desktop', 'MAS500', '
        bag_to_video', 'img_RGB', baseFileName);
54     % Display image name in the command window.
55     %fprintf(1, 'Now reading %s\n', fullFileName);
56     % Display image in an axes control.
57     thisimage = imread(fullFileName);
58     %imshow(thisimage); % Display image.
59     drawnow; % Force display to update immediately.
60     % Write this frame out to the AVI file.
61     writeVideo(writerObj, thisimage);
62 end
63 % Close down the video writer object to finish the file.

```

```
64 close(writerObj);
```

I. Arduino Scripts

I.1 GPS and Light Intensity Script

```
1
2 #include <TinyGPS++.h>
3 #include <SoftwareSerial.h>
4
5 float v_in=5.0;
6 float v_out;
7 float ldr_voltage;
8 float r=10000.0;
9 float ldr_input = A0;
10 float ldr_value;
11 float rldr;
12 float lux_scalar = 210074123.7;
13 float exponent= -1.59366;
14 float ldrlux;
15 float i;
16 float res_voltage;
17
18 static const int RXPin = 4, TXPin = 3;
19 static const uint32_t GPSBaud = 9600;
20
21
22 TinyGPSPlus gps;
23 SoftwareSerial ss(RXPin, TXPin);
24
25 void setup()
26 {
27   Serial.begin(115200);
28   ss.begin(GPSBaud);
29 }
30
31 void loop()
32 {
33   ldr_value = analogRead(ldr_input);
34   res_voltage= ldr_value* v_in/1023;
35   ldr_voltage = v_in - res_voltage;
36   rldr = ldr_voltage/res_voltage * r;
```

```
37  ldrlux = (lux_scalar)*pow(rldr , exponent);
38  while (ss.available() > 0){
39      gps.encode(ss.read());
40      if (gps.location.isUpdated()){
41          Serial.print(gps.location.lat(), 6);
42          Serial.print(" ");
43          Serial.print(gps.location.lng(), 6);
44          Serial.print(" ");
45          Serial.println(ldrlux, 2);
46      }
47
48  }
49
50
51 }
```