UNIVERSITY OF AGDER

# Modelling and Simulation of a Novel Liquamatic Fire Monitor

A fully automated fire recognition- and suppression system based on infrared machine vision technology

**Jared Hansen**
**Julian Løvlie Haugen**

**Supervisor**
Morten Kjeld Ebbesen

*This masters 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.*

University of Agder, 2015
Faculty of Technology and Science
Department of Engineering Sciences

# Summary

**Background:** Fire monitors are effective fire extinguishing apparatuses which combine high accuracy with long range. As part of the mechatronic trend, research has during recent years started to delve into the automation of fire monitors. This involves proper actuation and control of fire monitors in order to extinguish fire. Up to this point, however, research has primarily been concerned with indoor operation. The aim of the present thesis is to develop a system for an electrically actuated fire monitor which detects, localizes and suppresses fire in an outdoor environment without the need for manual operation.

**Solution & Experiments:** Fire is localized with computer analysis of IR stereo camera images. Based on the position of the fire, mathematical models found in literature are used to determine the optimum configuration of the monitor in order to extinguish the fire. Servomotors which actuate the monitor are modelled and simulated in real time using a HIL setup. A PLC is programmed to generate control signals to the servomotors.

The accuracy of the stereo vision system is tested experimentally by estimating the distance to a live fire at distances between 30 and 60 m. In addition, the system's ability to distinguish a fire from other hot objects is tested.

Liquid jet trajectory models are obtained from relevant research papers found in open literature. Parameters from these models are determined based on experiments conducted outdoors with a fire monitor where wind disturbances are measured.

**Results:** The stereo vision system exhibited a maximum error of 0.5 m or 1.6 %. The vision system is successful in distinguishing between a wooden fire, a person and a pot with boiling water.

The best model to predict jet trajectories found in literature yields an average error of 1.6 m from measured data with little wind present, and 9.8 m mean deviation with comparatively strong wind disturbances.

Simulations are carried out with only minor discrepancies with one of the models implemented on the PLC.

**Conclusion:** Computer algorithms which localize fire in conjunction with IR cameras has been designed.

The limiting factor with regards to the system's accuracy is precise predictions of the water jet's travel. The accuracy of the trajectory models as compared to experimental data measured under presented circumstances are of limited use. In addition, there are marginal differences between the presented trajectory models found in literature, and therefore either one may be used.

A PLC program has been created. HIL simulations are carried out with only minor discrepancies as compared to the predicted trajectories from one of the models.

# Preface

This thesis is written as a part of the Master's Programme in Mechatronics at the University of Agder. It has been an interesting, instructive and yet challenging process, where a solution to a problem formulated by the staff at FireProducts, located in Kristiansand Norway, has been proposed.

The reader should be made aware that the work done with this thesis has been done in parallel with another, related thesis. For practical considerations, some assumptions in this thesis are made with regards to design specifications, which are ultimately dependant on the results of the other thesis. Reference to the mentioned thesis will be given where appropriate.

The authors would like to thank Kristiansand Fire Department for their assistance in experiments in the context of liquid jets. Without their assistance, these experiments would have been difficult to carry out.

Finally, the authors would like to express their most profound gratitude to family and friends who have provided their support, love and tolerance throughout the past five years. Ariel Pedersen, Alexander Zimmermann Alsaker, Alexander Solan Gundersen: We made it! Tjobing!

Grimstad, May 20, 2015

Jared Hansen

Julian Løvlie Haugen

# Nomenclature

## Acronyms

| | |
|---|---|
| ADC | Analog to digital converter |
| AI | Analog input |
| CCD | Charge-coupled device |
| CFD | Computational fluid dynamics |
| DAQ | Data acquisition system |
| DIO | Digital input/output |
| DOF | Degree of freedom |
| FOV | Field of view |
| HIL | Hardware in the loop |
| IR | Infrared |
| IR3 | An infrared sensor which compares the ratios of spectral radiance at three different spectral bands |
| MEMS | Microelectromechanical sensors |
| MPS | Moving Particle Semi-implicit |
| PLC | Programmable logic controller |
| SIL | Safety Integrity Level |
| UV | Ultraviolet |

## Greek Letters

| | |
|---|---|
| $\alpha$ | Angle determined based on experimental measurements, $[rad]$ |
| $\alpha$ | Scaling factor determined during camera calibration |
| $\beta$ | Scaling factor determined during camera calibration |
| $\beta$ | Wind direction, $[rad]$ |
| $\delta$ | Angle computed during accuracy test, $[rad]$ |
| $\eta$ | Absorbance in IR sensitive films, $[1/m]$ |
| $\eta$ | Dynamic viscosity, $[Pa \cdot s]$ |
| $\gamma$ | Skew factor between the $u$ and $v$ axes, calculated during camera calibration |

| | |
|---|---|
| $\lambda$ | Scaling factor determined during camera calibration |
| $\lambda$ | Wavelength, $[m]$ |
| $\mu$ | Mean |
| $\nu$ | Frequency, $[Hz]$ |
| $\omega$ | Angular frequency, $[1/s]$ |
| $\rho$ | Volumetric mass density, $\left[kg/m^3\right]$ |
| $\tau$ | Time constant, $[s]$ |
| $\theta$ | Inclination angle, $[rad]$ |
| $\varphi$ | Azimuth angle, $[rad]$ |

**Roman Letters**

| | |
|---|---|
| $\mathbf{A}$ | Intrinsic matrix |
| $\mathbf{R}$ | Rotational matrix |
| $\mathbf{T}$ | Translational vector |
| $A$ | Area, $\left[m^2\right]$ |
| $A$ | Image |
| $a$ | Length measured during experiments, $[m]$ |
| $a_0$, $a_1$, $b_0$, $X$, $Y$, $k$, $k_1$, $k_2$, $n$ | Empirical constants used to determine jet trajectory or drag forces |
| $AE$ | Average error during parameter identification process, $[m]$ |
| $B$ | Baseline, i.e. distance between cameras, $[m]$ |
| $b$ | Length measured during experiments, $[m]$ |
| $b_W$ | Wien's displacement constant, $\left[2.89777 \cdot 10^{-3} m \cdot K\right]$ |
| $c$ | Speed of light in vacuum, 299 792 458 $[m/s]$ |
| $C_D$, $C_{Da}$ | Drag coefficient |
| $D$ | Diameter, $[m]$ |
| $d$ | Disparity, $[m]$ |
| $F$ | Force, $[N]$ |
| $f$ | Focal length, $[m]$ |
| $F_D$ | Drag force, $[N]$ |
| $Fr$ | Froude number |
| $G$ | Minimizing function used during parameter identification, $[m]$ |
| $G$ | Thermal conductivity, $[W/m \cdot K]$ |
| $g$ | Gravitational acceleration, $\left[m/s^2\right]$ |
| $G_{PI}$ | Transfer function of PI controller |

| | |
|---|---|
| $G_P$ | Proportional gain of PI controller |
| $H$ | Structuring element (image erosion) |
| $h$ | Planck's constant, $6.62607 \cdot 10^{-34}$ $[J \cdot s]$ |
| $I$ | Current, $[A]$ |
| $I$ | Mass moment of inertia, $\left[kg \cdot m^2\right]$ |
| $I$ | Pixel intensity |
| $K_1, K_2$ | Lengths used during accuracy tests, [m] |
| $k_B$ | The Boltzmann constant, $1.38065 \cdot 10^{-23}$ $\left[kg \cdot m^2 \cdot K^{-1} \cdot s^{-2}\right]$ |
| $L$ | Constant length used for reference during experiments, $[m]$ |
| $l$ | Throw length, $[m]$ |
| $m$ | Mass, $[kg]$ |
| $P$ | Fire position in $(x,y)$ coordinates |
| $P$ | Fire position, given as $(x, y)$ coordinate |
| $P$ | Power, $[W]$ |
| $P$ | Used as a point of reference in the pinhole model |
| $p$ | Pressure, $[Pa]$ |
| $Q$ | Used as a point of reference in the pinhole model |
| $Q$ | Volume flow, $\left[m^3/s\right]$ |
| $R$ | Outer radius, $[m]$ |
| $R$ | Resistance, $[\Omega]$ |
| $r$ | Inner radius, $[m]$ |
| $r$ | Radial distance, $[m]$ |
| $Re$ | Reynold's number |
| $S$ | Distance along jet, $[m]$ |
| $SR$ | Spectral radiance, $\left[W/sr \cdot m^2 \cdot m\right]$ |
| $T$ | Temperature, $[T]$ |
| $T$ | Threshold limit |
| $t$ | Time, $[s]$ |
| $T_I$ | Integration time of PI controller |
| $U$ | Wind speed, $[m/s]$ |
| $u$ | Horizontal pixel coordinate |
| $u_0$ | Horizontal principal point, i.e. center of the camera's coordinate system in pixels |

| | |
|---|---|
| $V$ | Voltage, $[V]$ |
| $v$ | Vertical pixel coordinate |
| $v_0$ | Vertical principal point, i.e. center of the camera's coordinate system in pixels |
| $x$, $y$, $z$ | Position in accordance with right hand coordinate system, $[m]$ |
| RMSD | Root-mean-square deviation |
| TCR | Temperature coefficient of resistance, $[1/K]$ |

**Subscripts**

| | |
|---|---|
| $L$ | Left |
| $R$ | Right |
| $x$, $y$, $z$ | Direction indicators |
| 0 | Initial |

# Contents

# List of Figures

# List of Tables

# CHAPTER 1

# Introduction

## 1.1 Background and Motivation

DURING the course of the past years, fire- detection and suppression and fire safety science in general have received an increased amount of attention due to a dramatic increase in fire accidents[1–3]. Progress in technology has allowed for exceedingly more sophisticated fire detection- and suppression methods, including detection of large scale fires from satellites, infrared- (IR), ultraviolet (UV) and color video detection methods as well as odor sensors, or a combination of the above, in conjunction with sprinklers, manual fire suppression or fire monitors (Figure 1.1). A key advantage of these systems is the capability of monitoring exceedingly larger areas, while promptly detecting fires and providing an effective response[2, 4–9].


Figure 1.1: A manually operated fire monitor.

Among the technologies developed to suppress flames, the fire monitor is considered an effective piece of equipment, mostly due to its relatively large flow capacity, precision and long range. Its characteristics is therefore well suited for modern firefighting in large open spaces. In turn, the mechatronic trend in the industry have called for more automation in these applications. A relatively recent development, is that machine vision based fire detection technology has been integrated with actuated fire monitors to provide fully automated fire protection and suppression systems[3, 4, 10, 11]. A major limitation of the work that has been done up to this point, however, is that it is primarily concerned with systems operating in controlled indoor environments with relatively small distances compared to the full throw lengths of high capacity monitors. A natural next step is to further develop existing systems for more general purpose use. The goal of the present thesis is to further develop currently existing systems by integrating machine vision based fire detection on an electrically actuated, high capacity fire monitor for outdoor use.

## 1.2 Problem statement

This thesis aims to model and simulate a fully automated fire detection and suppression system for outdoor operation, consisting of a fire monitor actuated by servomotors. The monitor used in this thesis is a 3 degree of freedom (DOF). The design monitor is presented in Figure 1.2, while drawings with some basic measurements are provided in Appendix A. The monitor is fitted with 3 servomotors which control its configuration in space, hereby denoted motor 1, 2 and 3 as illustrated by Figure 1.2. Motor 1 and 2 control the azimuth- and inclination angle respectively, while motor 3 is a linear actuator which adjusts the nozzle opening and thereby the proliferation of the jet. The details with respect to the actuators are beyond the scope of the present thesis, but are described in greater detail in [12].

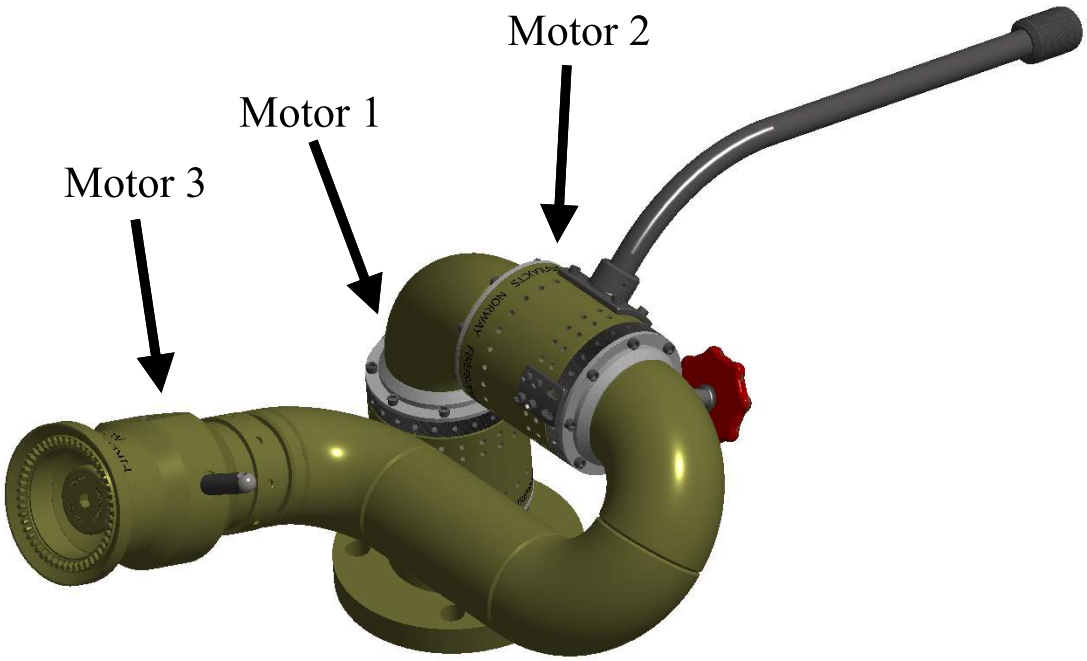



Figure 1.2: The fire monitor.

The problem in its essence is to firstly detect the spatial position of a fire, secondly provide control signals to a pressure source which yields the necessary water flow to the monitor, and lastly determine the necessary configuration of the motors in order to accurately hit a fire with the water discharge. Obviously, the accuracy of the water jet is crucial for successful fire suppression, which may be strongly influenced by errors in angular position, determination of the whereabouts of the fire and wind disturbances. However, assuming that the jet remains relatively coherent throughout its travel, the proliferation of the water spray as it lands provides some leeway in the systems accuracy. As long as the water delivered to the target is sufficient to extinguish the fire, the accuracy may be said to be sufficient. This translates to an accuracy of a few meters in the

longitudinal direction of the jet - perhaps in the order of 2-3 meters - and conceivably in the order of 1-2 m in the transverse direction, assuming an elliptical landing area and that there is some appreciable throw distance (Figure 1.3). These estimates are of course based on basic intuition and are dependant on a number of factors such as wind, range, etc., but provides some insight into how accurate the system can be while still exhibiting satisfactory performance. Consequently, in this thesis, a goal is set to hit a given position, with the centroid of the jet stream, within 2 meters.

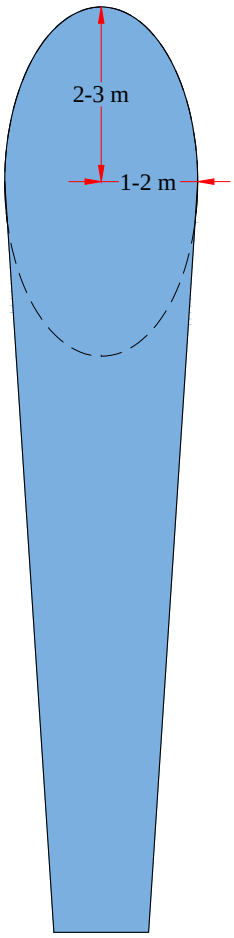



Figure 1.3: Illustration of jet proliferation.

To summarize, the goals of this thesis is the following:

1. Automatically detect and localize fire
2. Suppress the fire by spraying with the fire monitor and hitting the fire with an accuracy of $\leq$2 m

## 1.3 Key Assumptions and Limitations

The following assumptions are made with respect to the water discharge:

1. The water flows out from a uniform long pipe, i.e. the flow is fully developed
2. Pure water is used
3. Motor 3 (Figure 1.1) is assumed to be fixed at *full jet*, i.e. the narrowest jet, as opposed to *full fog* which is the widest position
4. The throw, and thus also the working range of the system, is between 25 and 50 meters

Assumption 1 through 3 are made for convenience. In practical applications, additives are often used in water which alters the fluid's properties and thereby the behaviour of the jet[13, 14]. The extent of these alterations will naturally be dependant on the type of additive and quantity, and therefore pure water is most often used in research. Furthermore, as will be discussed in Chapter 2, the state of the water stream also impacts the jet behaviour. In order to be able to find relevant subject material in literature, these simplifications are arguably necessary.

The throw lengths of high-capacity fire monitors may indeed be much greater than 50 m, but in this thesis the throw length is ultimately limited by the accuracy of the detection methods used, and longer ranges are therefore assumed to pose significantly lower accuracies.

The wind speed and direction are measured in the horizontal plane and is assumed to be representable by a uniform vector field. Wind is presumed to move approximately parallel to the ground, and thus vertical wind speed components are negligible.

Next, operating pressures $p$ are restricted to no more than 10 bar[15]. Larger pressures introduce instabilities and considerable stresses on the monitor's body, while contributions to range may be small, if any at all[3, 16]. Lastly, the monitor is assumed to be placed at ground level with surrounding flat terrain.

It must also be emphasized that no experiments in this thesis is conducted with the fire monitor presented in Figure 1.2. The difference between the fire monitor which is used in relevant experiments, such as the nozzle, is assumed to be negligible. However, in reality, there may be significant differences in for instance, throw length, flow characteristics etc. Therefore, the obtained results cannot be applied directly to the fire monitor in Figure 1.2, although the principles are the same.

With regards to fire detection, the following assumptions are made:

- The fire is the hottest object visible
- There is only one, coherent fire
- The fire is a wooden fire

The above limitations form the basis for the fire detection system.

## 1.4 Problem Solution

In order to achieve the goal set in the previous section, sufficiently accurate modelling of the water discharge is crucial for the system's accuracy. Therefore, obtaining mathematical models which can be used to predict a jet's travel in the air is imperative. This is solved by reviewing relevant papers found in the open literature. Subsequently, the models will be tested against experimental measurements.

Given that the system is to operate outdoors, it should be able to maintain as much accuracy as possible while compensating for wind disturbances. An important aspect of this work is therefore to explore the effects wind forces have on the trajectory of fluid jets. Headwinds and crosswinds especially are thought to severely impact the jet trajectory. Because of this, an important milestone is to investigate the possibility of predicting the trajectory of a liquid jet from a fire monitor under the influence of wind, and thereby being able to hit a target regardless of these disturbances. Measurements of wind will therefore be carried out in connection with experiments on jet trajectories.

Sufficiently accurate determination of the fire's whereabouts is also crucial. Therefore, much effort will be invested into fire detection and localization. In this thesis, a localization method which involves two IR cameras aligned in stereo is used. If successfully implemented, this allows determination of absolute coordinates of a fire. As with the jet modelling, efforts will also be made in reviewing relevant literature with

regards to IR stereo vision and fire detection.

In addition to the IR stereo cameras, an infrared sensor which measures the IR radiation at three different bandwidths (IR3 sensor) is added to verify the presence of a fire. Such sensors are available with a Safety Integrity Level (SIL) of 2 or 3, such that the presence of a fire can to a high degree of accuracy be determined.

Finally, the servomotors which actuate the monitor will be modelled and simulated in real time using a hardware-in-the-loop (HIL) setup with a platform running a dynamic model representing the fire monitor with actuators (Figure 1.4). A PLC will also be programmed to generate control signals to the servomotors.

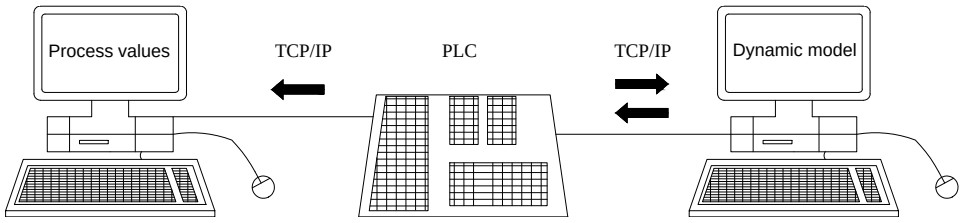



Figure 1.4: HIL setup.

The basic working principle of the system can be illustrated by Figure 1.5. Fire is initially detected by the IR3 sensor which computes a binary alarm signal. By default, the system is in a passive state, only acting with a positive output from the IR3 sensor. In turn, this activates the stereo vision system which determines absolute localization of the fire and forwards coordinates to the PLC, which estimates the proper configuration of the fire monitor based on fire localization and wind sensor data. Next, these signals are sent to the servomotors which properly poses the fire monitor. Finally, reference signals are sent to the pump which provides the necessary pressure and thus the necessary throw length. A visual summary of the working principles are presented in Figure 1.6.

To recap, the problem solution can be summarized as follows:

1. Using experiments and relevant literature, determine the most precise way to model the trajectory of a liquid jet
2. By experimentation and reviewing literature, synthesize an IR camera vision system to ascertain the position of a fire to a degree which satisfies the accuracy goal of the system
3. Create a PLC program which controls the fire monitor
4. Verify the entire system's accuracy by simulations, achieving a precision of $\leq$2 meters

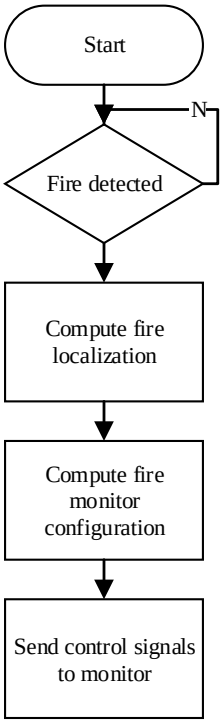



Figure 1.5: Basic working principle of system.

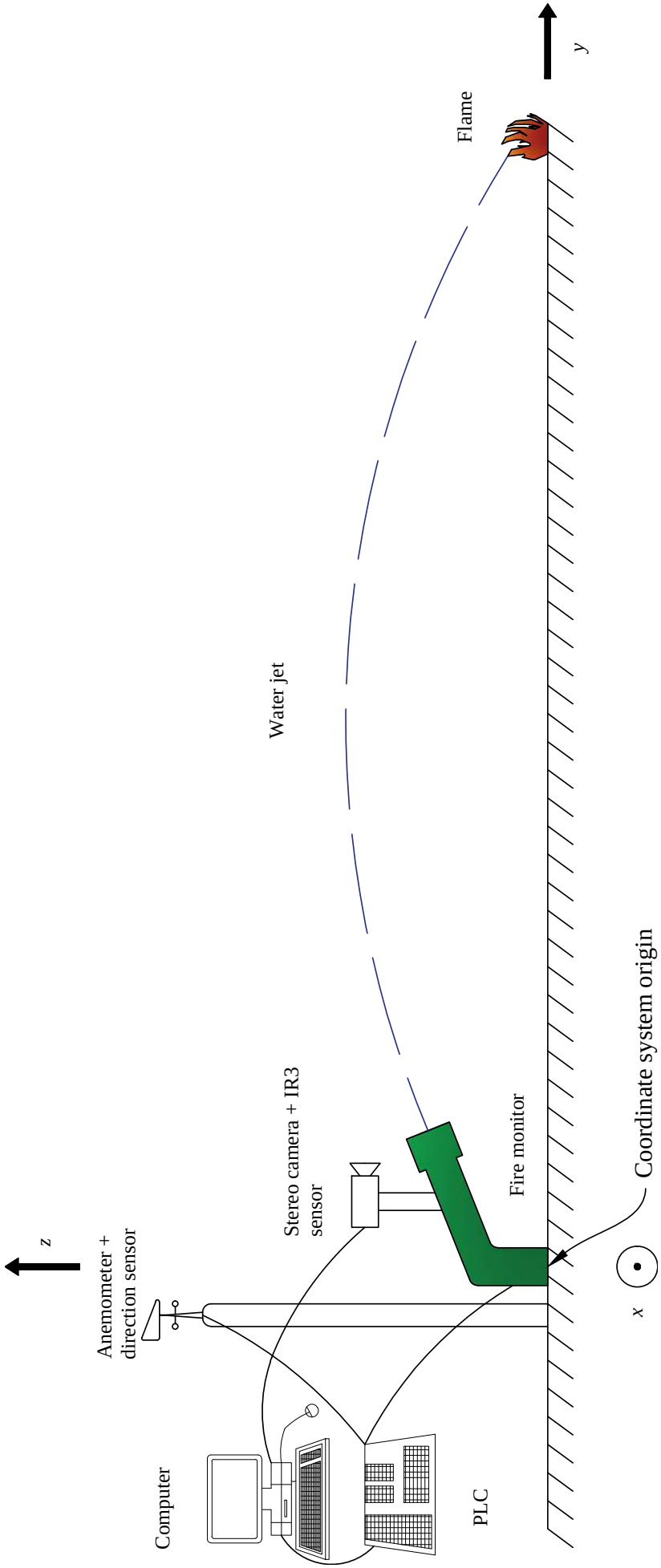



Figure 1.6: Sketch of working principle.

### 1.4.1 Stereo Vision Solution Ranking (a Side Note)

A large amount of research has been invested in fire detection with color video[2, 17] versus the comparatively small amount of literature on IR video fire analysis (e.g. [7, 18–20]). It is therefore appropriate to briefly summarize the traits of each approach. A brief review of each respective technology's advantages will be given here.

A comparison of the suitability of color video and IR video solutions for this problem is given in Table 1.1. Color video cameras score high for low cost, but require more complex image processing techniques in order to detect a fire. IR video therefore scores higher for simplicity. Long wavelength cameras have the ability to see through smoke, and is very efficient at detecting hot spots. Therefore, IR video is thought to be more precise in determination of a fire's location and scores higher for accuracy[17, 21]. An additional point could also have been dealt for resolution, but considering that for real time applications high resolution images often will have to be downsampled, this point is omitted.

Table 1.1: Solution ranking.

| Type | IR | Color video |
|---|---|---|
| Cost | 0 | 1 |
| Simplicity | 1 | 0 |
| Accuracy | 1 | 0 |
| Sum | 2 | 1 |

It should be noted however, that fire detection with IR cameras is not straightforward. Other sources of radiation such as people and vehicles may complicate discrimination of fire and other objects, and therefore additional techniques may be necessary in order to detect fire[19, 20]. Furthermore, humid environments offer a challenge since water features a high heat capacity, and therefore absorbs much of the radiation, potentially leaving IR cameras 'blind'. During times of heavy rain, for example, the ability to detect fires may be heavily impaired. Furthermore, the radiation from the sun may be a source of disturbance for any camera, although in the case of IR cameras, this depends on the spectral range of the camera (Figure 5.1). Lastly, highly reflective objects, for example shiny surfaces as aluminium, may introduce disturbances by reflecting stray radiation from other sources of light.

To conclude, the main drawback of IR cameras as opposed to color cameras is price, while on the other hand, IR cameras offer the following advantages[17, 21–25]:

1. Insensitive to small particles
2. Insensitive to changes in light conditions
3. IR radiation emitted by all kinds of objects

## 1.5 Report Outline

Chapter 2 reviews theoretical concepts and related work useful for fully grasping the contents of the present thesis. Chapter 3 shows how experiments are designed in order to solve the problems presented. Chapter 4 presents experimental results and analysis, and finally, Chapter 5 contains concluding remarks and suggestions for future work.

# CHAPTER 2

# Background

THIS chapter focuses on reviewing theoretical concepts and work relevant to solving the problems in this thesis. Section 2.1 reviews some definitions, 2.2 deals with relevant theoretical concepts, and section 2.3 reviews related works.

## 2.1 Definitions

The spatial positions in this thesis are described using the spherical coordinate system and right-handed cartesian coordinate system presented in Figure 2.1, with inclination $\theta$, azimuthal angle $\varphi$, radius $r$ and coordinates $x$, $y$, $z$ in three dimensional space. From the convention it follows that $r \in [0, \infty)$, $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$, and that the cartesian coordinates can be calculated with (2.1)-(2.3). Further, given that $\theta$ and $\varphi$ are constant, velocities and accelerations may be calculated by taking the time derivative in the normal fashion as in (2.4)-(2.9).

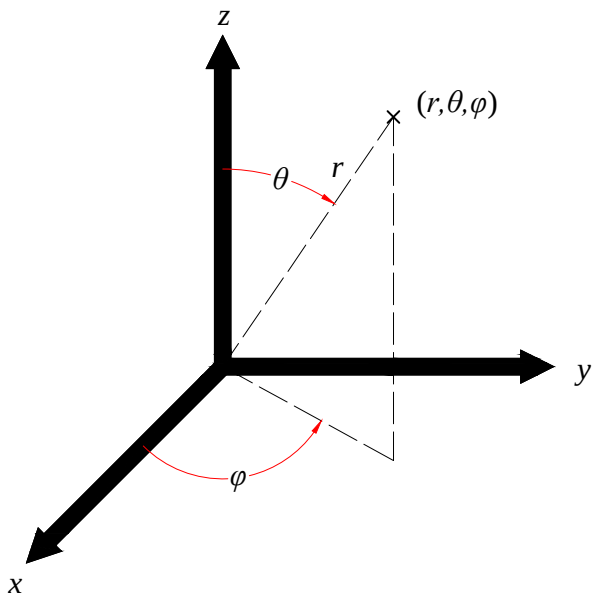



Figure 2.1: Spherical coordinate system.

$$x = r \cdot \sin\theta \cdot \cos\varphi \tag{2.1}$$
$$y = r \cdot \sin\theta \cdot \sin\varphi \tag{2.2}$$
$$z = r \cdot \cos\theta \tag{2.3}$$
$$\dot{x} = \dot{r} \cdot \sin\theta \cdot \cos\varphi \tag{2.4}$$
$$\dot{y} = \dot{r} \cdot \sin\theta \cdot \sin\varphi \tag{2.5}$$
$$\dot{z} = \dot{r} \cdot \cos\theta \tag{2.6}$$
$$\ddot{x} = \ddot{r} \cdot \sin\theta \cdot \cos\varphi \tag{2.7}$$
$$\ddot{y} = \ddot{r} \cdot \sin\theta \cdot \sin\varphi \tag{2.8}$$
$$\ddot{z} = \ddot{r} \cdot \cos\theta \tag{2.9}$$

### 2.1.1 Pinhole Model

In the present thesis, the pinhole camera model is used to describe the relationship between the camera plane and its three dimensional surroundings. The camera model describes the mathematical relationship between three-dimensional points and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point whereby no optics are used to focus incident light (Figure 2.2). The model is an approximation, and does not include distortions due to imperfections in geometry or blurring introduced by the camera's lens. In addition, it does not take into consideration the fact that digital cameras have only discrete image coordinates. As a consequence, the pinhole camera model is limited as an approximation which describes the transformation between three-dimensional world coordinates and a two-dimensional image.

The model's usefulness is ultimately dictated by the quality of the camera (i.e., optics and resolution), but some of the effects that the pinhole model does not take into consideration can be compensated for by applying coordinate transformations on the image coordinates, and some effects are sufficiently small such that they may be ignored. As a result, if a camera of sufficiently high quality is used, the pinhole model can be utilized as a reasonable approximation of three-dimensional space as viewed by the camera. The applications of the pinhole model will be discussed in further detail in Section 2.2 and 3.6.2.

Figure 2.3 shows the relations between a point $P$ in space and the coordinates at the image plane denoted by $u$ and $v$. By similar triangles it can be shown that a mathematical correlation between the point $P$ and $Q$ in the image plane is

$$\begin{bmatrix} u \\ v \end{bmatrix} = -\frac{f}{y} \begin{bmatrix} x \\ z \end{bmatrix} \tag{2.10}$$

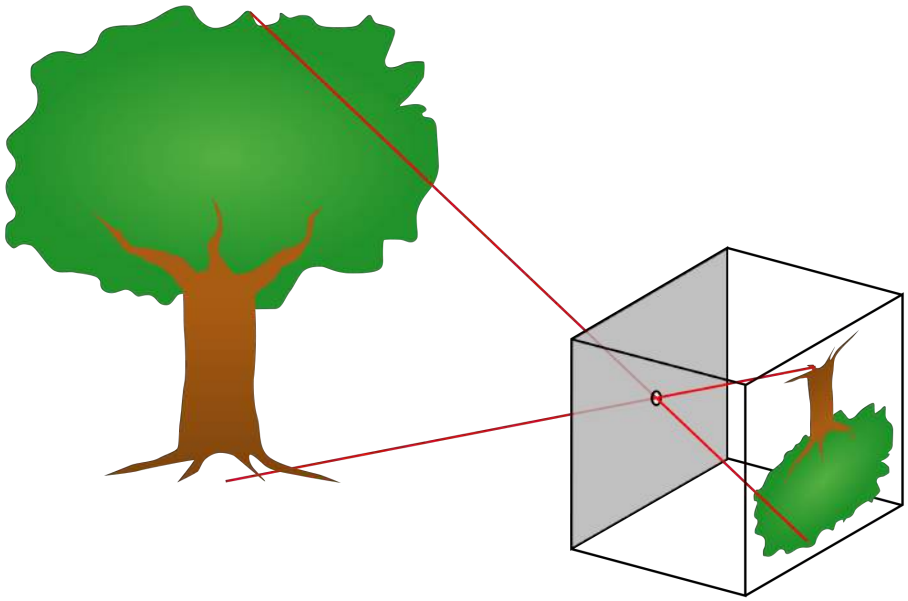


Figure 2.2: The pinhole model.

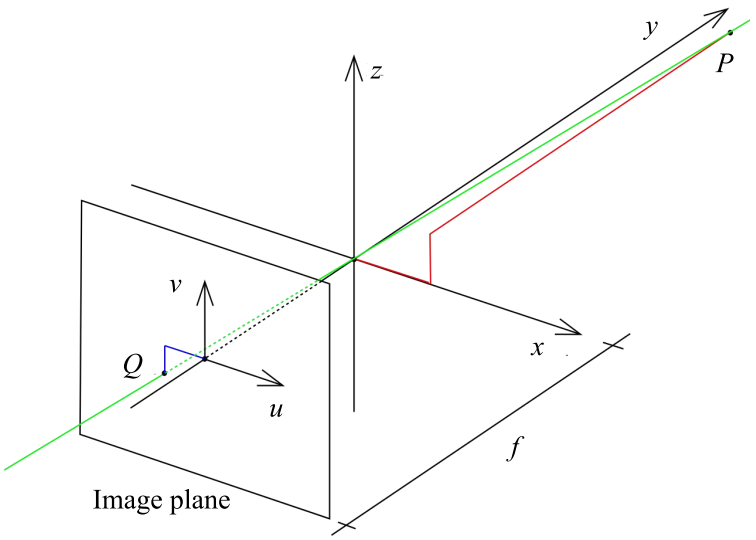



Figure 2.3: Geometric relations between world coordinates and image plane using the pinhole model.

## 2.2 Theory

### 2.2.1 Drag forces

As the reader is probably already familiar, a solid body travelling through a fluid will experience a drag force which opposes the motion of the body relative to the fluid. In general terms, the force is proportional to the relative velocity and the square of the relative velocity, which is expressed by Equation (2.11). The terms are known respectively as the *viscous* term and the *pressure term*[26].

$$F_D = -(k_1 \cdot \dot{r} + k_2 \cdot \dot{r}^2)\hat{\dot{r}} \tag{2.11}$$

In most cases, for instance when one is concerned with air drag, the viscous term is negligible and the pressure term is dominant. For a given situation, Equation (2.12) may then be used, where the drag coefficient $C_D$ is dependant on geometry and Reynold's number. The equation is valid for $Re \in [1 \cdot 10^3 \sim 2 \cdot 10^5]$. If the conditions persist, i.e. $Re$ is constant, the equation may be simplified as in Equation (2.13).

$$F_D = \frac{1}{2} \cdot \rho \cdot C_D \cdot A \cdot \dot{r}^2 \tag{2.12}$$

$$F_D = k \cdot \dot{r}^2 \tag{2.13}$$

### 2.2.2 Modelling Liquid Discharges

Equation (2.11) is not directly applicable for the purposes of the present thesis. As will be explained in greater detail in Section 2.3.2, the physics of free fall applied to a solid body, needless to say, is not sufficient to predict the behaviour of a liquid jet. As a consequence, one often tends to empiricism, meaning (2.11) is replaced by equations essentially dictated by common intuition and/or experience. These will, typically, contain parameters which depend on factors such as jet length, jet diameter etc.

The basic technique for predicting the position in any case is time integration of Newton's second law. For a liquid jet this is the force of gravity and the drag force due to air resistance (Figure 2.4).

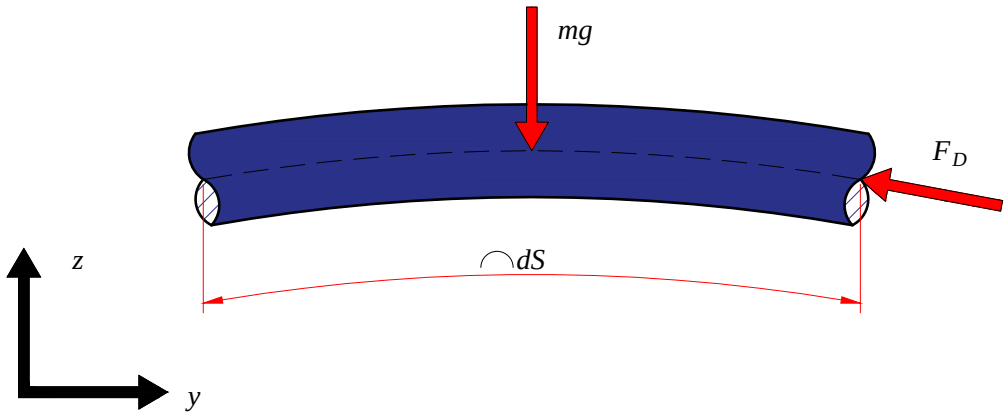



Figure 2.4: A strip $dS$ of a jet subjected to gravitational- and drag forces.

Newton's second law states that the sum of external forces $F$ is proportional to the acceleration of any body with mass $m$

$$\sum \vec{F} = m\vec{\ddot{r}} \tag{2.14}$$

Expanding the concept to three dimensions, this leads to the vector equation in (2.15), where the drag force $F_D$ will be the previously mentioned empirical drag force.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -F_{Dx} \\ -F_{Dy} \\ -mg - F_{Dz} \end{bmatrix} \tag{2.15}$$

In the present thesis, the mass $m$ will be set to 10 kg. This is mainly done for convenience as it reduces the complexity in the optimization process, where the free parameters of a given drag force model are determined, in addition to the fact that, needless to say, the mass of a jet is not easily ascertainable.

### 2.2.3 Stereo Vision

A stereo vision system has two cameras which are physically aligned with an overlapping field of view (FOV). For a given scene, pixels from one camera are matched to the corresponding pixels in the image of the other camera. Through geometric correlations, the offset of pixel coordinates in each camera's frame of reference can be converted to spatial coordinates using the camera properties, such as focal length, baseline, principal points and pixel size[27]. In this paper, the cameras are aligned in parallel, and the coordinates of a point in space relative to the left camera can be estimated with (2.16)-(2.19), expressed in terms of the baseline $B$, focal length $f$ and disparity $d$. An illustration of the working principle is given in Figure 2.5.

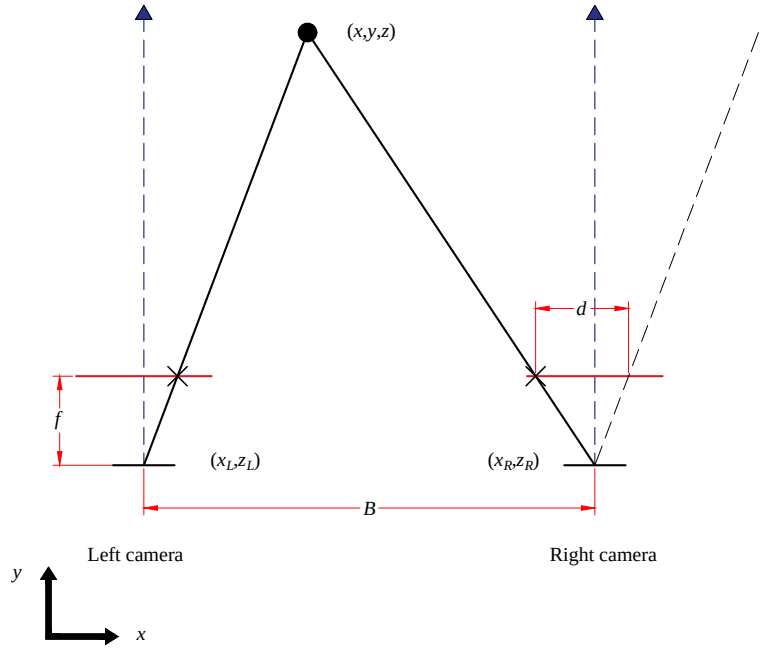



Figure 2.5: Stereo vision: Principle of operation for two perfectly aligned cameras.

$$d = x_L - x_R \tag{2.16}$$

$$x = \frac{B}{d} \cdot x_L \tag{2.17}$$

$$y = \frac{f \cdot B}{d} \tag{2.18}$$

$$z = \frac{B}{d} \cdot z_L \tag{2.19}$$

### 2.2.4 The TR Matrix

When dealing with rigid motion in three-dimensional space, it is customary to define a matrix which describes the orientation (translation and rotation) of a body in space. First, rotational matrices $\mathbf{R}$ are defined, one for each axis. Let the subscript denote axis of rotation with angular displacement $\theta$, then the rotational

matrices can be formulates as[28]

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.20}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.21}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

It can be observed that the matrices in (2.20)-(2.22) impose the following properties:

1. $\mathbf{R}^{-1} = \mathbf{R}^T$
2. The columns and rows are mutually orthogonal
3. Each column and row is a unit vector
4. $\det \mathbf{R} = 1$

Now, let $\theta$, $\phi$ and $\alpha$ denote rotation about the $x$, $y$ and $z$ axis respectively. Successive rotation about each axis then becomes

$$\mathbf{R} = \begin{bmatrix} \cos(\alpha)\cdot\cos(\phi) & -\cos(\phi)\cdot\sin(\alpha) & \sin(\phi) \\ \cos(\theta)\cdot\sin(\alpha)+\cos(\alpha)\cdot\sin(\theta)\cdot\sin(\phi) & \cos(\alpha)\cdot\cos(\theta)-\sin(\alpha)\cdot\sin(\theta)\cdot\sin(\phi) & -\cos(\phi)\cdot\sin(\theta) \\ \sin(\alpha)\cdot\sin(\theta)-\cos(\alpha)\cdot\cos(\theta)\cdot\sin(\phi) & \cos(\alpha)\cdot\sin(\theta)+\cos(\theta)\cdot\sin(\alpha)\cdot\sin(\phi) & \cos(\theta)\cdot\cos(\phi) \end{bmatrix} \tag{2.23}$$

Now one can define the **TR** matrix by including translation **t**, again with subscripts indicating direction. The **TR** matrix then becomes

$$\mathbf{TR} = [\mathbf{R}\ \mathbf{T}] = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} =$$

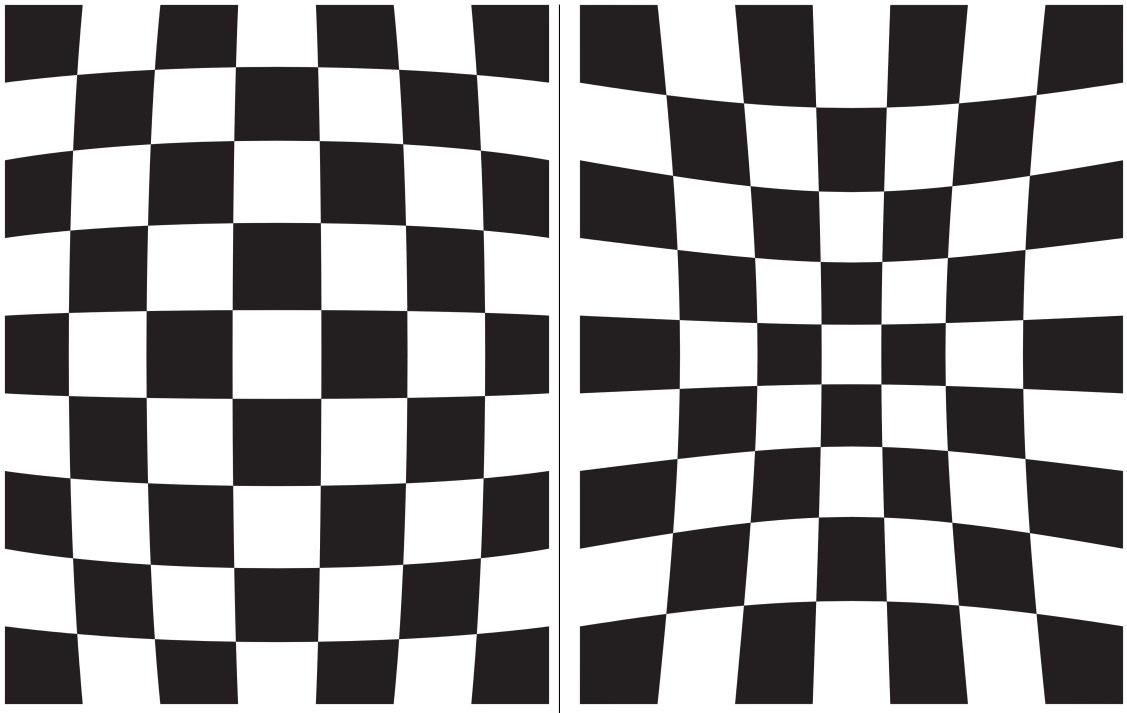
$$\begin{bmatrix} \cos(\alpha)\cdot\cos(\phi) & -\cos(\phi)\cdot\sin(\alpha) & \sin(\phi) & t_x \\ \cos(\theta)\cdot\sin(\alpha)+\cos(\alpha)\cdot\sin(\theta)\cdot\sin(\phi) & \cos(\alpha)\cdot\cos(\theta)-\sin(\alpha)\cdot\sin(\theta)\cdot\sin(\phi) & -\cos(\phi)\cdot\sin(\theta) & t_y \\ \sin(\alpha)\cdot\sin(\theta)-\cos(\alpha)\cdot\cos(\theta)\cdot\sin(\phi) & \cos(\alpha)\cdot\sin(\theta)+\cos(\theta)\cdot\sin(\alpha)\cdot\sin(\phi) & \cos(\theta)\cdot\cos(\phi) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

### 2.2.5 Camera Calibration

Although image resolution poses the ultimate limiting factor in terms of precision, lens distortion may introduce appreciable errors, especially when comparing 2D images[29]. Inexpensive wide angle lenses in particular tend to introduce radial distortion[30, 31](Figure 2.6). This type of distortion is often the most prevalent in IR cameras, but also tangential distortion (Figure 2.7) may be present[23, 32]. Camera calibration is therefore considered necessary in computer vision applications in order to virtually rectify optic imperfections and thus improve precision.

The general methodology is to photograph a calibration template with geometry which is known to a high degree of accuracy. The template is fastened to a planar surface, and the points on the template are related to global coordinates relative to the camera. The calibration procedure is as follows:

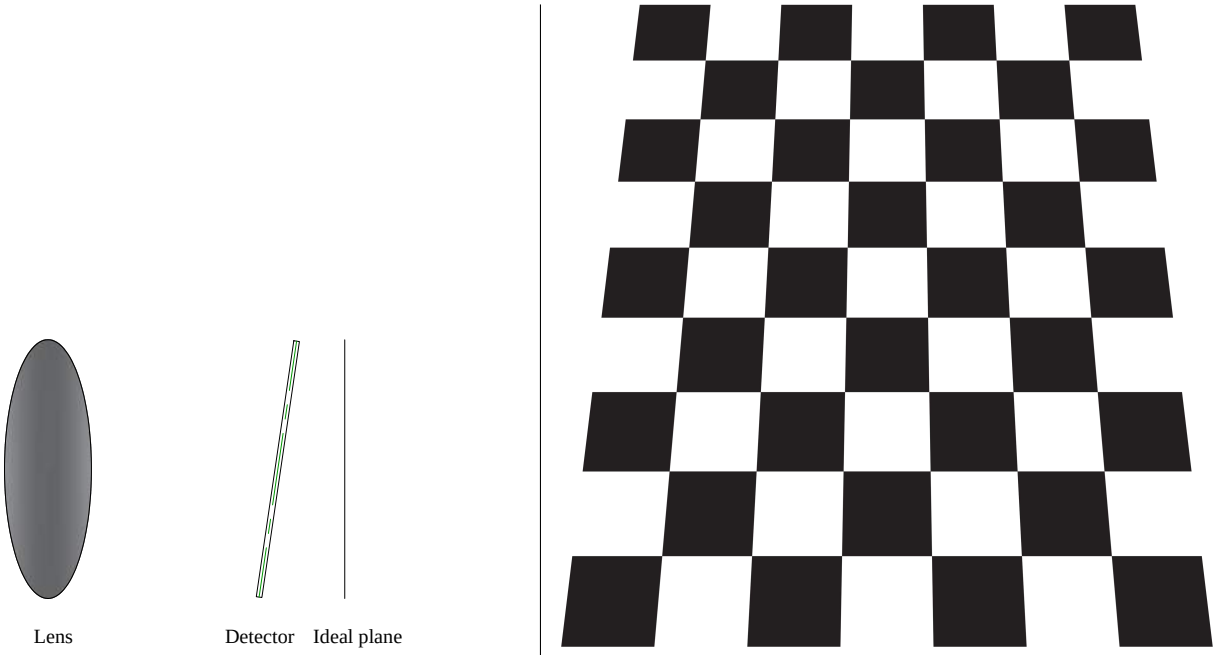
1. Attach a pattern (i.e. calibration template) to a flat surface.
2. Take 10-20 images of the template with various orientations by moving either the camera or the pattern.



(a) Barrel distortion: Positive radial distortion.

(b) Pincussion distortion: Negative radial distortion.

Figure 2.6: Radial distortion.




(a) Misalignment of detector chip.

(b) Resulting image due to lens misalignment.

Figure 2.7: Tangential distortion.

3. Detect the feature points in each picture.

4. Estimate intrinsic- and extrinsic parameters.

In this work, the *Camera Calibration Toolbox* in MATLAB is used, which refers to the work done by Zhang [31]. Therefore, the main contents of the mentioned paper will be reviewed in the following.

### Notation

Two-dimensional points in the image plane are denoted by $\mathbf{m} = [u\ v]^T$ and three-dimensional ones are denoted by $\mathbf{M} = [x\ y\ z]^T$. A tilde is added to indicate the augmented vector by adding 1 as the last element: $\widetilde{\mathbf{M}} = [x\ y\ z\ 1]^T$.

### Basic Equations

Now, with the pinhole model as basis (Section 2.1), the relation between world coordinates and the image projection is

$$s\widetilde{\mathbf{m}} = \mathbf{A}\,[\mathbf{R}\ \mathbf{T}]\,\widetilde{\mathbf{M}} \qquad ; \text{where } \mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

where $s$ is a scalar, $[\mathbf{R}\ \mathbf{T}]$ is the extrinsic parameters, consisting of the rotational matrix and translational vector relating the world coordinates to the camera plane (as reviewed in the preceding section). $\mathbf{A}$ is called the intrinsic matrix where $u_0$, $v_0$ are the coordinates to the principal point in the camera's coordinate system, $\alpha$ and $\beta$ are scale factors of the $u$ and $v$ axis respectively, and finally $\gamma$ describes the skew between the $u$ and $v$ axes.

Generalizing (2.25) while assuming the calibration template is at $z = 0$ and denoting the $i$'s column of the rotational matrix $\mathbf{R}$ by $\mathbf{r}_i$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3\ \mathbf{t}]\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.26}$$

And further defining $\mathbf{H}$ such that

$$s\widetilde{\mathbf{m}} = \mathbf{H}\widetilde{\mathbf{M}} \qquad ; \text{where } \mathbf{H} = \mathbf{A}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \tag{2.27}$$

Denoting $\mathbf{H}$ by $[\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3]$, (2.27) can be rewritten

$$[\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3] = \lambda \cdot \mathbf{A}\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3] \tag{2.28}$$

where $\lambda$ is a scaling factor. Further, since the rotational vectors $\mathbf{r}_i$ are orthonormal, one can simplify

$$\mathbf{h}_1 \cdot^T \mathbf{A}^{-T} \cdot \mathbf{A}^{-1} \cdot \mathbf{h}_2 = 0 \tag{2.29}$$

$$\mathbf{h}_1^T \cdot \mathbf{A}^{-T} \cdot \mathbf{A}^{-1} \cdot \mathbf{h}_1 = \mathbf{h}_2^T \cdot \mathbf{A}^{-T} \cdot \mathbf{A}^{-1} \cdot \mathbf{h}_2 \tag{2.30}$$

(2.29) and (2.30) represents the basic constraints on the intrinsic parameters.

### Analytic Solution

The closed form solution is

$$\mathbf{B} = \mathbf{A}^{-T} \cdot \mathbf{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2 \cdot \beta} & \frac{v_0 \cdot \gamma - u_0 \cdot \beta}{\alpha^2 \cdot \beta} \\ -\frac{\gamma}{\alpha^2 \cdot \beta} & \frac{\gamma^2}{\alpha^2 \cdot \beta^2} + \frac{1}{\beta^2} & -\frac{(v_0 \cdot \gamma - u_0 \cdot \beta)\gamma}{\alpha^2 \cdot \beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0 \cdot \gamma - u_0 \cdot \beta}{\alpha^2 \cdot \beta} & -\frac{(v_0 \cdot \gamma - u_0 \cdot \beta)\gamma}{\alpha^2 \cdot \beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0 \cdot \gamma - u_0 \cdot \beta)^2}{\alpha^2 \cdot \beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \tag{2.31}$$

which is a symmetric matrix defined by

$$\mathbf{b} = [B_{11}\ B_{12}\ B_{22}\ B_{13}\ B_{23}\ B_{33}]^T \tag{2.32}$$

If the $i$th column of $\mathbf{H}$ is denoted $\mathbf{h}_i = [h_{i1}\ h_{i2}\ h_{i3}]^T$, then

$$\mathbf{h}_i^T \cdot \mathbf{B} \cdot \mathbf{h}_j = \mathbf{v}_{ij}^T \cdot \mathbf{b} \tag{2.33}$$

where:

$$v_{ij} = [h_{i1} \cdot h_{j1}, h_{i1} \cdot h_{j2} + h_{i2} \cdot h_{j1}, h_{i2} \cdot h_{j2}, h_{i1} \cdot h_{j3} + h_{i3} \cdot h_{j1}, h_{i2} \cdot h_{j3} + h_{i3} \cdot h_{j2}, h_{i3} \cdot h_{j3}]^T$$

Given $\mathbf{H}$, (2.29) and (2.30) can be rewritten as

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = 0 \tag{2.34}$$

When $n$ images of a calibration template are observed, (2.34) can be written

$$\mathbf{V} \cdot \mathbf{b} = 0 \tag{2.35}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix.

Once $\mathbf{b}$ is estimated, all intrinsic parameters can be determined. This can be done by approximating $\mathbf{B}$ up to an arbitrary scaling factor

$$B = \lambda \cdot \mathbf{A}^{-T} \cdot \mathbf{A}^{-1}$$

It then follows that

$$v_0 = \frac{B_{12} \cdot B_{13} - B_{11} \cdot B_{23}}{B_{11} \cdot B_{22} - B_{12}^2}$$

$$\lambda = B_{33} - \left[ (B_{12} \cdot B_{13} - B_{11} \cdot B_{23})\, v_0 + B_{13}^2 \right] / B_{11}$$

$$\alpha = \sqrt{\frac{\lambda}{B_{11}}}$$

$$\beta = \sqrt{\frac{\lambda \mathrm{B}_{11}}{B_{11} \cdot B_{22} - B_{12}^2}}$$

$$\gamma = -\frac{\alpha^2 \cdot \beta \cdot B_{12}}{\lambda}$$

$$u_0 = \frac{\gamma \cdot v_0}{\alpha} - \frac{\alpha^2 \cdot B_{13}}{\lambda}$$

The extrinsic parameters may then be calculated from (2.27), with $\lambda = 1/||\mathbf{A}^{-1}\mathbf{h}_1||$:

$$\mathbf{r}_1 = \lambda \cdot \mathbf{A}^{-1}\mathbf{h}_1, \qquad \mathbf{r}_2 = \lambda \cdot \mathbf{A}^{-1}\mathbf{h}_2, \qquad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2, \qquad \mathbf{t} = \lambda \cdot \mathbf{A}^{-1}\mathbf{h}_3 \tag{2.36}$$

### Maximum-Likelihood Estimation

Assume $n$ images of a calibration template with $m$ points are given, which are biased by independent noise. The maximum-likelihood estimate can be obtained by minimizing the following function

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{m}_{ij} - \hat{\mathbf{m}}\left(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j\right)\|^2 \tag{2.37}$$

where $\hat{\mathbf{m}}\left(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j\right)$ is the projection of point $\mathbf{M}_i$ in accordance with (2.27).

In general, both the closed form solution in (2.31) and the maximum-likelihood estimate are used. Upon obtaining images of a calibration template in various orientations, the intrinsic and extrinsic parameters can be computed with the closed form solution, after which the parameters are refined by minimizing (2.37).

### 2.2.6 Image Processing

This section will describe relevant image processing techniques. Only grayscale images will be dealt with, as IR cameras does not produce cameras with color information.

#### Thresholding

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $I$ at $(u,v)$ is less than some fixed constant $T$. With 256 luminance levels, this can be programmatically formulated as

```
if I(u,v)<T
then 0
else 255
```

In essence, this means that each pixel is converted to either completely white or black. An example of a thresholding operation is presented in Figure 2.8.

(a) Input image. (b) Binary output image.

Figure 2.8: Illustration of thresholding.

#### Small Particle Removal

This process can be applied to binary images in order to remove particles which are small in comparison to other objects in the image. This is done by repeatedly applying erosion. Let $A$ be the input image and $H$ the structuring element (kernel). Then erosion can be defined by keeping only pixels $p \in A$ such that $H_p$ fits in $A$[33]:

$$A \ominus H = \{p | H_p \subseteq A\} \tag{2.38}$$

To put it more simply, if the structuring element is not identical to the image at the sampling point, the pixels will be deleted. An example is shown in Figure 2.9, where the input image has been eroded by a $3 \times 3$ matrix of ones.

(a) Input image. (b) Eroded image.

Figure 2.9: Erosion of a binary image.

### 2.2.7 Review: What Is IR?

This section will attempt to shed some light (no pun intended) on IR radiation, and how it relates to heat.

In short, IR radiation is invisible electromagnetic waves with longer wavelengths than that of visible light, extending from the red edge of the visible spectrum at ∼0.7 $\mu$m to ∼1 mm. All matter with temperatures above absolute zero emit some amount of radiation, and all matter emits radiation in the IR spectrum[23, 26]. This a consequence of conversion of kinetic energy to heat due to intermolecular collisions, which is released in the form which is known as thermal radiation. In general terms, all electromagnetic radiation is classified as shown in Figure 2.10: $\gamma$-rays, X-rays, UV, visible light, IR, microwaves, and radio waves. It may be useful to keep in mind that the energy of photons is proportional to the frequency. Therefore, high temperature sources will tend to release more thermal energy towards the left end of the spectrum and may thus also be visible to humans. On the other hand, objects with comparatively lower temperatures will still emit considerable amounts of radiation in the IR spectrum, and therefore heat signatures are easily detectable.

The temperature of a perfect black body can be determined with Planck's law, which states that spectral radiance (power per steradian per area per wavelength) emitted is a function of the absolute temperature $T$ and wavelength $\lambda$ (Equation (2.39), Figure 2.11). Clearly, the amount of radiance of a black body with a temperature of a few hundred degrees is greater in the IR spectrum than in the visible part, and will not be readily visible when the temperature is sufficiently low. However, as temperature increases, a black body will start to radiate more at wavelengths which are visible to humans and become 'red hot', before eventually appearing blue and violet. For an open, class A fire[1] the surface temperature is ∼400-500°C. If the fire is confined, the temperature may rise to ∼1100°C[1].

$$SR(\lambda, T) = \frac{2 \cdot c^2 \cdot h}{\lambda^5 \left( \exp\left( \frac{c \cdot h}{\lambda \cdot T \cdot k_B} \right) - 1 \right)} \tag{2.39}$$

The wavelengths at the peaks presented in Figure 2.11 is uniquely defined by Wien's displacement law,

[1] Class A fire: Solid material, usually organic (e.g. coal, paper, cardboard) which burn with the formation of glowing embers[34].

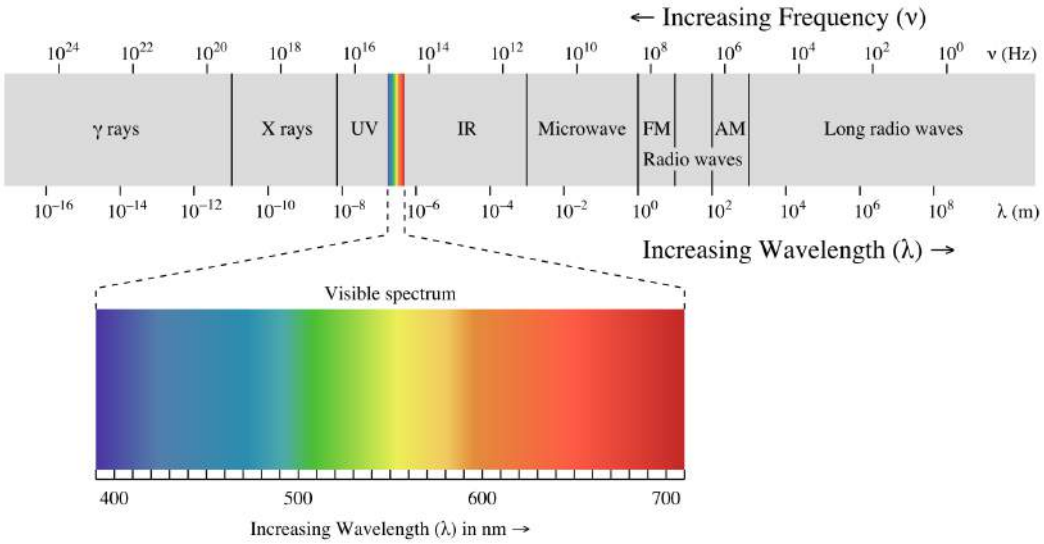



Figure 2.10: The electromagnetic spectrum.

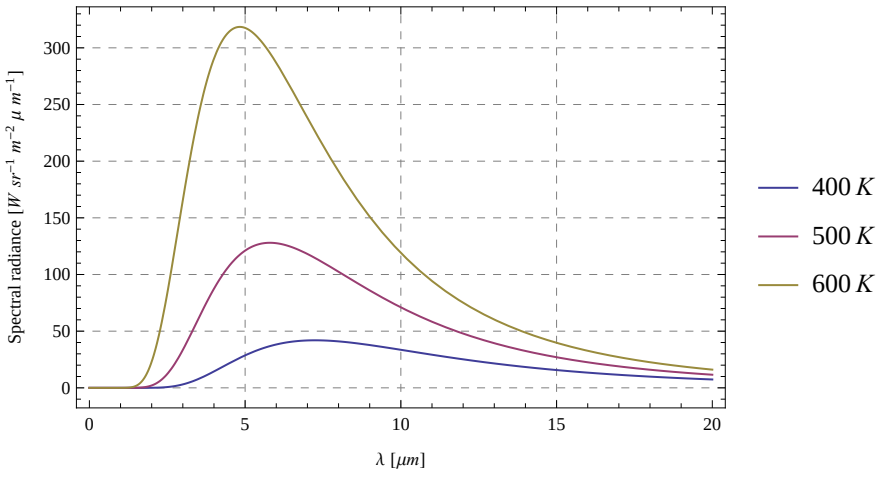



Figure 2.11: Spectral radiance of a perfect black body for various temperatures.

which states that the wavelength $\lambda_{max}$ is inversely proportional to temperature

$$\lambda_{max} = \frac{b_W}{T} \tag{2.40}$$

where $b_W$ is Wien's displacement constant. As a consequence, any detector will have to be made, depending on application, to be sensitive at the spectral range of interest. For the given curves in Figure 2.11, the peak wavelengths are 7.2, 5.8 and 4.8 $\mu$m respectively.

For a fire, the spectral intensity distribution may appear as shown by Figure 2.12. Three different curves for a hydrocarbon flame are presented; one for a perfect black body, and one for diffusion- and pre-mixed-flames. These are the two distinct types of flame. The pre-mixed flame is characterised by the blue flame which signifies that the combustion is complete. while the diffusion flame is characterized by a yellow flame in which soot is formed, indicating an incomplete combustion[35].

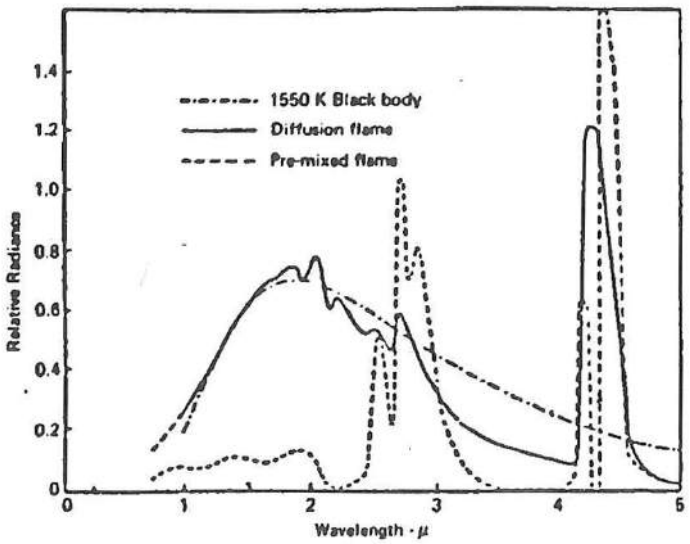



Figure 2.12: Typical spectral intensity distribution for a hydrocarbon flame[35]. Units in W/(sr · nm).

### 2.2.8 Microbolometers

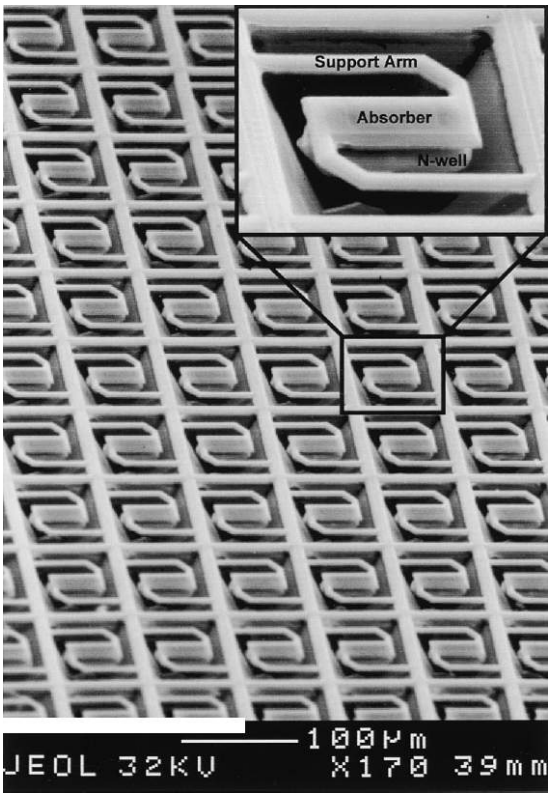
An IR camera, or thermographic camera, utilizes microbolometer detectors. In essence, each pixel on such a detector is a microscopic thermometer. The detectors' chip consist of an array of pixels with a slab of material, often doped amorphous silicon or vanadium oxide, which is heated by incident radiation (Figure 2.13). The resistance of the material subsequently changes, which makes it possible to measure alterations in heat in quantities as current or voltage[36, 37].

Unlike color cameras, IR cameras does not extract color information. Instead, the outputs are grayscale images where the luminance of each pixel depends on the amount of incident radiation (Figure 2.14).
The analysis of each pixel on an IR detector starts with the solution of the heat flow equation that describes the temperature increase in terms of the incident radiant power. The heat flow equation governing the heat increase $\Delta T$ in each pixel is[37]

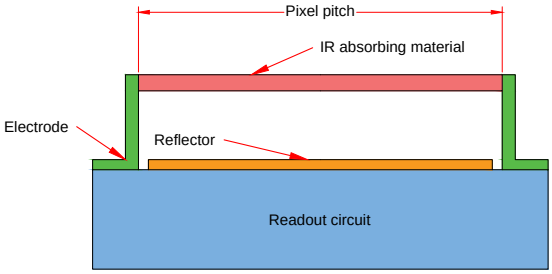
$$C\frac{\mathrm{d}(\Delta T)}{\mathrm{dt}} + G(\Delta T) = \eta \cdot P_0 \cdot e^{j\cdot\omega\cdot t} \tag{2.41}$$

And it's solution with respect to temperature change is

$$\Delta T = \frac{\eta \cdot P_0}{G\sqrt{\tau^2 \cdot \omega^2 + 1}} \tag{2.42}$$




(a) Pixel array[38].




(b) Sectional view of a single microbolometer pixel.

Figure 2.13: Microbolometer.

Figure 2.14: Color image versus IR. The heat source is a small wooden fire.

where $C$ is heat capacity of the sensitive area of the pixel, $G$ is the thermal conductance of the supporting pins, $P_0$ is the amplitude IR radiation power incident on the pixel, $\eta$ is the absorbance of IR sensitive films, $\tau$ is the thermal time constant and $\omega$ is the angular frequency of modulation of the incoming radiation.

When the resistance is linearly dependant on temperature, the resistance $R$ can be expressed as a function of temperature $T$:

$$R = R_0(\Delta T \cdot \mathrm{TCR} + 1) \tag{2.43}$$

where $R_0$ is the initial resistance and TCR is the temperature coefficient of resistance.

With the voltage $V$ proportional to the current $I$, and assuming $\Delta R \simeq R \cdot TCR \cdot \Delta T$, one obtains the following expression for the voltage across a pixel

$$V = \frac{\eta \cdot P_0 \cdot \mathrm{TCR} \cdot R}{G\sqrt{\tau^2 \cdot \omega^2 + 1}} I \tag{2.44}$$

It must be stressed that the solution here is the signal which is produced for a change in temperature, and is thus not a measure of absolute temperature, which requires precise calibration.

## 2.3 Review of Literature

Applicable literature has been found by searching relevant engineering- and physics databases, specifically ISI Web of Science, Scopus, IEEE Xplore and Google Scholar. Examples of search terms are presented in Table 2.1. Once a relevant article is identified, further relevant papers have also been discovered by reviewing the reference lists and by searching for related works using built in search algorithms in the Scopus and ISI Web of Knowledge databases.

Table 2.1: Literature search: Databases and search examples.

| Topic | Databases | Search examples |
|---|---|---|
| Water discharge trajectory | ISI Web of Science, Scopus, IEEE Xplore, Google Scholar | Water jet air, fluid projectile motion, fire water monitor discharge, discharge analysis, spray jet |
| Flame detection | ISI Web of Science, Scopus, IEEE Xplore, Google Scholar | Fire detection, infrared stereo vision, machine vision flame detection, video fire detection |
| Automatic fire extinguishing systems | ISI Web of Science, Scopus, IEEE Xplore, Google Scholar | Automatic fire suppression, automated fire detection, automated fire water monitor |

### 2.3.1 The Trajectories of Liquid Jets in Air

When the trajectory of water streams in air is of interest, most often it is modelled based on classic projectile motion (e.g. [39]) or it is neglected all together(e.g. [10, 11, 40]). For short throw lengths, classic modelling has arguably been shown to be appropriate. Goff and Liyanage [40] predicted the trajectory of a small-scale water stream for a few meters of horizontal distance while neglecting drag forces. This included using a small hose fitted with a protractor and white boards placed behind the water stream to measure the displacement. Varney and Gittes [39] suggested incorporating classical projectile motion in forensic practice to accurately determine the source of floor spatter in crime scene investigations. This included using (2.13) with $C_D = 0.5$ to model the drag forces and subsequently determine the source point. It was further demonstrated that droplets are relatively insensitive to air drag at this scale.

Edwards et al. [41] did experiments with water balloons over comparatively large distances which showed good agreement with classic projectile motion using (2.13) and $C_D = 0.55$. Theoretical throw lengths without air resistance overestimated horizontal displacement by some 20 m at most (Figure 2.15), and thus strongly indicate that drag forces are not negligible at such distances.

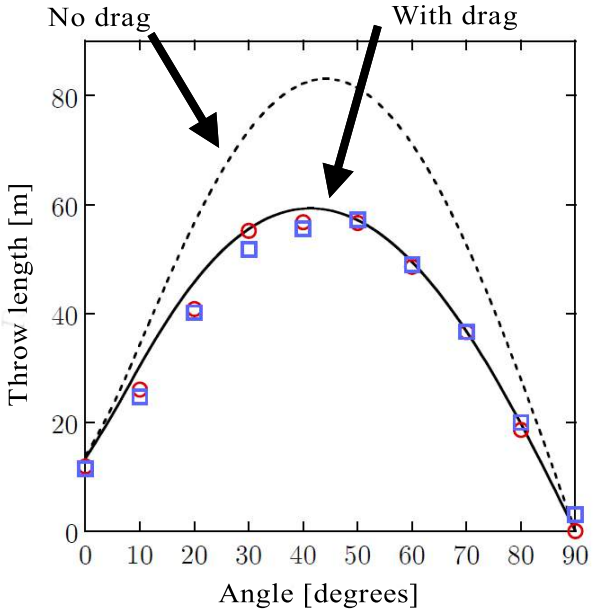



Figure 2.15: Experimental and theoretical horizontal range versus launch angle[41]. Experimental results are indicated by circles and squares.

For high capacity fire monitors, empirical or semi-empirical models are most often utilized in order to replicate the discharge trajectory[16, 42–44](Table 2.2). Due to the strong interaction between the water particles and atmosphere, deviations from an ideal trajectory based on the physics of free fall are, needless to say, substantial[44, 45]. A jet's behaviour in even quiescent air is particularly pronounced when observing high speed water jets (>85 m/s), let alone when wind affects are added[46, 47]. In addition, the breaking up of a jet during flight further complicates matters, since air resistance increases exponentially as the jet disintegrates to its constituent water particles (Section 2.2.2). Still, some efforts have been made in modelling discharge trajectories.

The rather comprehensive work conducted by Rouse et al. [44], although primarily concerned with nozzle and monitor geometry, resulted in the first experiments where jet trajectories were recorded known to the present authors. Hatton and Osborne [42] used these experimental results to fit two-dimensional simulations to the data utilizing a drag force proportional to the square of the velocity. It was proposed to make the proportionality constant a function of the Froude number, which in turn is a function of the initial velocity $\dot{r}_0$, initial jet diameter $D_0$ and gravitational acceleration $g$ as in (2.45). The results are presented in Figure 2.16, where it can be observed that the error between simulations and experimental data are in the order of a few meters at most. One obvious shortcoming of these results, however, is that the experimental trajectories have only been measured when a clear coherence in the jet could be observed, and thus the simulation model can only be verified for a limited amount of air travel. Furthermore, these results are only valid for approximately becalmed conditions, but no attempts are made at defining or documenting 'becalmed' conditions in this context.

$$Fr = \frac{\dot{r}_0}{\sqrt{g \cdot D_0}} \tag{2.45}$$

Hatton et al. [16] provided two dimensional simulations using an empirical exponential law (2.46). The free parameters $k$ and $b$ were automatically calculated and trajectories were compared with experimental data

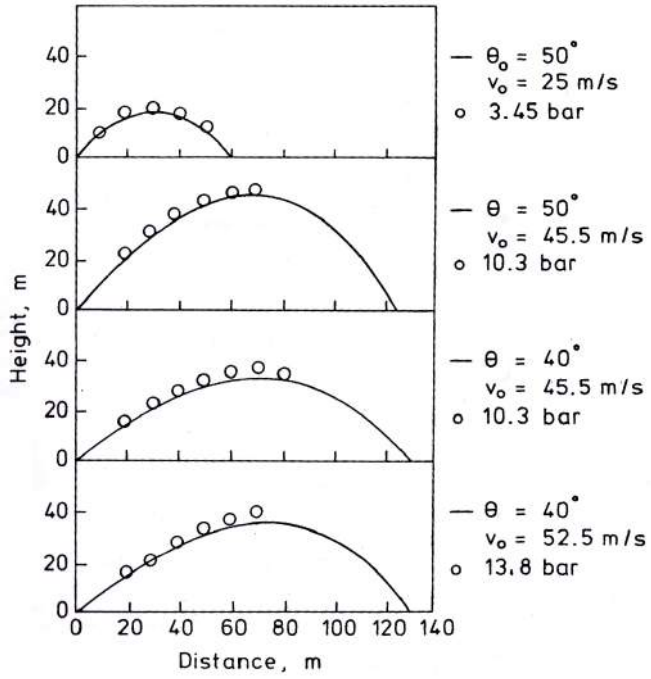



Figure 2.16: Comparison of experimental results from [44] (circled) and simulations (solid line)[42].

from the British Ship Research Association Laboratory, whereby water jets were photographed on a surveyed background (Figure 2.17). In addition, some simulation results in three dimensions using a modified version of (2.12), where the drag coefficient $C_D$ is given by (2.47), are shown in Figure 2.18 for various values of the free parameters $X$ and $Y$. A third model is also proposed, using a drag force which is proportional to powers of the distance along the jet path, $S$ (2.48). Some predicted trajectories according to the drag laws in (2.46) and a first order simplification of (2.48) is presented in Figure 2.19. Once more, however, predicted trajectories can only be verified for approximately wind still conditions.

$$F_D = k \cdot \dot{r}^2(1 + e^{b_0 \cdot S}) \tag{2.46}$$

$$C_D = C_{Da}\left[(X-1)\, e^{-Y \cdot S} + 1\right] \tag{2.47}$$

$$F_D = k \cdot \dot{r}^2\left(1 + a_0 \cdot S + a_1 \cdot S^2 + a_3 \cdot S^n\right) \tag{2.48}$$

Miyashita et al. [43] provided a three dimensional simulation model based on the Moving Particle Semi-implicit (MPS) method which reproduced results deviating by 3-14.7 % in range and 0-11.9 % in height (Figure 2.20). These results were achieved without sufficient knowledge of wind- and discharge conditions, however, and the authors of the cited paper considers the simulation results as verified with a deviation of less than 20 %. Subsequently, the authors suggested a spreadsheet model to predict the water trajectory in two dimensions, utilizing a third degree polynomial approximation with horizontal distance $x$ as variable. The height of the water jet $z$ is given by (2.49), where the inherent parameters are functions of wind speed in horizontal direction $U$, discharge pressure $p$ and elevation angle $\theta$ (here with baseline at the horizontal axis) (2.50)-(2.61). In addition, (2.49) contains a dimensionless parameter $M$ which represents the portion of particles projected along the discharge axis per flow, i.e. $M = 1$ indicates that 100 % of the discharge is delivered. For $M < 1$ parts of the jet's mass is torn from the bulk of the jet and hits the ground before the main portion of the water stream. For the purposes of this thesis, $M$ is set equal to unity, and the $n$ term in (2.50) can therefore be ignored. Inserting $\rho = 1000\ kg/m^3$ yields the expression in (2.62), which is valid for $U \in [-8, 8]\ m/s$, $\theta \in [30, 50]°$, $p \in [0.6, 0.9]$ MPa, $Q \in [10, 40]$ kl/min. Sample trajectories are presented in Figure 2.21, with $p = 0.7$ MPa, $U = 0$, $Q = 1.3$ kl/min and $z_0 = 0$

$$z(x) = a_0 \cdot M^{-n} \cdot G_3(\theta) \cdot G_3(Q) \cdot G_3(U) \cdot x^3 + b_0 \cdot G_2(\theta) \cdot G_2(p) \cdot G_2(Q) \cdot G_2(U) \cdot x^2 + c_0 \cdot G_1(\theta) \cdot x + z_0 \tag{2.49}$$

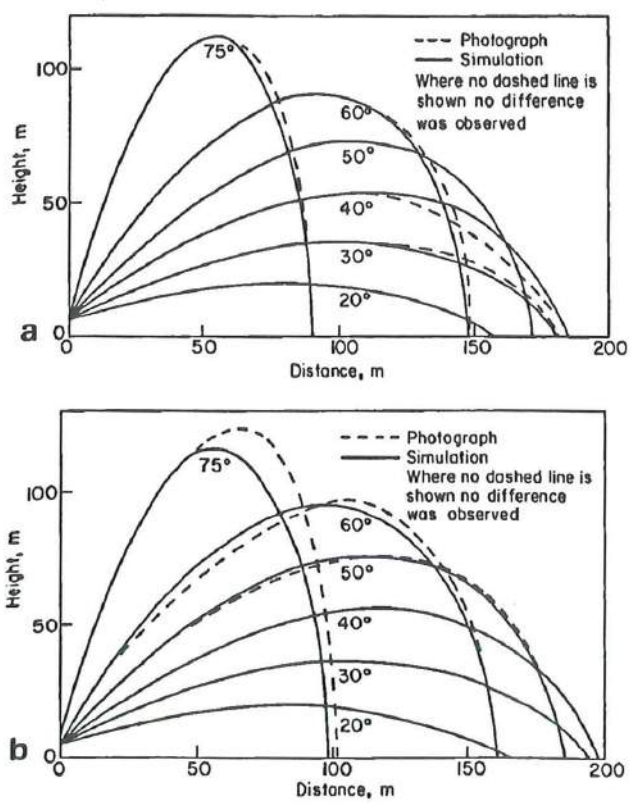



Figure 2.17: Comparison of computed and photographed trajectories: (a) $Q = 2400\ m^3/h$; (b) $Q = 3600\ m^3/h$[16].

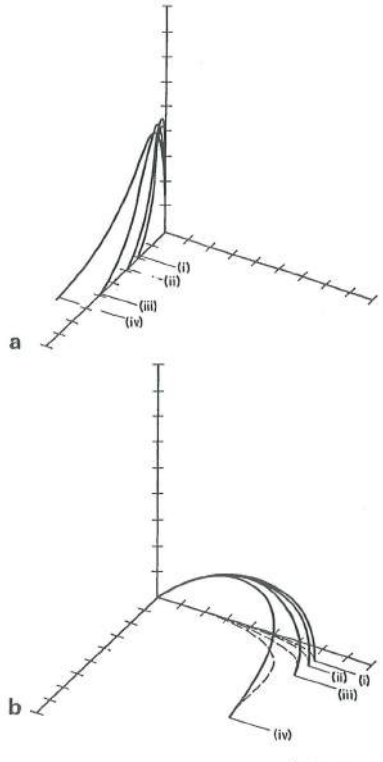



Figure 2.18: Influence of wind on simulations using (2.47) for various values of X and Y (25 m per division). Wind speed is 10 $m/s$ at 90° on initial jet angle[16].

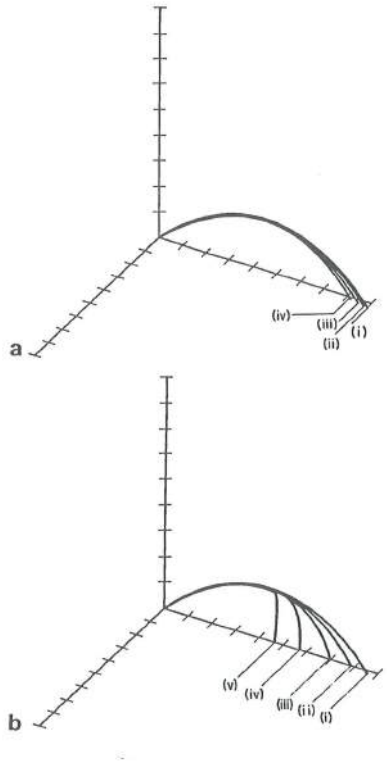



Figure 2.19: Predictions for windless conditions (25 m per division): (a) Using the drag law $k \cdot \dot{r}^2 \left(1 + a_0 \cdot S\right)$ for various values of $a_0$; (b) Using the drag law of $k \cdot \dot{r}^2 \left(1 + e^{b_0 \cdot S}\right)$ for various values of $b_0$[16].

$$n = \frac{2.1 \cdot U^2}{10^3} + \frac{1.9 \cdot U}{10^2} + 1.2 \tag{2.50}$$

$$G_3(U) = \frac{4.9 \cdot U^2}{10^3} - 0.15 \cdot U + 1 \tag{2.51}$$

$$G_2(U) = \frac{5 \cdot U}{10^2} + 1 \tag{2.52}$$

$$G_3(Q) = \frac{1}{Q^{0.8}} \tag{2.53}$$

$$G_2(Q) = 1 - \exp\left(-\frac{1}{5}(Q + 0.1)\right) \tag{2.54}$$

$$G_3(\theta) = 3 \cdot \theta^2 - 2.86 \cdot \theta + 1 \tag{2.55}$$

$$G_2(\theta) = \frac{1}{\cos^2(\theta)} \tag{2.56}$$

$$G_1(\theta) = \tan(\theta) \tag{2.57}$$

$$G_2(p) = \frac{1}{p} \tag{2.58}$$

$$a_0 = -\frac{7}{10^4} \tag{2.59}$$

$$b_0 = -\frac{2}{10^3} \tag{2.60}$$

$$c_0 = 1 \tag{2.61}$$

$$z(x) = -\frac{7\left(3 \cdot \theta^2 - 2.86\theta + 1\right)\left(0.0049 \cdot U^2 - 0.15U + 1\right)}{10000 \cdot Q^{0.8}} x^3 - \frac{\left(1 - e^{\frac{1}{5}(-Q-0.1)}\right)\left(\frac{U}{20} + 1\right)\sec^2(\theta)}{500 \cdot p} x^2 + \tan(\theta) \cdot x + z_0 \tag{2.62}$$

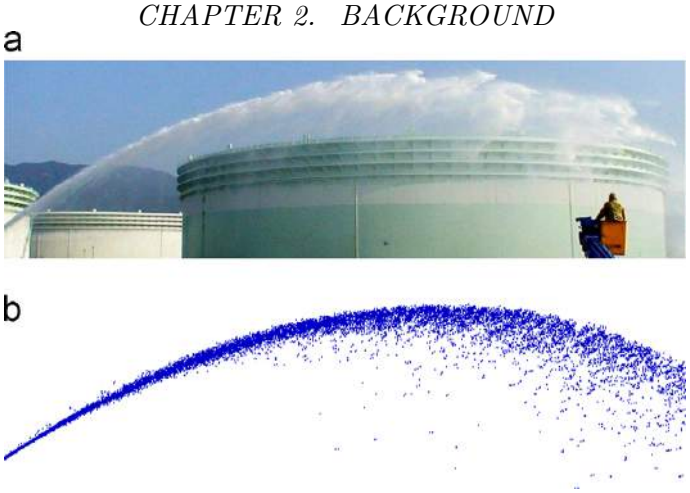



Figure 2.20: Comparison of side views with $Q = 20$ kl/min, $p = 0.7$ MPa, $\theta = 35°$ and 2.0 $m/s$ following wind: (a) Experimental trajectory; (b) Simulation using MPS method[43].

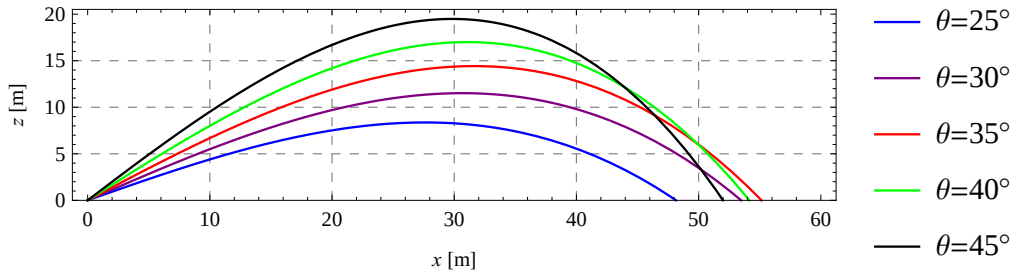



Figure 2.21: Sample trajectories of the spreadsheet model proposed by Miyashita et al. [43]. Note that $\theta$ is here measured from the $x$-axis.

It should also be noted that Long et al. [4] proposed modelling drag for fire suppression systems using a drag force proportional to velocity (Stoke's drag). However, compliance to practical applications or measurements is not evaluated. Furthermore, the protection radius of the designed system is significantly lower (18 meters) than the working range of fire monitors relevant for this thesis, and such drag models are therefore not considered applicable.

Holterman [48] suggested an empirical drag coefficient to be used in (2.12) which requires knowledge of Reynolds number $Re$, and therefore must be approximated with (2.63). Needless to say, $Re$ for a jet stream such as that presented in Figure 2.20 will not be constant and determination of a realistic value for the diameter $D$ would require multiple assumptions which are inherently uncertain. This drag model is hence regarded as irrelevant for the purposes of this thesis, although it may be of use for water sprays in agriculture, where the nozzles atomize the water to a greater extent and approximations on the size of the droplets may be more appropriately estimated.

$$Re = \frac{\rho \cdot \dot{r} \cdot D}{\eta} \tag{2.63}$$

Given the nature of propagation of a water jets in air, the present authors propose using a drag coefficient where the area of the jet is a function of the travelled distance. Typically, upon exit from the nozzle the jet has a relatively parallel travel as compared to the initial speed vector and an area approximately equal to the nozzle diameter. Eventually the jet will destabilize, break up and the area increases exponentially along with the drag force. Hence, the present authors hypothesize that using the travelled distance raised to some exponent may be an appropriate representation of the drag constant $k$ in (2.13). The expression for $k$ is given by (2.64), where $S$ is the distance along the jet. Note also that this makes the drag a function of time. With windless conditions, $\theta = 45°$, $\dot{r}_0 = 25 \cdot \sqrt{2}$, the constant $k = 0.1$, mass $m = 1$ and the exponent $n = 0.1$ the trajectory is as presented by Figure 2.22.

$$F_D = k \cdot S^n = k \left( \int \sqrt{y'(t)^2 + z'(t)^2}\, dt \right)^n \tag{2.64}$$

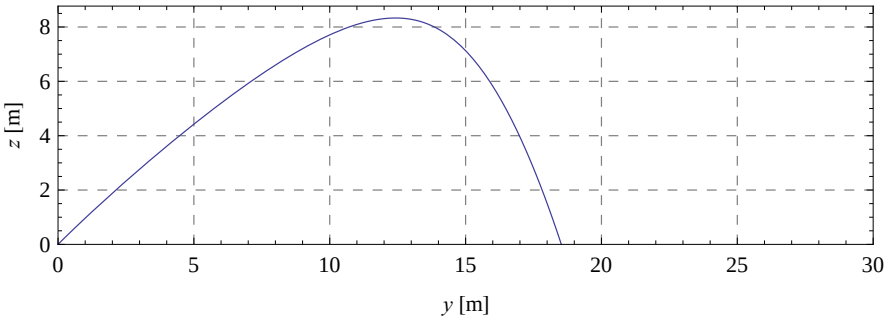



Figure 2.22: A sample trajectory with the proposed drag model.

A summary of all trajectory models is given in Table 2.2, where each model is assigned a number. Each model will hereafter be referred to as 'model x', where x is the assigned number. Note also that a classical drag model has been added for reference.

Table 2.2: Summary of trajectory models from open literature. Note that a classic drag model is added and will be used for reference.

| # | Source | Type of model | Model | Free parameters |
|---|---|---|---|---|
| I | Hatton et al. [16] | Drag force | $k \cdot \dot{r}^2 \left(1 + e^{b_0 \cdot S}\right)$ | $k$, $b_0$ |
| II | Hatton et al. [16] | Drag force | $k \cdot \dot{r}^2 \left(1 + a_0 \cdot S + a_1 \cdot S^2 + a_2 \cdot S^n\right)$ | $k$, $a_0$, $a_1$, $a_2$, $n$ |
| III | Hatton et al. [16] | Drag coefficient | As in (2.13), but with $k = C_{\mathrm{Da}}\left[(X-1)e^{-Y \cdot S} + 1\right]$ | $C_{\mathrm{Da}}$, $X$, $Y$ |
| IV | Present authors | Drag coefficient | As in (2.13), but $k$ is replaced by $k \cdot S^n$ | $k$, $n$ |
| V | Miyashita et al. [43] | Trajectory model | $z(x) = -\frac{7\left(3 \cdot \theta^2 - 2.86 \cdot \theta + 1\right)\left(0.0049 \cdot U^2 - 0.15 \cdot U + 1\right)}{10000 \cdot Q^{0.8}} x^3 - \frac{\left(1 - e^{\frac{1}{5}(-Q-0.1)}\right)\left(\frac{U}{20}+1\right)\sec^2(\theta)}{500 \cdot p} x^2 + \tan(\theta) \cdot x$ | None |
| VI | Hatton and Osborne [42] | Drag force | $(0.000273 \cdot Fr - 0.01)\,\dot{r}^2$ | None |
| VII | | Drag force | Turbulent drag, as in (2.13): $F_D = k \cdot v^2$ | $k$ |

### 2.3.2 Mechanisms of Jet Breakup

Although not the primary focus of this thesis, the breakup of water jets is a phenomenon which complicates accurate modelling of the trajectory, and thus deserves some basic reviewing. Here, a short discussion of some concepts and mechanisms regarding jet breakup will be held. For the purposes of this paper, the *jet breakup length* is the length along the jet at which the water jet is visibly coherent. Typically, a plot of the exit velocity versus the breakup length will appear as in Figure 2.23.

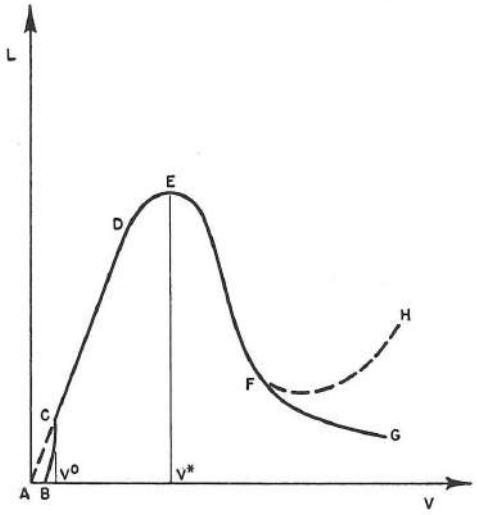



Figure 2.23: Typical shape of breakup curve, with breakup length on the vertical axis and exit velocity on the horizontal axis[49]. Dashed lines indicate areas where discrepancies may be observed.

The general mechanism of jet breakup is relatively straightforward: Turbulent and aerodynamic forces destabilize the jet as it emerges from the nozzle, eventually overpowering surface tension, leading to disintegration at some point. Clearly, the interaction between the fluid and the atmosphere has a significant impact on this phenomenon. The extent at which aerodynamic forces influences the jet trajectory is largely determined by the velocity and disturbances. For smooth (laminar) jets, the streams will not break up quickly since there are few surface disturbances. For turbulent jets, where there are irregularities on the surface, aerodynamic effects are more pronounced. It is therefore interesting to note that longer breakup lengths are often achieved with turbulent jet streams, and it may therefore be concluded that turbulence has a stabilizing effect on the jet stream (Figure 2.24). [45, 49–52]. As pointed out by Grant and Middleman [49], however, this statement may be confusing since this is with regards to *breakup length* and not *breakup time*, so the elapsed time it takes for a laminar jet to breakup may be longer than that of a turbulent one, but a turbulent jet may still have a longer breakup length. This may in part seem intuitive, since turbulence is related to speed. A jet which emerges at a high speed is obviously more likely to exhibit turbulent features, but possesses a larger momentum and may thus more easily break through the air, provided that it is sufficiently coherent. On the other hand, if the exit velocity is comparatively high, the drag forces lead to complete disintegration of the jet a shorter lengths, as illustrated by Figure 2.23.

It is only after the jet has become sufficiently disrupted that air effects have an appreciable impact on the disintegration process, although any jet will eventually disintegrate solely due to the interactions with the atmosphere[44, 53]. But for the purposes of minimizing the effects of wind on a jet, it may be advantageous for the flow to be fully laminar. Unfortunately, the nature of a jet stream is mostly governed by practical considerations, which in the context of firefighting with over relatively large distances, where high speeds are called for, is mostly turbulent[44]. Furthermore, as pointed out by Grant et al. [1], there exists a lower limit of water which must be sprayed onto a burning material in order to extinguish the flame, which means that the fire spray must have sufficient momentum to penetrate the fire and not vaporize before it reaches the source.

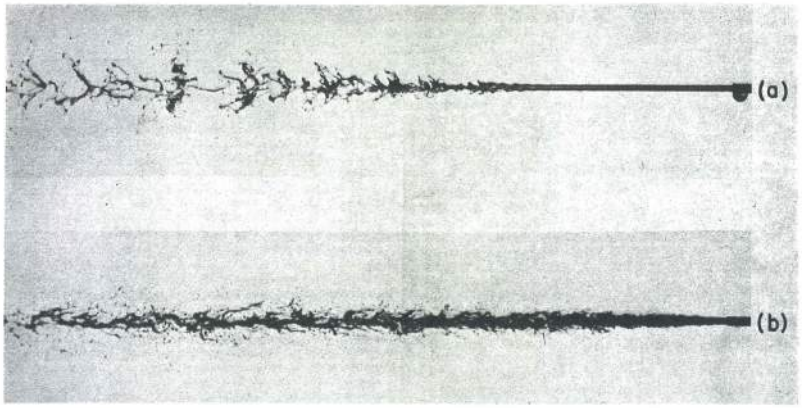



Figure 2.24: Comparison of laminar jet (a) and turbulent (b) under similar conditions[49].

Lee and Spencer [54] investigated the amount of jet disintegration on large quantities of photomicrographs, and showed that when other factors are constant, the following variables are reported to influence the degree of jet breakup:

- Increases with distance, assuming that the disintegration forces are not surpassed by viscous and capillary forces
- Increases with the density of air
- Increases with larger velocities and turbulence
- Decreases with increasing viscosity and surface tension
- Decreases with larger nozzle diameters

As first pointed out by Rouse et al. [44], however, the perhaps most influential factor with regards to the jet behaviour is the condition with which it emerges from the nozzle. As the water stream exits the nozzle it may be laminar, semiturbulent (turbulent core with laminar envelope) or turbulent, which greatly determines the aerodynamic forces on the jet. The design of the nozzle may introduce cavities, which in turn has a destabilizing effect. In addition, a coarse surface finish of the nozzle may introduce disturbances as well. The ladder point can be emphasized, perhaps counter to common intuition, by showing the effects the nozzle length has on turbulence. Longer nozzles produce more friction and therefore shorter breakup lengths are achieved (Figure 2.25). It should be mentioned that shorter nozzle exits does not necessarily correlate to longer breakup lengths, but experimental data suggests that there may be an 'optimum' nozzle length.[45].

The design of the fire monitor and the circumstances in which it is intended to operate in this thesis has been stipulated. It is therefore difficult to alter the breakup characteristics, although it may be argued that it is a challenging task with few changeable variables in the first place. However, it is important to appreciate the extent of the design's and other factors which impact flow behaviour and thus also the jet breakup characteristics. The objective of the present section is to clarify the difficulties in predicting turbulent forces on water jets and the limitations it may impose on trajectory models. Clearly, once a water jet is no longer coherent, the water particles will quickly disperse and a tangible position becomes difficult to determine. As a consequence, the jet breakup will ultimately affect a given trajectory model's accuracy. Finally, it must be emphasized that all the factors and how they interact during a jet's breakup is not fully understood[45].

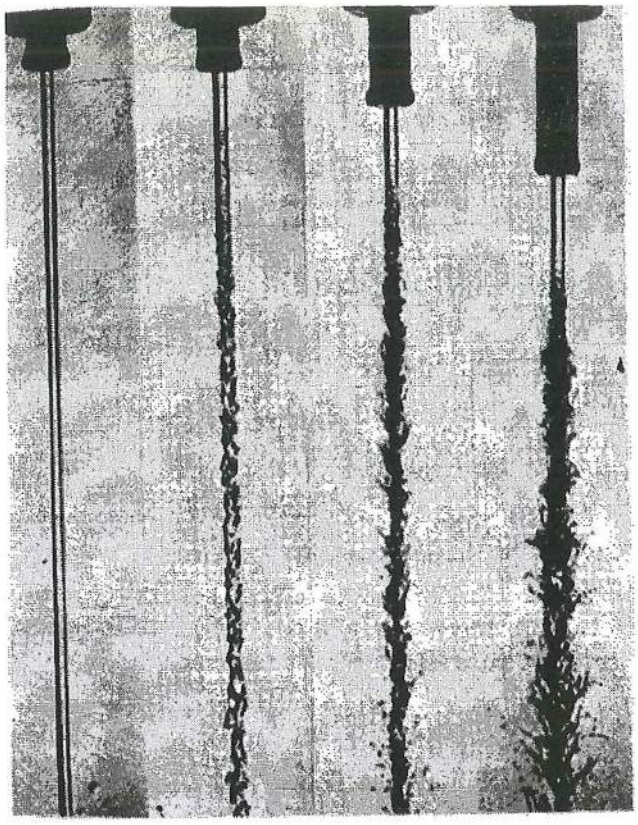


Figure 2.25: Comparison of flow characteristics with various exit lengths and conical nozzles[55].

### 2.3.3 IR Stereo Vision for Fire Localization

Little relevant literature has been found on the topic of fire localization. Most research available in the open literature is concerned with using cameras as fire detectors, most of which utilize color cameras (e.g. [9, 56–58]), and a some focus on IR cameras (e.g. [7, 18–20, 24, 59]). The cited papers are of limited relevance for the present thesis, since the stereo vision system here will be used for fire localization only. However, most of the aforementioned papers do utilize some imaging processing techniques which are applicable (Section 2.2.6), such as thresholding and erosion. Some also use analysis of flickering frequencies for object discrimination (e.g. [19, 60, 61]), which for a fire covers a band of 1-13 Hz[56].

Perhaps the most interesting paper with regards to the present thesis is the work done by Chen et al. [62], where a single CCD camera was used to estimate a fire's localization. The camera was fixed on a wall in a large hall, whereby fires were placed at known distances. The CCD camera estimated the distances to an accuracy of about 0.6 m at a horizantal distance of 20 meters. The relevance is limited, however, as the testing distances are not directly applicable to the scope of the present thesis, as well as the absence of IR cameras. Still, the results in [62] may serve as reference.

# CHAPTER 3

## Experimental Design

This chapter will describe the experiments which have been set up in order to solve the key problems in this thesis. Roughly speaking, the experiments are divided into the following categories:

1. Liquid jets (Section 3.1)
2. Vision (Section 3.2)
3. PLC and HIL simulation (Section 3.3)

The purpose of these experiments is the following:

1. To investigate to what degree it is possible to predict the trajectory of a liquid jet with and without wind disturbances. This will be in context with relevant literature review in Section 2.3.
2. To develop an IR stereo vision system which localizes a fire within the range specified in Chapter 1, and test the accuracy of the developed system
3. To develop and implement a control system on a PLC
4. To verify the system's performance as a whole with HIL simulations

### 3.1 Experiments on Liquid Jets

Two experiments have been set up with the purpose of measuring the landing points of a liquid jet, and thus determine the accuracy of the models presented in Table 2.2. The purpose of the first experiment is to determine the landing points of a liquid jet under the influence of wind, while the purpose of the second experiment is to determine the throwing distance with approximately quiescent conditions. The experiments, hereafter referred to as experiment I and II respectively, have been conducted on large open parking spaces in Kristiansand, southern Norway using a fire truck. A picture of the monitor used during experiments is given in Figure 3.1. The configurations of each experiment is summarized in Table 3.1, and weather conditions during experiments are presented in Table 3.2.

The experiments have been designed with the purpose of capturing the most practically applicable inclination angles, while at the same time attempting to capture the effects of tailwinds, headwinds and cross winds. The inclination angles which are thought to have the highest practical significance in this context is between 40 and 70 degrees, which corresponds to 20 and 50 degrees from the horizontal axis.

Table 3.1: Configurations of experiments.

| Experiment | $\theta$ [°] | $\varphi$ [°] | $Q$ [l/min] | $p$ [bar] |
|---|---|---|---|---|
| I | 40, 50, 60, 70 | 0, 45, 90, 135, 180 | 1200 | 10 |
| II | 45, 50, 55, 60, 65, 70 | | 1600 | 5 |

Figure 3.1: The monitor used during experiments.

Table 3.2: Weather conditions during experiments[63, 64].

| Experiment | Temperature[°C] | Weather conditions | Average relative humidity |
|---|---|---|---|
| I | ∼7 | Light rain | 92% |
| II | ∼8 | Clear | 77% |

The setup of experiment I is shown in Figure 3.2. For practical purposes, the anemometer and wind direction sensor is placed approximately 5 meters from the fire truck. A marker is placed at a known distance $L = 30$ m along the $y$ axis using a string which is pulled from the underneath the fire truck (Figure 3.3). The measurement method is shown in Figure 3.4, whereby the landing point of the jet is marked and the distances $a$ and $b$ are measured with tape measures. The angle $\alpha$ can then be calculated using (3.1) and the coordinates $x_i$ and $y_i$ can be computed using (3.2). A picture from experiment I is given in Figure 3.5.

$$\alpha = \arccos\left(\frac{a^2 + L^2 - b^2}{2 \cdot a \cdot L}\right) \tag{3.1}$$

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = a \begin{bmatrix} \cos(\pi/2 - \alpha) \\ \sin(\pi/2 - \alpha) \end{bmatrix} \tag{3.2}$$

Experiment II is conducted with comparatively little wind, and it is assumed that wind speed has negligible influence on the jet's trajectory, although the wind speed and direction are measured for documentation. The measurements of the jet's landing points are conducted in a similar manner as in experiment I, but since the direction of the jet is not affected by wind, the throw length $l$ can be determined directly (Figure 3.6). A picture from experiment II is presented in Figure 3.7.

The general measurement procedure for both experiments is as follows:

1. The monitor is set to the proper configuration

2. The pressure is ramped up to the working pressure which is recorded by an analog pressure gauge at the base of the monitor

3. The wind speed and direction is sampled and the landing point of the jet is marked before ramping down pressure (Figure 3.8)

4. Distance(s) are measured

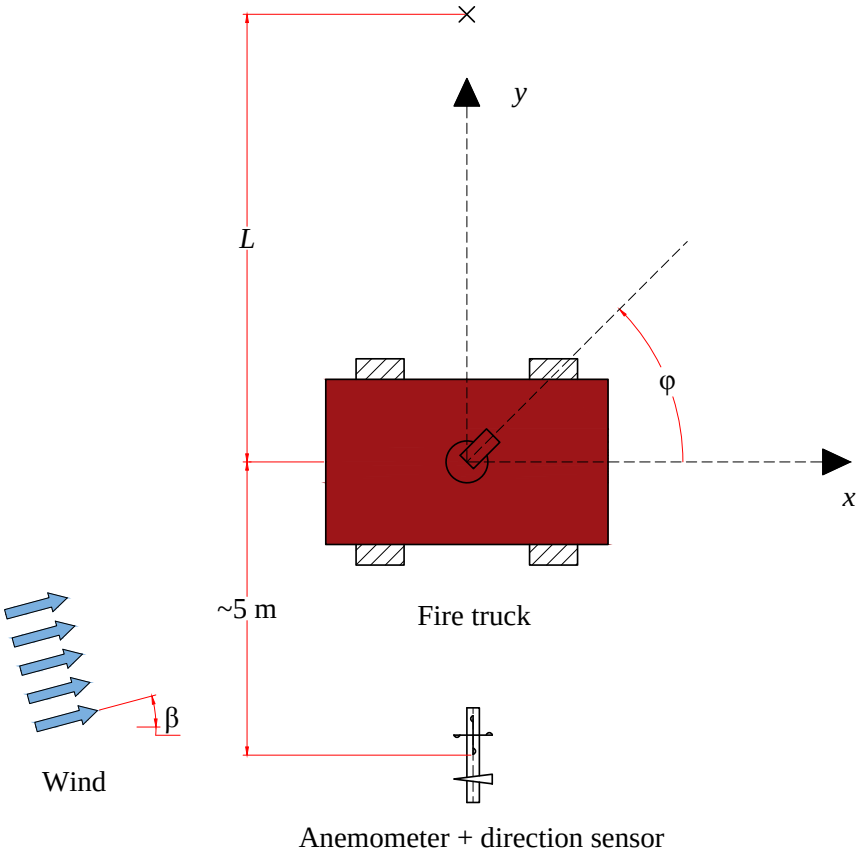



Figure 3.2: Setup of experiment I.

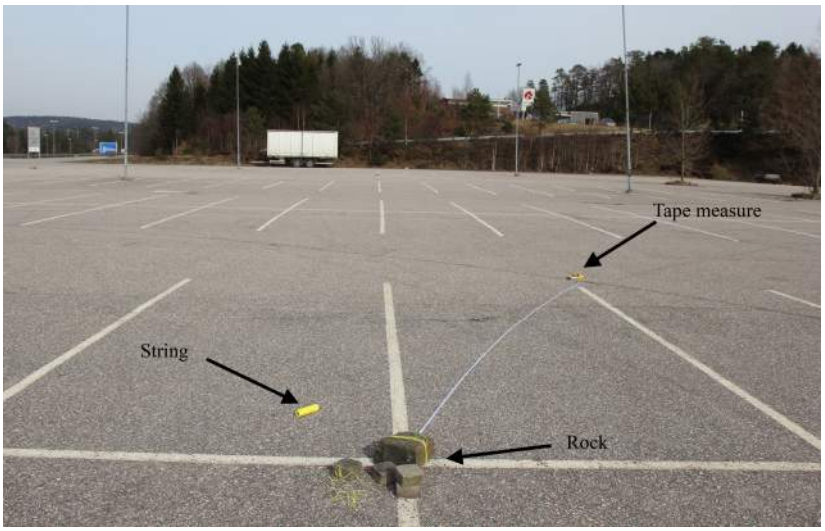



Figure 3.3: Measurement method.

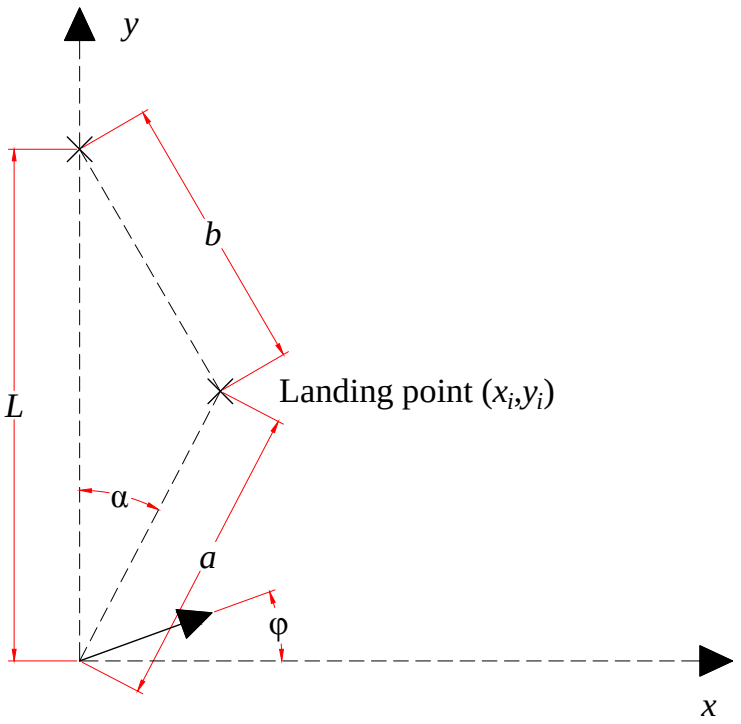



Figure 3.4: Measurement method during experiment I.

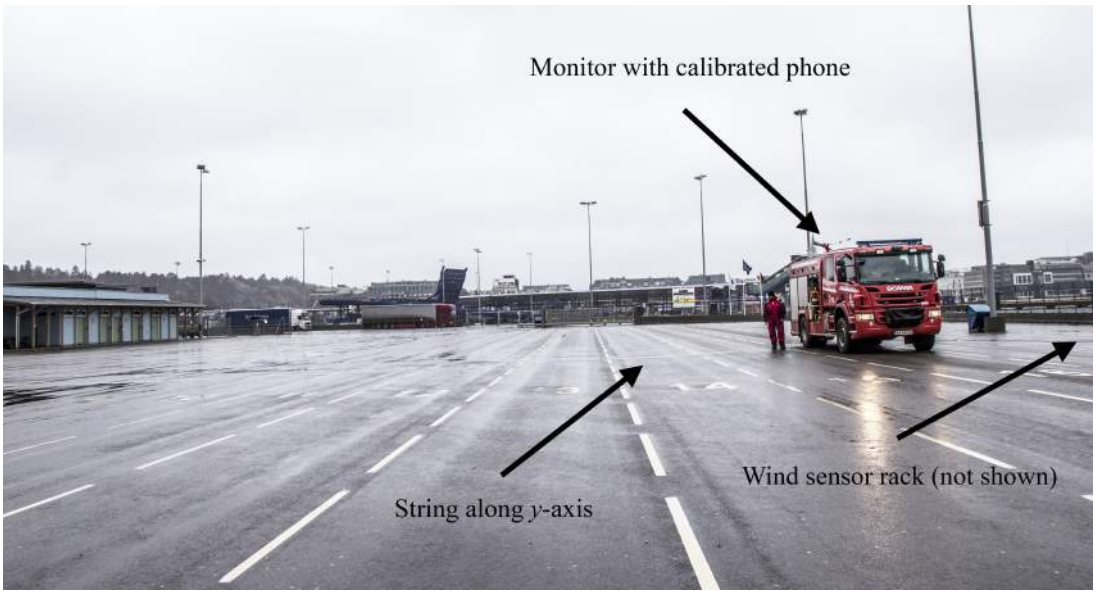



Figure 3.5: Picture from experiment I.

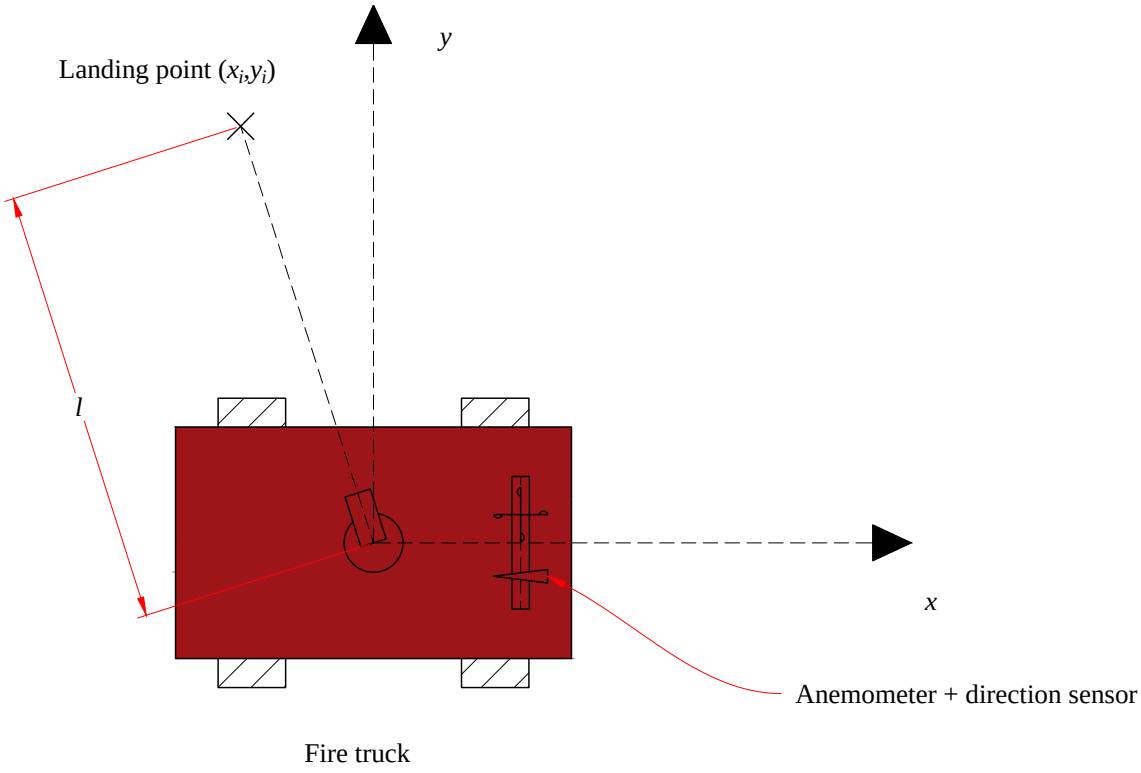



Figure 3.6: Setup of experiment II.


Figure 3.7: Picture from experiment II.


Figure 3.8: Measurement of landing points.

## 3.2 Stereo Vision

A simple test is devised in order to test the accuracy of the stereo vision system (Figure 3.9, 3.10). Since the most challenging aspect of a stereo vision system is to get sufficient depth accuracy, this is what the experiment will focus on[27]. Typically, the accuracy of a stereo system will be decrease as distance with higher distances due to the inverse proportionality with disparity (Equation 2.18).

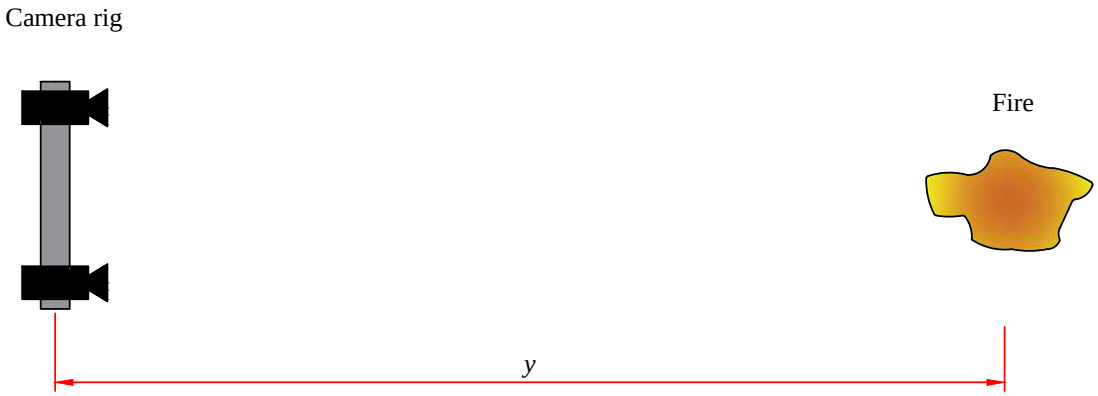



Figure 3.9: Stereo vision test.

The experiment will also test the object discrimination ability of the system by putting other heat sources in the FOV. This will be:

1. A person

2. A camping stove with a boiling pot of water, i.e. $\sim$100°C (Figure 3.11)

The values for $y$ during the experiment will be covering the working range of the system in 10 m increments, so the distance will be 30, 40, 50 and finally 60 m as a bonus.

Finally, the main source of a fire will in this case be a wooden fire with a size of approximately $30 \times 30$ cm $\approx$ 0.1 m$^2$ (Figure 3.12). The size of the fire is important since it relates to the accuracy of the IR3 sensor (Table 3.6).


Figure 3.10: Picture from stereo vision test.


Figure 3.11: Camping stove with pot of water.


Figure 3.12: Wooden fire with measuring tape (10 cm between main divides).

## 3.3 PLC & HIL Simulation

The purpose of the simulation experiment is to verify, at least in a virtual sense, the performance of the system as a whole. For this purpose, a virtual test scenario is set up, presented in Figure 3.13, which consists of three given areas ('zones'): A square-, rectangular- and a circular zone. The zones are distributed at 45°, 135° and 225° relative to the $x$-axis, whereby the circular area is at the innermost extremity of the working range, and the remaining two are situated at the outermost extremity. The idea is that, given an alarm signal in either of the defined zones, the system should be able to extinguish a fire in that zone.

Naturally, the system should be able to detect and extinguish a fire at any point within the working range, but this will be dependant on the configuration of the PLC program, which shall be reviewed in greater detail in Section 3.6.

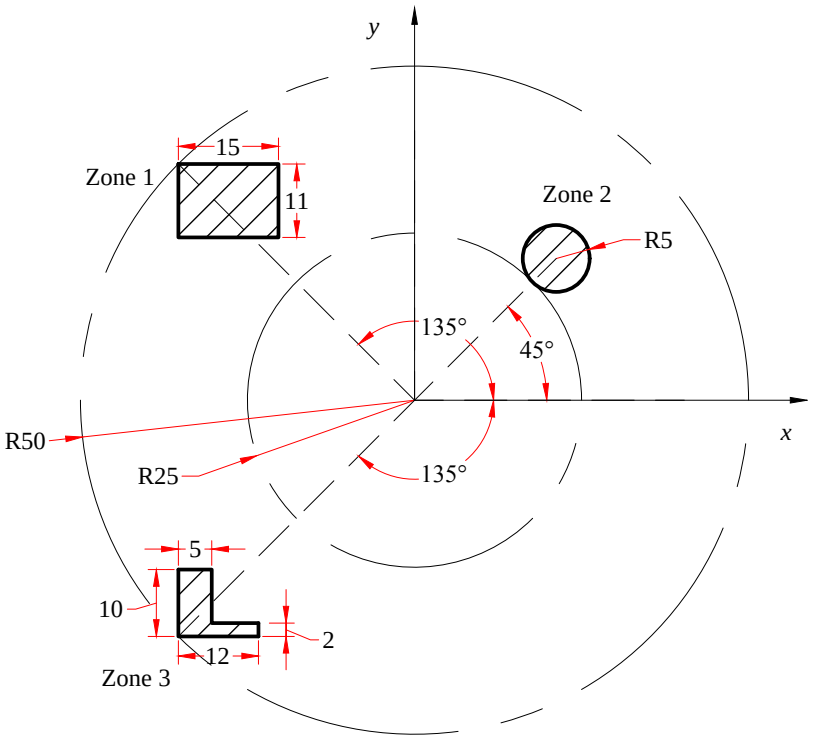



Figure 3.13: Defined zones. Units in m.

## 3.4 Requirements

The requirements of the system are listed in the following.

User requirements:

- The system shall be able to
  - Be operated manually, i.e. the configuration of the monitor and pressure source shall be manipulable electronically
  - Given an alarm signal in a given zone (e.g. smoke detectors), suppress fire in that zone
  - Automatically detect and suppress fire
- The system should have an emergency stop button

Technical requirements:

- 2×3 V DC power source for each IR camera (AA batteries)
- 2×220 V AC power outlet
- Computer with LabVIEW 2014 and Vision Development Module
- 2×frame grabbers, i.e. ADC with PAL-60 input format
- Custom camera rig (Appendix A) and tripod with 3/8-UNC head screw

Functional and non-functional requirements:

- When operated manually, the response shall be seemingly immediate
- User interface shall be orderly and clear
- User interface shall be operable, primarily, with buttons
- Graphical representation of current monitor configuration and setpoints, i.e. pressure, angles and landing point of jet

## 3.5 Design Specifications

This section will summarize the technical specifications of equipment used during the experiments in the present thesis. At the end of the section, a summary will be presented with the uncertainties in measurements. Note, however, that some of these uncertainties are due to human errors, e.g. calibration and initial setup, such that some of the uncertainties are obtained, admittedly, by mere estimation, but the presented errors should in the very least give some insight into what uncertainties are present during measurements presented in this thesis.

### 3.5.1 HIL Simulation

The computers used during HIL simulations are HP Compaq 8100 Elite Convertible Minitower desktop PCs with the specifications listen in Table 3.3.

Table 3.3: Computer specifications.

| Entity | Specification |
|---|---|
| Processor | Intel Core i5-660 3.33 GHz |
| RAM | 8 GB DDR3 1333 MHz |
| Operating system | Windows 7 64-bit |
| Network card | National Instruments GigE Vision Adapter |

The simulations are run in LabVIEW with the specifications listen in Table 3.4.

Table 3.4: Simulation software specifications.

| Entity | Specification |
|---|---|
| Simulation software | LabVIEW 2014 with Vision Development Module 2014 |
| Solver | Runge-Kutta 4 |
| Step time | 1 ms |

Specifications of the PLC and programming application used are given in Table 3.5.

Table 3.5: PLC specifications.

| Feature | Specification |
|---|---|
| Model | Siemens SIMATIC ET 200S |
| Cycle time | 10ms |
| Communication | PROFINET |
| PLC programming | TIA Portal V12 |

Table 3.6: Specifications for the IR3 sensor.

| Feature | Specification |
|---|---|
| Manufacturer | Simtronics |
| Model | MultiFlame 3xIR Long Range |
| Operational range (0.1 $m^2$ fire) | ≤65 m (gasoline fire), ≤80 m ($n$-heptane fire) |
| Horizontal FOV | 104° |
| Response time | 3.0 s |
| SIL certificate | SIL3 |

### 3.5.2 Cameras and Sensors

The specifications of the IR3 sensor is presented in Table 3.6.

The specifications of the IR cameras, anemometer and wind direction sensor are given in Table 3.7-3.9 respectively. This equipment is in turn fastened to a Manfrotto 190XPROB tripod.

Table 3.7: Camera specifications

| Feature | Specification |
|---|---|
| Supplier | Sun Creative Technologies Inc. |
| Model | M700 |
| Resolution/pixel pitch | 384×288/17$\mu$m |
| Interface | RS232 |
| Output | Analog, RCA cable |
| Power supply | 3 V DC |
| Spectral range | 8∼14 $\mu$m |
| Lens | 19mm/f 0.9 |
| Horizontal FOV | 20° |
| Vertical FOV | 15° |
| Sensitivity | <80mK at f/1.0 and 300K |

The analog output from each IR camera is captured and converted with frame grabbers with specifications listed in Table 3.10.

Sensory signals are acquired using a National Instruments USB-6008 multifunction data acquisition (DAQ) device, with specifications listed in Table 3.11.

An iPhone (Table 3.12) will be used to measure the angles $\varphi$ and $\theta$ using an integrated microelectromechanical (MEMS) gyro. In order to test the error in measurement, a simple experiment is set up (Figure 3.14). The angle $\delta$ can be calculated as

$$\delta = \arcsin\left(\frac{K_2}{K_1}\right) \tag{3.3}$$

For an angle of 15.8°, the iPhone computed 16°. The resolution, however, is 1°. The total uncertainty in angular measurements is therefore concluded to be about 0.5°. However, this does not account for errors

Table 3.8: Anemometer specifications.

| Feature | Specification |
|---|---|
| Supplier | Opto-Electrical Technology Co., Ltd. |
| Model | FY-FS wind speed sensor |
| Type | 3 arm cup |
| Measuring range | 0∼70 m/s |
| Start wind speed | <0.3 m/s |
| Output/interface | 0-5 V DC |
| Power supply | 12 V DC |
| Uncertainty in measurement | ±3 % of measurement |

Table 3.9: Wind direction sensor specifications.

| Feature | Specification |
|---|---|
| Supplier | Opto-Electrical Technology Co., Ltd. |
| Model | FX1 Wind Direction Sensor |
| Measuring range | 0°-360° |
| Dead angle | 5° ± 1° |
| Output | 0-5 V DC |
| Power supply | 12 V DC |
| Uncertainty in measurement | ±3° |

Table 3.10: Frame grabbers specifications.

| Entity | Specification |
|---|---|
| Manufacturer | Pinnacle |
| Model | Dazzle Video Capture DVC100 |
| Inputs | Composite video (yellow RCA plug), S-Video, Stereo audio (red/white RCA plugs) |
| Resolutions | From $160 \times 120$ up to $720 \times 526$ |
| Frame rate | 25 or 30 FPS |

Table 3.11: DAQ device specifications.

| Feature | Specification |
|---|---|
| AI resolution | 12 bits differential, 11 bits single-ended |
| Maximum AI sample rate, single channel | 10 kS/s |
| Maximum AI sample rate, multiple channels (aggregate) | 10kS/s |
| DIO configuration | Open collector |

in calibration. In essence, the azimuth angle $\varphi$ is calibrated with eyeballing when placing the equipment (Figure 3.2, 3.6), which may lead to a slight angular offset with respect to the coordinate system. The absolute uncertainty in $\varphi$ is therefore assumed to be in the order of 5°. $\theta$ is zeroed by using a level, and the uncertainty is therefore smaller, although the calibration is still done by hand. The absolute uncertainty in $\theta$ is consequently assumed to be in the order of half the $\varphi$ uncertainty, namely 2.5°.

Table 3.12: iPhone specs.

| Feature | Specification |
|---|---|
| Model | Apple iPhone 5s |
| Operating system | iOS 8.3 |
| Application | 'Gyroscope', developer: Acrossair |

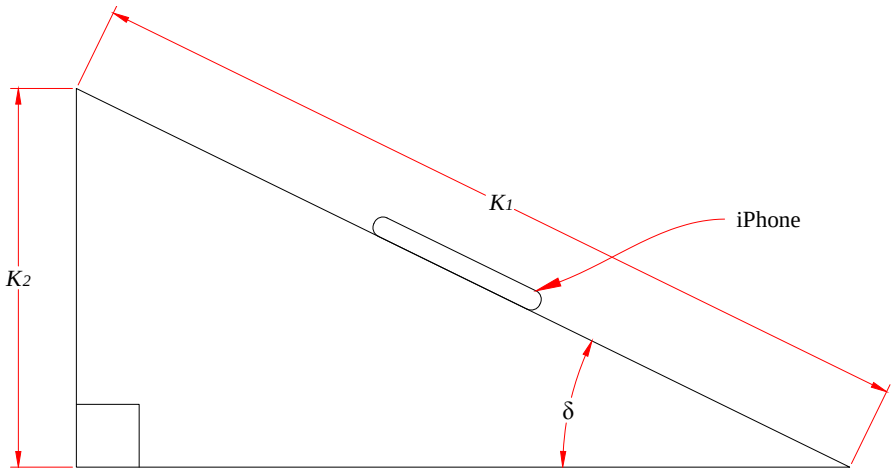



Figure 3.14: Test of iPhone accuracy.

The uncertainties in measurement are summarized in Table 3.13. Note that an additional 5° has been added to the uncertainty in $\beta$ due to error in initial positioning relative to the coordinate system (Figure 3.2, 3.6), which is due to the same considerations taken into account when estimating the error in $\varphi$.

Table 3.13: Summary of uncertainties in measurements.

| Parameter | Uncertainty in measurement |
|---|---|
| $U$ | 3 % of measurement |
| $\beta$ | ±8° |
| $\theta$ | ±2.5° |
| $\varphi$ | ±5° |

## 3.6 Implementation

### 3.6.1 Experiments on Liquid Jets

#### Sensor Rig & Data Processing

A custom rack is made in order to fix the anemometer and wind direction sensor (Appendix A). The sensors are bolted onto a steel plate with an arbitrary distance between, although sufficient to avoid direct contact. The steel plate is in turn mounted on a tripod (Figure 3.15).

Figure 3.15: Sensor rack with 2 m yardstick.

The anemometer and wind direction sensor are connected to a 12 V DC power supply, and the outputs are acquired using a multifunction DAQ, which acquires and converts the analog voltage signals. See Figure 3.16.

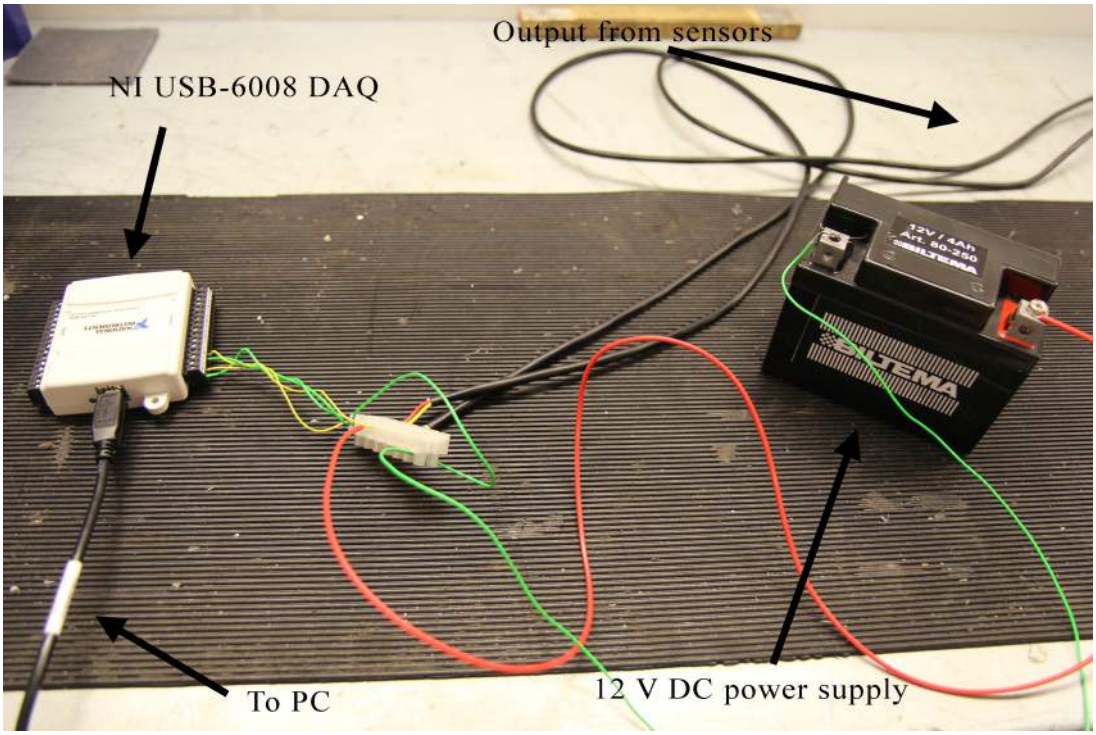



Figure 3.16: Implementation of anemometer and wind direction sensor.

The anemometer and wind direction sensor used to measure wind speed $U$ and direction $\beta$ both yield an output voltage between 0 and 5 V which is proportional to the process variable. The values are recorded instantaneously from real time sampling in LabVIEW (Appendix B).

The recorded voltages are filtered with an averaging filter, which is set to calculate the running mean from the past 18 and 25 samples for the wind direction and wind speed respectively, whereby the average sampling frequency is 70.9 Hz for experiment I and 83.9 Hz for experiment II. The parameters used during calculation of the mean have been found by experimentation, such that stable outputs are achieved, but with a seemingly immediate response. Let the $i$th sample of a process value be denoted $x_i$ up to $n$ samples, then the mean $\mu$ is computed as

$$\mu = \frac{1}{n} \sum_{i=0}^{n-1} x_i \tag{3.4}$$

The angles $\theta$ and $\varphi$ are measured using an iPhone and the built in MEMS gyroscope. The phone is calibrated using a level (Figure 3.17). The phone is taped onto the side of the monitor in a horizontal position before being zeroed and the inclination angle of the monitor is adjusted. For experiment I, the azimuth angle $\varphi$ is calibrated with eyeballing, i.e. the direction of the monitor is set in the positive direction of the $x$-axis by visual inspection.

Figure 3.17: Calibration of iPhone.

### Optimizing Trajectory Model Parameters

Based on the landing points and boundary conditions found during experiment I and II, an optimization process is carried out in order to identify the best-fit parameters of the models presented in Table 2.2. This process includes model I-V, as well as model VII for reference. Model VI is omitted, since it requires knowledge of the Froude number (Equation 2.45) which is valid for a solid jet. Model V will only be tested against data from experiment II, since it only allows for one-dimensional wind disturbances.

The optimization process is done by minimizing the sum of euclidean distances between the measured- and predicted landing points of a given model. Let $\hat{x}_i,\ \hat{y}_i$ denote the measured landing points, $x_i,\ y_i$ denote the predicted landing points of each model for $N$ measurements, then the minimizing function $G$ becomes

$$G = \sum_{i=1}^{N} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} \tag{3.5}$$

And the average error $AE$ of each model is calculated as

$$AE = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} \tag{3.6}$$

The parameter identification process is carried out in MATLAB (Appendix C), where a genetic algorithm is used. Genetic algorithms are evolutionary, stochastic search techniques based on the mechanisms of natural selection[65, 66]. In general, the algorithms start with an initial set of random solutions in a 'population' set. Each individual is a 'chromosome' which is a potential solution. The chromosomes 'evolve' through each iteration, or each 'generation', after which the chromosomes are measured to some 'fitness' level. The next generation is formed by merging two existing chromosomes (crossover) or by mutating an existing one. The population size is kept constant by rejecting chromosomes with the lowest measure of fitness. This process is repeated until the solution converges, i.e. the change between generations is sufficiently small, as compared to the most fit chromosome. A flowchart of the general optimization process using genetic algorithms is presented in Figure 3.18.

After the minimization process using the genetic algorithm is complete, the parameters are used as initial guesses in a gradient based minimization algorithm to determine the minimas with more accuracy. This is done in part due to the fact that, although the genetic algorithm generally is robust with regards to avoiding local minima, on the other hand it does not pinpoint the exact global minima.

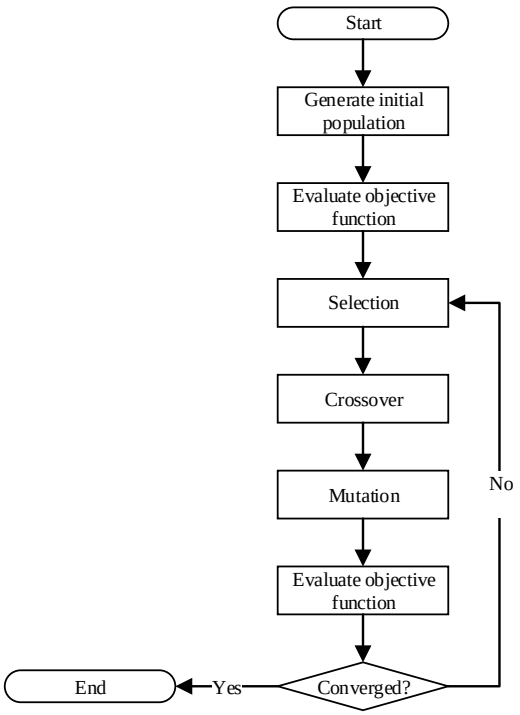



Figure 3.18: A flowchart illustrating the genetic algorithm optimizing process.

### Constraints

In order to ensure that meaningful results are obtained, proper constraints during the optimization process is crucial. In this specific case, it is imperative that the basic form of (2.11) holds, i.e. that the drag force always opposes the motion. With that in mind, the constraints, at least in principle, can be constrained as shown in Table 3.14.

Table 3.14: Constraints during optimization process.

| **Model #** | **Constraints** |
|---|---|
| I | $k > 0$, $-\infty < b_0 < \infty$ |
| II | $c > 0$, $a_0 > 0$, $a_1 > 0$, $a_2 > 0$, $-\infty < n < \infty$ |
| III | $1 < X < \infty$, $C_{Da} > 0$, $-\infty < Y < \infty$ |
| IV | $k > 0$, $n > 0$ |
| VII | $k > 0$ |

The size of the constraints during an optimization process is a decisive factor in terms of the required computing power, and therefore the constraints listed in Table 3.14 are, of course, not limited enough. Essentially, setting proper constraints is a trade-off: Too general constraints may results in unreasonably long computation times, while too narrow limitations may miss out on the best results. Furthermore, since the minimizing function is fairly complex, the size of the interval in which the optimization parameters are allowed to reside, will exponentially increase computation time by itself. As a rule, the constraints are first set by an initial guess, guided by general intuition, and, based on the initial results, the constraints are refined. The general procedure is summarized by Figure 3.19.

The parameter identification process is carried out in two different ways:

- With parameters in all models independent in each direction, e.g. drag parameters different in the $x$

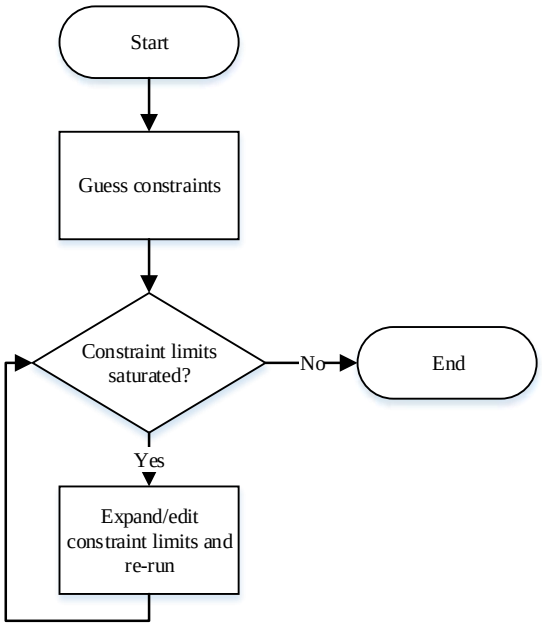



Figure 3.19: General optimization procedure.

and $y$ direction, hereby denoted 'non-uniform' parameters

- In the usual way, i.e. with parameters equal in all directions, hereby referred to as 'uniform' parameters

This is done in order to examine whether it is possible to improve the results by introducing some extra leeway in each model. However, in doing this, the behaviour of each parametric solution is modified relative to each other, and some abnormal behaviour in the results is therefore to be expected. Moreover, the minimization process is considerably complexified, leading to a three-fold increase in optimization parameters. But still, considering the importance of a sufficiently accurate model, this step is carried out.

### Determination of Initial Velocity

The velocity of the jet as it leaves the nozzle is crucial with regards to throw length. Therefore, steady state computational fluid dynamics (CFD) analyses are carried out using Autodesk Simulation CFD in order to provide useful approximations. The simulations are run until the automatic detection methods in the software detects convergence. Summaries of the settings of the CFD analysis are given in Table 3.15-3.17.

The boundary conditions are set up as shown in Figure 3.20. The inlet pressure and flow is set to 5 bar and 1600 l/min respectively in accordance with experiment II, while the material of the nozzle body and fluid is set to steel and water in the given order.

The initial speed may be estimated by Bernoulli's principle, as is done in [43] (Equation 3.7). However, this does not take into account frictional losses and the discharge coefficient of the monitor. Still, it is assumed that the speed of the fluid at the nozzle outlet follows the same basic relation, and is thus proportional to the square root of the pressure differential (3.8). What remains is determination of the proportionality constant.

$$\dot{r}_0 = \sqrt{\frac{2}{\rho} \cdot \Delta \mathrm{p}} \tag{3.7}$$

$$\Downarrow$$


Figure 3.20: CFD model of fluid volume from the fire monitor with boundary conditions.

$$\dot{r}_0 \propto \sqrt{\Delta \mathrm{p}} \tag{3.8}$$

The exit speed is essentially determined by probing the speed component which is parallel to the longitudinal direction of the nozzle (Figure 3.21). The probed values are near the walls of the nozzle, which is where the speed is expected to be highest. Naturally, since the end result of the CFD analysis depends on probing, there is some inherent error. After a few tests, the variability in probed speeds are observed to be about 1 m/s.

Table 3.15: Physics settings of CFD analysis.

| Variable | Setting |
|---|---|
| Flow compressibility | Incompressible |
| Heat transfer | Off |
| Auto forced convection | Off |
| Gravity | Off |
| Radiation | Off |
| Scalar | No scalar |
| Turbulence | On |

As pointed out in Chapter 1, it should be stressed that the monitor used during experiments and the one used in the CFD analysis (Figure 1.2), is not the same. However, the difference between the two is assumed to be negligible.

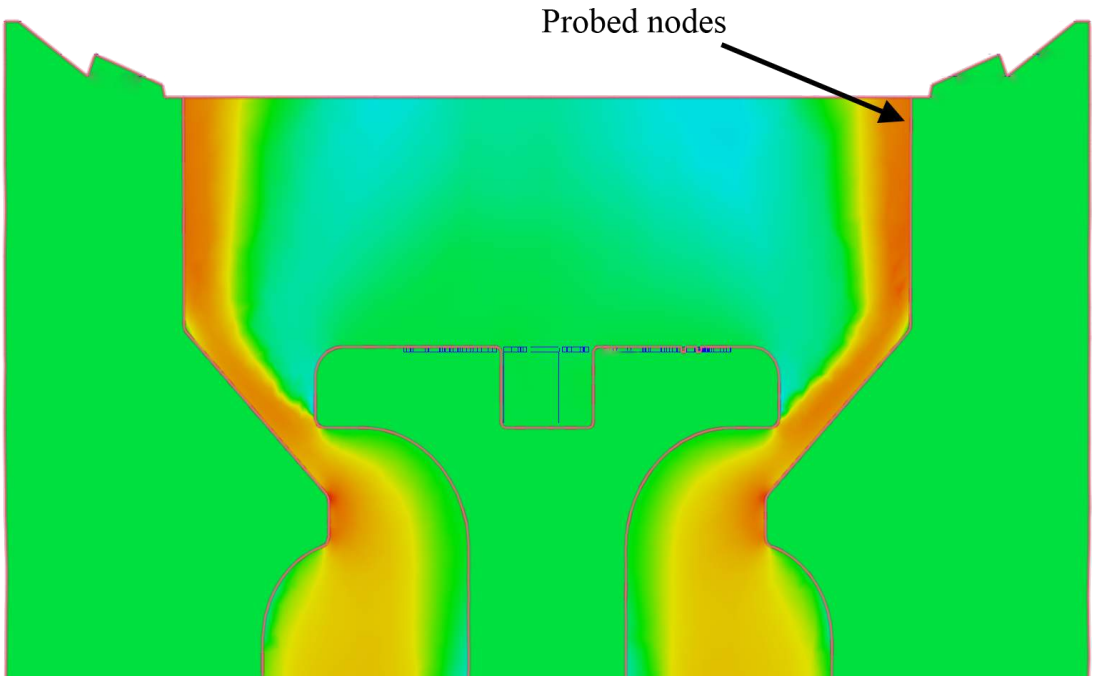



Figure 3.21: Contour plot of vertical speeds on the CFD model and approximate location for probed nodes.

Table 3.16: Solver settings of CFD analysis.

| Variable | Setting |
|---|---|
| Solution mode | Steady state |
| Intelligent solution control | On |
| Turbulence model | $\kappa$-$\varepsilon$ |

Table 3.17: Mesh settings of CFD analysis.

| Variable | Setting |
|---|---|
| Surface refinement | 0 |
| Gap refinement | 0 |
| Resolution factor | 1 |
| Edge growth rate | 1.1 |
| Minimum points on edge | 2 |
| Points on longest edge | 10 |
| Surface limiting aspect ratio | 20 |
| Mesh enhancement | 1 |
| Enhancement blending | 0 |
| Number of layers | 3 |
| Layer factor | 0.45 |
| Layer gradation | 1.05 |

### 3.6.2 Stereo Vision

#### Camera Calibration

As previously discussed in Section 2.2.5, camera calibration is a necessary step in order to improve the accuracy of a stereo vision system. In this thesis, this will be done using a standard checkered pattern (Figure 3.22) in MATLAB using the *Camera Calibrator* toolbox[67].

The main difficulty in this thesis with regards to calibration is the fact that the cameras are IR. This is challenging because camera calibration requires that the edges of the calibration template is easily detectable, and thus one requires sufficient contrast in the images. That being said, this is not straightforward since the images acquired from an IR camera is clearly dependant on the amount of IR radiation it receives, and not the color of the calibration template or reflected visible light. In the literature it has been suggested to replace the checkered grid with a pattern of heat resistors[32], replacing the black squares in Figure 3.22 with a material which reflects a heat source toward the camera[68], putting a metal grid in front of a heat source[25], or make a custom calibration template with heated elements[23]. All of the mentioned methods should, in theory and based on results in the cited papers, work in their own regard.

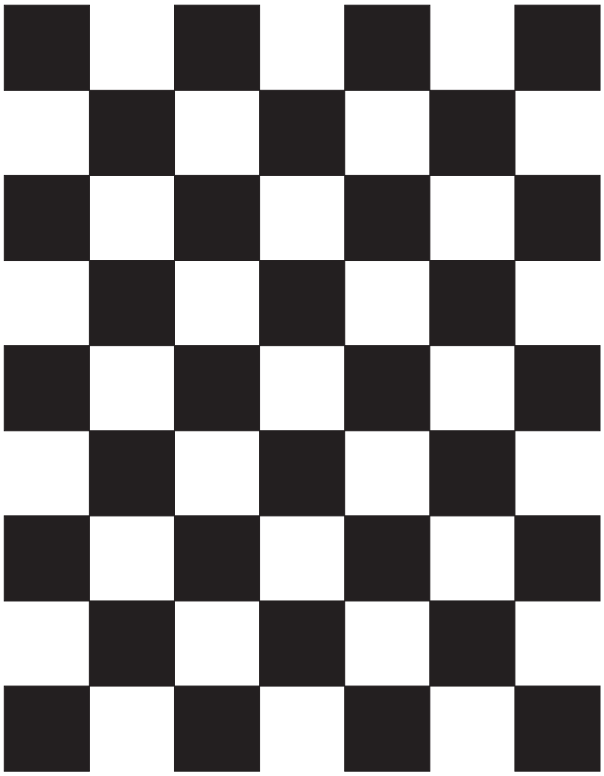


Figure 3.22: Chess pattern.

In the present thesis, however, the authors would like to suggest using a more straightforward approach, which entails using the standard checkered pattern with a low-emissivity material glued to surface of the calibration template. In this case, the calibration template is setup as shown in Figure 3.23, using a cardboard plate as basis with pieces of aluminium foil as reflective insulation.

The advantage of doing it this way is that the standardized calibration pattern can be used directly with the toolbox in MATLAB. It is also cheap and convenient. The downside is that the aluminium foil squares are cut out by hand, and thus require tedious manual work which will have to be done with high precision. Since calibration requires precise knowledge of the calibration template's geometry, an inaccurately prepared template may lead to a sub-optimal calibration. What is more, if one wishes to use a calibration template with circles instead of squares, this may be difficult to do with sufficient accuracy. Nevertheless, once an

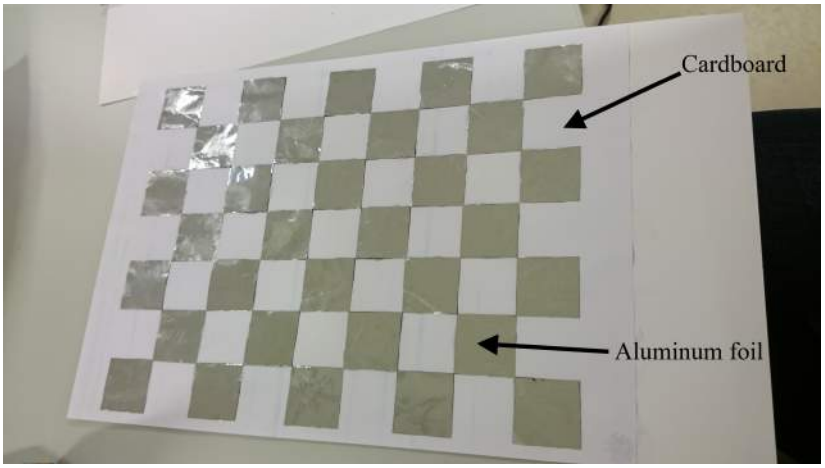



Figure 3.23: Calibration template.

adequate template is prepared, the rest of the process should be convenient.

With the calibration template in place, one can move ahead with the calibration process, which in MATLAB is best achieved by taking 10-20 pictures. The software then relates the corners of the calibration pattern in world coordinates and calculates the extrinsic and intrinsic parameters of the camera, as discussed in Section 2.2.5 (See Figure 3.24 for further illustration). During the calibration process with the presented calibration template (Figure 3.23), one needs to heat the cardboard plate sufficiently such that the aluminium squares are clearly visible. During the calibration process in the present thesis, this is done using a roasting pan filled with hot water (Figure 3.25).

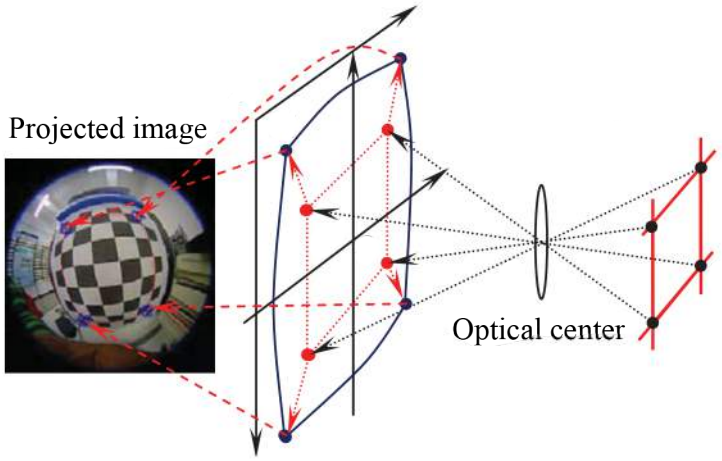



Figure 3.24: Camera calibration with the pinhole model. Figure adapted from [69].

A picture from the calibration process in MATLAB is presented in Figure 3.26, which shows the detected corners on the calibration template, the location of the pictures taken relative to the calibration template and the overall mean error, which is the euclidean distance in pixels between a keypoint detected in an image, and a corresponding world point projected onto the same image (Figure 3.27). The reprojection error is thus a quantitative measure of calibration image quality.

Once the intrinsic camera parameters from the calibration process in MATLAB are computed, these are implemented in LabVIEW using a set of block diagrams which rectifies the images taken in real time (Appendix B). An example image is presented in Figure 3.28.


Figure 3.25: Picture from calibration process.

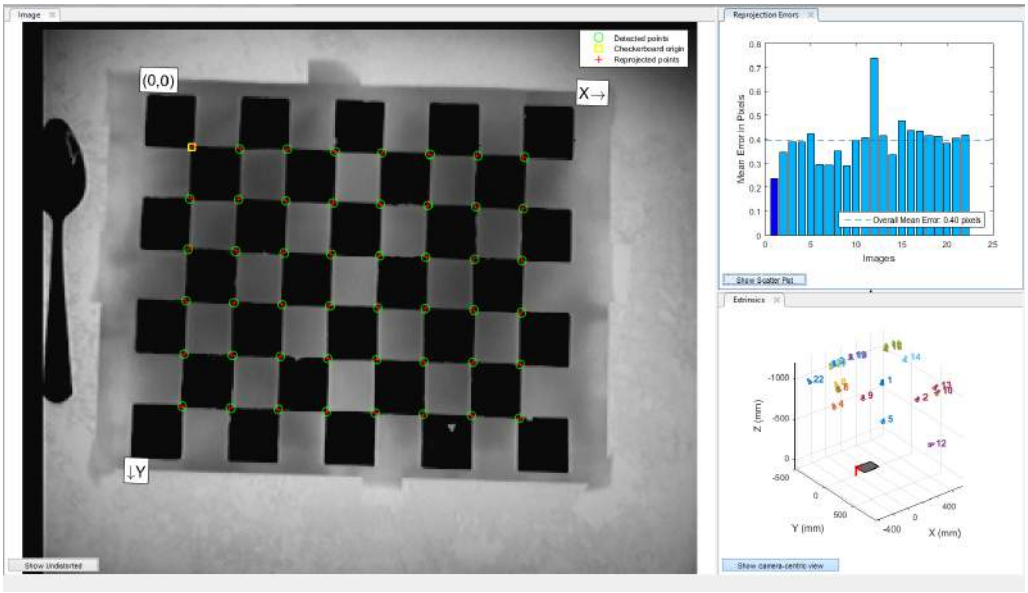



Figure 3.26: Camera calibration in MATLAB.

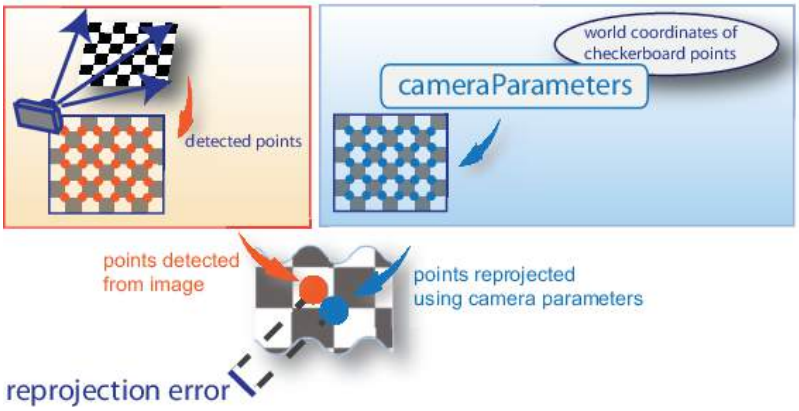



Figure 3.27: Illustration of reprojection error.

(a) Input image. (b) Undistorted image.

Figure 3.28: Image calibration.

### Image Processing

The image processing techniques used in order to determine the fire's position will be described here. A visual summary of the image processing steps is presented in Figure 3.29 and 3.30. In general terms, the image processing steps are divided into the following sequence:

- **Image acquisition:** The analog video signal is digitized by the frame grabber which is set up to capture 25 FPS at the cameras' resolution $352 \times 288$ pixels. The images are stored in 32-bit RGB format, although the images are technically grey-scale.

- **Image rectification:** The parameters determined during the calibration procedure in MATLAB are used to rearrange the image and rectify it with regards to tangential- and radial distortion as well as eccentricity in the principal point.

- **Intensity extraction:** The intensity of each pixel in the 32-bit image is extracted in order to create a 8-bit image and reduce the size.

- **Thresholding:** In these images, the hottest objects are the ones that appear the brightest. Therefore, a thresholding algorithm is applied in order to excrete the points of interest and reduce the image to a binary image. The initial thresholding value is set to 225, which, in the experiments conducted in the present thesis, is typically a little too high, and the value will often therefore be reduced somewhat by the algorithm (Figure 3.29).

- **Small particle removal:** After the thresholding operation is applied, there may be small 'hot-spots' left in the image. An algorithm is therefore applied which, based on the relative size of the non-zero areas in the binary image, erodes the size of the smaller areas, thereby removing smaller objects.

- **Particle analysis:** This step counts the number of coherent areas in the image. If there are more or less than one area left in the image, the thresholding value is adjusted accordingly and the imageing process loops back to the thresholding algorithm. Once there is a single non-zero area left in the image, the centroid of that area is calculated in terms of pixel coordinates.

- **Disparity computation:** The disparity is now calculated by subtracting the $u$ coordinate of the centroid found in each camera's frame of reference. The distance to the fire is then computed using (2.18) before an averaging filter is applied to reduce noise.

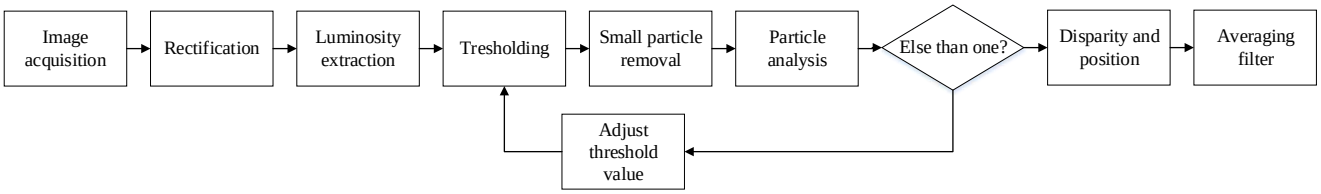
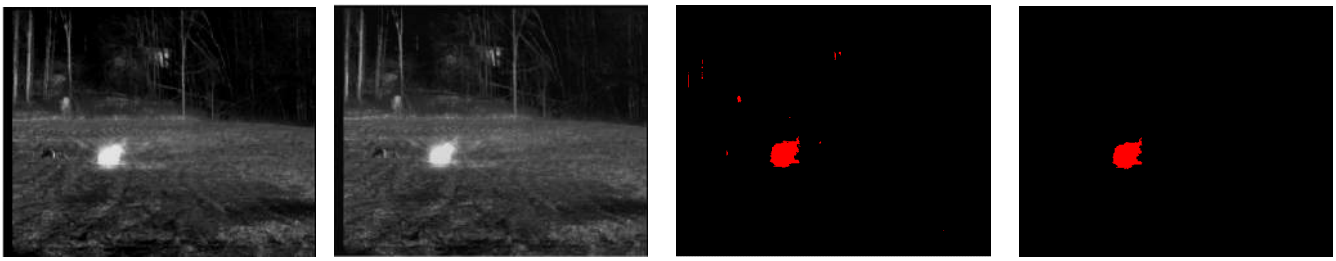



Figure 3.29: Threshold adjustment.



(a) Image acquisition. (b) Image rectification. (c) Thresholding. (d) Small particle removal.

Figure 3.30: Illustration image processing steps with example image.

### Camera Rig

A custom camera rig is made for the IR cameras (Appendix A, Figure 3.31). The IR cameras are fitted to a similarly designed rack as the wind sensor rack, but with the added option of adjusting the baseline between the cameras. The distance between the cameras must be carefully determined, since a large baseline is essential in order to achieve higher depth-accuracy, but should still be sufficiently small such that the cameras' FOV overlap, and key features are visible in both cameras. That being said, the baseline should be unreasonably large in order not to have an overlapping FOV for the working range specified in this thesis, so the main focus in the design of the camera rig is to determine a baseline which yields sufficient accuracy within practical reason.

In this design, the camera rig is designed with a baseline of 1.2 m. Inserting numerical values from Table 3.7 and solving (2.18) for the disparity $d$ at the extremities of the working range (25 and 50 m), one obtains

$$d \in [26.824, 53.647] \tag{3.9}$$

The relationship between the disparity and distance is presented in Figure 3.32, where each point represents an integer integer value of $d$ to indicate the change in resolution, which is approximately a three fold decrease at long range.
The IR cameras are in turn connected to a computer with LabVIEW via frame grabbers (Figure 3.33). The cameras also come supplied with manual brightness/contrast adjustment controls, but these have not been used in the present thesis.

Figure 3.31: Camera rack with 2 m yardstick.

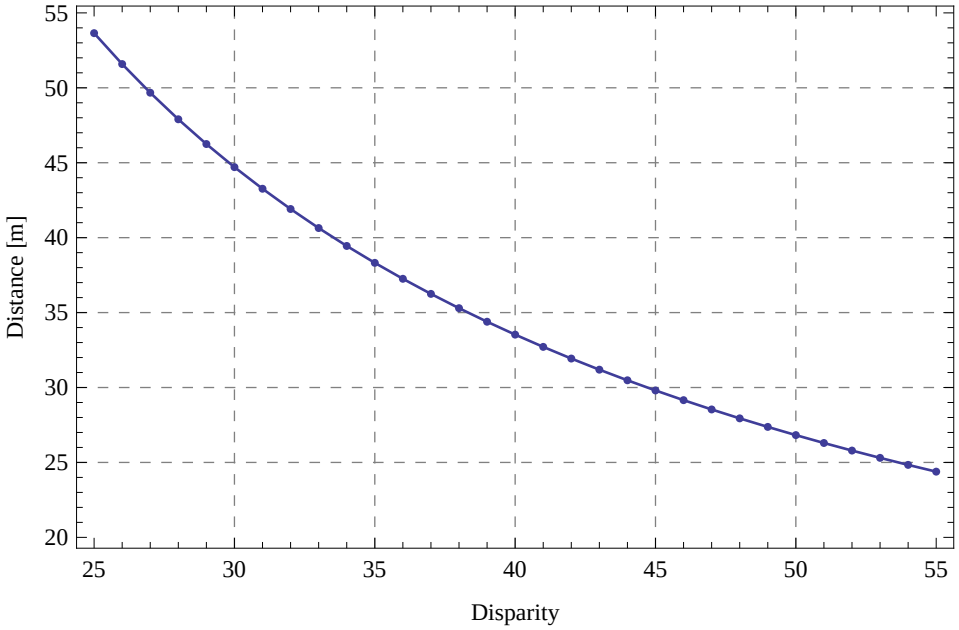



Figure 3.32: Disparity versus distance.

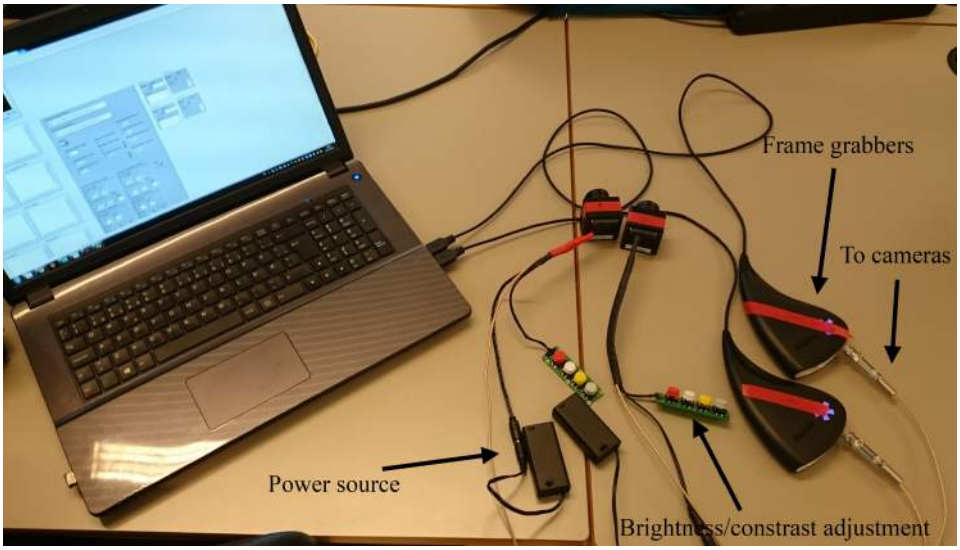



Figure 3.33: IR cameras and PC.

### 3.6.3 PLC

The structure of the PLC program (Appendix D) which controls the fire monitor is shown in Figure 3.34 with the defined transitions (marked 'T') and functions (marked 'F') which form the basis of the program. To begin with, the system is in a passive state, until the operator activates one of three binary inputs which initiates one of the control system's three parts (modes):

1. **Manual mode:** This mode activates a user panel with buttons that allows the operator to manually steer each actuator on the monitor individually. This includes motor 1 and 2 (Figure 1.2) as well as initiate the pressure source and modulate the pressure between $0 \dots 10$ bar. In essence, this mode is programmed based on binary inputs from the user panel, where buttons are assigned to increase or decrease the angular positions $\theta$ and $\varphi$. When a given button is pressed, small increments are added or subtracted such that a displacement $\Delta\varphi = 90°$ is done in 10 seconds and a displacement of $\Delta\theta = 30°$ in 10 seconds. The pressure source is controlled with an analog input. The user panel will be described in more detail shortly.

2. **Zone protection mode:** This mode is activated by external fire alarms in predefined zones. If an alarm signal in a given zone is received, the monitor will follow a set of predefined setpoints to extinguish the fire located in the zone. The specifics of the extinguishing sequence will be reviewed shortly.

3. **Fully automatic mode:** This mode is initiated by the IR3 sensor. After an alarm signal is received, the exact whereabouts of the fire will be determined by the IR stereo vision system, after which a suitable spraying sequence to extinguish the fire will be initiated.

These modes will hereafter be referred to as mode 1, 2 and 3 respectively.

#### Setpoint Generation

The following 2 sections will describe how setpoints are generated in the PLC, which involves control of three variables: $\varphi$, $\theta$ and $p$. In general, the ultimate purpose of the setpoint generating process is to extinguish the fire. However, given the nature of the system, this will be dealt with in following two ways:

1. Direct fire suppression, i.e. spraying the fire source with water.

2. Spraying the surrounding area.

Directly hitting the fire source with the monitor can be challenging in several ways: Sufficiently accurate models to predict the landing points of the jet, wind disturbances and sufficiently accurate measurement of fire localization. To account for this, it may be beneficial not only target the fire but also to spray the nearby area, and thus, while hopefully extinguishing the fire, also prevent the fire from spreading further.

#### Mode 2

The general fire extinguishing method for mode 2 is presented in Figure 3.35. Given the geometry and position of a zone, the zone is 'painted' by the jet of the fire monitor.

A zone is defined in the program by sets vertices (Figure 3.35a). A vector is then constructed between each of the subsequent points, and the setpoints are computed along the path of these vectors (Figure 3.35b).

The space between each pass of the jet is set to 1 meter in order to be able to cover the entire zone. It is assumed that the proliferation of the jet (Figure 1.3) is sufficiently high that 1 meter separation will suffice, i.e. the fire cannot 'hide'. After completion of each loop, the monitor will repeat the process until the time limit of 60 seconds is reached.

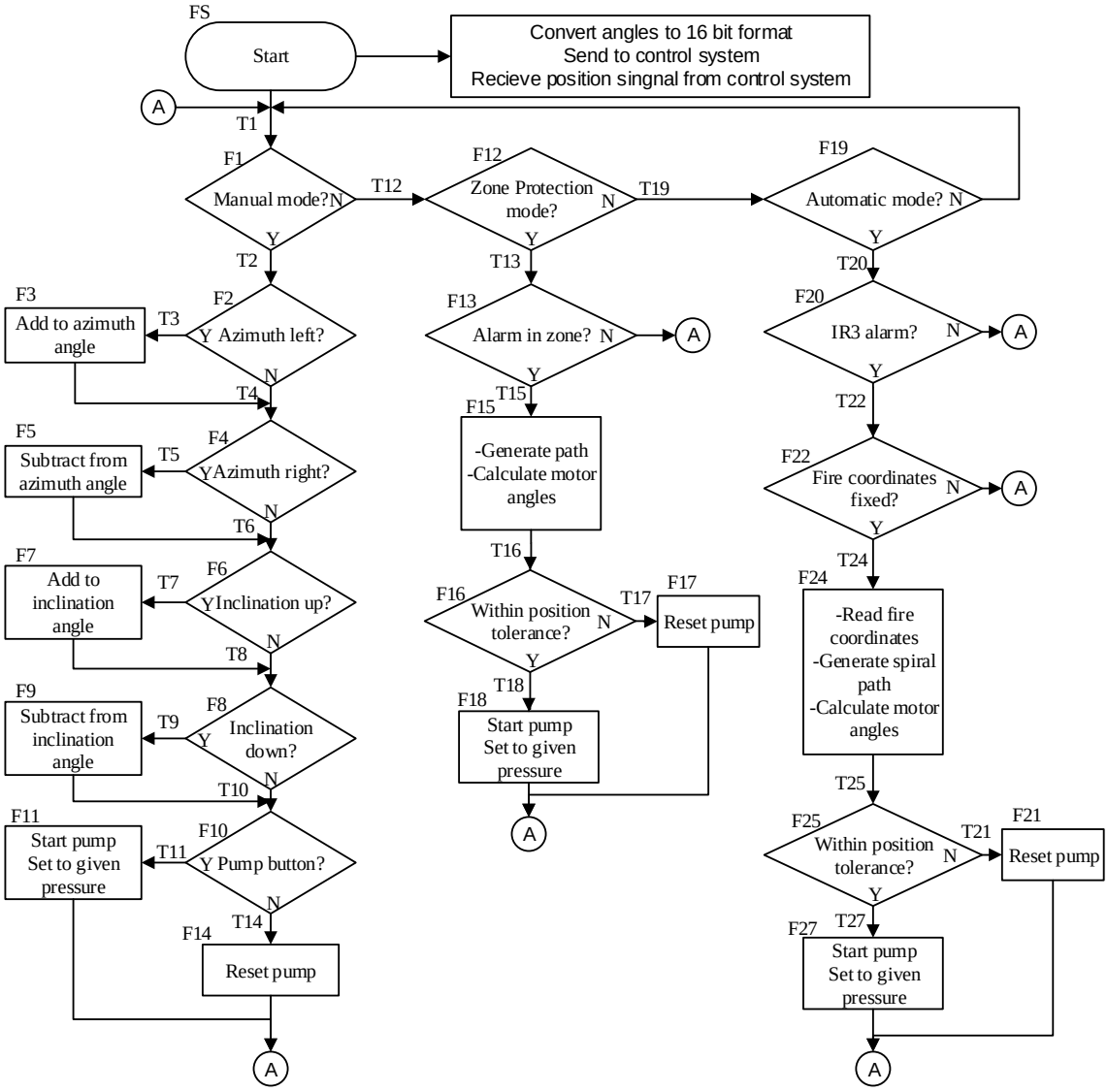
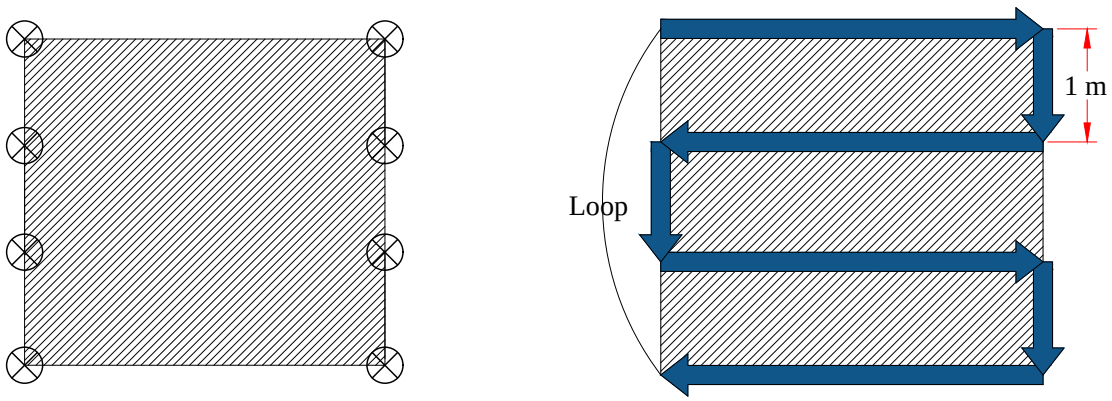



Figure 3.34: Flowchart of PLC program structure.




(a) Example of a zone with vertices.

(b) Zone with vectors added.

Figure 3.35: Fire extinguishing in mode 2.

**Mode 3**

The setpoint generation in mode 3 is similar to that in mode 2, but with a smaller distance between each swipe (Figure 3.36). Since this mode is triggered by a positive signal from the IR3 sensor, which is sensitive to relatively small fire sources (Table 3.6), the fires detected in this mode is likely not as developed as those which may be encountered in mode 2, and therefore setpoints are closer to the fire source than in mode 2.

In addition, the setpoints in this mode is generated by a spiraling path originating at the fire position.

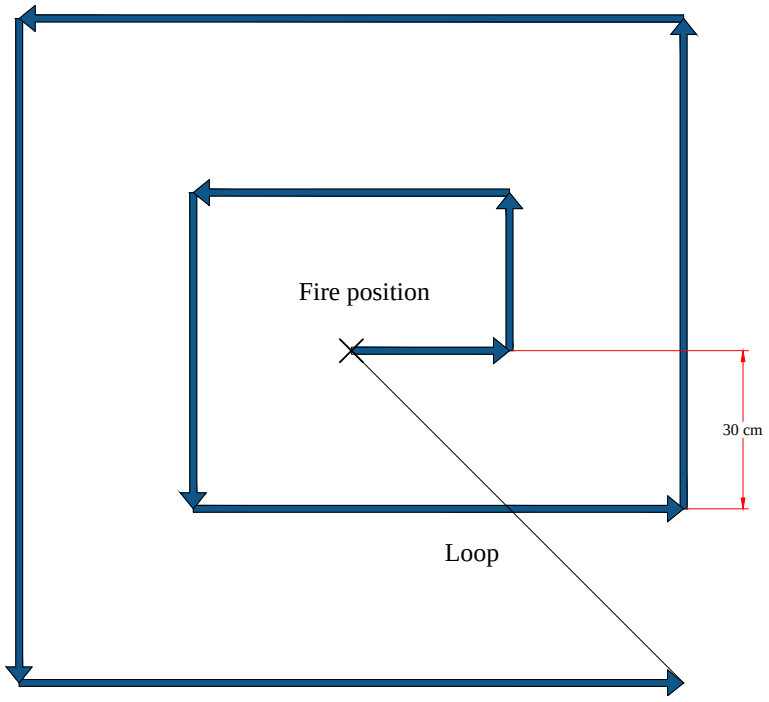



Figure 3.36: General fire extinguishing method in mode 3.

**Hysteresis: Discretization of Pressure**

For simplicity's sake, it is advantageous to modulate the pressure source in discreet levels instead of continuous manipulation. For instance, for throw lengths up to $l_1$, a pressure $p_1$ may be used, and for throw lengths which surpass $l_1$ a pressure $p_2$ may be used, while only controlling the configuration of the monitor to accurately hit a given target.

One challenge in this regard, however, is that a zone may be located between $l_1$ and $l_2$ such that the pressure will make frequent jumps during operation. A way to work around this issue, is by implementing a hysteretic behaviour in pressure levels (Figure 3.37). Initially, the pressure $p_1$ is used to hit targets within throw range $l_2$ before pressure is increased to $p_2$. After the pressure level is changed, a new lower limit is created, $l_1$ such that when working at distances slightly above and below $l_2$, which may happen during spraying of large zones, does not alter the pressure level until the new threshold is reached.

In this setup, $p_1$=5 bar, $p_2$=10 bar, $l_1$= 45 m and $l_2$=47 m. 5 bar is used for most of the range in order to make the system more energy efficient, in addition to being the pressure used during experiment II (Section 3.1). The 2 m gap between $l_1$ and $l_2$ is arbitrarily chosen.

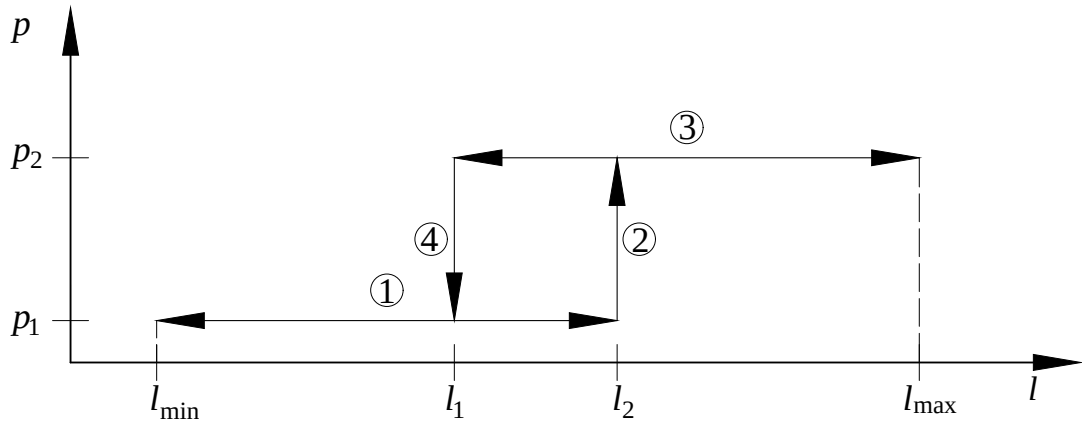



Figure 3.37: Hysteretic pressure modulation. An example sequence of how pressure can be changed from lower to upper limit is and back is indicated by circled numbers.

### Polynomial Approximation

The preceding section describes the way the different models are optimized to fit measured data. Ultimately, however, the fitted parameters are part of a differential equation (Equation 2.15). Needless to say, implementing an ODE solver on a PLC running in real time is excessively resource demanding and therefore difficult to do in practice. Because of this, polynomial approximations are made of the solutions to the differential equations. The purpose is to map the proper configuration of the fire monitor to the input variables, $U$, $\beta$ and fire position $P$. A simple illustration is presented in Figure 3.38, where the function $F$ represents the polynomial approximation.

The regression procedure is done in MATLAB, which approximates solutions to the ODEs by determining

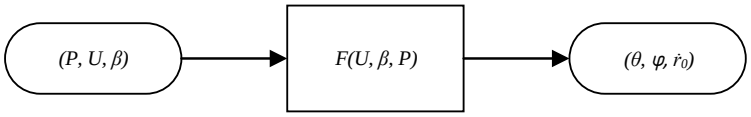



Figure 3.38: Illustration of mapping from input variables to monitor configuration.

polynomial coefficients via the Vandermonde matrix. In the case where there is no wind disturbances, one can set up the equations in the following way, where $\theta$ is a polynomial function of throw length $l$ of degree $n$:

$$\begin{bmatrix} 1 & l_0 & l_0^2 & \dots & l_0^{n-1} & l_0^n \\ 1 & l_1 & l_1^2 & \dots & l_1^{n-1} & l_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & l_{n-1} & l_{n-1}^2 & \dots & l_{n-1}^{n-1} & l_{n-1}^n \\ 1 & l_n & l_n^2 & \dots & l_n^{n-1} & l_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{n-1} \\ \theta_n \end{bmatrix}$$

Or just

$$\mathbf{V} \cdot \mathbf{a} = \mathbf{b} \tag{3.10}$$

where $\mathbf{V}$ is the Vandermonde matrix, $\mathbf{a}$ is a vector of coefficients and $\mathbf{b}$ is a vector consisting of the actual values. (3.10) can be solved in the following way:

$$\mathbf{a} = (\mathbf{V}^T \cdot \mathbf{V})^{-1} \cdot \mathbf{V}^T \cdot \mathbf{b} \tag{3.11}$$

The quality of the estimator function will be quantized with the root-mean-square deviation (RMSD). Let

$\hat{y}$ be the predicted value, $y$ be the true value of $N$ predictions, then the RMSD is computed as

$$RMSD = \sqrt{\frac{\sum_{n=1}^{N} (\hat{y} - y)^2}{N}} \tag{3.12}$$

### User Interface

The PLC's user panel features 8 on/off buttons and 2 potentiometers which functions as analog inputs, yielding 0...10 V (Figure 3.39). The three leftmost buttons set the mode the system is in, mode 1, 2 and 3 respectively. The next three buttons are used in mode 1 to change the orientation of the monitor, which incrementally changes the monitor configuration when pressed. 'Left' indicates a positive rotation of $\varphi$, and 'Right' a negative one. 'Up' will orient the monitor upwards, i.e. decrease the value of $\theta$, and 'Down' the opposite. The final button is the alarm signal which initiates the fire extinguishing sequences, depending on which mode the PLC is in. Next, the left analog input is used to modulate the pressure source between 0 and 10 bar, such that an input of 10 V corresponds with 10 bar, and 0 V yields 0 bar. Finally, the right potentiometer is used select in which zone the alarm signal is from in mode 2. An input of 0-3.3 V indicates zone 1, 3.3-6.7 specifies zone 2 and 6.7-10 V is zone 3.


Figure 3.39: PLC with analog and digital inputs.

Even though there is no emergency stop button on the user panel, the program is built in such a way (Figure 3.34), that when no mode is active, the monitor becomes passive. So, if the PLC is in the fully automatic mode for instance, simply disengaging that mode will effectively stop the monitor and the pressure source.

## 3.6.4 HIL Simulation

A dynamic model of the system is obtained by modeling in SimulationX (Figure 3.40). However, it should be noted that, since the specifications of the system at the present time is not known, but the basic topology is[12], some assumptions are made. Firstly, a standard servomotor model from the SimulationX library is

used, which has the specifications listed in Table 3.18. The motor is connected, via a gearing connection with ratio $i = 20$, to a load representing the inertia of the monitor, of 0.15 kg· m$^2$. Secondly, friction is neglected. This is done for convenience, but also since the actual friction present is not known. Lastly, a simple feedback control system with a PI controller is connected to the servomotor. The transfer function $G_{PI}(s)$ of the controller is

$$G_{PI}(s) = G_P + \frac{1}{T_I \cdot s} \tag{3.13}$$

Consequently, the 'D' term in Figure 3.40 is set to zero.

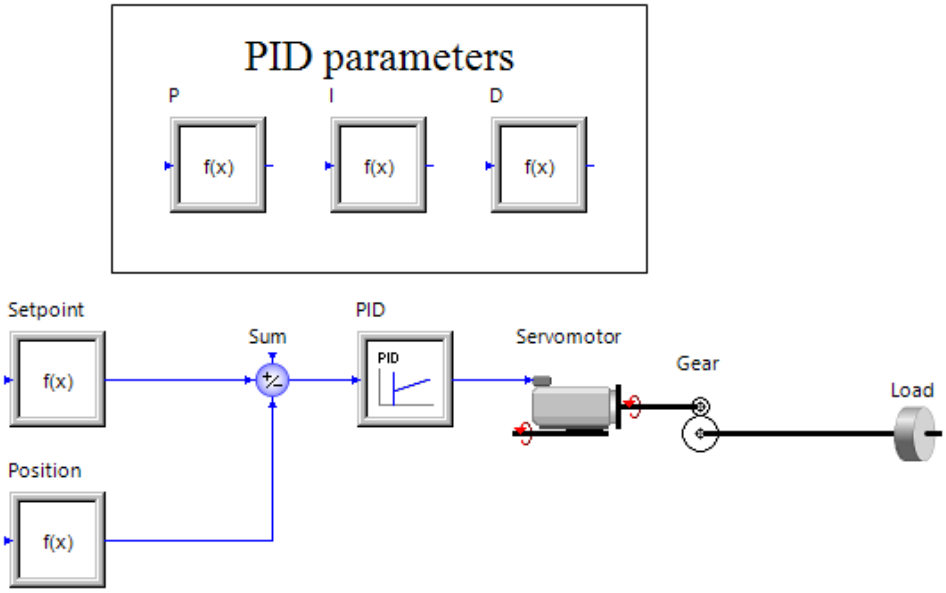



Figure 3.40: Simulation model in SimulationX.

Table 3.18: Servomotor specifications.

| Parameter | Value |
|---|---|
| Speed controller gain | 240 |
| Speed controller integration time | 0.0005 s |
| Current control gain | 0.004 A/(rad/s) |
| Current control rise time | 0.003 s |
| Maximum motor current | 5.56 A |
| Motor torque constant | 1.44 Nm/A |
| Rotor inertia | 0.79 kg · cm$^2$ |

The load can be approximated, for instance by assuming the body of the monitor to be approximately a hollow cylinder. Then the mass moment of inertia $I$ (for motor 1, Figure 1.2) with mass $m$, outer radius $R$ and inner radius $r$ can be calculated as

$$I = \frac{1}{2} \cdot m \left(R^2 + r^2\right) \tag{3.14}$$

Now assuming $m = 10$ kg, $R = 0.15$ m and $r = 0.10$ m, $I$ becomes 0.1625 kg·m$^2$. Finally, for simplicity, the inertia is rounded off to the nearest number, so it becomes 0.15. This inertia is now assumed to be the load of motor 1 and 2.

The models from SimulationX, one for motor 1 and 2, are then exported as *.dll* files and imported into LabVIEW, with angular position, controller gains $G_P$ and $T_I$ as input and angular position as output.

## 3.7 Validation and Testing

Final simulations are carried out as illustrated by Figure 1.4, where the PLC is connected to a real time target which runs the model representing the system dynamics, and another computer for readouts and data logging. The actual setup can be seen in Figure 3.41.

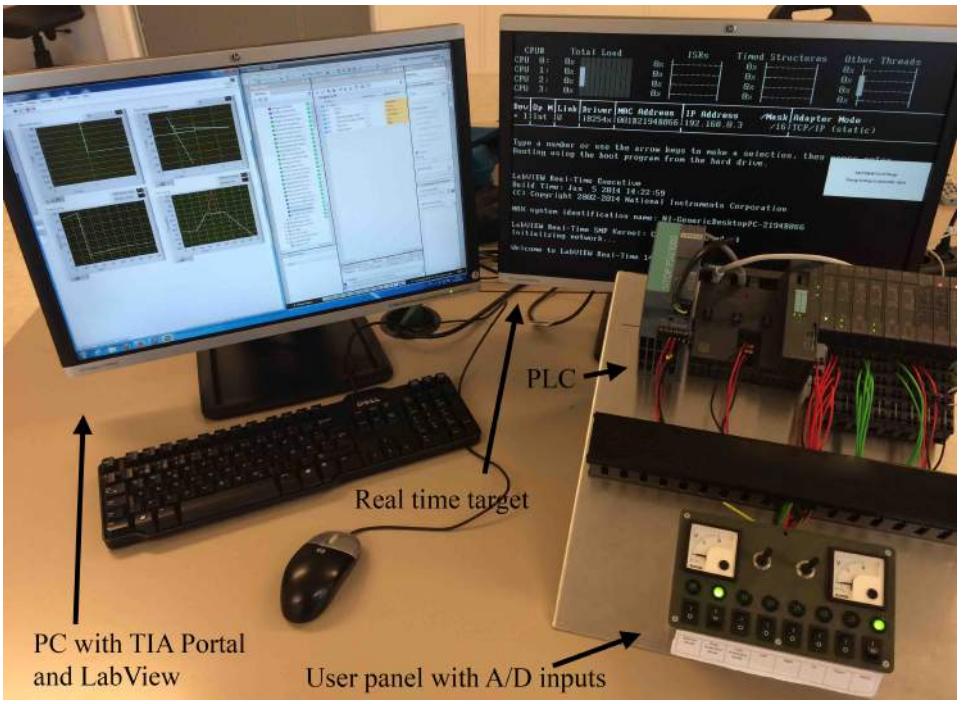



Figure 3.41: HIL setup.

# CHAPTER 4

# Results and Analysis

## 4.1 Liquid Jet Modelling

### 4.1.1 Experiment I

RESULTS of measurements of the landing points in experiment I is presented in Figure 4.1, where the discharge point is centered at the origin. An uncertainty in measurement of ±3 m is allowed for. The relatively large uncertainties are due to the fact that, during testing, winds were quite strong (∼5-10 m/s, Appendix E), which means that the centroid of the landing points are challenging to determine. All in all it must be said, that conditions during experiment I, made measurements an awkward process.

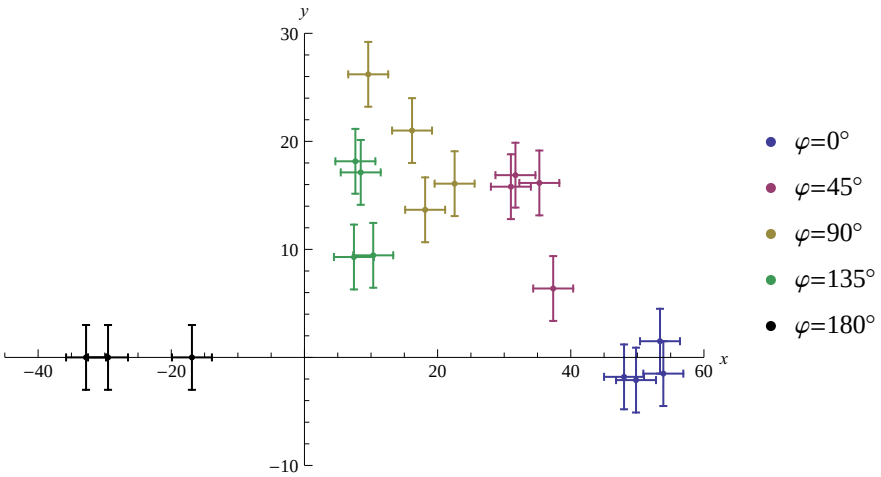



Figure 4.1: Landing points for various values of $\varphi$.

From Figure 4.1 it can be seen that the deflection of the jet is quite substantial. Even at $\varphi = 135°$ the wind, which generally blew in the direction of the $x$-axis, would bend the trajectory of the jet such that landing points are in the first quadrant. It can also be observed that one sample at $\varphi = 180°$ has not been included in the dataset since the jet was completely disintegrated, and a landing point could not be determined. This particular sample was at the utmost vertical configuration, i.e. $\theta = 20°$. This agrees with the general observation of the landing points during testing with headwinds and crosswinds, namely that the higher the inclination, the more pronounced the wind effects in the sense that the jet is more disintegrated in it's travel. Landing points for various values of $\theta$ is presented in Figure 4.2.

Wind data corresponding to Figure 4.1 and 4.2 are presented in Figure 4.3. The uncertainties are as listed in Table 3.13. Continuous measurements are presented in Appendix E.

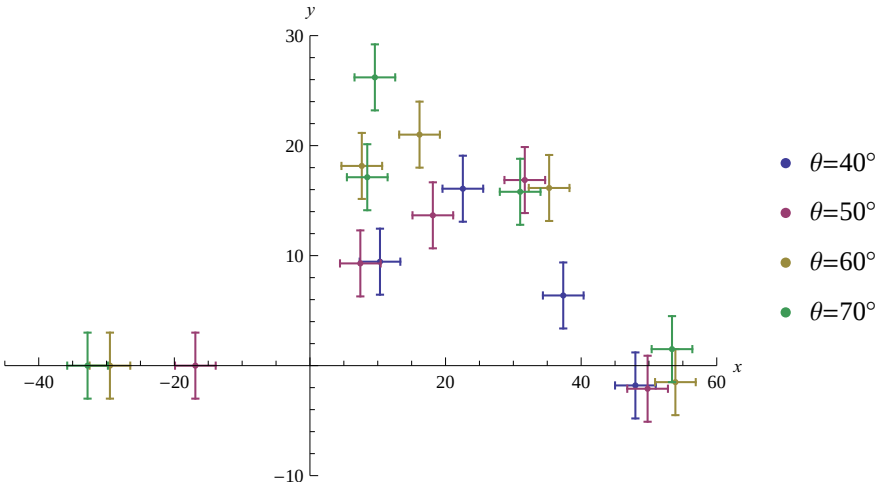



Figure 4.2: Landing points for various values of $\theta$.

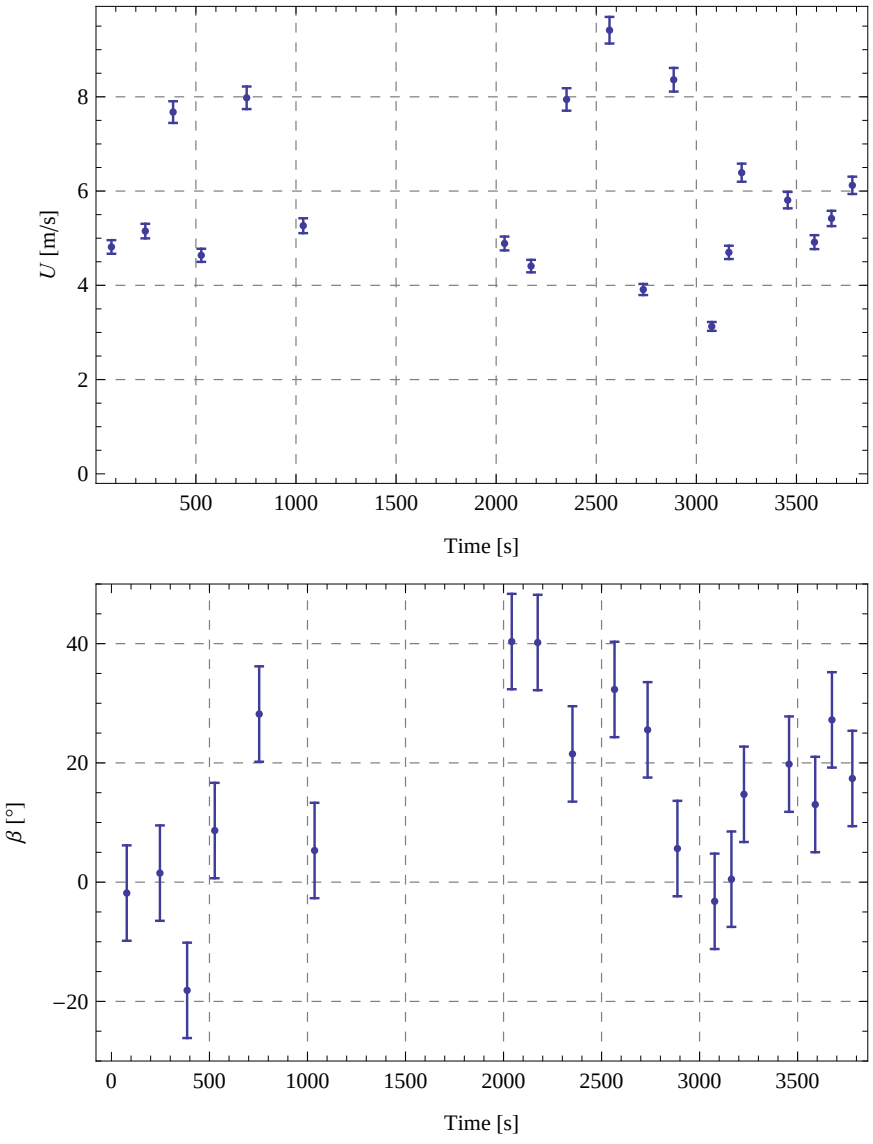



Figure 4.3: Wind measurements for experiment I.

### 4.1.2 Experiment II

Measurements of throw lengths for various values of $\theta$ during experiment II is presented in Figure 4.4. As discussed in Section 3.1, experiment II is conducted with, in relation to experiment I, comparatively little wind, and it is assumed that the wind effects are negligible. Therefore, $\varphi$ is not of interest in this experiment.

The uncertainties in this experiment is set to $\pm 1$ m. By comparison to experiment I, this is a modest uncertainty, which is mainly due to the fact the coherence in the jet was considerably improved in this experiment, and thus marking the landing points is much simpler.

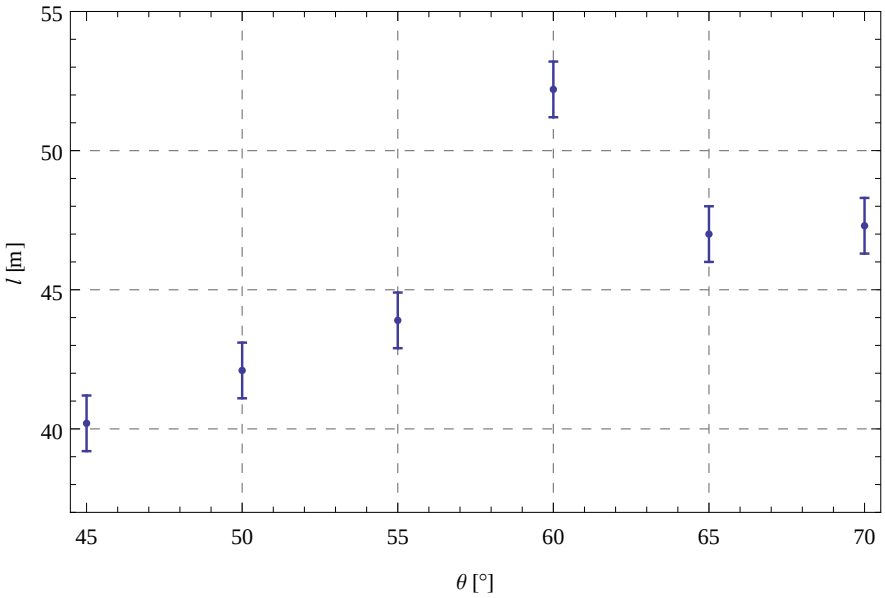



Figure 4.4: Measurements of throw lengths for various values of $\theta$ (experiment II).

The results presented in Figure 4.4 may be of interest. Since we know that the maximum throw length for a classic projectile occurs at about 45° relative to the horizontal, the agreement with the data shown here is limited, also when taking into account the uncertainty in $\theta$[41](Figure 2.15). This may be somewhat expected, however, since the two are clearly not the same, as should be clear from Section 2.3.

The maximum throw length occurs at $\theta = 60°$. If the discharge angle is altered by 5°, however, at least 10 % of the throw length is lost. Furthermore, the discharge angle $\theta = 45°$, actually yields the lowest throw length in this dataset.

The compliance with data in [16] is also limited, which shows the maximum throw length of a jet to occur at $\theta = 50°$ (Figure 2.17). In the cited paper, there is little difference in throw length between $\theta = 50°$ and $\theta = 60°$, however, so there seems to be some relation at the very least. The incongruity with the results presented here may be due to a variety of factors, which will be discussed shortly.

The measurement data from experiment II is presented in Figure 4.5. Although the wind data does not serve any purpose except for reference.

Figure 4.5 may give some insight with regards to assessing the reasonability of the assumptions taken during this experiment. The wind speed during most of the samples are between 1.5 and 2.0 m/s, and the highest wind speed is about 2.8 m/s, which are clearly lower than that of experiment I (Figure 4.3). Ultimately, however, it is of course up to the reader to interpret these results.

It should be noted that, during this experiment, there was, for the majority of the time, not sufficient wind present to rotate the indicator of the wind direction sensor, and therefore the wind direction data is of quite

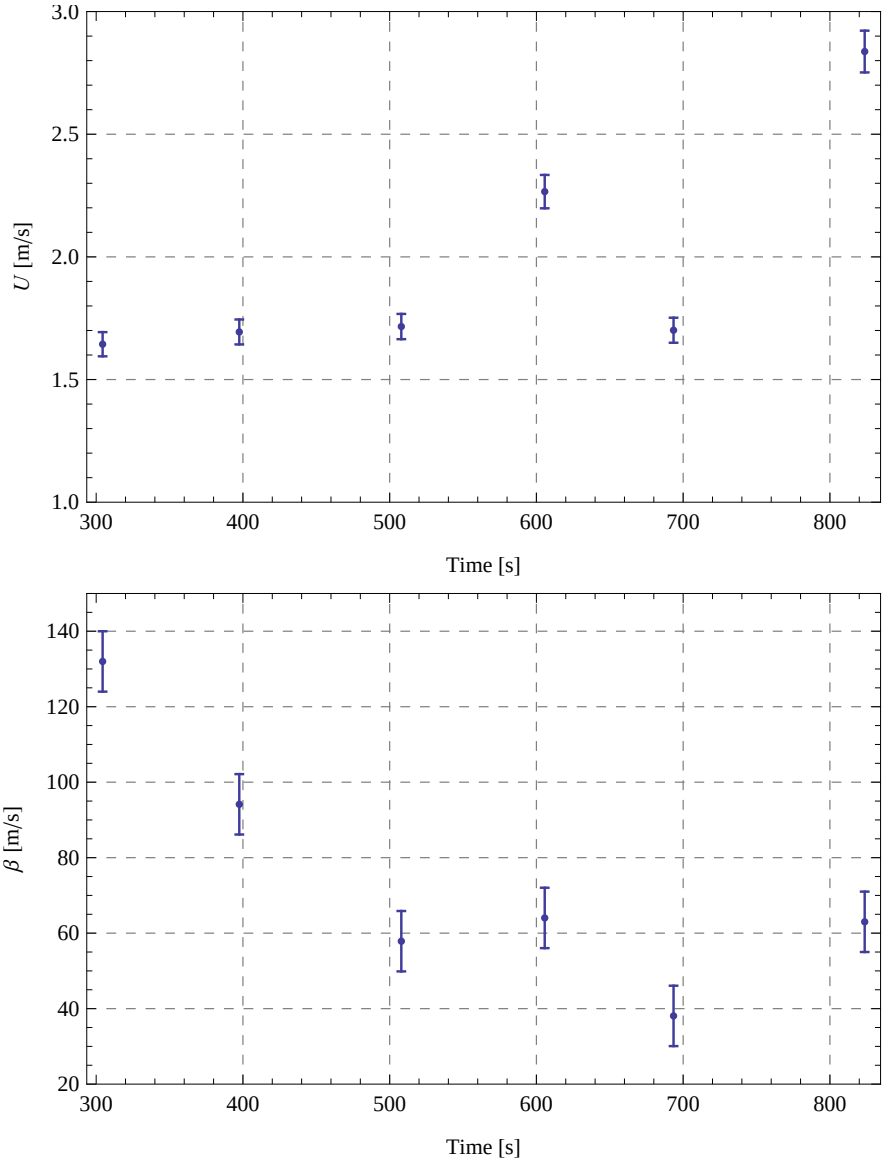



Figure 4.5: Wind speed and direction during experiment II.

limited value.

### 4.1.3 Determination of Initial Speed

A contour plot of the fluid's speed in the longitudinal direction of the nozzle from the CFD analysis is presented in Figure 4.6. To verify that the solution is indeed a steady-state solution, the convergence of the average speed components of the fluid can be visualized as presented by Figure 4.7.

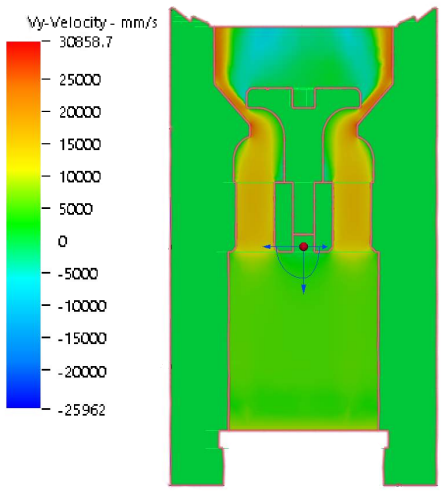



Figure 4.6: Contour plot of the fluid speed from CFD analysis.

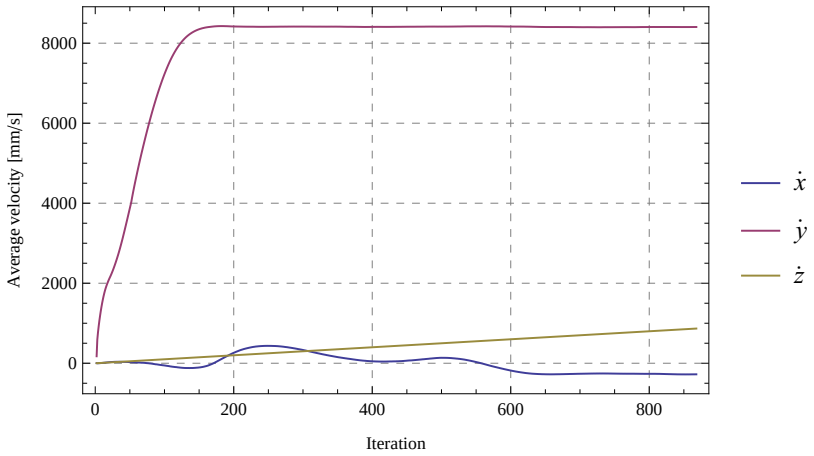



Figure 4.7: Convergence curves from CFD model.

The proportionality constant from (3.8) is then determined by matching the result from the CFD analysis (Figure 4.8). The numeric value is given in (4.1), where the pressure is in bar and $\dot{r}_0$ is in m/s.

$$\dot{r}_0 = 9.86631\sqrt{p} \tag{4.1}$$

### 4.1.4 Parameter Identification

#### Non-uniform Parameters

Outcomes from the parameter identification process carried out based on the results from experiments I and II is presented in Table 4.1 and 4.2 respectively, where the mean deviation is computed from (3.6).

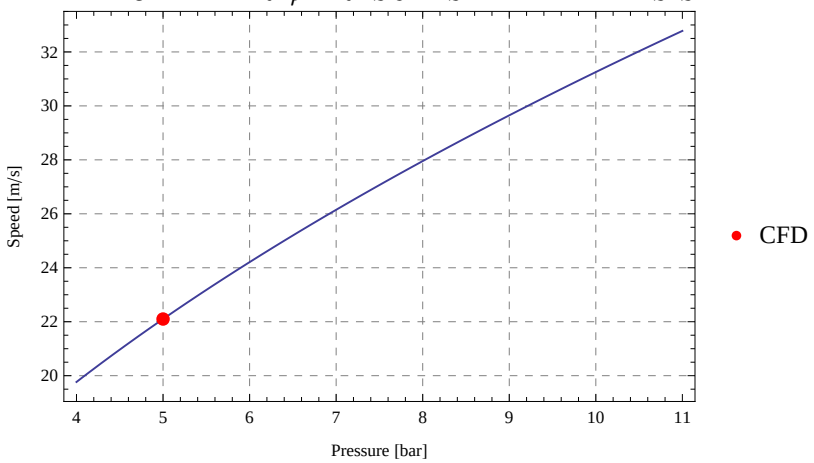



Figure 4.8: The estimated function of the initial speed and result from the CFD analysis.

Table 4.1: Comparison of non-uniform models (experiment I).

| **Model #:** | I | II | III | IV | VII |
|---|---|---|---|---|---|
| Mean deviation [m] | 9.84 | 10.6 | 11.0 | 10.4 | 12.7 |
| Maximum deviation [m] | 18.5 | 18.6 | 18.3 | 20.0 | 18.3 |

Trajectories from experiment I as reproduced by model I are as shown in Figure 4.9.

The trajectories from experiment II as recreated by model I is presented in Figure 4.10.

### Uniform Parameters

Results from the parameter identification process based on experiment I and II is presented in Table 4.3 and 4.4 respectively. Some reproduced trajectories are presented in Figure 4.11 and 4.12 using the models with the lowest average error.

### Analysis

As expected, the results from the non-uniform parameter models tend to exhibit non-physical behaviour. From Figure 4.9 and 4.10 it can be observed that the trajectories are 'stretched', due to the fact that the models are allowed to have different drag forces in each direction. On the plus side, the non-uniform models have achieved the best results in terms of error, with the best mean deviation at 9.84 m for experiment I and 1.63 m for experiment II (Table 4.1-4.4).

The uniform parameter models undoubtedly exhibit more intuitive behaviour (Figure 4.12 versus Figure 4.10), but on the other hand yield higher errors. This is probably due to the fact that the models are much more limited in terms of free parameters.

It can observed from the errors in Table 4.1-4.4 that there exist some outliers in each model's results which deviate quite considerably from the mean error. Moreover, the results are seemingly flat, e.g. Table 4.1. In

Table 4.2: Comparison of non-uniform models as compared to data from experiment II.

| **Model #:** | I | II | III | IV | VII |
|---|---|---|---|---|---|
| Mean deviation [m] | 1.6 | 3.0 | 1.7 | 1.8 | 9.5 |
| Max deviation [m] | 4.5 | 6.6 | 5.0 | 5.0 | 15.1 |

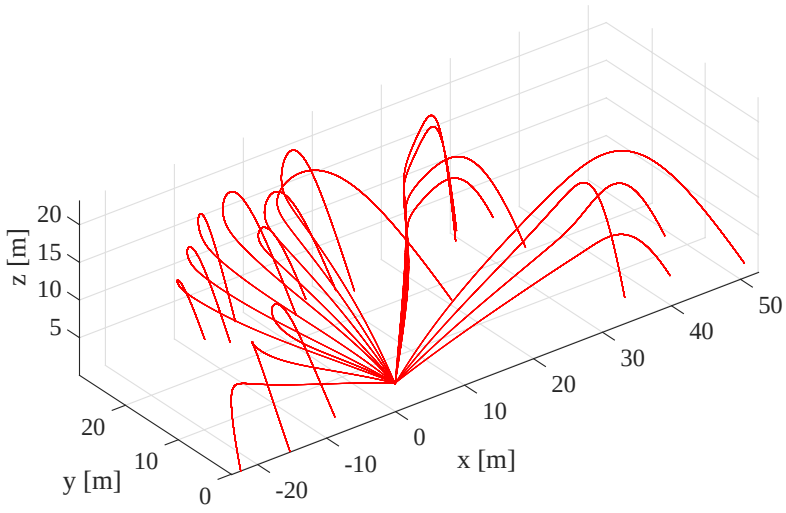



Figure 4.9: Trajectories reproduced by model I with non-uniform parameters based on initial conditions in experiment I.

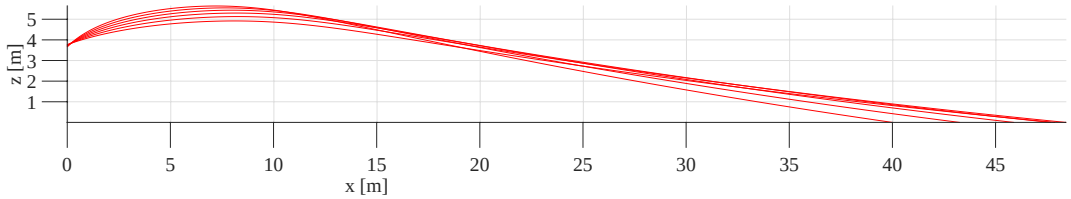



Figure 4.10: Trajectories from experiment II reproduced by model I with non-uniform parameters.

Table 4.3: Comparison of uniform parameter models as compared to data from experiment I.

| **Model #:** | I | II | III | IV | VII |
|---|---|---|---|---|---|
| Mean deviation [m] | 13.2 | 12.9 | 12.8 | 12.9 | 13.7 |
| Max deviation [m] | 22.3 | 22.9 | 22.8 | 23.1 | 22.0 |

Table 4.4: Comparison of uniform parameter models as compared to data from experiment II.

| **Model #:** | I | II | III | IV | V | VII |
|---|---|---|---|---|---|---|
| Mean deviation [m] | 3.9 | 4.3 | 4.6 | 4.5 | 8.9 | 6.2 |
| Max deviation [m] | 5.9 | 6.3 | 6.6 | 6.4 | 13.8 | 9.2 |

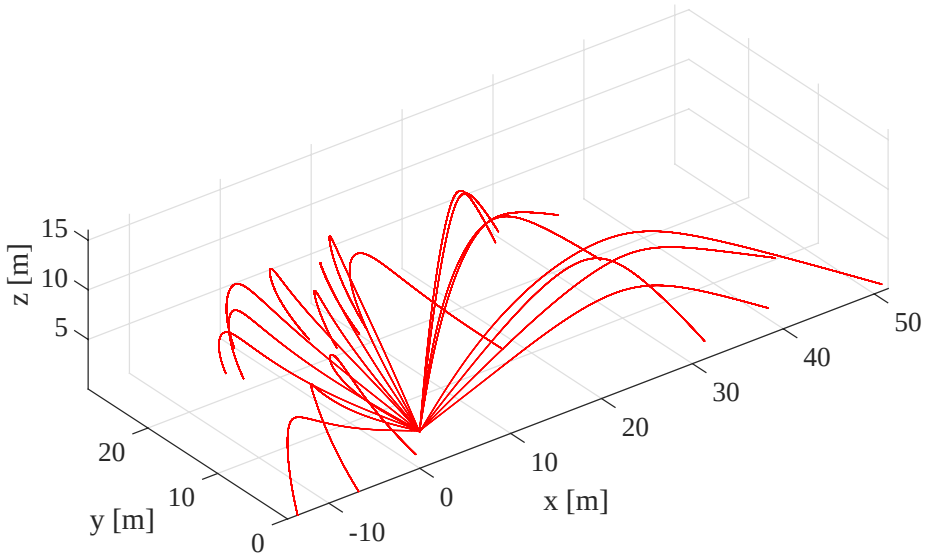



Figure 4.11: Trajectories reproduced by model III with the same initial conditions as experiment I.

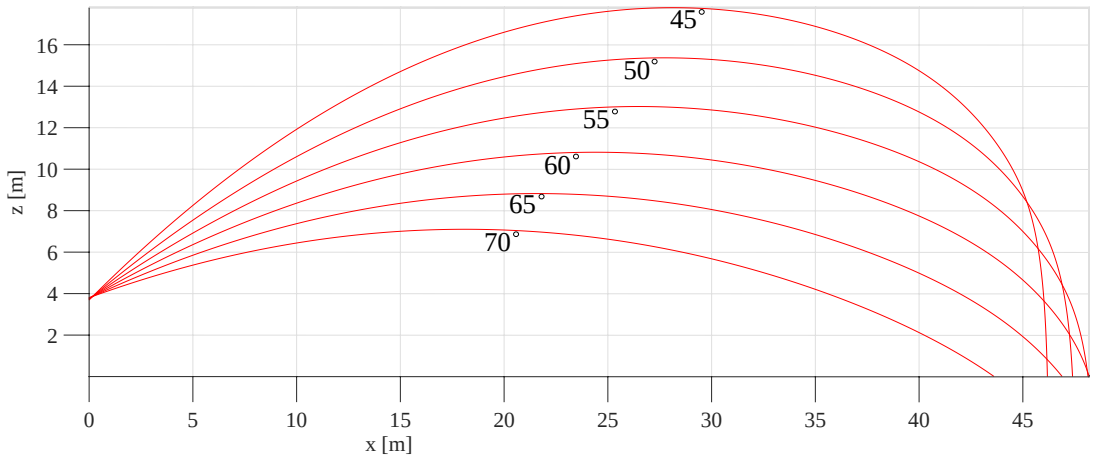



Figure 4.12: Trajectories with boundary conditions of experiment II reproduced by model I.

other words, there are quite marginal differences in the applicability of the models tested. These results may thus indicate that each model is structured in a useful way. In addition, perhaps as expected, the classical drag model (model VII) yields the poorest results in terms of mean deviation.

To summarise, it must be said that the majority of the models with the estimated parameters fail to yield satisfactory results. This may be due to a number of reasons.

Firstly, the optimization problem in itself is arbitrarily difficult. The minimization of (3.5) represents a numerical nonlinear global optimization problem which involves solving nonlinear ODEs. In general, the ODEs include free parameters multiplied with exponential terms also raised to some free parameters. During the optimization process large variations in the results were observed, and most notably, a large span in all the free parameters, either favouring the exponential terms, or the constant terms. Therefore, it is difficult to say with certainty that the results found are optimum, although considerable efforts have been invested in the optimization process.

One way of improving results would be to have more data available. In this thesis, only landing points have been measured. While, in the works by Hatton et al. [16] for example, data from the air travel is also taken into account, which may undoubtedly be helpful in weighting the constant terms and exponential terms.

Secondly, the measurements of wind, particularly those during experiment I may not be optimal. The wind rack was placed at some distance from the discharge point (Figure 3.2). The reason for this was to shelter technical equipment from bad weather. This distance may have lead to readings which were not sufficiently accurate. Furthermore, experiment I was conducted during relatively strong, turbulent winds (Figure E.1,E.2) which, at times, changed rapidly in magnitude and direction. Therefore, the winds at the time of measurement may be substantially different than occurring ones, and may thus introduce far greater uncertainties in measurement than those imposed by the measurement instruments themselves.

The above statements does not take into account the limitations specified in Chapter 1, which are key for these measurement methods to work in the first place. Clearly, the neglected effects from a non-uniform wind field may not be justifiable, and the fact that the wind is, of course, not completely parallel to the $xy$ plane. The ladder may be accommodated somewhat by adding an additional wind sensor to record the vertical wind speeds as well. Non-uniformity in the wind speed may be to some degree accounted for by using multiple sensors placed appropriately, but would also complicate the problem considerably.

Thirdly, it must be emphasized, that the only truly relevant research in the open literature with regards to the present thesis is the work done by Hatton et al. [16] and Miyashita et al. [43], who have presented, at least by comparison, useful results with respect to reconstructing the trajectories of a liquid jet. The relevance of [16] with the present thesis is limited, however, since the nozzle used during experiments, which is also the case in general in literature, is a conical nozzle (Figure 4.13). These are effectively the inverse of what is installed on the monitor used during experiment I and II, and indeed also the one for which this thesis is intended (Figure 1.2). As should be clear from 2.2.2, the state of the jet as it emerges from the nozzle is decisive for the throw length of the jet. In a conical nozzle, the outlet is smaller than the inlet, so the water flow is contracted, as opposed to scattered to the sides (Figure 3.21). In addition, the throw lengths in [16] is far greater than the working range of the system in this thesis (Figure 2.17).

The applicability of the work done by Miyashita et al. [43] in this context is, unfortunately, also limited, since the cited paper is concerned with comparatively heavy duty fire monitors. As a consequence, the limitations on the domains of the model are not in compliance with the experiments carried out in the present thesis. The model presented in [43] is valid for $U \in [-8, 8]$ $m/s$, $\theta \in [40, 60]^\circ$, $p \in [0.6, 0.9]$ MPa and $Q \in [10, 40]$ kl/min (Section 2.3). The flow used in experiments in this thesis are significantly lower, which may explain some of the discrepancies between experiment II and model V (Table 4.4).

Finally, the exit speed of the monitor may be another source of error. The CFD analysis carried out have been conducted with a model of the fire monitor in Figure 1.2, while the experiments, of course, have been

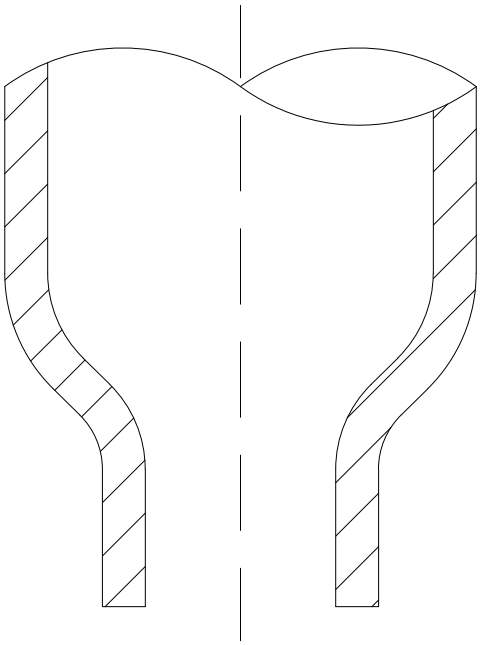


Figure 4.13: Conical nozzle.

done with the monitor shown in Figure 3.1. It is assumed that the geometries are sufficiently equal so that any difference in jet characteristics can be ignored. Admittedly, however, there is most likely some variances between the two, and so applying the exit speed determined in (4.1) in the parameter identification process may adversely affect the end results.

In the following sections, model I with uniform parameters and wind-still conditions will be used, which is the model with the best results in terms of error with uniform parameters. The principles in what follows, however, is the same for any other model.

### Polynomial Approximation

As previously mentioned, the pressure source is discretized in to 5 and 10 bar. In addition, when carrying out the polynomial approximation process, $\theta$ is restricted to $60...85°$. These limits allow the monitor to utilize the whole working range with the specified pressures, while making implementation of the hysteretic behaviour in pressure possible. Moreover, as compared with other limitations on $\theta$, the RMSDs tended to be greater than those listed in Table 4.5

Polynomial approximations are made with 4th order polynomials, yielding RMSDs as specified in Table 4.5. Initially, the goal of the approximation was a RMSD $< 0.1$ m, however, for the case where the pressure is 5 bar the RMSD is 0.110. Visual confirmation of the approximation is presented in Figure 4.14, where it can be seen that there is an overlap between the two curves, which makes it possible to implement the hysteretic pressure behaviour as described in Section 3.6.3.

Table 4.5: Polynomial adaptation

| Pressure [bar] | RMSD [m] |
|---|---|
| 5 | 0.110 |
| 10 | 0.016 |

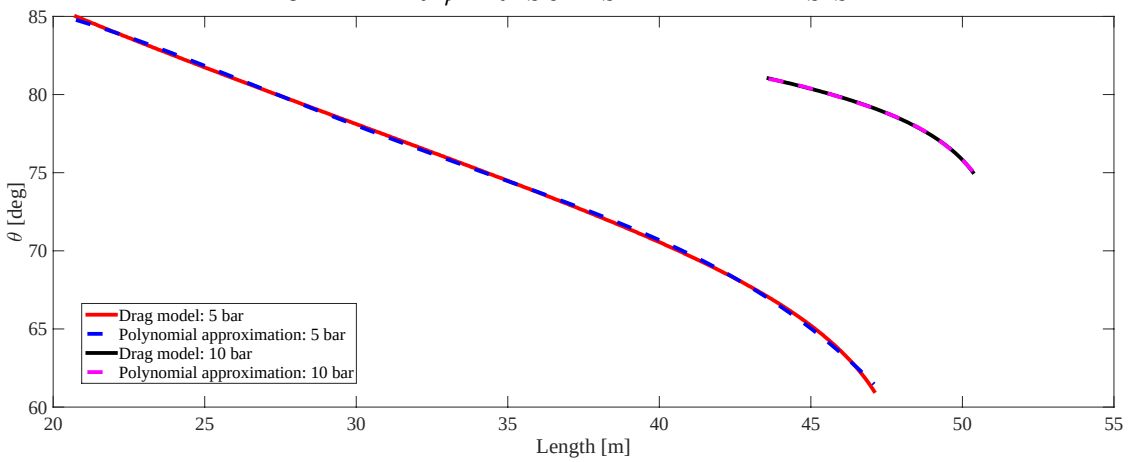



Figure 4.14: Polynomial adaptation to the dataset form the confirmed simulation model

## 4.2 Stereo Vision

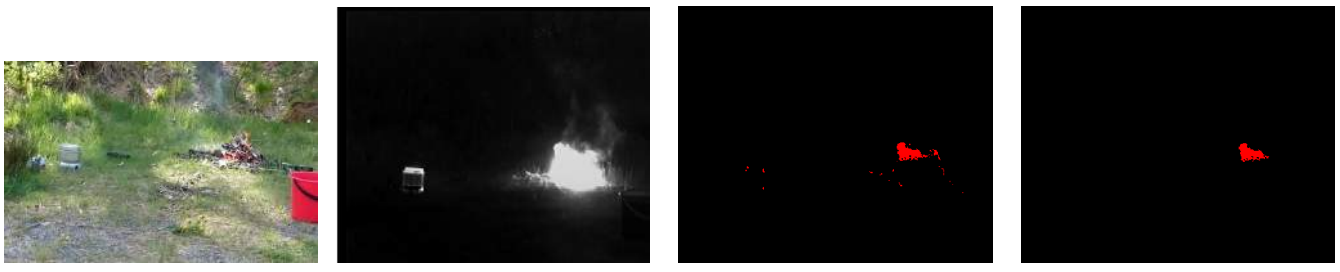
The results from the stereo vision test is presented in Table 4.6. It can be seen that the results are quite stable, remaining at ∼0.5 m error. This is somewhat unexpected, since, typically, the accuracy will be high at close range and decrease with distance (Figure 3.32). These results may thus be an indication that the resolution of the cameras and the image processing techniques are suitable for the working range for the system.

Table 4.6: Results from stereo vision test.

| **Distance [m]** | **Measured distance [m]** | **Error [m]** | **Error [%]** |
|---|---|---|---|
| 30 | 30.5 | 0.5 | 1.64 |
| 40 | 40.5 | 0.5 | 1.23 |
| 50 | 49.7 | -0.3 | -0.60 |
| 60 | 59.5 | -0.5 | -0.84 |

### 4.2.1 Object Discrimination

The vision system is successful in distinguishing a fire and a camping stove with a boiling pot of water (Figure 4.15). The shown image is taken approximately 10 meters from the fire, this time with a somewhat smaller fire in order to make the sizes of the two objects more comparable.



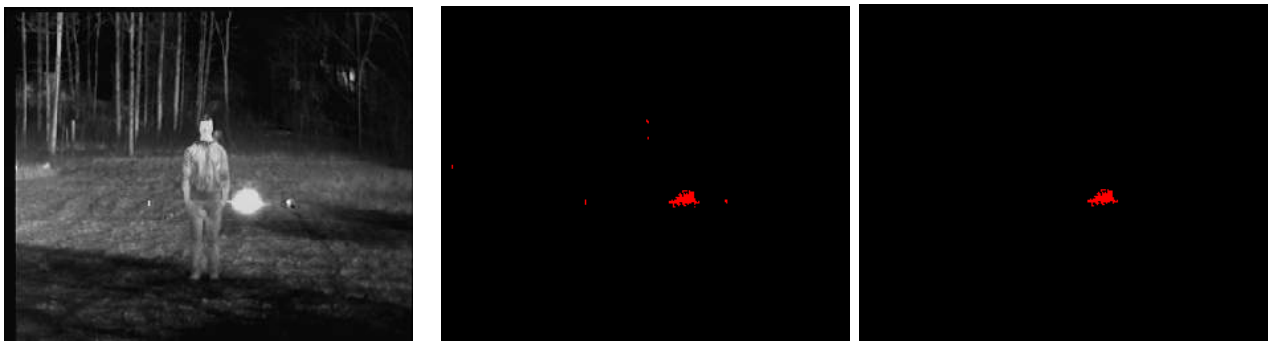
(a) Color image. (b) Calibrated IR image. (c) Results after automatic thresholding algorithm. (d) Resulting image after small particle removal.

Figure 4.15: Discrimination of wooden fire and camping stove with boiling pot of water.

It is uncertain, however, whether it is the temperature difference between the objects (∼ 400°) or the small object removal process which ultimately removes the camping stove from the image. Clearly, the fire is the brightest object in the image, and after the thresholding is applied, only seemingly small hotspots remain

of the stove, which are removed in the last image processing step. Tests where other hot objects of some size may therefore be interesting to test as well.

The vision system also successfully distinguishes between a fire and person (Figure 4.16). In this image, the sun shines onto the surroundings of the fire, leaving some hot spots on tree trunks and surrounding ground as can be seen in Figure 4.16b.



(a) Calibrated input image. (b) Thresholded image. (c) Final image.

Figure 4.16: Object discrimination: Person and wooden fire.

## 4.3 Simulation Results

This section will review simulation results from the HIL setup. It is important to note that the results presented here is the output from the polynomial approximations deduced in Section 4.1, so in reality there is an inherent error which comes from the model, which in this case is 3.9 m on average. On the contrary, the results presented here is meant to verify the performance of the system, assuming that the model is accurate.

### 4.3.1 Mode 1

Although results from mode 1 may be conceived as somewhat trivial, they may further demonstrate the working principles of the PLC's user panel (Figure 3.39). For a pressure of 5 bar, the setpoints of the fire monitor is as shown by Figure 4.17 by manipulating $\theta$ and $\varphi$ as shown in Figure 4.18a and 4.18b.

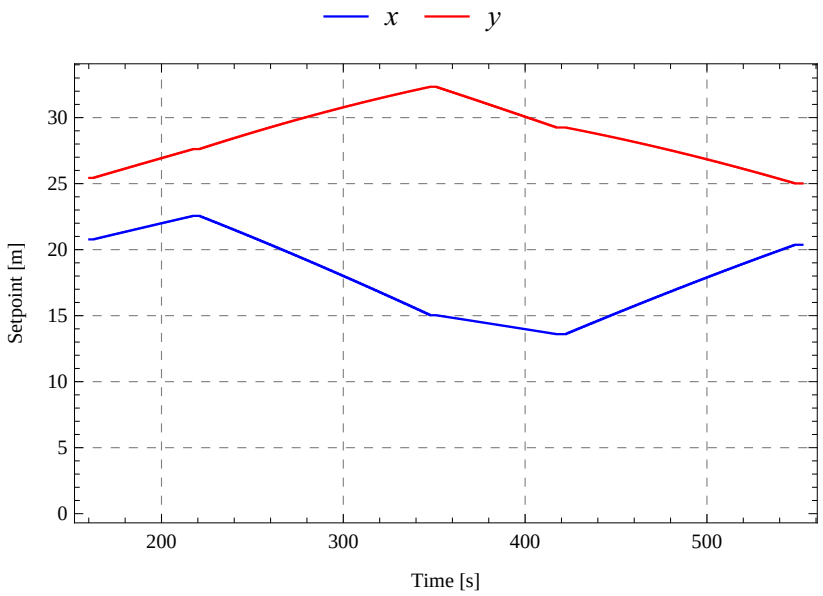
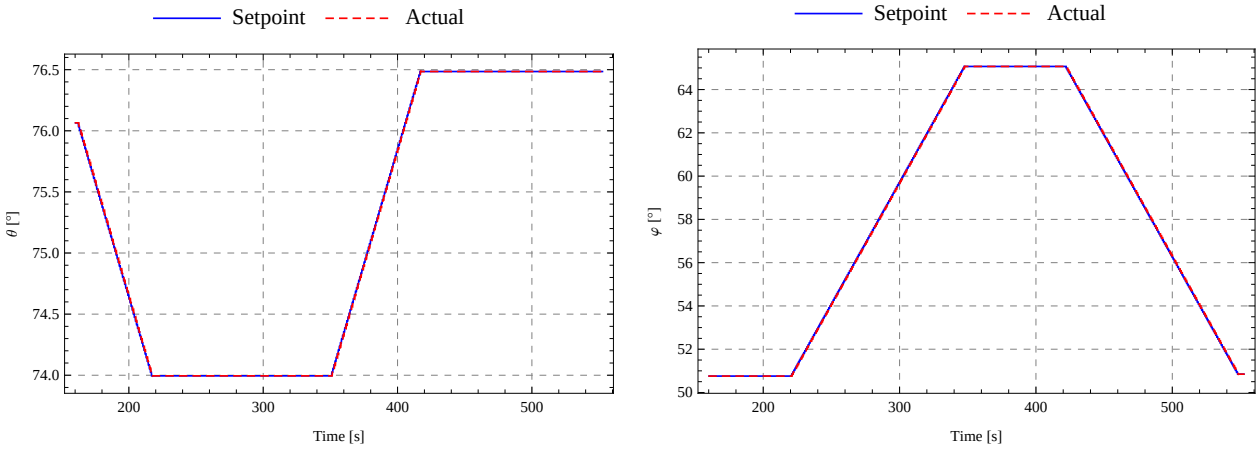



Figure 4.17: Setpoints of the jet in mode 1 with manual manipulation of $\theta$ and $\varphi$.




(a) Setpoint and process values for $\theta$ in mode 1.

(b) Setpoint and process values for $\varphi$ in mode 1.

Figure 4.18: Setpoints and process values for mode 1.

### 4.3.2 Mode 2

The simulation results presented here have been achieved with the controller parameters presented in Table 4.7. Results are presented in Figure 4.19, 4.21 and 4.23 for zone 1, 2 and 3 respectively. In addition, the setpoints of $\theta$ and $\varphi$ for each zone is presented in Figure 4.20, 4.22 and 4.24 with reference signals in the sequential order.

Table 4.7: Parameters of PI controller.

| Parameter | Value |
|---|---|
| $G_P$ | 250 |
| $T_I$ | 1 ms |

Overall, the results are satisfactory, except from minor discrepancies between the reference signals and the actual values. These discrepancies is likely caused by the error in the polynomial approximations. Also, by observation, it can be seen that the reference signals in general are quite sharp. As a result, the motors will have to follow instantaneous changes position, leading to some deviations at certain points, e.g. Figure 4.20b and 4.24b. The cause of the instantaneous changes in reference signals is the fact that throw length is a function of pressure, which as described previously is discretisized, and thus sudden changes in pressure levels results in immediate changes in setpoint values.

That being said, the accuracy of the system seems to be within what is necessary, considering the fact that the jet is likely to proliferate to a degree which makes an accuracy of less than a few meters essentially redundant (Figure 1.3). But in any case, it is clear that the main limitation is accurate models of the jet trajectory. The lack of which, may be partially redeemed by spraying sequences as the ones presented here. For mode 3, however, where more accuracy is necessary, this may obviously pose a great challenge and limit the effectiveness of the system.

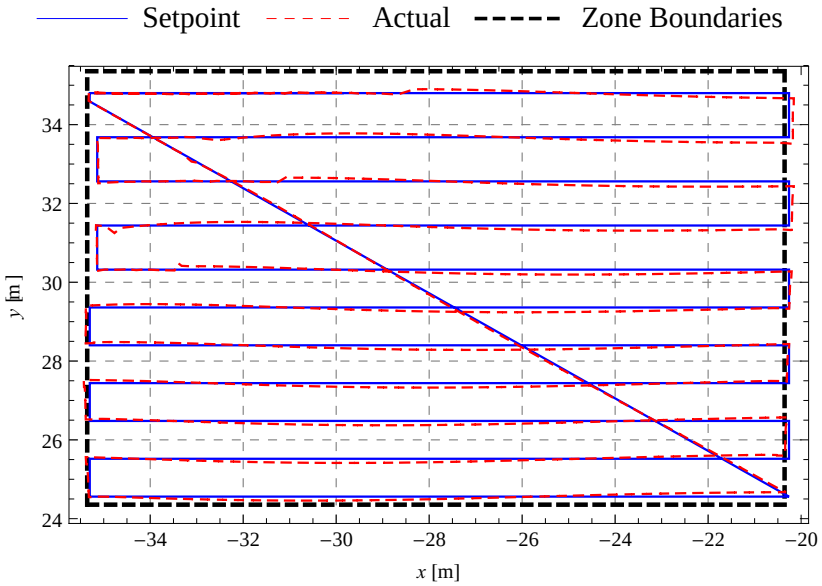
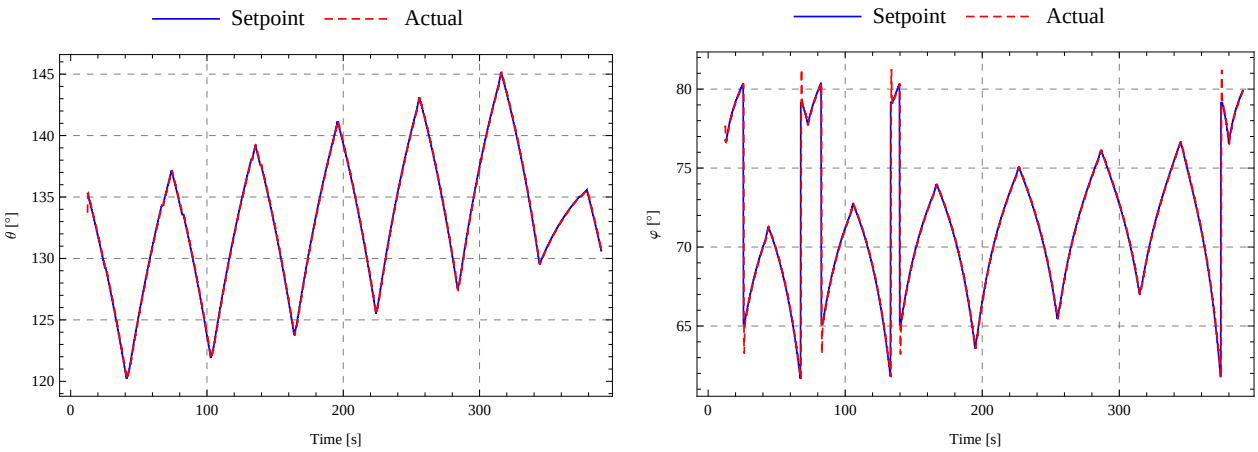



Figure 4.19: Zone 1: Comparison of setpoints and landing points of the jet.

### 4.3.3 Mode 3

The simulation results from mode 3 is obtained by giving an arbitrary coordinate to the PLC, which in practice would come supplied from the vision system. In this case, a coordinate of $(x, y) = (20, 20)$ is used. A plot of the setpoints and the actual landing points as predicted by the model is presented in Figure 4.25.




(a) Setpoint and process values for $\theta$ in zone 1.

(b) Setpoint and process values for $\varphi$ in zone 1.

Figure 4.20: Setpoints and process values for zone 1.

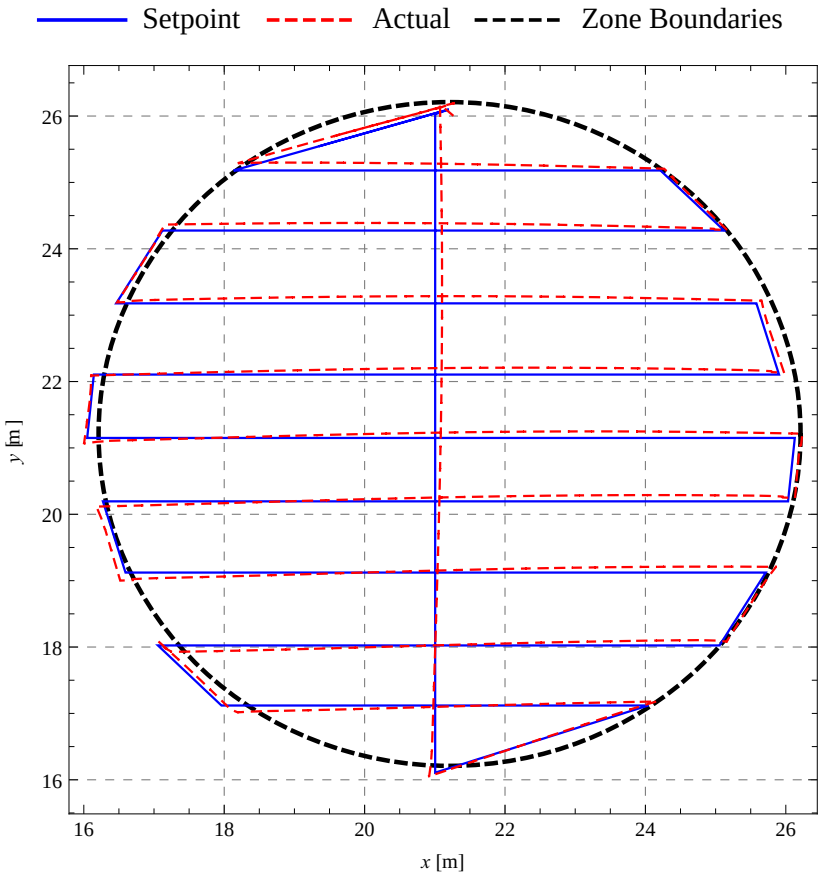
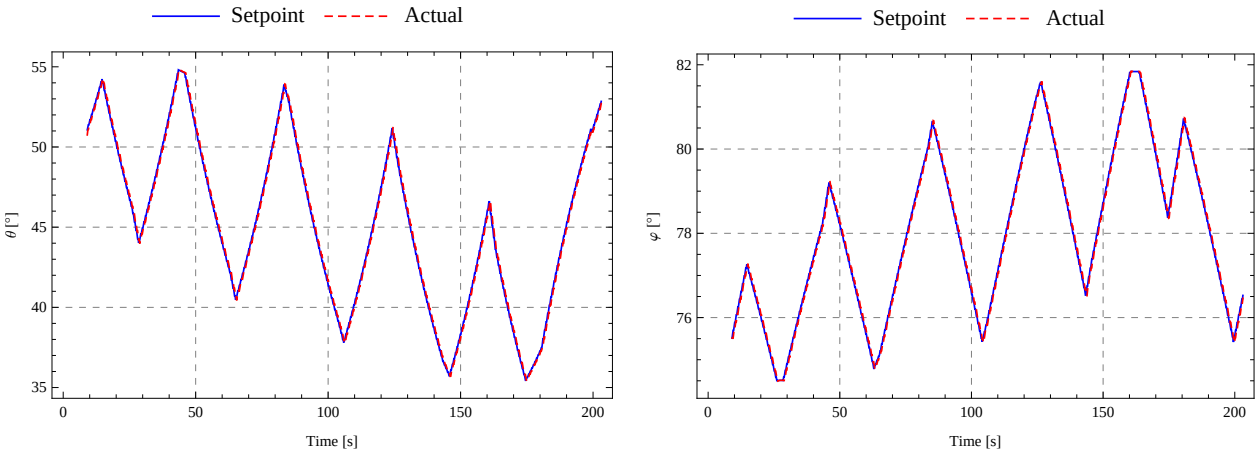



Figure 4.21: Simulation results for zone 2.




(a) Setpoint and process values for $\theta$ in zone 2.

(b) Setpoint and process values for $\varphi$ in zone 2.

Figure 4.22: Setpoints and process values for zone 2.

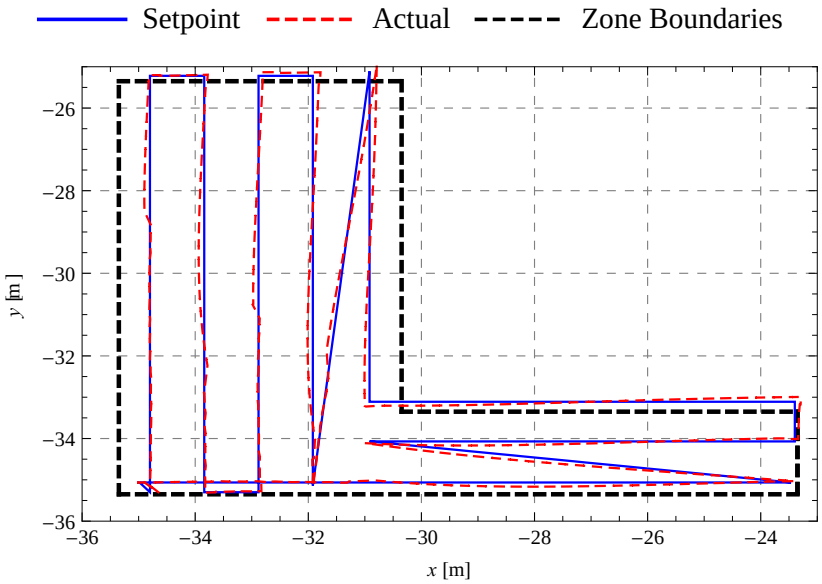
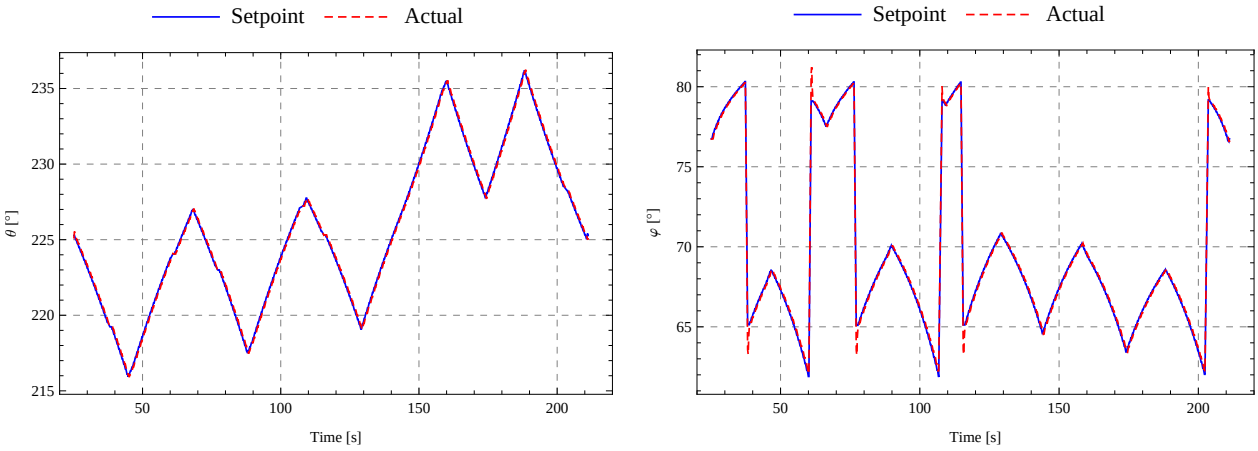



Figure 4.23: Simulation results for zone 3.




(a) Setpoint and process values for $\theta$ in zone 3.

(b) Setpoint and process values for $\varphi$ in zone 3.

Figure 4.24: Setpoints and process values for zone 3.

Plots of the setpoints and actual values of the angles $\theta$ and $\varphi$ is given in Figure 4.26.

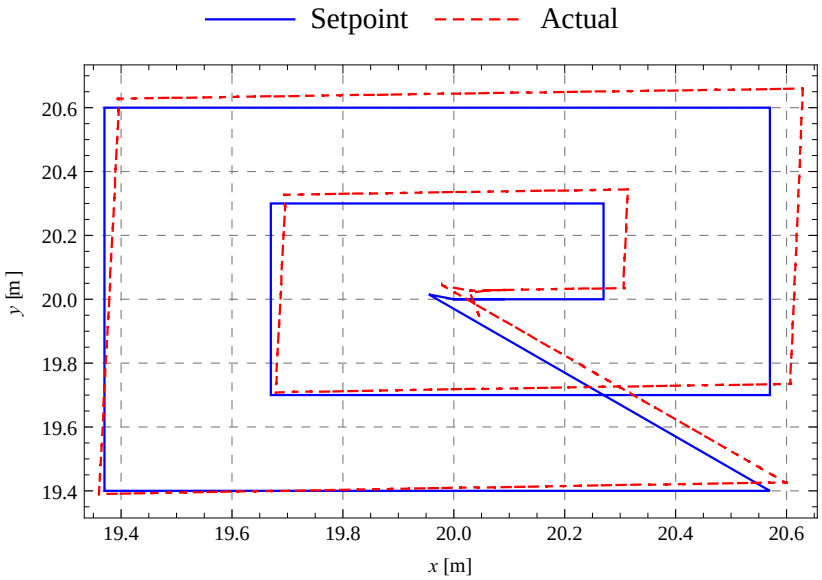
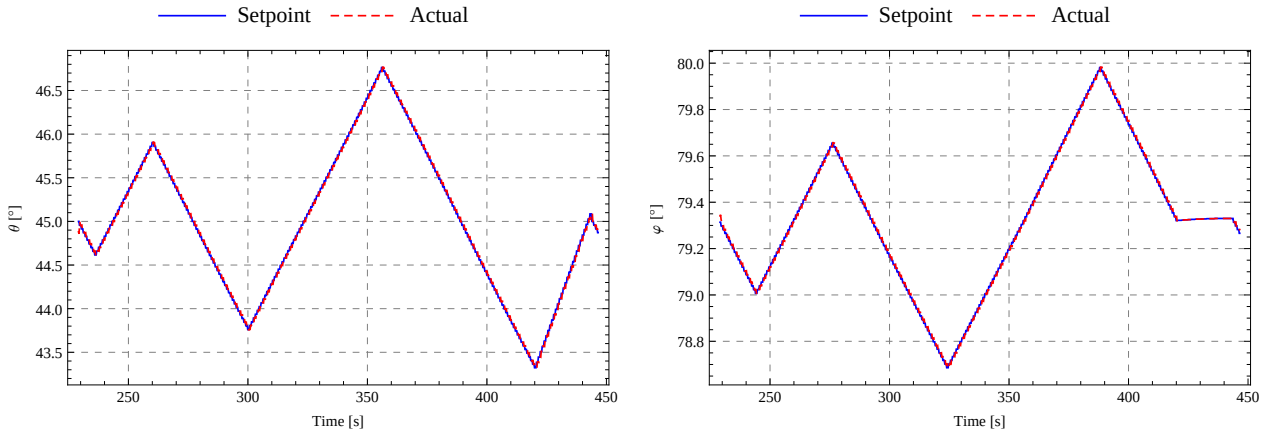



Figure 4.25: Results from mode 3.




(a) Setpoint and process values for $\theta$ in mode 3.

(b) Setpoint and process values for $\varphi$ in mode 3.

Figure 4.26: Setpoints and process values for mode 3.

The results in mode 3 is similar to those of mode 2 in terms of accuracy. Although the 'actual' values presented in Figure 4.25 seem skewed, at the same time the distances are considerably smaller in this mode, and therefore the results are satisfactory.

What may seem somewhat curious, however, is that the entire plot is translated slightly to the left, in addition to a small abnormality around the center. It is not readily clear why this happens, but most likely there is some unwanted modification done to the setpoints during generation in the PLC program. Still, the overall accuracy is passable.

# CHAPTER 5

# Conclusion & Future Work

## 5.1 Jet Trajectory Prediction

RELEVANT literature has been reviewed, and the most applicable trajectory models have been optimized. The most accurate model to predict the landing points of a jet with wind disturbances under the circumstances described in Section 3.1 is model I (using non-uniform parameters) and III using uniform parameters. In both cases, however, the difference between the models is marginal (Table 4.1-4.3), and therefore any of models I-IV may be used equally. That being said, the presented results cannot be said to be satisfactory, with the best average error being about 10 m with non-uniform parameters and 12.8 m using uniform parameters . As specified in Chapter 1, the end goal is to be able to hit a target with an accuracy of about 2 m, which is clearly not reached in this case. From these data it is evident that the major limiting factor is, perhaps as expected, accurate modelling of the jets' trajectories.

The most accurate model to predict a jet's landing points with approximately wind still conditions (Figure E.3) with the circumstances described in Section 3.1 is model I, with and without uniform parameters (Table 4.2 and 4.4). In this case, the target of less than 2 m error is fulfilled, since the smallest error is about 1.6 m

In order to overcome these challenges, the authors would like to propose to conduct many more experiments without wind present, such as that presented in Figure 4.4, with the actual fire monitor (Figure 1.2). This will, hopefully yield more accurate results and make it possible to, at the very least, predict jet trajectories in becalmed conditions.

If a model which is sufficiently accurate with regards to wind disturbances is to be determined, it is likely that the problem must be attacked with more ambition than it has in the present thesis. For one, experiments should naturally be conducted with the actual fire monitor (Figure 1.2). The reason this has not been done in this thesis is for practical purposes. Admittedly, in retrospect, the authors of the present thesis must admit that the problem of wind compensation has taken somewhat lightly. In reality, the problem is arbitrarily challenging, especially when conducting outdoor experiments with relatively strong winds. In order to improve on the results presented in this thesis, the following suggestions are made:

1. More precise measurements of wind and landing points
2. Conduct experiments in more controlled environments, perhaps using a wind tunnel or indoors in large open areas
3. Simplify the minimizing function, i.e. (3.5)
4. Test different optimization methods
5. Obtain coordinate of the jet's air travel

More precise measurements of the wind direction as well as measurements of the vertical wind speeds may improve the outcome of the optimization process. In this thesis, the uncertainty in wind direction measurements is about $\pm 8°$, and no measurements are made of the vertical wind speeds. The impacts these

limitations have on the outcome of the parameter identification process is not known.

One possible way to conduct similar experiments in more controlled environments is to use a wind tunnel, where the wind speed and direction is known to great accuracy, although the applicability between such an experiment, and actual testing outdoors may be limited. Another possibility is to conduct experiments in an indoor environment, such as large production halls etc., which should make it possible to obtain quality data and predict the landing points in wind still conditions, at the very least.

The minimization process itself may be simplified by squaring each side of (3.5) which may make the optimization process faster, and possibly yield better solutions. In addition, several other global optimization algorithms can be tested, e.g. the Nelder-Mead-, differential evolution-, simulated annealing- and random search method.

As is done in [16], the optimization procedure can be done with datapoints from the jet's air travel as opposed to optimization with respect to the landing points alone. This could potentially give much better results, especially with quality data.

## 5.2 Stereo Vision Fire Detection

A stereo system consisting of 2 IR cameras has been developed. The developed vision system is able to detect a fire with an accuracy of about 0.5 m throughout the working range (Table 4.6). Although the vision system has not been tested at a distance of 25 m, it is highly unlikely that the accuracy would decrease at lower distance.

The vision system is also able to discriminate between a wooden fire, a person and a camping stove with a boiling pot of water (Figure 4.15 and 4.16).

The stereo system has, however, not been tested with hot objects of large size, which could be an additional test in terms of object discrimination. Such an object could for instance be vehicles or other objects which have been heated by the sun. Furthermore, the system has not been tested with significant amount direct sunlight disturbances, which may be significant. In theory, the cameras used in this thesis has a spectral band which overlap with the spectral radiance emitted by the sun (Figure 5.1), and therefore the IR cameras may potentially be 'blinded'.

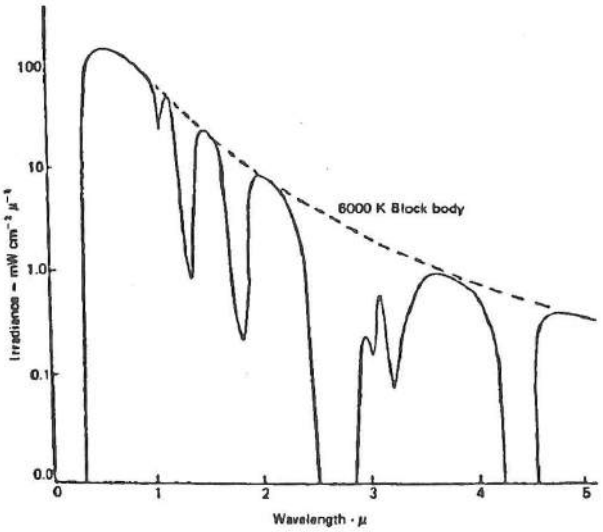



Figure 5.1: Solar spectrum at 45° zenith[35].

On that last note, it must be mentioned that the authors of the present thesis had envisioned, prior to commencing experiments, that more challenges would be encountered while detecting fire with other warm objects present. However, in practice, objects which reflect heat radiation has been more problematic (Figure 4.16).

One suggestion for future work is stereo calibration. This has not been done in the work of this thesis, but could improve the accuracy further. In addition, it has been observed during experiments, that the system is quite sensitive to minor physical disturbances during measurements, e.g. if the relative rotation between the cameras is changed. Effects of such imperfections in camera alignment may mitigated with a stereo calibration.

To improve the robustness of the vision system, calibrated IR (radiometric) cameras may be used. Thermal IR cameras with radiometric calibration allow direct temperature mapping, and thus the system may be programmed to distinguish objects in specified temperature intervals[24]. Using radiometric cameras are not without challenges, however. Objects with low emissivity, i.e. objects which cannot be approximated as a perfect black body, may still be difficult to detect. In any case, frequency analysis of fire may offer a useful solution in terms of object discrimination.

The IR3 sensor which is thought to be implemented in the system has not been tested. Furthermore, the sensor is certified for gasoline and $n$-heptane fires, which is clearly not the same as the source of fire that has been tested for in the experiments in the present thesis. Consequently, to ensure that the sensor is usable in this context, proper testing under similar circumstances may be wise.

## 5.3 PLC & Simulation

A control system using a PLC has been made (Appendix D), whereby HIL simulation results have been obtained using a drag model with zero wind. The structure of the program is divided into three main modes: One which allows the operator to manipulate the configuration of the monitor manually, as well as a mode which protects given areas in space based on alarm signals sent from that area, and finally a fully automatic mode which detects the whereabouts and suppresses fire (Figure 3.34).

A proper user panel which allows the operator to easily interact with the settings of the PLC has been made (Figure 3.39).

The performance of the PLC program is largely satisfactory (Figure 4.17-4.26). Some discrepancies are observed, but this is most likely due to errors in polynomial approximations of the wind model, which yielded a RMSD specified in Table 4.5.

In order to make it simpler for the system to follow the prescribed setpoints, it would be advantageous to 'smooth out' the generated setpoints, which are quite pointy (Figure 3.13) thus may be hard to follow. Furthermore, it is clear that during setpoint generation, the PLC protocol often translates the setpoints somewhat (e.g. Figure 4.21). The reason for this is unclear. However, the overall functioning of the PLC program is satisfactory and is obviously not a major limiting factor with regards to the system's accuracy.

## 5.4 Concluding Notes

The core of the problem presented in this thesis is the difficulty of hitting a target accurately. In many other systems, disturbances are rejected and accuracy improved by the implementation of feedback controls. Obviously, in this case, this is particularly challenging, and therefore the proposed solution focuses on robust open-loop control by predicting the trajectories based on the disturbances. One possible way of implementing a feedback which may work, however, is presented in Figure 5.2, where the landing point of

a jet is determined by a stereo rig. If IR cameras are used, the jet should appear as a quite cold object and therefore be distinguishable from the surroundings, particularly if the surroundings are sufficiently hot.

Clearly, this method would be challenging to implement in a number of ways, but if successful, it would render jet trajectory models obsolete and make accurate fire extinguishing far more easy.

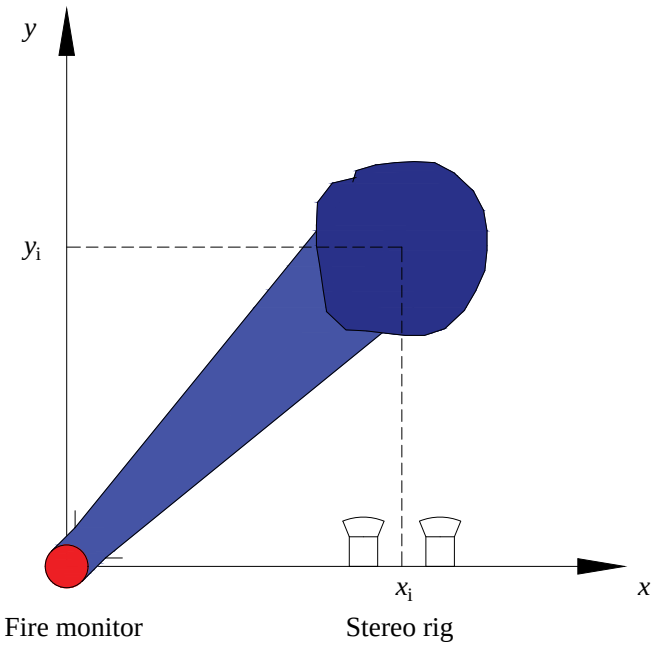



Figure 5.2: Possible solution for feedback control.

# Bibliography


[1] G Grant, J Brenton, and D Drysdale. Fire suppression by water sprays. *Progress in energy and combustion science*, 26(2):79–130, 2000.

[2] Elham Mahdipour and Chitra Dadkhah. Automatic fire detection based on soft computing techniques: review from 2000 to 2010. *Artificial Intelligence Review*, 42(4):895–934, 2014.

[3] Guoliang Hu, Ming Long, Juxing Liang, and Weihua Li. Analysis of jet characteristics and structural optimization of a liquamatic fire water monitor with self-swinging mechanism. *The International Journal of Advanced Manufacturing Technology*, 59(5-8):805–813, 2012.

[4] Ming Long, Guo Liang Hu, and Zhong Li. Analysis of water jet trajectory of auto-targeting fire sprinkler system in interior large space. In *Advanced Materials Research*, volume 490, pages 171–175. Trans Tech Publ, 2012.

[5] Yoshiaki Okayama. Approach to detection of fires in their very early stage by odor sensors and neural net. *Journal of Applied Fire Science*, 2(3):205–217, 1992.

[6] Majid Bahrepour, Nirvana Meratnia, and Paul JM Havinga. Automatic fire detection: A survey from wireless sensor network perspective. 2008.

[7] Won-Ho Kim. Dsp embedded early fire detection method using ir thermal video. *KSII Transactions on Internet and Information Systems (TIIS)*, 8(10):3475–3489, 2014.

[8] I Bosch, A Serrano, and L Vergara. Multisensor network system for wildfire detection using infrared image processing. *The Scientific World Journal*, 2013, 2013.

[9] Mei Zhibin, Yu Chunyu, and Zhang Xi. Machine vision based fire flame detection using multi-features. In *Control and Decision Conference (CCDC), 2012 24th Chinese*, pages 2844–2848. IEEE, 2012.

[10] Tao Chen, Hongyong Yuan, Guofeng Su, and Weicheng Fan. An automatic fire searching and suppression system for large spaces. *Fire safety journal*, 39(4):297–307, 2004.

[11] Feiniu Yuan. An integrated fire detection and suppression system based on widely available video surveillance. *Machine Vision and Applications*, 21(6):941–948, 2010.

[12] Mats Strande, Henrik Wikander, and Marina L. Aanonsen. Redesign og automatisering av vannkanon for brannslukking. Bachelor thesis, University of Agder, 2015.

[13] N Brook and David A Summers. The penetration of rock by high-speed water jets. In *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, volume 6, pages 249–258. Elsevier, 1969.

[14] JW Hoyt. Effect of polymer additives on jet cavitation. *Journal of Fluids Engineering*, 98(1):106–111, 1976.

[15] Kjetil Sivertsen. Personal communication. Research and Development, Fireproducts AS.

[16] AP Hatton, CM Leech, and MJ Osborne. Computer simulation of the trajectories of large water jets. *International journal of heat and fluid flow*, 6(2):137–141, 1985.

[17] A Enis Çetin, Kosmas Dimitropoulos, Benedict Gouverneur, Nikos Grammalidis, Osman Günay, Y Hakan Habibolu, B Uur Töreyin, and Steven Verstockt. Video fire detection–review. *Digital Signal Processing*, 23(6):1827–1843, 2013.

[18] Jeffrey C Owrutsky, Daniel A Steinhurst, Christian P Minor, Susan L Rose-Pehrsson, Frederick W Williams, and Daniel T Gottuk. Long wavelength video detection of fire in ship compartments. *Fire safety journal*, 41(4):315–320, 2006.

[19] Behcet Uğur Töreyin, Ramazan Gökberk Cinbiş, Yiğithan Dedeoğlu, and Ahmet Enis Çetin. Fire detection in infrared video using wavelet analysis. *Optical Engineering*, 46(6):067204–067204, 2007.

[20] Ignacio Bosch, Soledad Gomez, Raquel Molina, and Ramón Miralles. Object discrimination by infrared image processing. In *Bioinspired Applications in Artificial and Natural Computation*, pages 30–40. Springer, 2009.

[21] Joseph W Starr and BY Lattimer. Evaluation of navigation sensors in fire smoke environments. *Fire Technology*, 50(6):1459–1481, 2014.

[22] Jong-Hwan Kim, Joseph W Starr, and Brian Y Lattimer. Firefighting robot stereo infrared vision and radar sensor fusion for imaging through smoke. *Fire Technology*, pages 1–23.

[23] Zhang Yu, Shen Lincheng, Zhou Dianle, Zhang Daibing, and Yan Chengping. Camera calibration of thermal-infrared stereo vision system. In *Intelligent Systems Design and Engineering Applications, 2013 Fourth International Conference on*, pages 197–201. IEEE, 2013.

[24] Joseph W Starr and BY Lattimer. Application of thermal infrared stereo vision in fire environments. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 1675–1680. IEEE, 2013.

[25] Joseph W Starr and B Lattimer. A comparison of ir stereo vision and lidar for use in fire environments. In *2012 IEEE sensors 2012 conference*, 2012.

[26] Hugh D Young and Roger A Freedman. *Sears and Zemansky's university physics*, volume 1. Pearson education, 2006.

[27] Manaf A Mahammed, Amera I Melhum, and Faris A Kochery. Object distance measurement by stereo vision. *International Journal of Science and Applied Information Technology (IJSAIT) Vol*, 2:05–08, 2013.

[28] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[29] Ting-Wei Lin and Cheng-Yuan Chang. Enhanced calibration method for camera distortion. In *ICCAS-SICE, 2009*, pages 1115–1120. IEEE, 2009.

[30] Donald G Bailey. A new approach to lens distortion correction. *Proceedings Image and Vision Computing New Zealand 2002*, pages 59–64, 2002.

[31] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[32] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Infrared camera calibration for dense depth map construction. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 857–862. IEEE, 2011.

[33] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):532–550, 1987.

[34] British Standards Institution. BS EN 2: Classification of fires. 1992.

[35] JF Middleton. Developments in flame detectors. *Fire safety journal*, 6(3):175–182, 1983.

[36] JL Tissot, C Trouilleau, B Fieque, A Crastes, and O Legras. Uncooled microbolometer detector: recent developments at ulis. *Opto-Electronics Review*, 14(1):25–32, 2006.

[37] RK Bhan, RS Saxena, CR Jalwania, and SK Lomash. Uncooled infrared microbolometer arrays and their characterisation techniques (review paper). *Defence Science Journal*, 59(6):580–589, 2009.

[38] D Sabuncuoglu Tezcan, Selim Eminoglu, and Tayfun Akin. A low-cost uncooled infrared microbolometer detector in standard cmos technology. *Electron Devices, IEEE Transactions on*, 50(2):494–502, 2003.

[39] Christopher R Varney and Fred Gittes. Locating the source of projectile fluid droplets. *American Journal of Physics*, 79(8):838–842, 2011.

[40] John Eric Goff and Chinthaka Liyanage. Projectile motion gets the hose. *The Physics Teacher*, 49(7): 432–433, 2011.

[41] Boyd F Edwards, David D Sam, William A Booth, Leslie O Jessup, and Michael A Christensen. Angry birds realized: water balloon launcher for teaching projectile motion with drag. *European Journal of Physics*, 35(3), 2014.

[42] AP Hatton and MJ Osborne. The trajectories of large fire fighting jets. *International Journal of Heat and Fluid Flow*, 1(1):37–41, 1979.

[43] Tatsuya Miyashita, Osami Sugawa, Tomohiko Imamura, Kyoko Kamiya, and Yasuo Kawaguchi. Modeling and analysis of water discharge trajectory with large capacity monitor. *Fire Safety Journal*, 63: 1–8, 2014.

[44] Hunter Rouse, Joseph Warner Howe, and Donald E Metzler. Experimental investigation of fire monitors and nozzles. *Transactions of the American Society of Civil Engineers*, 117(1):1147–1175, 1952.

[45] Madjid Birouk and Nebojsa Lekic. Liquid jet breakup in quiescent atmosphere: A review. *Atomization and Sprays*, 19(6), 2009.

[46] N Rajaratnam, SAH Rizvi, PM Steffler, and PR Smy. An experimental study of very high velocity circular water jets in air. *Journal of Hydraulic Research*, 32(3):461–470, 1994.

[47] Takao Inamura and Nobuki Nagai. Spray characteristics of liquid jet traversing subsonic airstreams. *Journal of Propulsion and Power*, 13(2):250–256, 1997.

[48] HJ Holterman. *Kinetics and evaporation of water drops in air*, volume 2012. IMAG, 2003.

[49] Rollin Peter Grant and Stanley Middleman. Newtonian jet stability. *AIChE Journal*, 12(4):669–678, 1966.

[50] Adel Mansour and Norman Chigier. Effect of turbulence on the stability of liquid jets and the resulting droplet size distributions. *Atomization and Sprays*, 4(5):583–604, 1994.

[51] Ralph E Phinney. Breakup of a turbulent liquid jet in a low-pressure atmosphere. *AIChE Journal*, 21 (5):996–999, 1975.

[52] EG Arato, DA Crow, and DS Miller. *Investigations of a high performance water nozzle.* 1970.

[53] PH Schweitzer. Mechanism of disintegration of liquid jets. *Journal of Applied Physics*, 8(8):513–521, 1937.

[54] Dana W Lee and Robert C Spencer. Photomicrographic studies of fuel sprays. 1934.

[55] MJ McCarthy and NA Molloy. Review of stability of liquid jets and the influence of nozzle design. *The Chemical Engineering Journal*, 7(1):1–20, 1974.

[56] Fatih Erden, B Ugur Toreyin, E Birey Soyer, Ihsan Inac, Osman Gunay, Kivanc Kose, and A Enis Cetin. Wavelet based flickering flame detector using differential pir sensors. *Fire Safety Journal*, 53: 13–18, 2012.

[57] Ti Nguyen-Ti, Thuan Nguyen-Phuc, and Tuan Do-Hong. Fire detection based on video processing method. In *Advanced Technologies for Communications (ATC), 2013 International Conference on*, pages 106–110. IEEE, 2013.

[58] Byoung Chul Ko, Kwang-Ho Cheong, and Jae-Yeal Nam. Fire detection based on vision sensor and support vector machines. *Fire Safety Journal*, 44(3):322–329, 2009.

[59] I Bosch, S Gómez, and L Vergara. A ground system for early forest fire detection based on infrared signal processing. *International journal of remote sensing*, 32(17):4857–4870, 2011.

[60] Pedro Gomes, Pedro Santana, and José Barata. A vision-based approach to fire detection. *Int J Adv Robot Syst*, 11:149, 2014.

[61] Osman Günay, Kasım Taşdemir, B Uğur Töreyin, and A Enis Çetin. Video based wildfire detection at night. *Fire Safety Journal*, 44(6):860–868, 2009.

[62] Tao Chen, Hongyong Yuan, Guofeng Su, and Weicheng Fan. An automatic fire searching and suppression system for large spaces. *Fire Safety Journal*, 39(4):297 – 307, 2004. ISSN 0379-7112. doi: http://dx.doi.org/10.1016/j.firesaf.2003.11.007. URL http://www.sciencedirect.com/science/article/pii/S0379711203001383.

[63] Wolframalpha: Weather data kristiansand, norway. http://www.wolframalpha.com/input/?i=weather+data+kristiansand+07%2F03%2F2015, . Accessed: 2015-03-26.

[64] Wolframalpha: Weather data kristiansand, norway. http://www.wolframalpha.com/input/?i=weather+data+kristiansand+19%2F03%2F2015, . Accessed: 2015-03-26.

[65] Pengfei Guo, Xuezhi Wang, and Yingshi Han. The enhanced genetic algorithms for the optimization design. In *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, volume 7, pages 2990–2994. IEEE, 2010.

[66] Daniel S Weile and Eric Michielssen. Genetic algorithm optimization applied to electromagnetics: A review. *Antennas and Propagation, IEEE Transactions on*, 45(3):343–353, 1997.

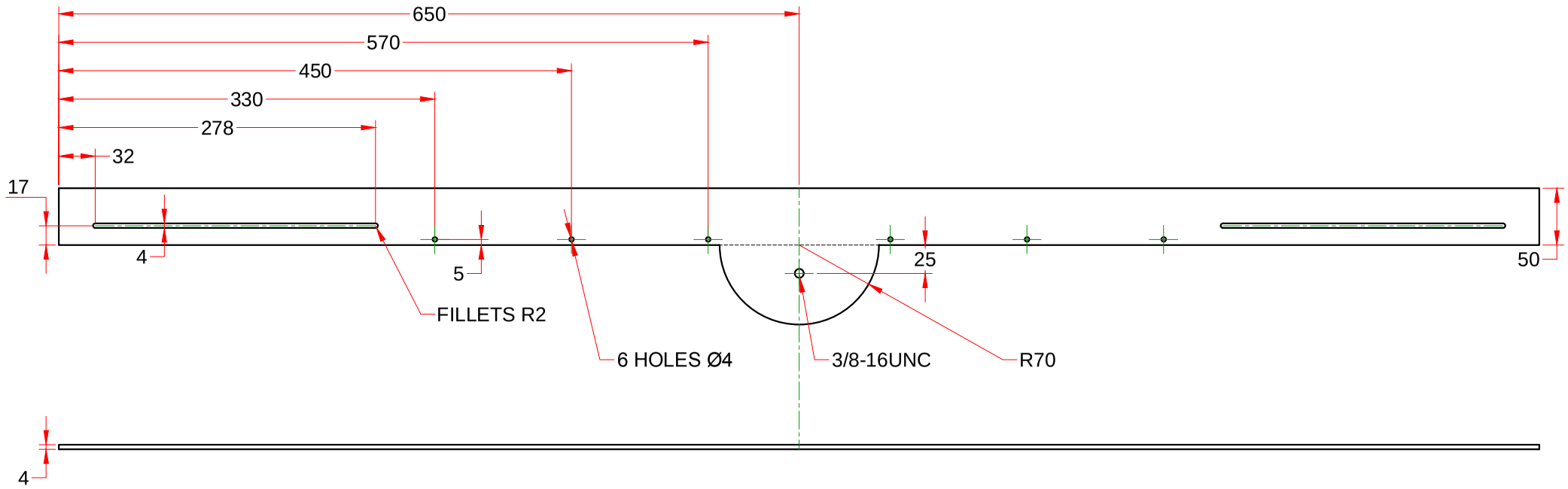
[67] Camera calibration with matlab. http://se.mathworks.com/videos/camera-calibration-with-matlab-81233.html. Accessed: 2015-04-13.

[68] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Hélene Laurent, and Christophe Rosenberger. Human detection with a multi-sensors stereovision system. In *Image and Signal Processing*, pages 228–235. Springer, 2010.

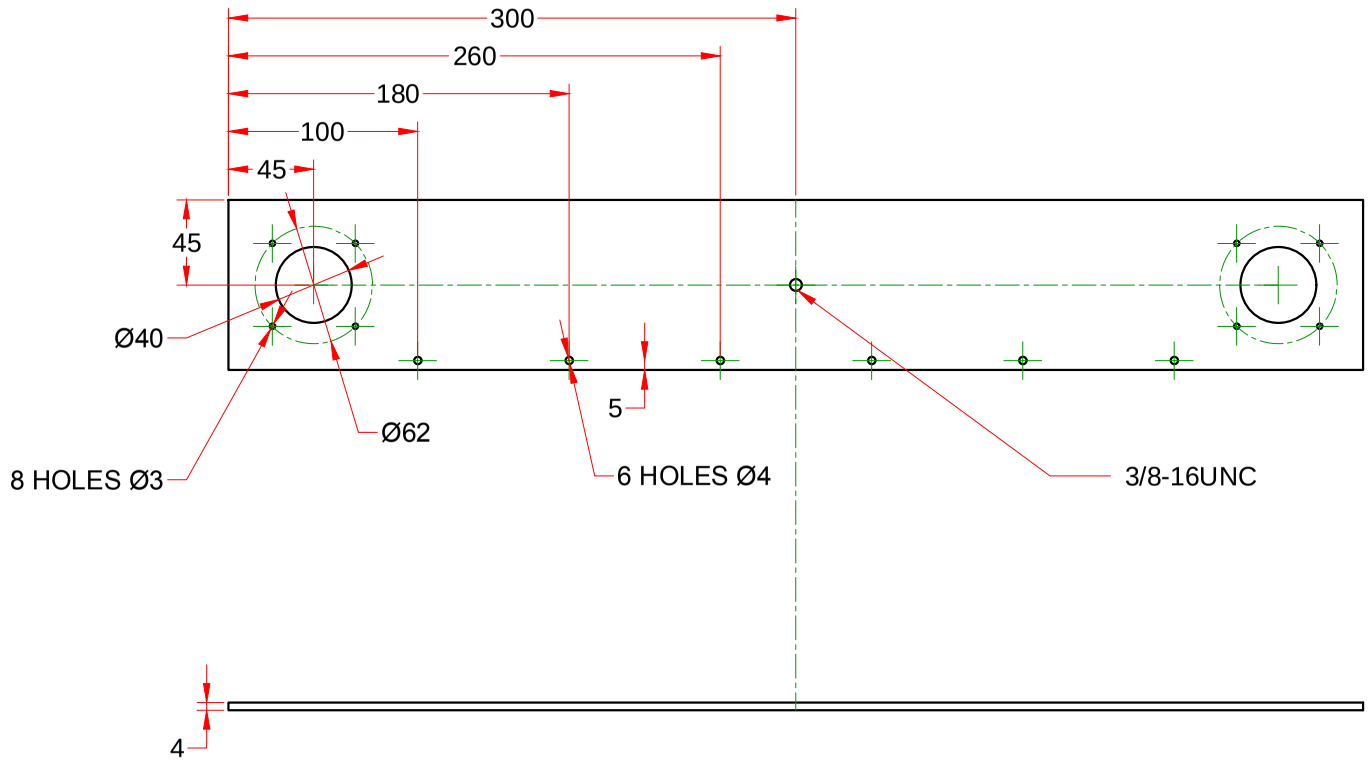
[69] Ke Wang, Lijun Zhao, and Ruifeng Li. Fisheye omnidirectional camera calibration pinhole or spherical model? *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference*, pages 873–877, 2014.

# APPENDIX A

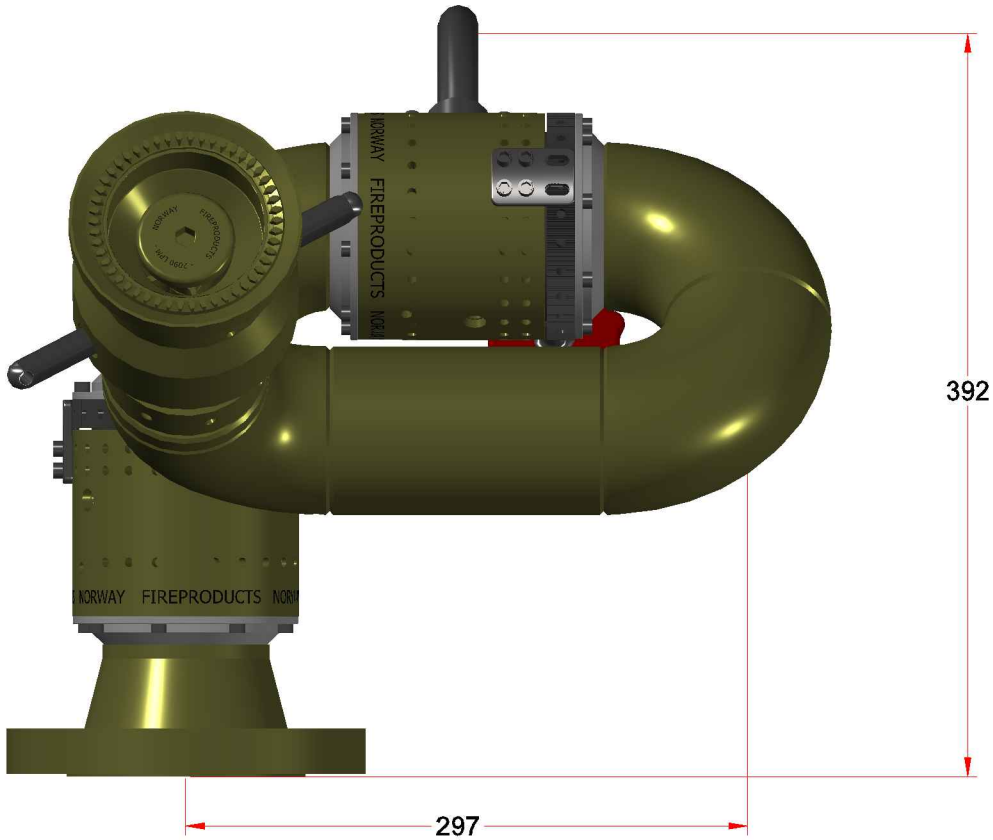
## Drawings



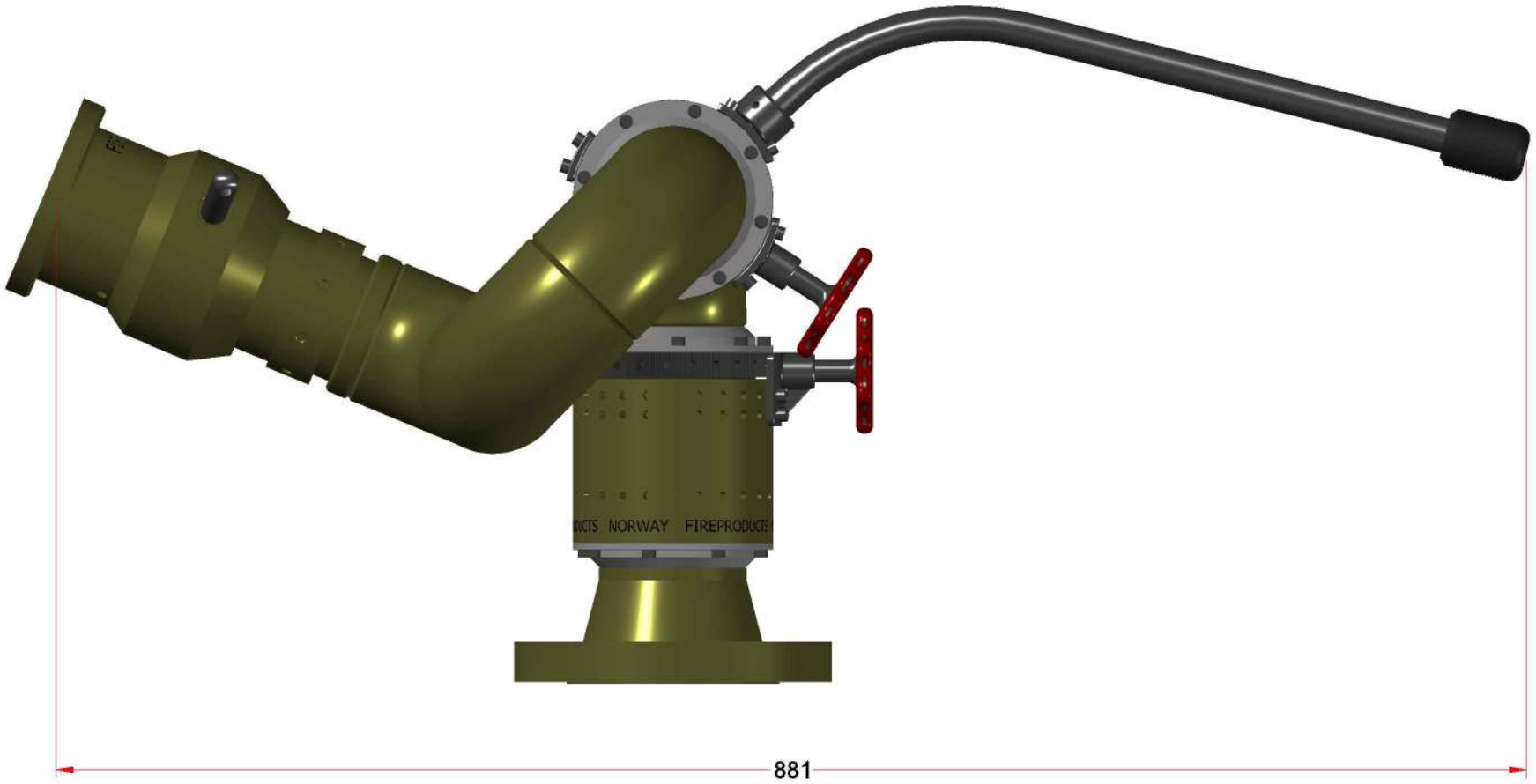

| | | | | | | |
|---|---|---|---|---|---|---|
| UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS TOLERANCES IN ACCORDANCE WITH DIN ISO 2768 T1 (MEDIUM) | FINISH: | | DEBUR AND BREAK SHARP EDGES | DO NOT SCALE | | REVISION: 1 |
| | NAME | SIGNATURE | DATE | TITLE: CAMERA RACK | | |
| DRAWN | JULIAN | JLH | 15/4-15 | | | |
| CHK'D | JARED | JH | 15/4-15 | | | |
| APPR'VD | | | | | | |
| MFG | | | | | | |
| Q.A | | | | | | |
| | | | | MATERIAL: STEEL | DWG NO. 1 | A4 |
| | | | | MASS: 2.3 kg | SCALE=1:5 | SHEET 1 OF 1 |




| | | | | | | |
|---|---|---|---|---|---|---|
| UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS TOLERANCES IN ACCORDANCE WITH DIN ISO 2768 T1 (MEDIUM) | FINISH: | | DEBUR AND BREAK SHARP EDGES | DO NOT SCALE | | REVISION: 1 |
| | NAME | SIGNATURE | DATE | | | |
| DRAWN | JULIAN | JLH | 15/4-15 | TITLE: SENSOR RACK | | |
| CHK'D | JARED | JH | 15/4-15 | | | |
| APPR'VD | | | | | | |
| MFG | | | | | | |
| Q.A | | | | MATERIAL: STEEL | DWG NO. 2 | A4 |
| | | | | MASS: 1.8 kg | SCALE=1:4 | SHEET 1 OF 1 |




| | | | | | | |
|---|---|---|---|---|---|---|
| UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS TOLERANCES IN ACCORDANCE WITH DIN ISO 2768 T1 (MEDIUM) | FINISH: | | | | DEBUR AND BREAK SHARP EDGES | DO NOT SCALE / REVISION: 1 |
| | NAME | SIGNATURE | DATE | | | TITLE: FIRE MONITOR ASSEMBLY VIEW 1 |
| DRAWN | JULIAN | JLH | 21/4-15 | | | |
| CHK'D | JARED | JH | 21/4-15 | | | |
| APPR'VD | | | | | | |
| MFG | | | | | | |
| Q.A | | | | | | |
| | | | | MATERIAL: VARIOUS | | DWG NO. 3 / A4 |
| | | | | MASS: APPROXIMATELY 50 kg | | SCALE=1:4 / SHEET 1 OF 1 |




| | | | | | | |
|---|---|---|---|---|---|---|
| UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS TOLERANCES IN ACCORDANCE WITH DIN ISO 2768 T1 (MEDIUM) | FINISH: | | | | DEBUR AND BREAK SHARP EDGES | DO NOT SCALE / REVISION: 1 |
| | NAME | SIGNATURE | DATE | | | TITLE: FIRE MONITOR ASSEMBLY VIEW 2 |
| DRAWN | JULIAN | JLH | 21/4-15 | | | |
| CHK'D | JARED | JH | 21/4-15 | | | |
| APPR'VD | | | | | | |
| MFG | | | | | | |
| Q.A | | | | | | |
| | | | | MATERIAL: | | DWG NO. 3 / A4 |
| | | | | MASS: APPROXIMATELY 50 kg | | SCALE=1:4 / SHEET 1 OF 1 |

# APPENDIX B

## LabVIEW

Relevant parts of the LabVIEW programs will be shown here. Parts which are not relevant for the function is left out.

## B.1 Wind Data Processing

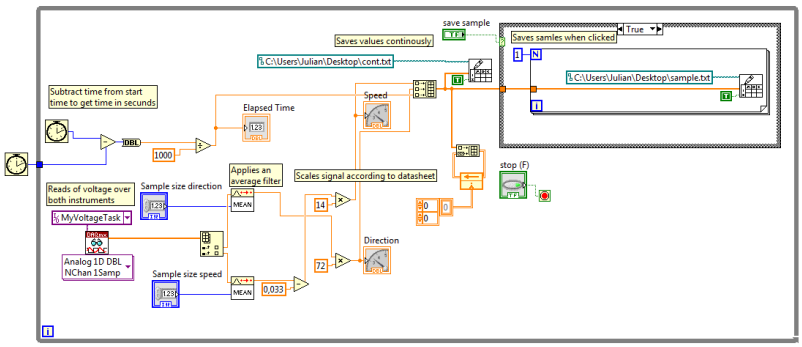



Figure B.1: Data processing in LabVIEW.

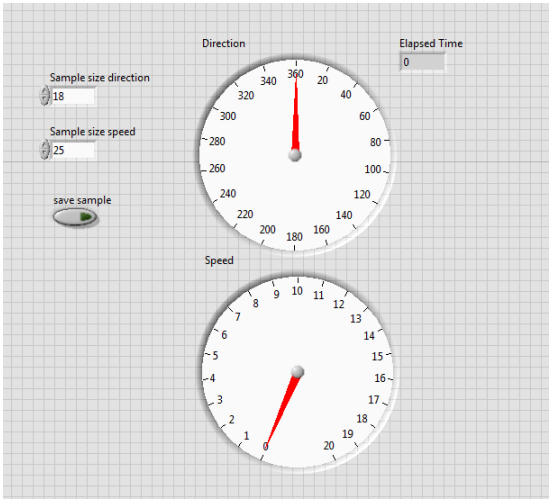



Figure B.2: Front panel for data processing in LabVIEW.

## B.2 Communication between PLC and model

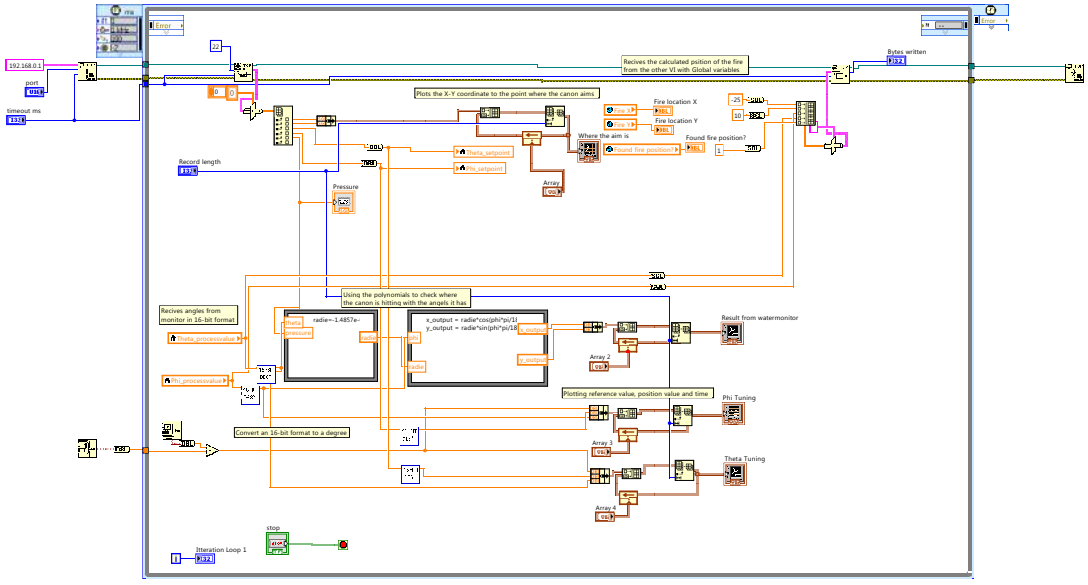



Figure B.3: Block diagram for communication end verification in LabVIEW.

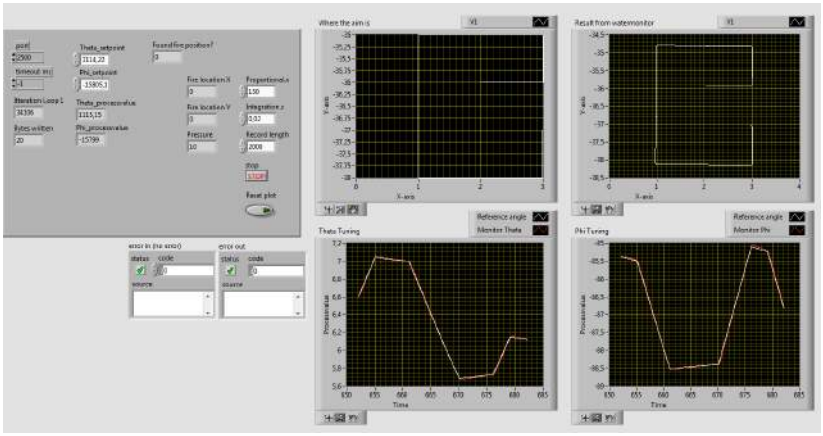



Figure B.4: Front panel for communication end verification in LabVIEW.

## B.3 Image processing

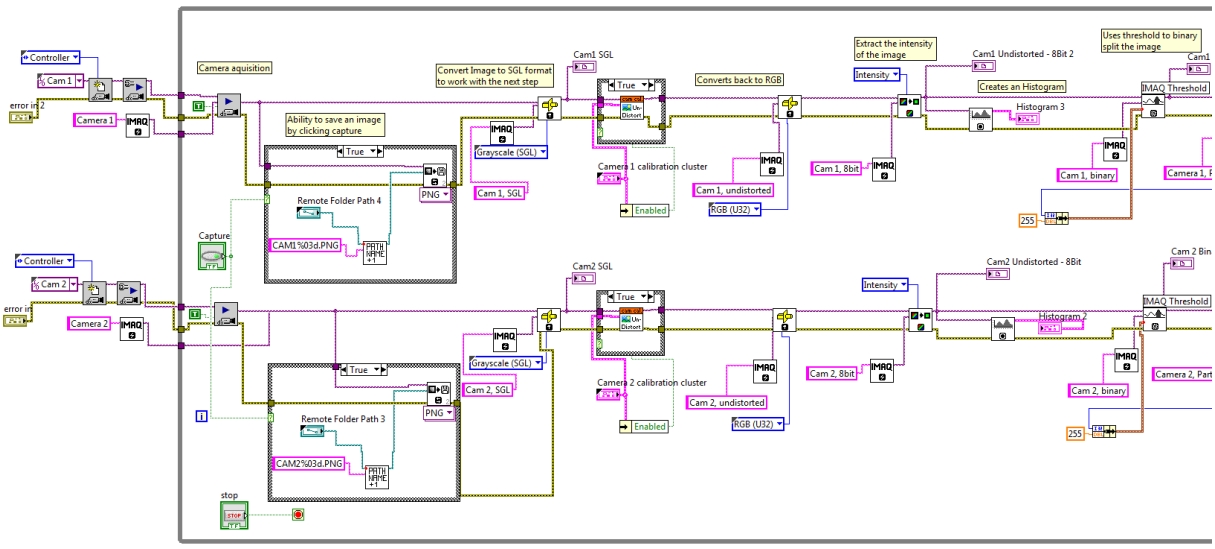



Figure B.5: Left part of block diagram for image acquisition and processing in LabVIEW

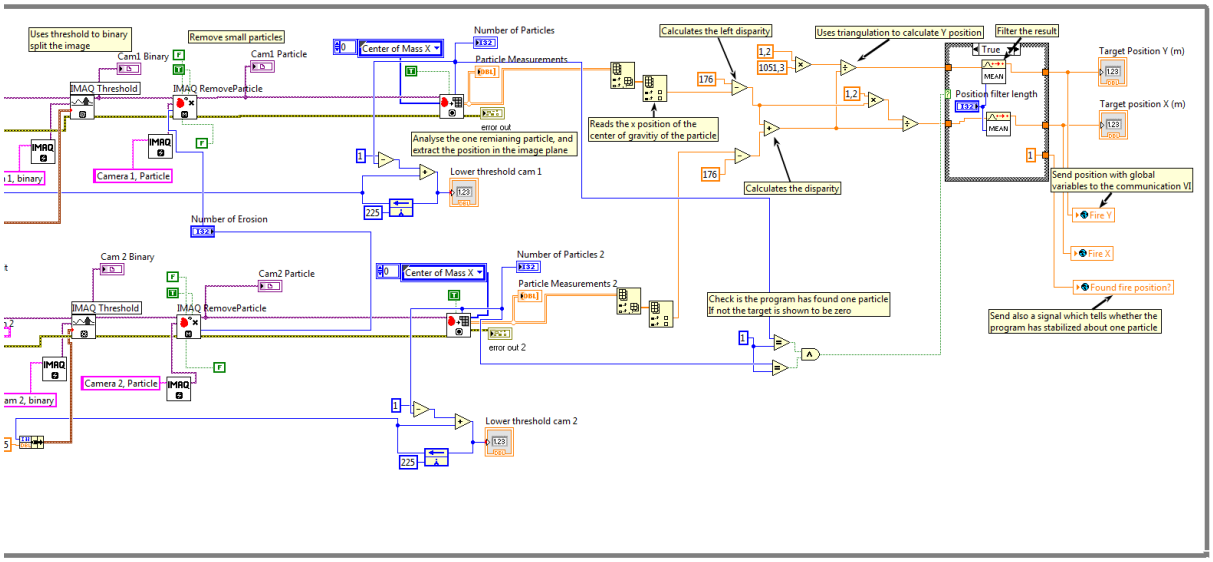



Figure B.6: Right part of block diagram for image acquisition and processing in LabVIEW.

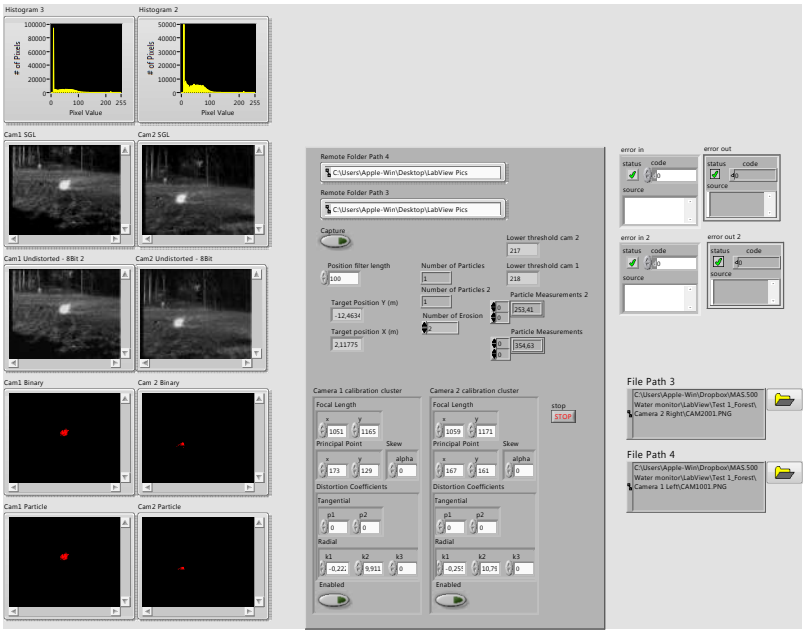



Figure B.7: Front panel for image acquisition and processing in LabVIEW.

## B.4 Dynamic model

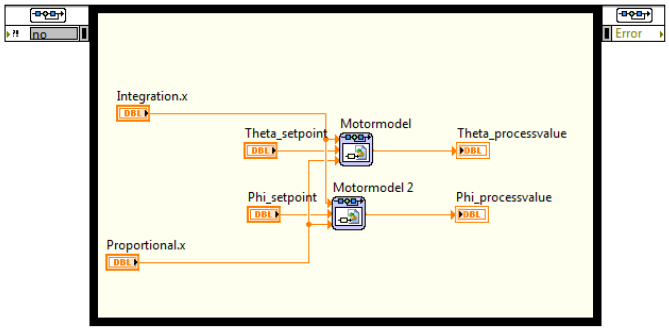



Figure B.8: Block diagram for model simulation in LabVIEW.

## B.5 16-bit conversion

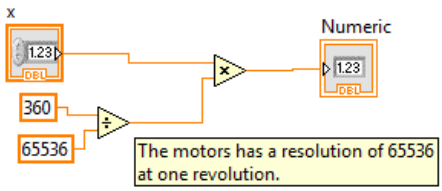



Figure B.9: Block diagram for signal conversion in LabVIEW.

## B.6 Name index generator

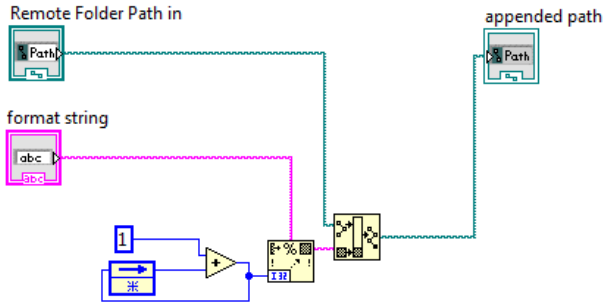



Figure B.10: Block diagram for name generator in LabVIEW.

## B.7 Image rectification

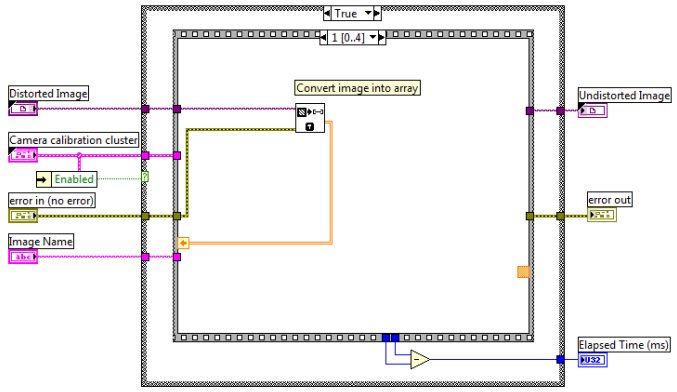



Figure B.11: Block diagram for rectification of images, window 1.

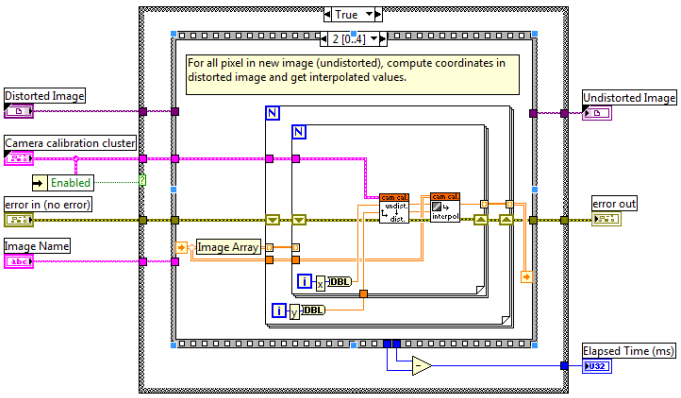



Figure B.12: Block diagram for rectification of images, window 2.

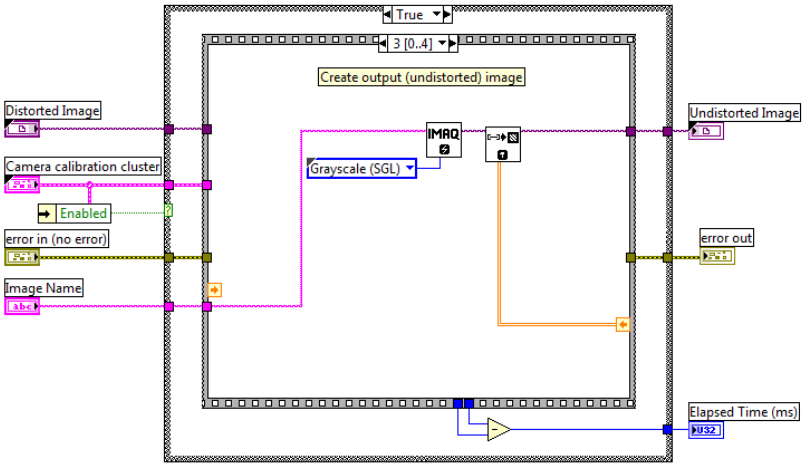



Figure B.13: Block diagram for rectification of images, window 3.

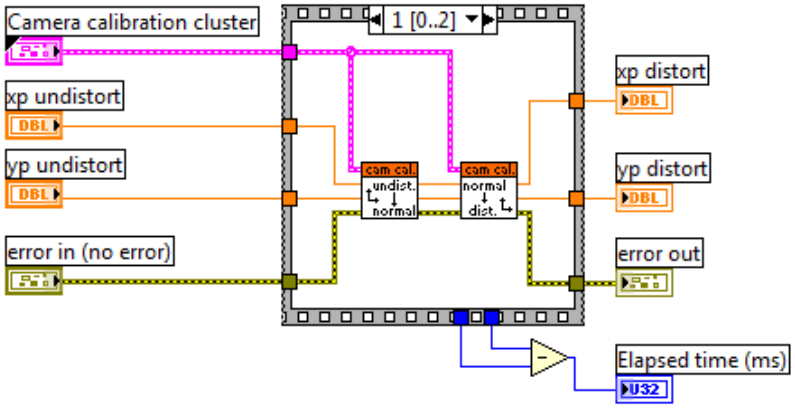



Figure B.14: Block diagram for rectification of images, window 4.

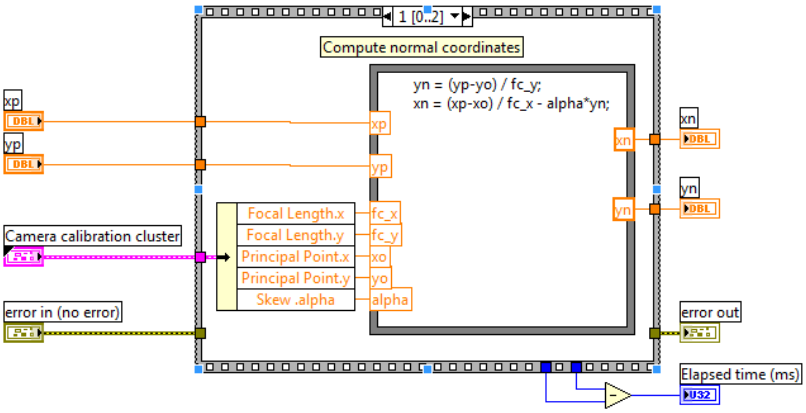



Figure B.15: Block diagram for rectification of images, window 5.

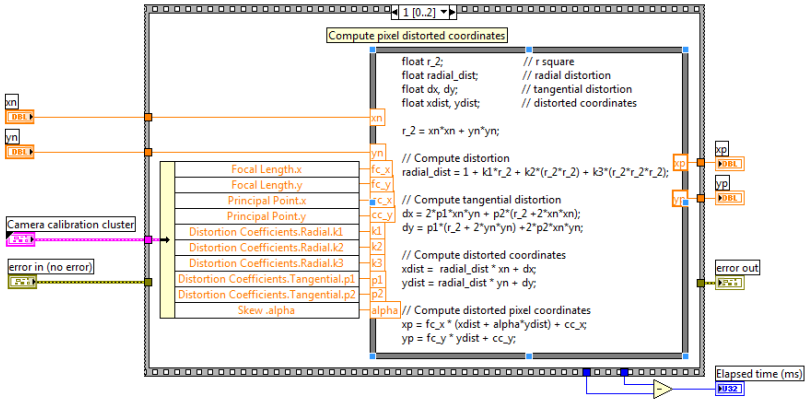



Figure B.16: Block diagram for rectification of images, window 6.

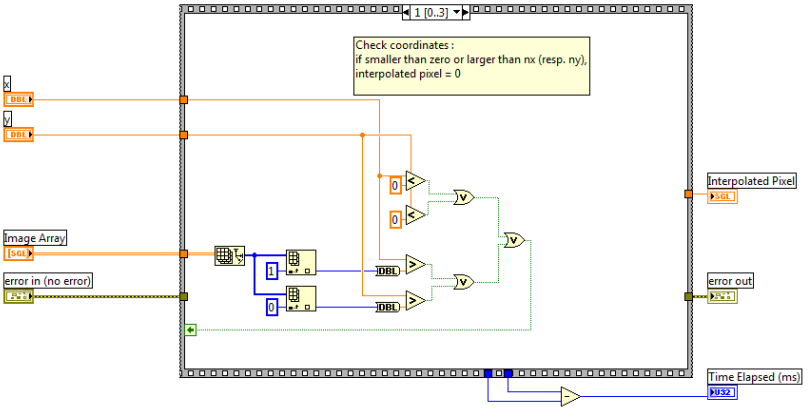



Figure B.17: Block diagram for rectification of images, window 7.

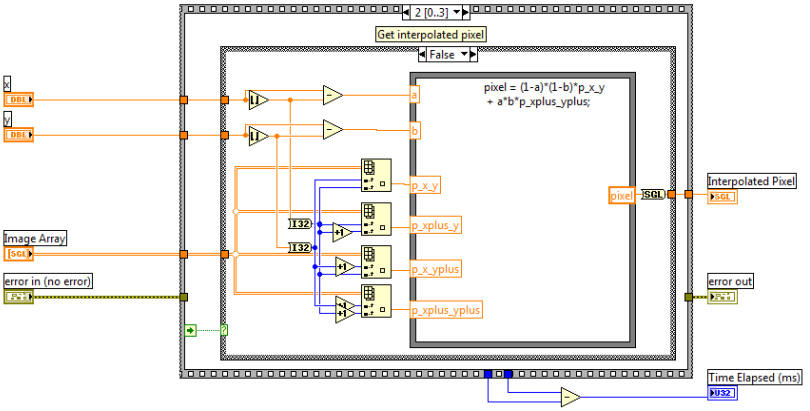



Figure B.18: Block diagram for rectification of images, window 8.

# APPENDIX C

## MATLAB

```
%%% Minimization process: GA and Gradient solver%%%
close all
clear all
clc
%Import sensory data from text file
adapt_data = dlmread('measureddata3.txt');

%%%GA solver%%%
%Defines minimizer function
funObj = @(guess)Parameter_objective(guess,adapt_data);

%Defines upper- and lower bounds
Lb = [0 -5];
Ub = [5 5];
%Sets options for GA.
options = gaoptimset('Display','iter',...
    'TolFun',1e-3,'PopulationSize',1000);
%Calls the GA function
[result,fvall] = ga(funObj,2,[],[],[],[],Lb,Ub,[],options);

%Recieves the result from the minimized function to plot the result and
%calculat the deviation.
R1 = result(1)
R2 = result(2)

%%%Gradient solver%%%
%Defines minimizer function
funObj = @(guess)Parameter_objective(guess,adapt_data);
%Sets options for fminsearch()
options = optimset('MaxFunEvals',3000,'MaxIter',1000,'TolFun',1e-6,...
    'Display','iter','Algorithm','interior-point');
guess = [R1 R2];
lb = [0 0];
ub = [0.1 0.4]
[result,fvall] = fmincon(funObj,guess,[],[],[],[],lb,ub,[],options);

%Recieves the result from the minimized function to plot the result and
%calculat the deviation.
k = result(1);
A = result(2);

%%%%%%%%% Showing the results graphically %%%%%%%%
%Imports the test data
t=adapt_data(1:size(adapt_data,1),1);
wind_speed=adapt_data(1:size(adapt_data,1),2);
wind_direction=adapt_data(1:size(adapt_data,1),3);
x_goal=adapt_data(1:size(adapt_data,1),4);
y_goal=adapt_data(1:size(adapt_data,1),5);
theta_opt=adapt_data(1:size(adapt_data,1),7);
phi_opt=adapt_data(1:size(adapt_data,1),8);
initial_speed=adapt_data(1:size(adapt_data,1),9);
pressure=adapt_data(1:size(adapt_data,1),10);

%Basic data
g=9.81;
m=10;
%Air resitance data
for i=1:size(adapt_data,1);
clear x_Plot_drag;
clear y_Plot_drag;
clear z_Plot_drag;

%Water stream conditions
v=10.5152*sqrt(pressure(i));
```

```
theta=(90-theta_opt(i))*pi/180;
phi=(phi_opt(i))*pi/180;

%Wind parameters
% v_wx=cos((90-wind_direction(i))*(pi/180))*wind_speed(i);
v_wx=0;
v_wy=0;
v_wz=0;

%Initializes parameters
x_Init=0;
xDot_Init=v*cos(phi)*sin(theta);
y_Init=0;
yDot_Init=v*sin(phi)*sin(theta);
z_Init=3.35+0.45*sin(theta);
zDot_Init=v*cos(theta);

%Initialize
x=x_Init;
xDot=xDot_Init;
y=y_Init;
yDot=yDot_Init;
z=z_Init;
zDot=zDot_Init;

%Initialize counters so that plot data is only saved once pr. a number of
%time steps corresponding to ReportInterval
ReportCounter=0;
ReportInterval=1;
Counter=ReportInterval;

%Start time integration
Time=0;
StepTime=1e-3;
distance=0;
while z>0
    v_rel_x=xDot-v_wx;
    v_rel_y=yDot-v_wy;
    v_rel_z=zDot-v_wz;

    speed=sqrt(xDot^2+yDot^2+zDot^2);

    s=distance;

%     Compute Air drag
%     F_Dx=-sign(v_rel_x)*k*(v_rel_x)^2;
%     F_Dy=-sign(v_rel_y)*k*(v_rel_y)^2;
%     F_Dz=-sign(v_rel_z)*k*(v_rel_z)^2;
%
    F_Dx=-sign(v_rel_x)*k*(v_rel_x)^2*(1+exp(A*s));
    F_Dy=-sign(v_rel_y)*k*(v_rel_y)^2*(1+exp(A*s));
    F_Dz=-sign(v_rel_z)*k*(v_rel_z)^2*(1+exp(A*s));

%     F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2*(1+exp(A1*s));
%     F_Dy=-sign(v_rel_y)*k2*(v_rel_y)^2*(1+exp(A2*s));
%     F_Dz=-sign(v_rel_z)*k3*(v_rel_z)^2*(1+exp(A3*s));

%     F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2*(1+A1*s+B1*s^2+C1*s^n1);
%     F_Dy=-sign(v_rel_y)*k1*(v_rel_y)^2*(1+A1*s+B1*s^2+C1*s^n1);
%     F_Dz=-sign(v_rel_z)*k1*(v_rel_z)^2*(1+A1*s+B1*s^2+C1*s^n1);
%
%     F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2*(1+A1*s+B1*s^2+C1*s^n1);
%     F_Dy=-sign(v_rel_y)*k2*(v_rel_y)^2*(1+A2*s+B2*s^2+C2*s^n2);
%     F_Dz=-sign(v_rel_z)*k3*(v_rel_z)^2*(1+A3*s+B3*s^2+C3*s^n3);
```

```
%     F_Dx=-sign(v_rel_x)*(v_rel_x)^2*C1*((X1-1)*exp(-Y1*s)+1);
%     F_Dy=-sign(v_rel_y)*(v_rel_y)^2*C2*((X2-1)*exp(-Y2*s)+1);
%     F_Dz=-sign(v_rel_z)*(v_rel_z)^2*C3*((X3-1)*exp(-Y3*s)+1);

%     F_Dx=-sign(v_rel_x)*(v_rel_x)^2*k*s^n;
%     F_Dy=-sign(v_rel_y)*(v_rel_y)^2*k*s^n;
%     F_Dz=-sign(v_rel_z)*(v_rel_z)^2*k*s^n;

    %Compute accelerations
    xDotDot=0+F_Dx/m;
    yDotDot=0+F_Dy/m;
    zDotDot=-g+F_Dz/m;

    if z<0;
        yDot=0;
        xDot=0;
        zDot=0;
    end


    %report
    if Counter==ReportInterval
        Counter=0;
        ReportCounter=ReportCounter+1;
        Time_Plot_drag(ReportCounter)=Time;
        x_Plot_drag(ReportCounter)=x;
        xDot_Plot_drag(ReportCounter)=xDot;
        y_Plot_drag(ReportCounter)=y;
        yDot_Plot_drag(ReportCounter)=yDot;
        z_Plot_drag(ReportCounter)=z;
        zDot_Plot_drag(ReportCounter)=zDot;

    end;
    %Time integrate
    x=x+xDot*StepTime;
    xDot=xDot+xDotDot*StepTime;
    y=y+yDot*StepTime;
    yDot=yDot+yDotDot*StepTime;
    z=z+zDot*StepTime;
    zDot=zDot+zDotDot*StepTime;
    Time=Time+StepTime;
    distance=distance+speed*StepTime;
    Counter=Counter+1;
end;
%Calculates the radial deviation from measured results
e_x=x-x_goal(i);
e_y=y-y_goal(i);
r(i,1) = sqrt(e_x^2+e_y^2);

%Plots the trajectories
plot3(x_Plot_drag,y_Plot_drag,z_Plot_drag,'r')
axis equal
xlabel('x-axis');
ylabel('y-axis');
zlabel('z-axis');
grid on;
hold on;
i;


end
%Compute deviation
deviation=r
```

```
%Compute mean of deviation
average=mean(r)
%Show optimized parameters
res=result'
```

```
%%% Objective function to minimize%%%
function E=Parameter_objective(guess,adapt_data)
k1 = guess(1);
k2 = guess(2);
k3 = guess(3);
% C = guess(4);
% n = guess(5);

%Imports the test data
t=adapt_data(1:size(adapt_data,1),1);
wind_speed=adapt_data(1:size(adapt_data,1),2);
wind_direction=adapt_data(1:size(adapt_data,1),3);
x_goal=adapt_data(1:size(adapt_data,1),4);
y_goal=adapt_data(1:size(adapt_data,1),5);
theta_opt=adapt_data(1:size(adapt_data,1),7);
phi_opt=adapt_data(1:size(adapt_data,1),8);
initial_speed=adapt_data(1:size(adapt_data,1),9);
pressure=adapt_data(1:size(adapt_data,1),10);

%Basic data
g=9.81;
m=10;
r = zeros(1,size(adapt_data,1));
for i=1:size(adapt_data,1);
clear x_Plot_drag;
clear y_Plot_drag;
clear z_Plot_drag;

%Water stream conditions
v=initial_speed(i);
theta=(90-theta_opt(i))*pi/180;
phi=(phi_opt(i))*pi/180;

%Wind parameters
v_wx=cos((wind_direction(i))*(pi/180))*wind_speed(i);
v_wy=sin((wind_direction(i))*(pi/180))*wind_speed(i);
% v_wx=0;
% v_wy=0;
v_wz=0;

%Initialize the parameters
x_Init=0;
xDot_Init=v*cos(phi)*sin(theta);
y_Init=0;
yDot_Init=v*sin(phi)*sin(theta);
z_Init=3.35+0.45*sin(theta);
zDot_Init=v*cos(theta);

%Initialize
x=x_Init;
xDot=xDot_Init;
y=y_Init;
yDot=yDot_Init;
z=z_Init;
zDot=zDot_Init;

%Initialize counters so that plot data is only saved once pr. a number of
%time steps corresponding to ReportInterval
ReportCounter=0;
ReportInterval=10;
Counter=ReportInterval;

%Start time integration
Time=0;
```

```
StepTime=1e-3;
distance=0;
while z>0
    v_rel_x=xDot-v_wx;
    v_rel_y=yDot-v_wy;
    v_rel_z=zDot-v_wz;

    speed=sqrt(xDot^2+yDot^2+zDot^2);

    s=distance;

    %Compute Air drag


    F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2;
    F_Dy=-sign(v_rel_y)*k2*(v_rel_y)^2;
    F_Dz=-sign(v_rel_z)*k3*(v_rel_z)^2;

%     F_Dx=-sign(v_rel_x)*k*(v_rel_x)^2*(1+exp(A*s));
%     F_Dy=-sign(v_rel_y)*k*(v_rel_y)^2*(1+exp(A*s));
%     F_Dz=-sign(v_rel_z)*k*(v_rel_z)^2*(1+exp(A*s));

%     F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2*(1+exp(A1*s));
%     F_Dy=-sign(v_rel_y)*k2*(v_rel_y)^2*(1+exp(A2*s));
%     F_Dz=-sign(v_rel_z)*k3*(v_rel_z)^2*(1+exp(A3*s));

%     F_Dx=-sign(v_rel_x)*k1*(v_rel_x)^2*(1+A1*s+B1*s^2+C1*s^n1);
%     F_Dy=-sign(v_rel_y)*k1*(v_rel_y)^2*(1+A1*s+B1*s^2+C1*s^n1);
%     F_Dz=-sign(v_rel_z)*k1*(v_rel_z)^2*(1+A1*s+B1*s^2+C1*s^n1);

%     F_Dx=-sign(v_rel_x)*k*(v_rel_x)^2*(1+A*s+B*s^2+C*s^n);
%     F_Dy=-sign(v_rel_y)*k*(v_rel_y)^2*(1+A*s+B*s^2+C*s^n);
%     F_Dz=-sign(v_rel_z)*k*(v_rel_z)^2*(1+A*s+B*s^2+C*s^n);

%     F_Dx=-sign(v_rel_x)*(v_rel_x)^2*C*((X-1)*exp(-Y*s)+1);
%     F_Dy=-sign(v_rel_y)*(v_rel_y)^2*C*((X-1)*exp(-Y*s)+1);
%     F_Dz=-sign(v_rel_z)*(v_rel_z)^2*C*((X-1)*exp(-Y*s)+1);

%     F_Dx=-sign(v_rel_x)*(v_rel_x)^2*k*s^n;
%     F_Dy=-sign(v_rel_y)*(v_rel_y)^2*k*s^n;
%     F_Dz=-sign(v_rel_z)*(v_rel_z)^2*k*s^n;

    %Compute accelerations
    xDotDot=0+F_Dx/m;
    yDotDot=0+F_Dy/m;
    zDotDot=-g+F_Dz/m;

    if z<0;
        yDot=0;
        xDot=0;
        zDot=0;
    end

    %Time integrate
    x=x+xDot*StepTime;
    xDot=xDot+xDotDot*StepTime;
    y=y+yDot*StepTime;
    yDot=yDot+yDotDot*StepTime;
    z=z+zDot*StepTime;
    zDot=zDot+zDotDot*StepTime;
    Time=Time+StepTime;
    distance=distance+speed*StepTime;
end;
%Calculates the error in each direction, and radial error
```

```
e_x = x-x_goal(i);
e_y = y-y_goal(i);

r(i) = sqrt(e_x^2+e_y^2);
end
%Calculates the function error
E = r*r';
```

```
%%% Make trajectory lengt vs. angle for validated model%%%
close all
clear all
clc

%Pressure to make polynomials of
pressure=[5 10];
%Angles to make polynomais of
theta_x=linspace(85,45,1000)';
%Preallocate space in memory
eqnarray(1:2000,1:3) = 0;


for j=1:length(pressure);
%Wind parameters
v_wx=0;
v_wy=0;
v_wz=0;

v=10.5152*sqrt(pressure(j));
p=pressure(j);
for o=1:length(theta_x);
%Coefficient from minimized model
fac=[3.14456184529967e-06;0.289873924387303];
k=fac(1);
A=fac(2);


%Basic data
g=9.81;
m=10;

%Water stream conditions
theta_d=theta_x(o);
theta=theta_x(o)*pi/180;
phi=(0)*pi/180;

%Initializes conditions
x_Init=0;
xDot_Init=v*cos(phi)*sin(theta);
y_Init=0;
yDot_Init=v*sin(phi)*sin(theta);
z_Init=1.5+0.53*sin(theta);
zDot_Init=v*cos(theta);

%Initialize
x=x_Init;
xDot=xDot_Init;
y=y_Init;
yDot=yDot_Init;
z=z_Init;
zDot=zDot_Init;

%Initialize counters so that plot data is only saved once pr. a number of
%time steps corresponding to ReportInterval
ReportCounter=0;
ReportInterval=1;
Counter=ReportInterval;

%Start time integration
Time=0;
StepTime=1e-3;
distance=0;
```

```
while z>0
    %Calculates relative velocity
    v_rel_x=xDot-v_wx;
    v_rel_y=yDot-v_wy;
    v_rel_z=zDot-v_wz;
    %Calculates speed
    speed=sqrt(xDot^2+yDot^2+zDot^2);
    %Calculates traveled distance
    s=distance;

    %Compute Air drag with minimized model
    F_Dx=-sign(v_rel_x)*k*(v_rel_x)^2*(1+exp(A*s));
    F_Dy=-sign(v_rel_y)*k*(v_rel_y)^2*(1+exp(A*s));
    F_Dz=-sign(v_rel_z)*k*(v_rel_z)^2*(1+exp(A*s));

    %Compute accelerations
    xDotDot=0+F_Dx/m;
    yDotDot=0+F_Dy/m;
    zDotDot=-g+F_Dz/m;

    if z<0;
        yDot=0;
        xDot=0;
        zDot=0;
    end

    %report
    if Counter==ReportInterval
        Counter=0;
        ReportCounter=ReportCounter+1;
        Time_Plot_drag(ReportCounter)=Time;
        x_Plot_drag(ReportCounter)=x;
        xDot_Plot_drag(ReportCounter)=xDot;
        y_Plot_drag(ReportCounter)=y;
        yDot_Plot_drag(ReportCounter)=yDot;
        z_Plot_drag(ReportCounter)=z;
        zDot_Plot_drag(ReportCounter)=zDot;

    end;
    %Time integrate
    x=x+xDot*StepTime;
    xDot=xDot+xDotDot*StepTime;
    y=y+yDot*StepTime;
    yDot=yDot+yDotDot*StepTime;
    z=z+zDot*StepTime;
    zDot=zDot+zDotDot*StepTime;
    Time=Time+StepTime;
    distance=distance+speed*StepTime;
    Counter=Counter+1;
end;
%Put into common matrix
eqnarray((o+length(theta_x)*j-length(theta_x)),1)=p; %Pressure in bars
eqnarray((o+length(theta_x)*j-length(theta_x)),2)=theta_d; %Theta in degrees
eqnarray((o+length(theta_x)*j-length(theta_x)),3)=x; %Landing spot in meters
end
end
%Export all data
dlmwrite('eqnarray',eqnarray)
```

```
%%% Make polynomial out of points%%%
close all
clear all
clc
%Sets MatLab to write 10 digits answear
format long
%Import data that is to be fitted
adapt_data = dlmread('eqnarray');

%Read input file, and parts them into an matrix for each curve
t1=adapt_data(1:600,1:3);
%Defines the parameters in the function
pressure1=t1(:,1); %Pressure
theta1=t1(:,2); %Theta
length1=t1(:,3); %Length

t2=adapt_data(1100:1250,1:3);
%Defines the parameters in the function
pressure2=t2(:,1); %Pressure
theta2=t2(:,2); %Theta
length2=t2(:,3); %Length

%Uses function fitting
p1 = polyfitn(length1,theta1,4)
p2 = polyfitn(length2,theta2,4)

%Shows polynomials in symbolic form
if exist('sympoly') == 2
  polyn2sympoly(p1)
  polyn2sympoly(p2)
end

%Plots
%Evaluates function values for fitted polynomials
theta_fit1 = polyvaln(p1,length1);
theta_fit2 = polyvaln(p2,length2);

%Plots the input value 5 bar
plot(length1,theta1,'r','LineWidth',4)
hold on;
%Plots the polynomial 5 bar
plot(length1,theta_fit1,'b--','LineWidth',4);
%Plots the input value 10 bar
plot(length2,theta2,'k','LineWidth',4)
%Plots the polynomial 10 bar
plot(length2,theta_fit2,'m--','LineWidth',4);
legend('Drag model: 5 bar','Polynomial approximation: 5 bar','Drag model: 10
bar','Polynomial approximation: 10 bar')
xlabel('Length [m]');
ylabel('Theta [deg]');
```

# APPENDIX D

## PLC

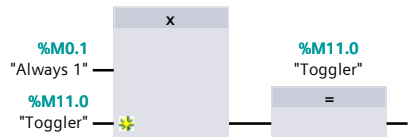
# Program blocks

## Program [OB35]

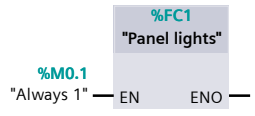
| Program Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Program | **Number** | 35 | **Type** | OB |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | "Cyclic Interrupt" | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Temp | | |
| OB35_EV_CLASS | Byte | 0.0 |
| OB35_STRT_INF | Byte | 1.0 |
| OB35_PRIORITY | Byte | 2.0 |
| OB35_OB_NUMBR | Byte | 3.0 |
| OB35_RESERVED_1 | Byte | 4.0 |
| OB35_RESERVED_2 | Byte | 5.0 |
| OB35_PHASE_OFFSET | Word | 6.0 |
| OB35_RESERVED_3 | Int | 8.0 |
| OB35_EXC_FREQ | Int | 10.0 |
| OB35_DATE_TIME | Date_And_Time | 12.0 |

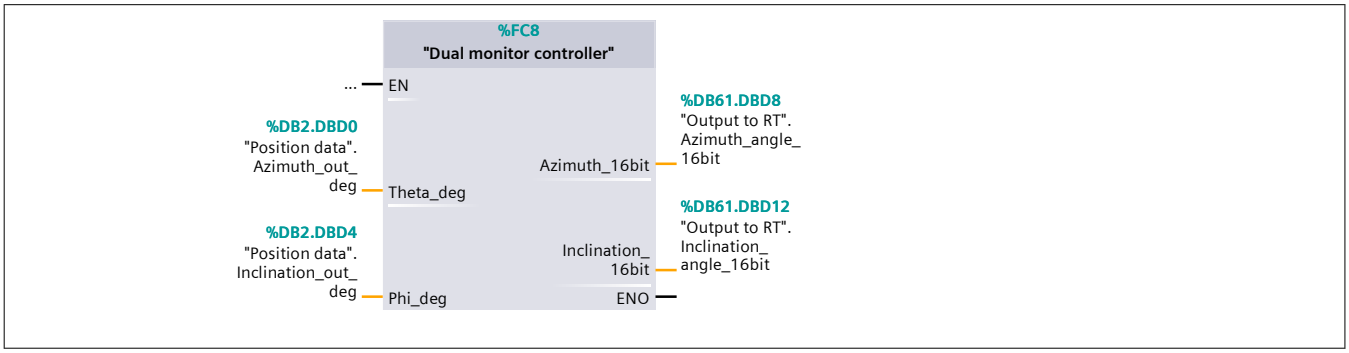
### Network 1: Network for toggling Read/Write to LabView



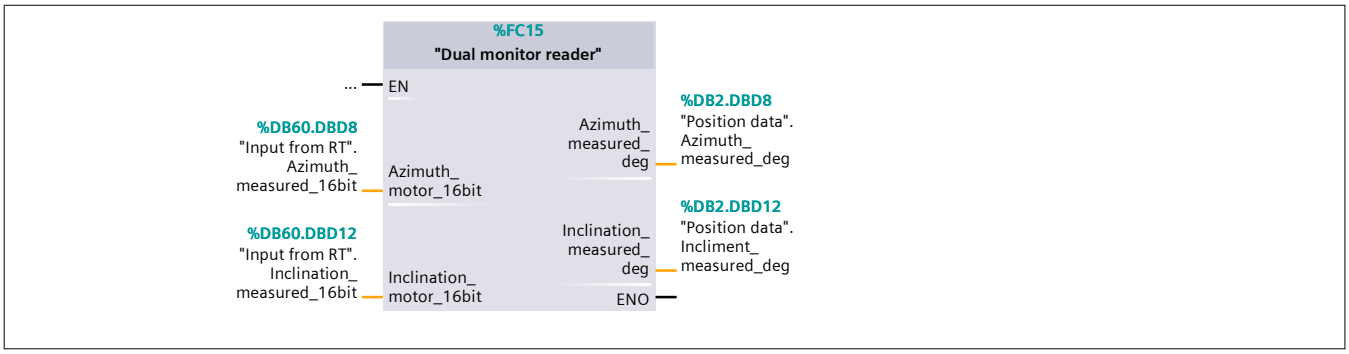

### Network 2: Panel lights



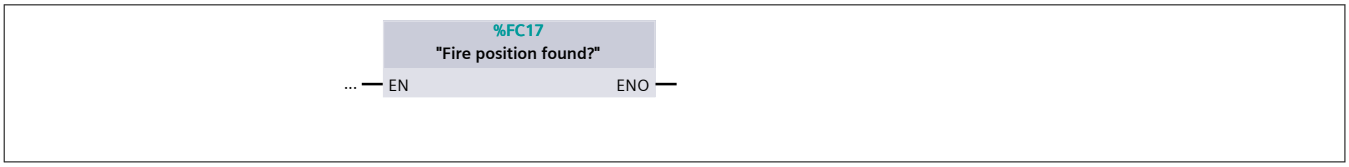

### Network 3: Converts degrees to 16-bit format for setpoint



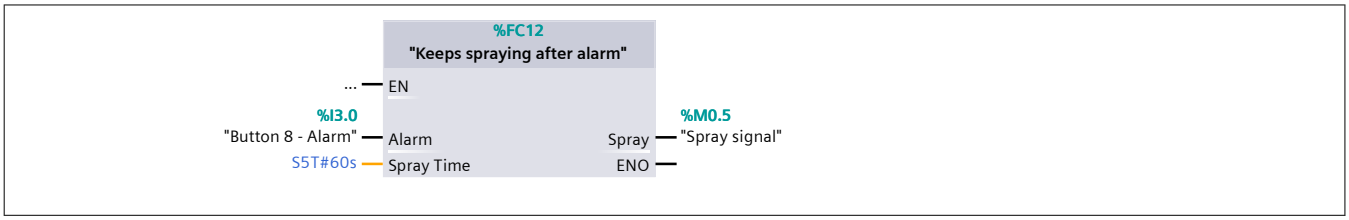

## Network 4: Converts 16-bit format as processvalue to degrees



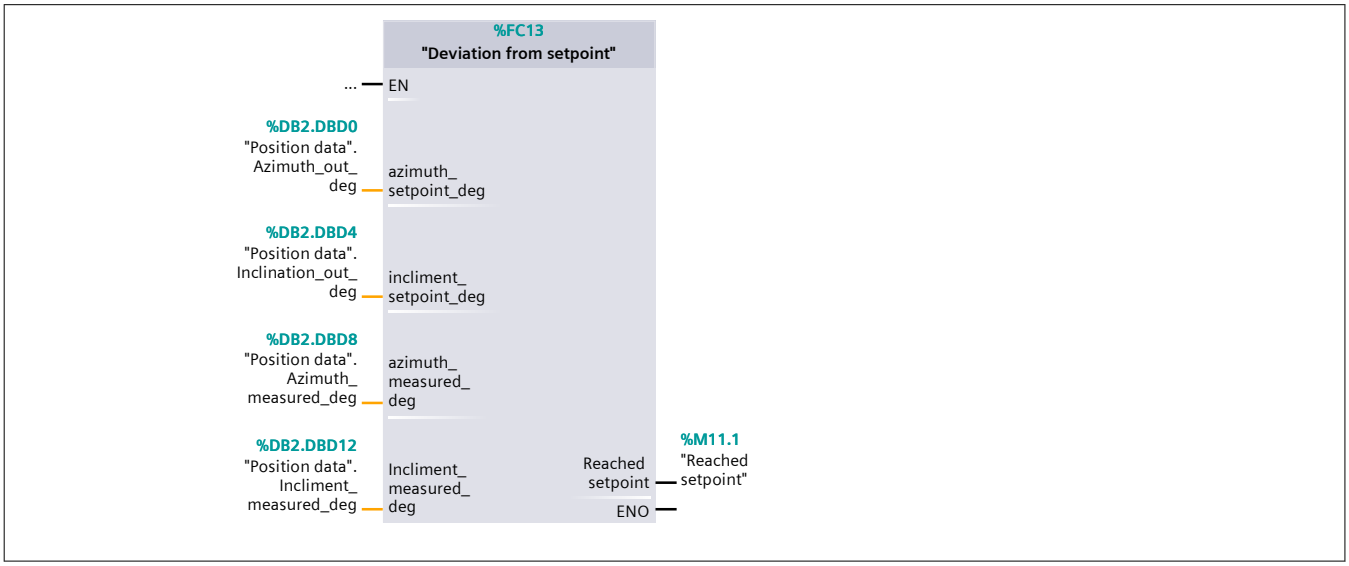

## Network 5: Writes bit if fire position is found




## Network 6: Times that holds the alarm high for a while




## Network 7: Deviation from setpoint

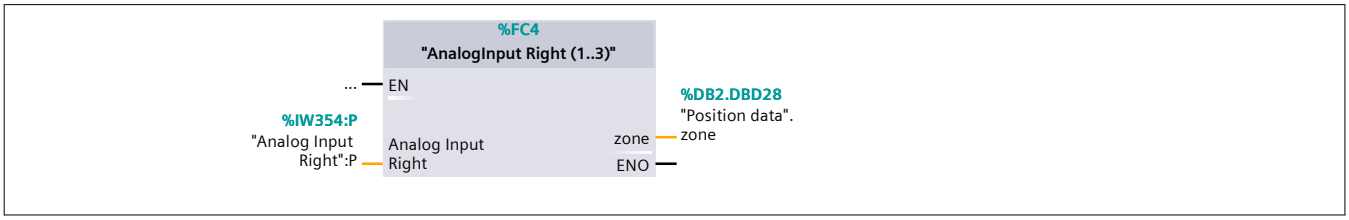



## Network 8: Transitions


## Network 9: Functions


## Network 10: Outputs


## Network 11: Reads right analog input



# Program blocks

## Input from RT [DB60]

| Input from RT Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Input from RT | **Number** | 60 | **Type** | DB |
| **Language** | DB | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Start value | Retain |
|---|---|---|---|
| ▼ Static | | | |
| Vision Fire X | Real | 0.0 | True |
| Vision Fire Y | Real | 0.0 | True |
| Azimuth_measured_16bit | Real | 0.0 | True |
| Inclination_measured_16bit | Real | 0.0 | True |
| Found fire position? | Real | 0.0 | True |

# Program blocks

## Output to RT [DB61]

| Output to RT Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Output to RT | **Number** | 61 | **Type** | DB |
| **Language** | DB | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Start value | Retain |
|---|---|---|---|
| ▼ Static | | | |
| P2P X | Real | 0.0 | True |
| P2P Y | Real | 0.0 | True |
| Azimuth_angle_16bit | Real | 0.0 | True |
| Inclination_angle_16bit | Real | 0.0 | True |
| Pressure | Real | 0.0 | True |
| Start pump | Bool | false | True |

# Program blocks

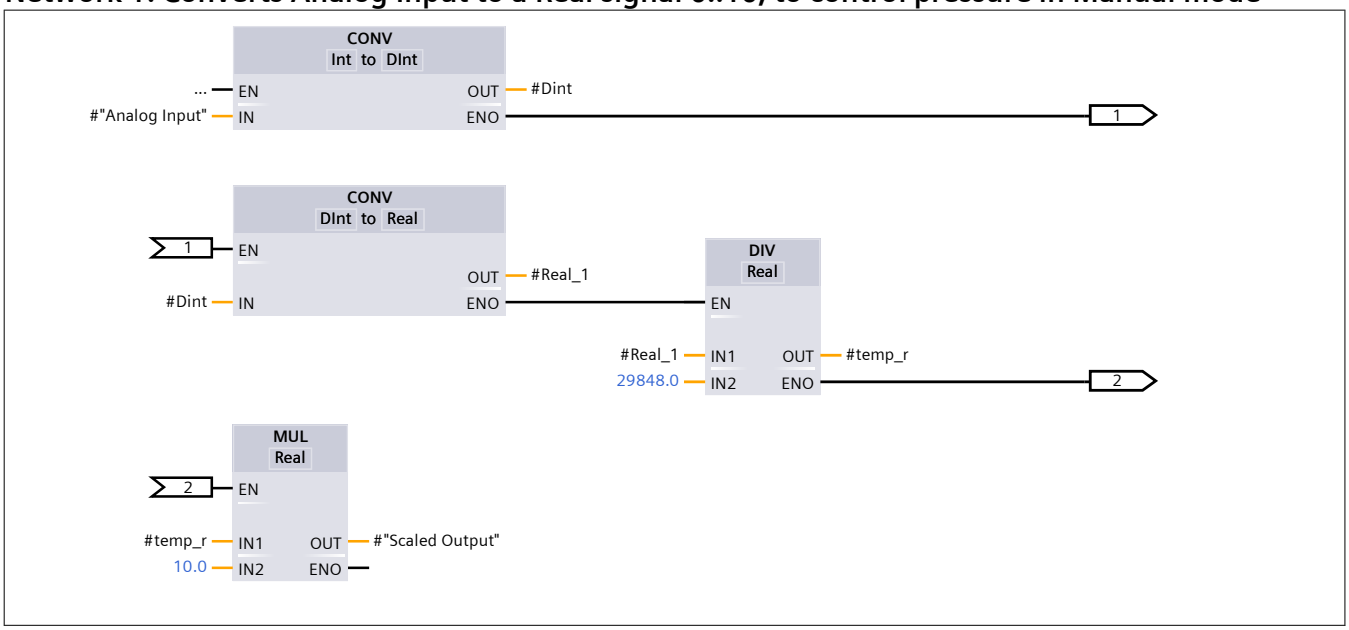
## AnalogInput Left (0..10) [FC2]

| AnalogInput Left (0..10) Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | AnalogInput Left (0..10) | **Number** | 2 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Analog Input | Int | |
| ▼ Output | | |
| Scaled Output | Real | |
| InOut | | |
| ▼ Temp | | |
| Dint | DInt | 0.0 |
| Real_1 | Real | 4.0 |
| temp_r | Real | 8.0 |
| temp_r2 | Real | 12.0 |
| ▼ Return | | |
| AnalogInput Left (0..10) | Void | |

### Network 1: Converts Analog Input to a Real signal 0..10, to control pressure in Manual mode

### Network 1: Converts Analog Input to a Real signal 0..10, to control pressure in Manual mode



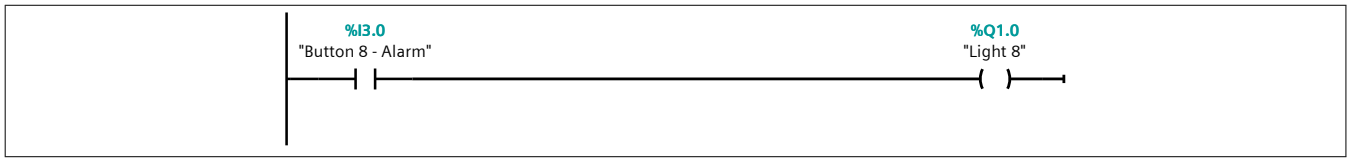
# Program blocks

## Panel lights [FC1]

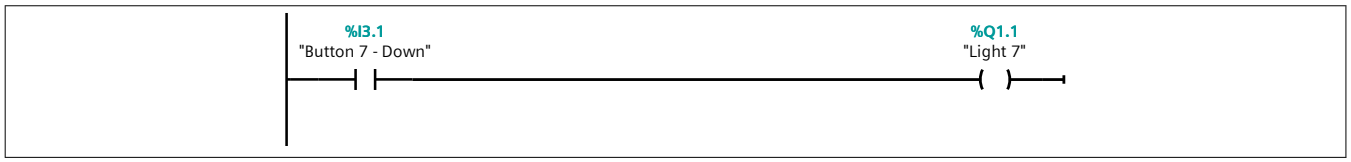
| Panel lights Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Panel lights | **Number** | 1 | **Type** | FC |
| **Language** | LAD | | | | |
| **Information** | | | | | |
| **Title** | Activates the light corre-spondig to each button in panel | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| Output | | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Panel lights | Void | |

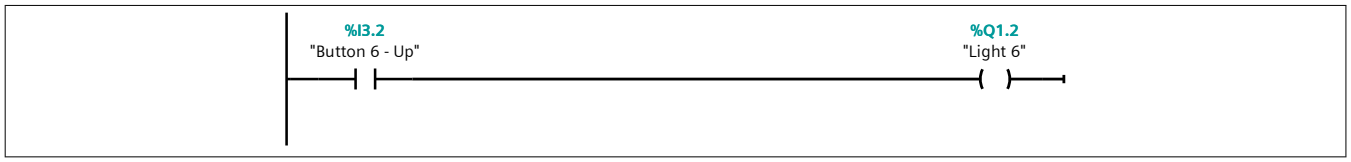
### Network 1:



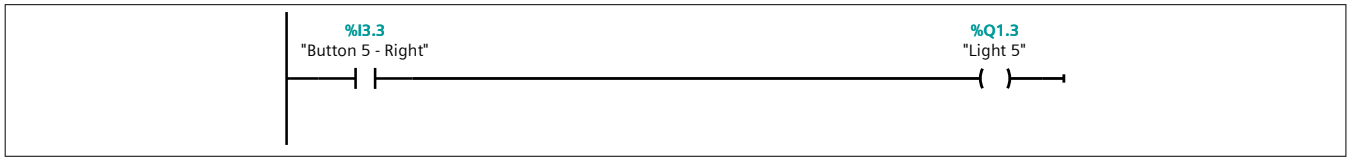

### Network 2:



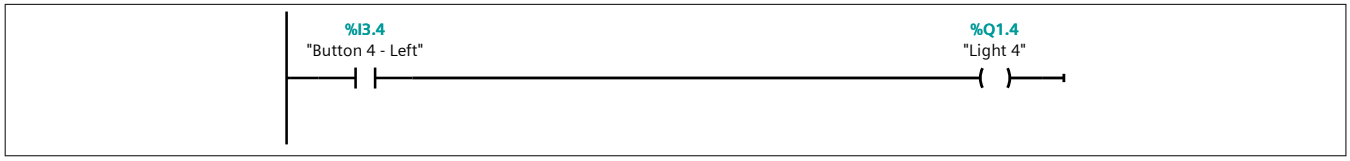

### Network 3:



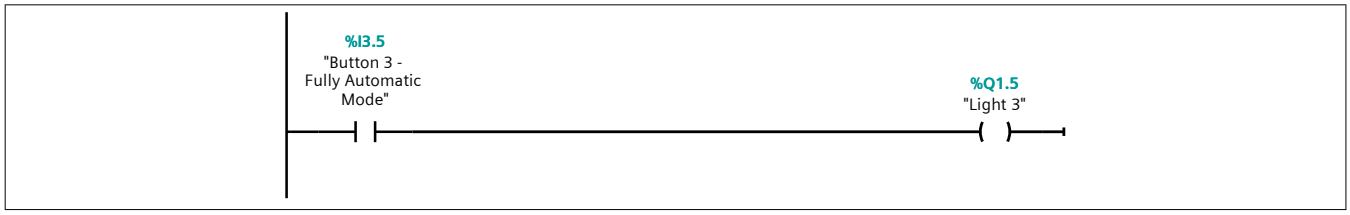

### Network 4:




## Network 5:




## Network 6:




## Network 7:


## Network 8:

# Program blocks

## Angle to monitor signal [FC3]

| Angle to monitor signal Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Angle to monitor signal | **Number** | 3 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Angle_deg | Real | |
| ▼ Output | | |
| Motor_signal_16bit | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Angle to monitor signal | Void | |

```
0001 //
0002 #Motor_signal_16bit:=#Angle_deg*(2**16)/360;
```

# Program blocks

## Position data [DB2]

| Position data Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Position data | **Number** | 2 | **Type** | DB |
| **Language** | DB | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Start value | Retain |
|---|---|---|---|
| ▼ Static | | | |
| Azimuth_out_deg | Real | 0.0 | False |
| Inclination_out_deg | Real | 0.0 | False |
| Azimuth_measured_deg | Real | 0.0 | False |
| Incliment_measured_deg | Real | 0.0 | False |
| s_uni_zone | Real | 0.0 | False |
| s_uni_auto | Real | 0.0 | False |
| Pressure | Real | 5.0 | False |
| zone | Real | 0.0 | False |

# Program blocks

## Cartesian to polar coordinates [FC5]

| Cartesian to polar coordinates Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Cartesian to polar coordinates | **Number** | 5 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| xpoint | Real | |
| ypoint | Real | |
| ▼ Output | | |
| Radie | Real | |
| Phi | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Cartesian to polar coordinates | Void | |

```
0001 //
0002
0003 IF #xpoint>0
0004 THEN #Phi:=ATAN(#ypoint/#xpoint)*(180/3.1416);
0005 ELSIF #xpoint<0
0006 THEN #Phi:=(ATAN(#ypoint/#xpoint)+3.1416)*(180/3.1416);
0007 END_IF;
0008
0009 #Radie:=SQRT(#xpoint**2+#ypoint**2);
```

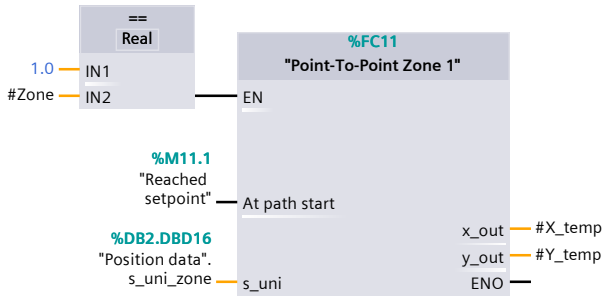
# Program blocks

## Zone Protection Mode [FC6]

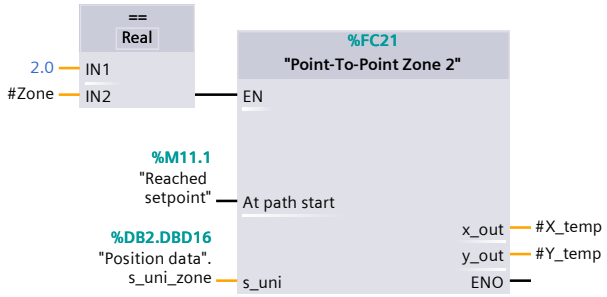
| Zone Protection Mode Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Zone Protection Mode | **Number** | 6 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | In Zone Protection Mode the monitor "paints" a prdifes fiel that corre-spon with a alarm for this field. | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Zone | Real | |
| ▼ Output | | |
| Phi | Real | |
| Theta | Real | |
| x_labview | Real | |
| y_labview | Real | |
| ▼ InOut | | |
| Pressure | Real | |
| ▼ Temp | | |
| X_temp | Real | 0.0 |
| Y_temp | Real | 4.0 |
| Radie_temp | Real | 8.0 |
| Phi_temp | Real | 12.0 |
| Theta_temp | Real | 16.0 |
| ▼ Return | | |
| Zone Protection Mode | Void | |

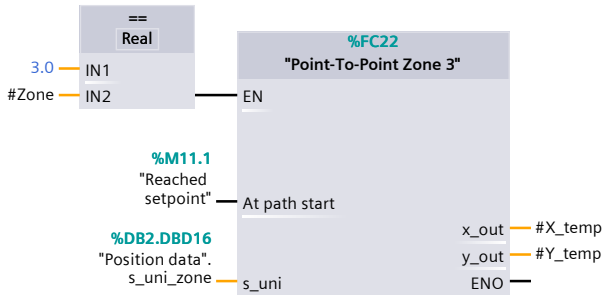
### Network 1: Generates a path around recieved fire position



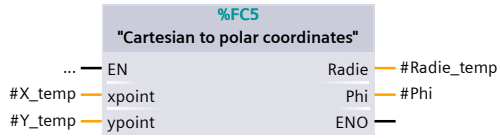

### Network 2:



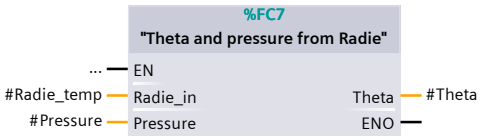

## Network 3:



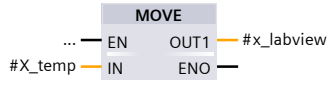

## Network 4: Converts cartesian coordinates to polar coordinates



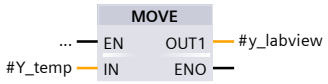

## Network 5: Calculates Theta and Pressure from radie




## Network 6: Sends path in cartesian coordinates to Plot for reference check




## Network 7: Sends path in cartesian coordinates to Plot for reference check


MOVE
...
EN
OUT1
#y_labview
#Y_temp
IN
ENO

# Program blocks

## Theta and pressure from Radie [FC7]

| Theta and pressure from Radie Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Theta and pressure from Radie | **Number** | 7 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Radie_in | Real | |
| ▼ Output | | |
| Theta | Real | |
| ▼ InOut | | |
| Pressure | Real | |
| ▼ Temp | | |
| Radie | Real | 0.0 |
| Pressure_temp | Real | 4.0 |
| ▼ Return | | |
| Theta and pressure from Radie | Void | |

```
0001 //Defines the allowed range of radie for monitor
0002 IF #Radie_in<25
0003 THEN #Radie:=25;
0004 ELSIF #Radie_in>50
0005 THEN #Radie:=50;
0006 ELSE #Radie:=#Radie_in;
0007 END_IF;
0008 //
0009 IF #Pressure<7.5 & #Radie<47
0010 THEN #Pressure_temp:=5;
0011      #Theta:=-7.99991111056179e-05*#Radie**4 + 0.00977388311668905*#Radie**3
     - 0.440517398978581*#Radie**2 + 7.93925052320138*#Radie + 37.2138715347672;
0012
0013 ELSIF #Pressure<7.5 & #Radie>47
0014 THEN #Pressure_temp:=10;
0015      #Theta:=-0.00382588242799651*#Radie**4 + 0.7014581726548*#Radie**3 -
     48.2565245501124*#Radie**2 + 1475.72620650573*#Radie - 16839.7026876163;
0016 END_IF;
0017
0018   IF #Pressure>7.5 & #Radie>45
0019 THEN #Pressure_temp:=10;
0020      #Theta:=-0.00382588242799651*#Radie**4 + 0.7014581726548*#Radie**3 -
     48.2565245501124*#Radie**2 + 1475.72620650573*#Radie - 16839.7026876163;
0021 ELSIF #Pressure>7.5 & #Radie<45
0022 THEN #Pressure_temp:=5;
0023      #Theta:=-7.99991111056179e-05*#Radie**4 + 0.00977388311668905*#Radie**3
     - 0.440517398978581*#Radie**2 + 7.93925052320138*#Radie + 37.2138715347672;
0024 END_IF;
0025
```

```
0026 #Pressure:=#Pressure_temp;
0027
0028
```

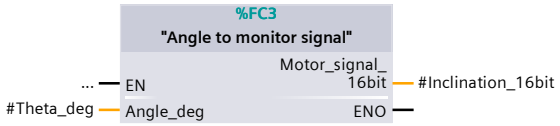
# Program blocks

## Dual monitor controller [FC8]

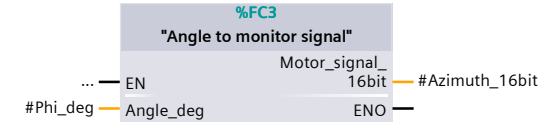
| Dual monitor controller Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Dual monitor controller | **Number** | 8 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | Scales up angles to motorsignal in 16-bit format | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Theta_deg | Real | |
| Phi_deg | Real | |
| ▼ Output | | |
| Azimuth_16bit | Real | |
| Inclination_16bit | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Dual monitor controller | Void | |

### Network 1: Scales up Inclination angle to 16 bit format




### Network 2: Scales up Azimuth angle to 16 bit format



# Program blocks

## Point-to-Point Spiral around fire [FC9]

| Point-to-Point Spiral around fire Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Point-to-Point Spiral around fire | **Number** | 9 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| FirespotX | Real | |
| FirespotY | Real | |
| Reached_setpoint | Bool | |
| ▼ Output | | |
| x_out | Real | |
| y_out | Real | |
| ▼ InOut | | |
| s_uni | Real | |
| ▼ Temp | | |
| ▼ X_array | Array [1..11] of Real | 0.0 |
| X_array[1] | Real | 0.0 |
| X_array[2] | Real | 4.0 |
| X_array[3] | Real | 8.0 |
| X_array[4] | Real | 12.0 |
| X_array[5] | Real | 16.0 |
| X_array[6] | Real | 20.0 |
| X_array[7] | Real | 24.0 |
| X_array[8] | Real | 28.0 |
| X_array[9] | Real | 32.0 |
| X_array[10] | Real | 36.0 |
| X_array[11] | Real | 40.0 |
| ▼ Y_array | Array [1..11] of Real | 44.0 |
| Y_array[1] | Real | 0.0 |
| Y_array[2] | Real | 4.0 |
| Y_array[3] | Real | 8.0 |
| Y_array[4] | Real | 12.0 |
| Y_array[5] | Real | 16.0 |
| Y_array[6] | Real | 20.0 |
| Y_array[7] | Real | 24.0 |
| Y_array[8] | Real | 28.0 |
| Y_array[9] | Real | 32.0 |
| Y_array[10] | Real | 36.0 |
| Y_array[11] | Real | 40.0 |
| ▼ s_dx | Array [1..11] of Real | 88.0 |
| s_dx[1] | Real | 0.0 |
| s_dx[2] | Real | 4.0 |

| Name | Data type | Offset |
|---|---|---|
| s_dx[3] | Real | 8.0 |
| s_dx[4] | Real | 12.0 |
| s_dx[5] | Real | 16.0 |
| s_dx[6] | Real | 20.0 |
| s_dx[7] | Real | 24.0 |
| s_dx[8] | Real | 28.0 |
| s_dx[9] | Real | 32.0 |
| s_dx[10] | Real | 36.0 |
| s_dx[11] | Real | 40.0 |
| ▼ s_dy | Array [1..11] of Real | 132.0 |
| s_dy[1] | Real | 0.0 |
| s_dy[2] | Real | 4.0 |
| s_dy[3] | Real | 8.0 |
| s_dy[4] | Real | 12.0 |
| s_dy[5] | Real | 16.0 |
| s_dy[6] | Real | 20.0 |
| s_dy[7] | Real | 24.0 |
| s_dy[8] | Real | 28.0 |
| s_dy[9] | Real | 32.0 |
| s_dy[10] | Real | 36.0 |
| s_dy[11] | Real | 40.0 |
| ▼ s_length | Array [1..11] of Real | 176.0 |
| s_length[1] | Real | 0.0 |
| s_length[2] | Real | 4.0 |
| s_length[3] | Real | 8.0 |
| s_length[4] | Real | 12.0 |
| s_length[5] | Real | 16.0 |
| s_length[6] | Real | 20.0 |
| s_length[7] | Real | 24.0 |
| s_length[8] | Real | 28.0 |
| s_length[9] | Real | 32.0 |
| s_length[10] | Real | 36.0 |
| s_length[11] | Real | 40.0 |
| i | Int | 220.0 |
| s | Real | 222.0 |
| v | Real | 226.0 |
| x | Real | 230.0 |
| y | Real | 234.0 |
| dt | Real | 238.0 |
| v_x | Real | 242.0 |
| v_y | Real | 246.0 |
| v_update | Real | 250.0 |
| ▼ Return | | |
| Point-to-Point Spiral around fire | Void | |

```
0001 //Defines each corner in a square spiral shape around the calculates fire po-
     sition, X- and Y coordinate correspond to a point.
0002 //Defines x-points
0003 #X_array[1]:=#FirespotX;
0004 #X_array[2]:=#FirespotX + 0.3;
0005 #X_array[3]:=#FirespotX + 0.3;
0006 #X_array[4]:=#FirespotX - 0.3;
0007 #X_array[5]:=#FirespotX - 0.3;
```

```
#X_array[6]:=#FirespotX + 0.6;
#X_array[7]:=#FirespotX + 0.6;
#X_array[8]:=#FirespotX - 0.6;
#X_array[9]:=#FirespotX - 0.6;
#X_array[10]:=#FirespotX + 0.6;
#X_array[11]:=#FirespotX;
//Defines y-points
#Y_array[1]:=#FirespotY;
#Y_array[2]:=#FirespotY;
#Y_array[3]:=#FirespotY + 0.3;
#Y_array[4]:=#FirespotY + 0.3;
#Y_array[5]:=#FirespotY - 0.3;
#Y_array[6]:=#FirespotY - 0.3;
#Y_array[7]:=#FirespotY + 0.6;
#Y_array[8]:=#FirespotY + 0.6;
#Y_array[9]:=#FirespotY - 0.6;
#Y_array[10]:=#FirespotY - 0.6;
#Y_array[11]:=#FirespotY;

//Path speed
#v:=0.15;
//Interupt time for OB35 in sec
#dt:=0.010;

#x_out:=#X_array[1]+#x;
#y_out:=#Y_array[1]+#y;

#s:=#s_uni;

IF #Reached_setpoint
THEN
//Calculate the vector orientation and length from a point to the next one
FOR #i := 1 TO 10
 DO  #s_dx[#i]:= #X_array[(#i+1)]-#X_array[(#i)];
     #s_dy[#i]:= #Y_array[(#i+1)]-#Y_array[(#i)];
     #s_length[#i]:=SQRT(#s_dx[#i]**2+#s_dy[#i]**2);
END_FOR;

//If on first strech, then go this direction and update the postion for each
cycle
//Do this for every strech..
  IF #s<#s_length[1]
THEN
    #v_x:=(#s_dx[1]/#s_length[1])*#v;
    #v_y:=(#s_dy[1]/#s_length[1])*#v;
    #x:=#x+#v_x*#dt;
    #y:=#y+#v_y*#dt;

ELSIF  #s<(#s_length[1]+#s_length[2])
THEN
    #v_x:=(#s_dx[2]/#s_length[2])*#v;
    #v_y:=(#s_dy[2]/#s_length[2])*#v;
    #x:=#x+#v_x*#dt;
    #y:=#y+#v_y*#dt;

ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3])
THEN
    #v_x:=(#s_dx[3]/#s_length[3])*#v;
    #v_y:=(#s_dy[3]/#s_length[3])*#v;
```

```
0066      #x:=#x+#v_x*#dt;
0067      #y:=#y+#v_y*#dt;
0068  ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4])
0069  THEN
0070      #v_x:=(#s_dx[4]/#s_length[4])*#v;
0071      #v_y:=(#s_dy[4]/#s_length[4])*#v;
0072      #x:=#x+#v_x*#dt;
0073      #y:=#y+#v_y*#dt;
0074  ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5])
0075  THEN
0076      #v_x:=(#s_dx[5]/#s_length[5])*#v;
0077      #v_y:=(#s_dy[5]/#s_length[5])*#v;
0078      #x:=#x+#v_x*#dt;
0079      #y:=#y+#v_y*#dt;
0080  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6])
0081  THEN
0082      #v_x:=(#s_dx[6]/#s_length[6])*#v;
0083      #v_y:=(#s_dy[6]/#s_length[6])*#v;
0084      #x:=#x+#v_x*#dt;
0085      #y:=#y+#v_y*#dt;
0086  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7])
0087  THEN
0088      #v_x:=(#s_dx[7]/#s_length[7])*#v;
0089      #v_y:=(#s_dy[7]/#s_length[7])*#v;
0090      #x:=#x+#v_x*#dt;
0091      #y:=#y+#v_y*#dt;
0092  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8])
0093  THEN
0094      #v_x:=(#s_dx[8]/#s_length[8])*#v;
0095      #v_y:=(#s_dy[8]/#s_length[8])*#v;
0096      #x:=#x+#v_x*#dt;
0097      #y:=#y+#v_y*#dt;
0098  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9])
0099  THEN
0100      #v_x:=(#s_dx[9]/#s_length[9])*#v;
0101      #v_y:=(#s_dy[9]/#s_length[9])*#v;
0102      #x:=#x+#v_x*#dt;
0103      #y:=#y+#v_y*#dt;
0104  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10])
0105  THEN
0106      #v_x:=(#s_dx[10]/#s_length[10])*#v;
0107      #v_y:=(#s_dy[10]/#s_length[10])*#v;
0108      #x:=#x+#v_x*#dt;
0109      #y:=#y+#v_y*#dt;
0110    END_IF;
0111    IF
      #s>#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length[
      6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]
0112  THEN
```

```
0113    #s:=0;
0114    #x:=0;
0115    #y:=0;
0116  END_IF;
0117  //Send out path position values
0118  //Update velocity and distance gone
0119  #x_out:=#X_array[1]+#x;
0120  #y_out:=#Y_array[1]+#y;
0121  #v_update:=SQRT(#v_x**2+#v_y**2);
0122  #s_uni:=#s+#v_update*#dt;
0123  END_IF;
0124
```

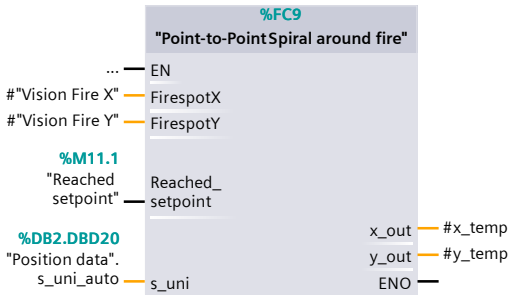
# Program blocks

## Fully Automatic Mode [FC10]

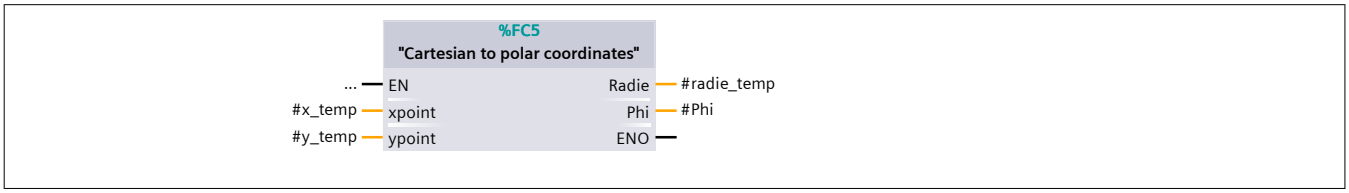
| Fully Automatic Mode Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Fully Automatic Mode | **Number** | 10 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | This mode generates a path around measured fire position if alarm signal is high | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Vision Fire X | Real | |
| Vision Fire Y | Real | |
| ▼ Output | | |
| Phi | Real | |
| Theta | Real | |
| x_labview | Real | |
| y_labview | Real | |
| ▼ InOut | | |
| Pressure | Real | |
| ▼ Temp | | |
| x_temp | Real | 0.0 |
| y_temp | Real | 4.0 |
| radie_temp | Real | 8.0 |
| phi_temp | Real | 12.0 |
| theta_temp | Real | 16.0 |
| ▼ Return | | |
| Fully Automatic Mode | Void | |

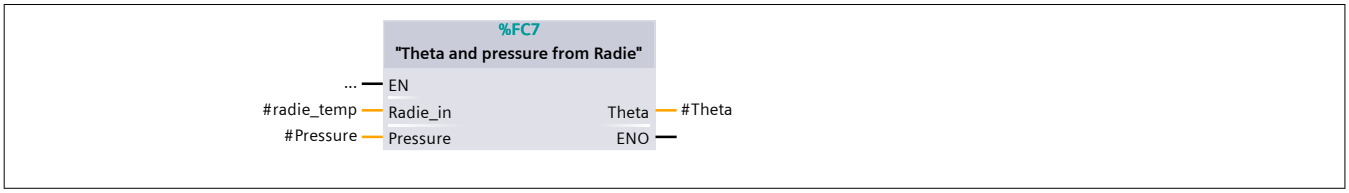
### Network 1: Generates a path around recieved fire position



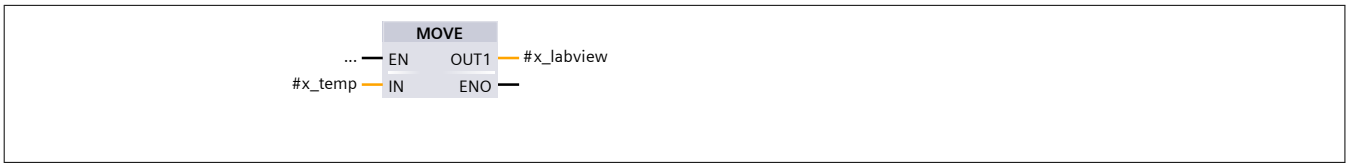

### Network 2: Converts cartesian coordinates to polar coordinates



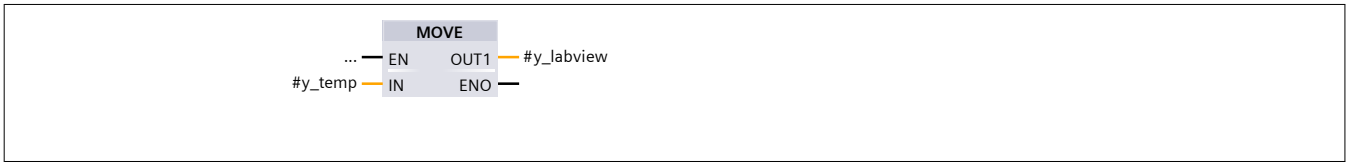

## Network 3: Calculates Theta and Pressure from radie




## Network 4: Sends path in cartesian coordinates to Plot for reference check




## Network 5: Sends path in cartesian coordinates to Plot for reference check



# Program blocks

## Point-To-Point Zone 1 [FC11]

| Point-To-Point Zone 1 Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Point-To-Point Zone 1 | **Number** | 11 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| At path start | Bool | |
| ▼ Output | | |
| x_out | Real | |
| y_out | Real | |
| ▼ InOut | | |
| s_uni | Real | |
| ▼ Temp | | |
| ▼ X_array | Array [1..23] of Real | 0.0 |
| X_array[1] | Real | 0.0 |
| X_array[2] | Real | 4.0 |
| X_array[3] | Real | 8.0 |
| X_array[4] | Real | 12.0 |
| X_array[5] | Real | 16.0 |
| X_array[6] | Real | 20.0 |
| X_array[7] | Real | 24.0 |
| X_array[8] | Real | 28.0 |
| X_array[9] | Real | 32.0 |
| X_array[10] | Real | 36.0 |
| X_array[11] | Real | 40.0 |
| X_array[12] | Real | 44.0 |
| X_array[13] | Real | 48.0 |
| X_array[14] | Real | 52.0 |
| X_array[15] | Real | 56.0 |
| X_array[16] | Real | 60.0 |
| X_array[17] | Real | 64.0 |
| X_array[18] | Real | 68.0 |
| X_array[19] | Real | 72.0 |
| X_array[20] | Real | 76.0 |
| X_array[21] | Real | 80.0 |
| X_array[22] | Real | 84.0 |
| X_array[23] | Real | 88.0 |
| ▼ Y_array | Array [1..23] of Real | 92.0 |
| Y_array[1] | Real | 0.0 |
| Y_array[2] | Real | 4.0 |
| Y_array[3] | Real | 8.0 |
| Y_array[4] | Real | 12.0 |
| Y_array[5] | Real | 16.0 |

| Name | Data type | Offset |
|---|---|---|
| Y_array[6] | Real | 20.0 |
| Y_array[7] | Real | 24.0 |
| Y_array[8] | Real | 28.0 |
| Y_array[9] | Real | 32.0 |
| Y_array[10] | Real | 36.0 |
| Y_array[11] | Real | 40.0 |
| Y_array[12] | Real | 44.0 |
| Y_array[13] | Real | 48.0 |
| Y_array[14] | Real | 52.0 |
| Y_array[15] | Real | 56.0 |
| Y_array[16] | Real | 60.0 |
| Y_array[17] | Real | 64.0 |
| Y_array[18] | Real | 68.0 |
| Y_array[19] | Real | 72.0 |
| Y_array[20] | Real | 76.0 |
| Y_array[21] | Real | 80.0 |
| Y_array[22] | Real | 84.0 |
| Y_array[23] | Real | 88.0 |
| ▼ s_dx | Array [1..23] of Real | 184.0 |
| s_dx[1] | Real | 0.0 |
| s_dx[2] | Real | 4.0 |
| s_dx[3] | Real | 8.0 |
| s_dx[4] | Real | 12.0 |
| s_dx[5] | Real | 16.0 |
| s_dx[6] | Real | 20.0 |
| s_dx[7] | Real | 24.0 |
| s_dx[8] | Real | 28.0 |
| s_dx[9] | Real | 32.0 |
| s_dx[10] | Real | 36.0 |
| s_dx[11] | Real | 40.0 |
| s_dx[12] | Real | 44.0 |
| s_dx[13] | Real | 48.0 |
| s_dx[14] | Real | 52.0 |
| s_dx[15] | Real | 56.0 |
| s_dx[16] | Real | 60.0 |
| s_dx[17] | Real | 64.0 |
| s_dx[18] | Real | 68.0 |
| s_dx[19] | Real | 72.0 |
| s_dx[20] | Real | 76.0 |
| s_dx[21] | Real | 80.0 |
| s_dx[22] | Real | 84.0 |
| s_dx[23] | Real | 88.0 |
| ▼ s_dy | Array [1..23] of Real | 276.0 |
| s_dy[1] | Real | 0.0 |
| s_dy[2] | Real | 4.0 |
| s_dy[3] | Real | 8.0 |
| s_dy[4] | Real | 12.0 |
| s_dy[5] | Real | 16.0 |
| s_dy[6] | Real | 20.0 |
| s_dy[7] | Real | 24.0 |
| s_dy[8] | Real | 28.0 |
| s_dy[9] | Real | 32.0 |

| Name | Data type | Offset |
|---|---|---|
| s_dy[10] | Real | 36.0 |
| s_dy[11] | Real | 40.0 |
| s_dy[12] | Real | 44.0 |
| s_dy[13] | Real | 48.0 |
| s_dy[14] | Real | 52.0 |
| s_dy[15] | Real | 56.0 |
| s_dy[16] | Real | 60.0 |
| s_dy[17] | Real | 64.0 |
| s_dy[18] | Real | 68.0 |
| s_dy[19] | Real | 72.0 |
| s_dy[20] | Real | 76.0 |
| s_dy[21] | Real | 80.0 |
| s_dy[22] | Real | 84.0 |
| s_dy[23] | Real | 88.0 |
| ▼ s_length | Array [1..23] of Real | 368.0 |
| s_length[1] | Real | 0.0 |
| s_length[2] | Real | 4.0 |
| s_length[3] | Real | 8.0 |
| s_length[4] | Real | 12.0 |
| s_length[5] | Real | 16.0 |
| s_length[6] | Real | 20.0 |
| s_length[7] | Real | 24.0 |
| s_length[8] | Real | 28.0 |
| s_length[9] | Real | 32.0 |
| s_length[10] | Real | 36.0 |
| s_length[11] | Real | 40.0 |
| s_length[12] | Real | 44.0 |
| s_length[13] | Real | 48.0 |
| s_length[14] | Real | 52.0 |
| s_length[15] | Real | 56.0 |
| s_length[16] | Real | 60.0 |
| s_length[17] | Real | 64.0 |
| s_length[18] | Real | 68.0 |
| s_length[19] | Real | 72.0 |
| s_length[20] | Real | 76.0 |
| s_length[21] | Real | 80.0 |
| s_length[22] | Real | 84.0 |
| s_length[23] | Real | 88.0 |
| i | Int | 460.0 |
| s | Real | 462.0 |
| v | Real | 466.0 |
| x | Real | 470.0 |
| y | Real | 474.0 |
| dt | Real | 478.0 |
| v_x | Real | 482.0 |
| v_y | Real | 486.0 |
| v_update | Real | 490.0 |
| ▼ Return | | |
| Point-To-Point Zone 1 | Void | |

```
0001 //Defines each corner in a square spiral shape around the calculates fire po-
     sition, X- and Y coordinate correspond to a point.
0002 //Defines x-points
```

```
0003 #X_array[1]:=-35.3;
0004 #X_array[2]:=-20.3;
0005 #X_array[3]:=-20.3;
0006 #X_array[4]:=-35.3;
0007 #X_array[5]:=-35.3;
0008 #X_array[6]:=-20.3;
0009 #X_array[7]:=-20.3;
0010 #X_array[8]:=-35.3;
0011 #X_array[9]:=-35.3;
0012 #X_array[10]:=-20.3;
0013 #X_array[11]:=-20.3;
0014 #X_array[12]:=-35.3;
0015 #X_array[13]:=-35.3;
0016 #X_array[14]:=-20.3;
0017 #X_array[15]:=-20.3;
0018 #X_array[16]:=-35.3;
0019 #X_array[17]:=-35.3;
0020 #X_array[18]:=-20.3;
0021 #X_array[19]:=-20.3;
0022 #X_array[20]:=-35.3;
0023 #X_array[21]:=-35.3;
0024 #X_array[22]:=-20.3;
0025 #X_array[23]:=-35.3;
0026 //Defines y-points
0027 #Y_array[1]:=34.8;
0028 #Y_array[2]:=34.8;
0029 #Y_array[3]:=33.8;
0030 #Y_array[4]:=33.8;
0031 #Y_array[5]:=32.8;
0032 #Y_array[6]:=32.8;
0033 #Y_array[7]:=31.8;
0034 #Y_array[8]:=31.8;
0035 #Y_array[9]:=30.8;
0036 #Y_array[10]:=30.8;
0037 #Y_array[11]:=29.8;
0038 #Y_array[12]:=29.8;
0039 #Y_array[13]:=28.8;
0040 #Y_array[14]:=28.8;
0041 #Y_array[15]:=27.8;
0042 #Y_array[16]:=27.8;
0043 #Y_array[17]:=26.8;
0044 #Y_array[18]:=26.8;
0045 #Y_array[19]:=25.8;
0046 #Y_array[20]:=25.8;
0047 #Y_array[21]:=24.8;
0048 #Y_array[22]:=24.8;
0049 #Y_array[23]:=34.8;
0050
0051 //Path speed
0052 #v:=0.8;
0053 //Interupt time for OB35 in sec
0054 #dt:=0.010
0055 ;
0056
0057 #x_out:=#X_array[1]+#x;
0058 #y_out:=#Y_array[1]+#y;
0059
0060 #s:=#s_uni;
0061 IF #"At path start"
```

```
0062 THEN
0063 //Calculate the vector orientation and length from a point to the next one
0064 FOR #i := 1 TO 23
0065  DO  #s_dx[#i]:= #X_array[(#i+1)]-#X_array[(#i)];
0066      #s_dy[#i]:= #Y_array[(#i+1)]-#Y_array[(#i)];
0067      #s_length[#i]:=SQRT(#s_dx[#i]**2+#s_dy[#i]**2);
0068 END_FOR;
0069
0070 //If on first strech, then go this direction and update the postion for each
     cycle
0071 //Do this for every strech..
0072   IF #s<#s_length[1]
0073 THEN
0074     #v_x:=(#s_dx[1]/#s_length[1])*#v;
0075     #v_y:=(#s_dy[1]/#s_length[1])*#v;
0076     #x:=#x+#v_x*#dt;
0077     #y:=#y+#v_y*#dt;
0078
0079 ELSIF  #s<(#s_length[1]+#s_length[2])
0080 THEN
0081     #v_x:=(#s_dx[2]/#s_length[2])*#v;
0082     #v_y:=(#s_dy[2]/#s_length[2])*#v;
0083     #x:=#x+#v_x*#dt;
0084     #y:=#y+#v_y*#dt;
0085
0086 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3])
0087 THEN
0088     #v_x:=(#s_dx[3]/#s_length[3])*#v;
0089     #v_y:=(#s_dy[3]/#s_length[3])*#v;
0090     #x:=#x+#v_x*#dt;
0091     #y:=#y+#v_y*#dt;
0092 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4])
0093 THEN
0094     #v_x:=(#s_dx[4]/#s_length[4])*#v;
0095     #v_y:=(#s_dy[4]/#s_length[4])*#v;
0096     #x:=#x+#v_x*#dt;
0097     #y:=#y+#v_y*#dt;
0098 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5])
0099 THEN
0100     #v_x:=(#s_dx[5]/#s_length[5])*#v;
0101     #v_y:=(#s_dy[5]/#s_length[5])*#v;
0102     #x:=#x+#v_x*#dt;
0103     #y:=#y+#v_y*#dt;
0104 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6])
0105 THEN
0106     #v_x:=(#s_dx[6]/#s_length[6])*#v;
0107     #v_y:=(#s_dy[6]/#s_length[6])*#v;
0108     #x:=#x+#v_x*#dt;
0109     #y:=#y+#v_y*#dt;
0110 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7])
0111 THEN
0112     #v_x:=(#s_dx[7]/#s_length[7])*#v;
0113     #v_y:=(#s_dy[7]/#s_length[7])*#v;
0114     #x:=#x+#v_x*#dt;
0115     #y:=#y+#v_y*#dt;
```

```
0116 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8])
0117 THEN
0118     #v_x:=(#s_dx[8]/#s_length[8])*#v;
0119     #v_y:=(#s_dy[8]/#s_length[8])*#v;
0120     #x:=#x+#v_x*#dt;
0121     #y:=#y+#v_y*#dt;
0122 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9])
0123 THEN
0124     #v_x:=(#s_dx[9]/#s_length[9])*#v;
0125     #v_y:=(#s_dy[9]/#s_length[9])*#v;
0126     #x:=#x+#v_x*#dt;
0127     #y:=#y+#v_y*#dt;
0128 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10])
0129 THEN
0130     #v_x:=(#s_dx[10]/#s_length[10])*#v;
0131     #v_y:=(#s_dy[10]/#s_length[10])*#v;
0132     #x:=#x+#v_x*#dt;
0133     #y:=#y+#v_y*#dt;
0134 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11])
0135 THEN
0136     #v_x:=(#s_dx[11]/#s_length[11])*#v;
0137     #v_y:=(#s_dy[11]/#s_length[11])*#v;
0138     #x:=#x+#v_x*#dt;
0139     #y:=#y+#v_y*#dt;
0140 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12])
0141 THEN
0142     #v_x:=(#s_dx[12]/#s_length[12])*#v;
0143     #v_y:=(#s_dy[12]/#s_length[12])*#v;
0144     #x:=#x+#v_x*#dt;
0145     #y:=#y+#v_y*#dt;
0146 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13])
0147 THEN
0148     #v_x:=(#s_dx[13]/#s_length[13])*#v;
0149     #v_y:=(#s_dy[13]/#s_length[13])*#v;
0150     #x:=#x+#v_x*#dt;
0151     #y:=#y+#v_y*#dt;
0152 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14])
0153 THEN
0154     #v_x:=(#s_dx[14]/#s_length[14])*#v;
0155     #v_y:=(#s_dy[14]/#s_length[14])*#v;
0156     #x:=#x+#v_x*#dt;
0157     #y:=#y+#v_y*#dt;
```

```
0158 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15])
0159 THEN
0160     #v_x:=(#s_dx[15]/#s_length[15])*#v;
0161     #v_y:=(#s_dy[15]/#s_length[15])*#v;
0162     #x:=#x+#v_x*#dt;
0163     #y:=#y+#v_y*#dt;
0164 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16])
0165 THEN
0166     #v_x:=(#s_dx[16]/#s_length[16])*#v;
0167     #v_y:=(#s_dy[16]/#s_length[16])*#v;
0168     #x:=#x+#v_x*#dt;
0169     #y:=#y+#v_y*#dt;
0170 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17])
0171 THEN
0172     #v_x:=(#s_dx[17]/#s_length[17])*#v;
0173     #v_y:=(#s_dy[17]/#s_length[17])*#v;
0174     #x:=#x+#v_x*#dt;
0175     #y:=#y+#v_y*#dt;
0176 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18])
0177 THEN
0178     #v_x:=(#s_dx[18]/#s_length[18])*#v;
0179     #v_y:=(#s_dy[18]/#s_length[18])*#v;
0180     #x:=#x+#v_x*#dt;
0181     #y:=#y+#v_y*#dt;
0182 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18]+#s_length[19])
0183 THEN
0184     #v_x:=(#s_dx[19]/#s_length[19])*#v;
0185     #v_y:=(#s_dy[19]/#s_length[19])*#v;
0186     #x:=#x+#v_x*#dt;
0187     #y:=#y+#v_y*#dt;
0188 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18]+#s_length[19]+#s_length[20])
0189 THEN
0190     #v_x:=(#s_dx[20]/#s_length[20])*#v;
0191     #v_y:=(#s_dy[20]/#s_length[20])*#v;
0192     #x:=#x+#v_x*#dt;
0193     #y:=#y+#v_y*#dt;
0194 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
```

```
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18]+#s_length[19]+#s_length[20]+#s_length[21])
0195 THEN
0196     #v_x:=(#s_dx[21]/#s_length[21])*#v;
0197     #v_y:=(#s_dy[21]/#s_length[21])*#v;
0198     #x:=#x+#v_x*#dt;
0199     #y:=#y+#v_y*#dt;
0200 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18]+#s_length[19]+#s_length[20]+#s_length[21]+#s_length[22])
0201 THEN
0202     #v_x:=(#s_dx[22]/#s_length[22])*#v;
0203     #v_y:=(#s_dy[22]/#s_length[22])*#v;
0204     #x:=#x+#v_x*#dt;
0205     #y:=#y+#v_y*#dt;
0206
0207 END_IF;
0208 //Reset when whole route is gone
0209   IF
     #s>(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
     s_length[18]+#s_length[19]+#s_length[20]+#s_length[21]+#s_length[22])
0210 THEN
0211   #s:=0;
0212   #x:=0;
0213   #y:=0;
0214 END_IF;
0215
0216 //Send out path position values
0217 //Update velocity and distance gone
0218 #x_out:=#X_array[1]+#x;
0219 #y_out:=#Y_array[1]+#y;
0220 #v_update:=SQRT(#v_x**2+#v_y**2);
0221 #s_uni:=#s+#v_update*#dt;
0222 END_IF;
```

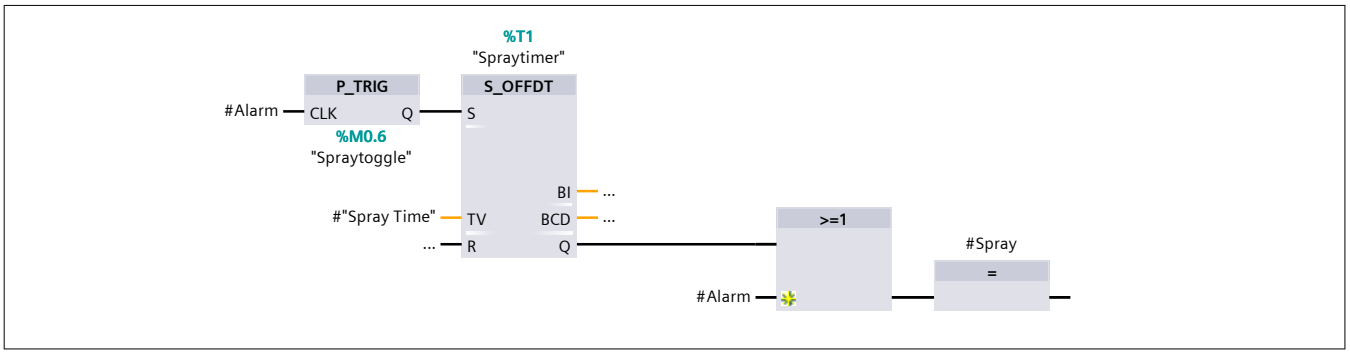
# Program blocks

## Keeps spraying after alarm [FC12]

| Keeps spraying after alarm Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Keeps spraying after alarm | **Number** | 12 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | This FC makes sure that the monitor spray for a given time, even though the alarm is low immedi-atey | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Alarm | Bool | |
| Spray Time | S5Time | |
| ▼ Output | | |
| Spray | Bool | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Keeps spraying after alarm | Void | |

### Network 1: Timer



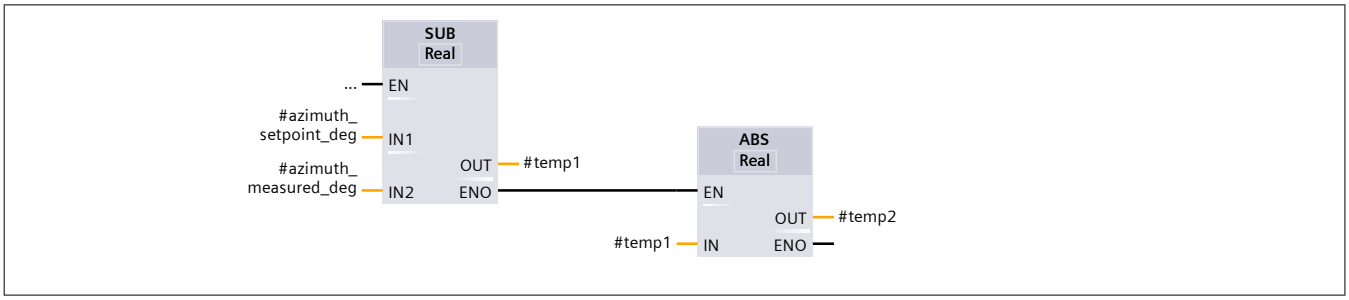
# Program blocks

## Deviation from setpoint [FC13]

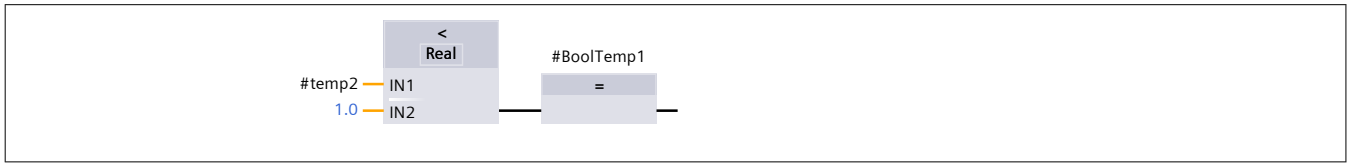
| Deviation from setpoint Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Deviation from setpoint | **Number** | 13 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | Creates a signal that pauses the path generator if the processvaule is to far away. | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| azimuth_setpoint_deg | Real | |
| incliment_setpoint_deg | Real | |
| azimuth_measured_deg | Real | |
| Incliment_measured_deg | Real | |
| Output | | |
| Reached setpoint | Bool | |
| InOut | | |
| Temp | | |
| temp1 | Real | 0.0 |
| temp2 | Real | 4.0 |
| temp3 | Real | 8.0 |
| temp4 | Real | 12.0 |
| BoolTemp1 | Bool | 16.0 |
| BoolTemp2 | Bool | 16.1 |
| Return | | |
| Deviation from setpoint | Void | |

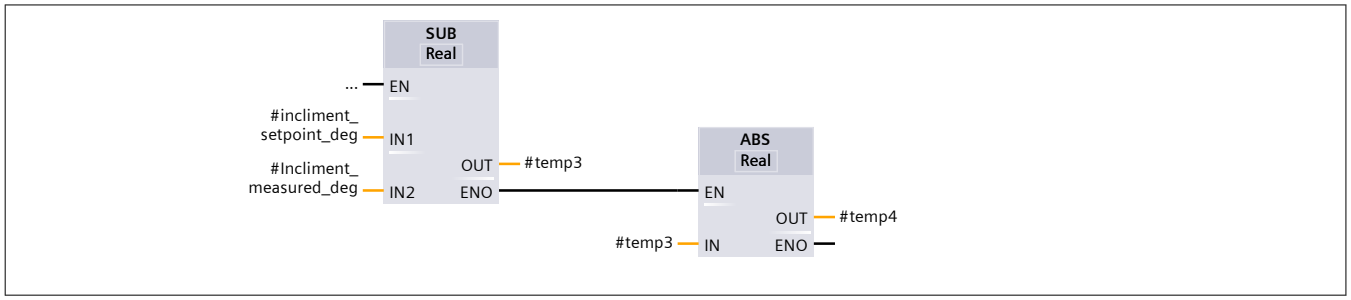
### Network 1: Checks differnece between setpoint and processvaule at azimuth angle.



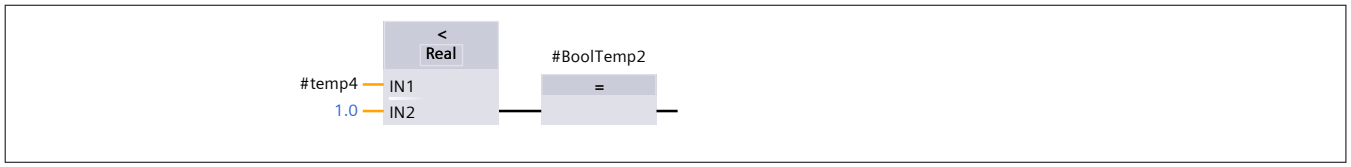

### Network 2: Checks this difference with allowed deviation



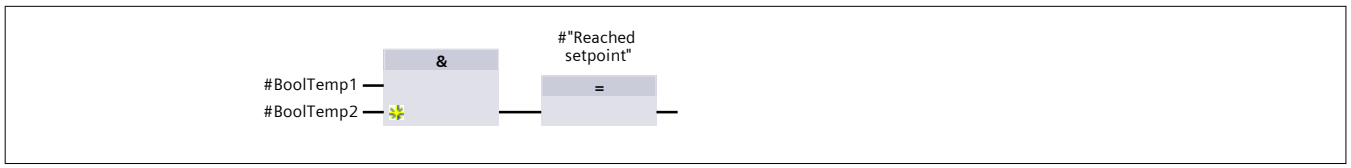

## Network 3: Checks differnece between setpoint and processvaule at inclination angle.




## Network 4: Checks this difference with allowed deviation




## Network 5: Gives signal if both angles is within allowed deviation



# Program blocks

## Monitor signal to angle [FC14]

| Monitor signal to angle Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Monitor signal to angle | **Number** | 14 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Motor_measured_16bit | Real | |
| ▼ Output | | |
| Monitor_measured_deg | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Monitor signal to angle | Void | |

```
0001 //
0002 #Monitor_measured_deg:=#Motor_measured_16bit*360/(2**16);
```

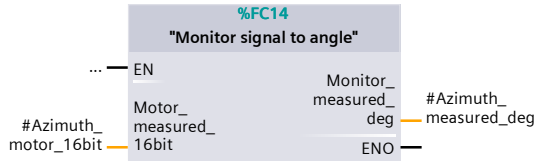
# Program blocks

## Dual monitor reader [FC15]

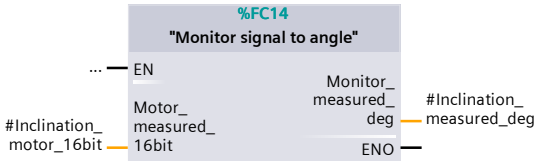
| Dual monitor reader Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Dual monitor reader | **Number** | 15 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | Scales down feedback values from motor controller from 16 bit value to degrees. | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Azimuth_motor_16bit | Real | |
| Inclination_motor_16bit | Real | |
| ▼ Output | | |
| Azimuth_measured_deg | Real | |
| Inclination_measured_deg | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Dual monitor reader | Void | |

### Network 1: Scales down Azimuth angle




### Network 2: Scales down Azimuth angle



# Program blocks

## Stepped_pressure [FC16]

| Stepped_pressure Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Stepped_pressure | **Number** | 16 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Pressure_cont | Real | |
| ▼ Output | | |
| Pressure_stepped | Real | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Stepped_pressure | Void | |

```
0001 //
0002 IF #Pressure_cont<7.5
0003 THEN #Pressure_stepped:=5.0;
0004 ELSE #Pressure_stepped:=10;
0005 END_IF;
```

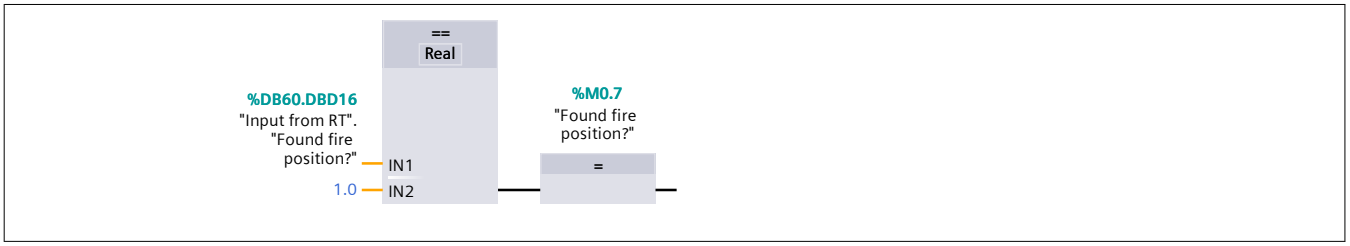
# Program blocks

## Fire position found? [FC17]

| Fire position found? Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Fire position found? | **Number** | 17 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | Recieves signal if fire position is found | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| Output | | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Fire position found? | Void | |

### Network 1: 1 is found, else is not found



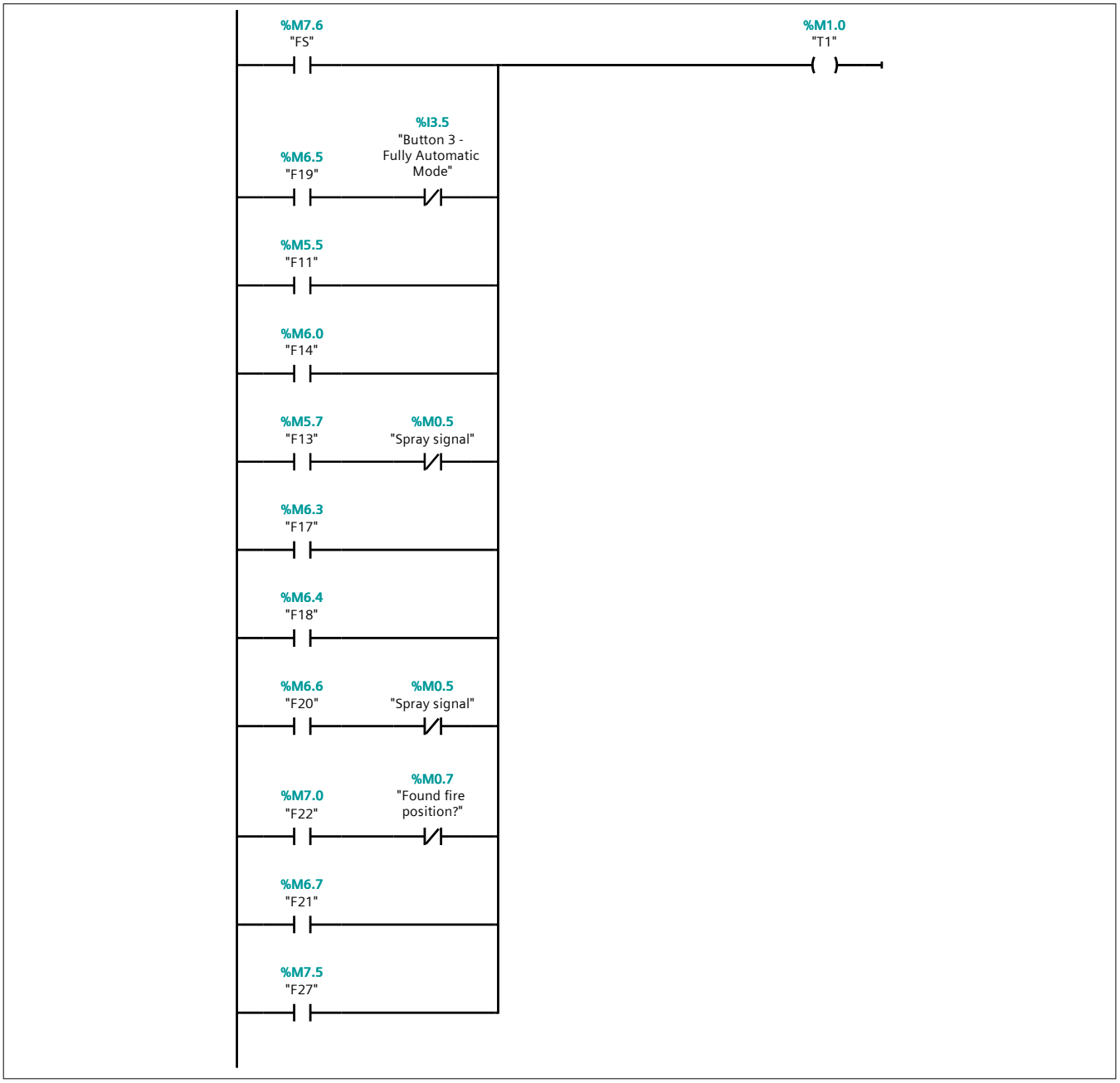
# Program blocks

## Transitions [FC18]

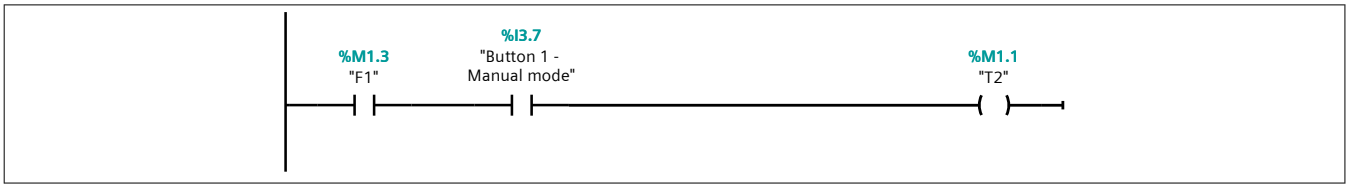
| Transitions Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Transitions | **Number** | 18 | **Type** | FC |
| **Language** | LAD | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| Output | | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Transitions | Void | |

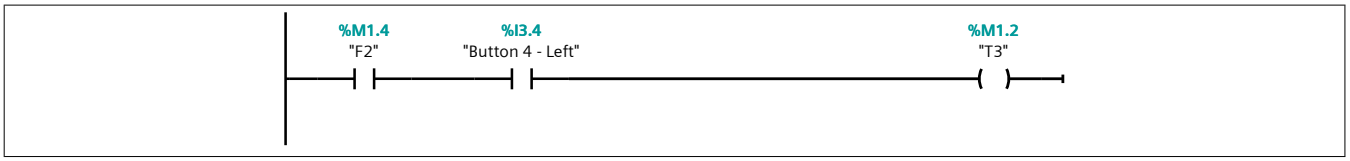
**Network 1:**



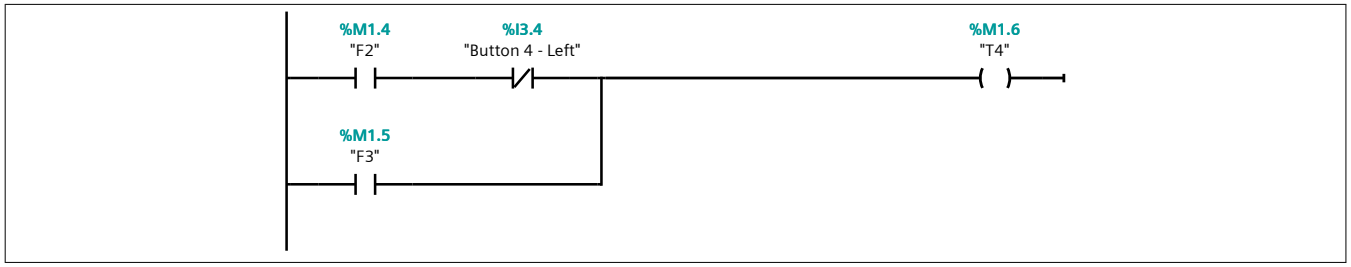

## Network 2:



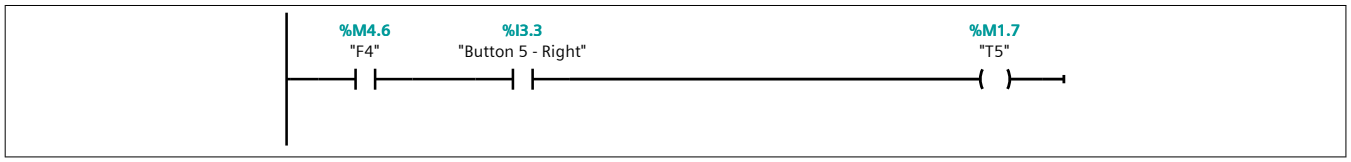

## Network 3:



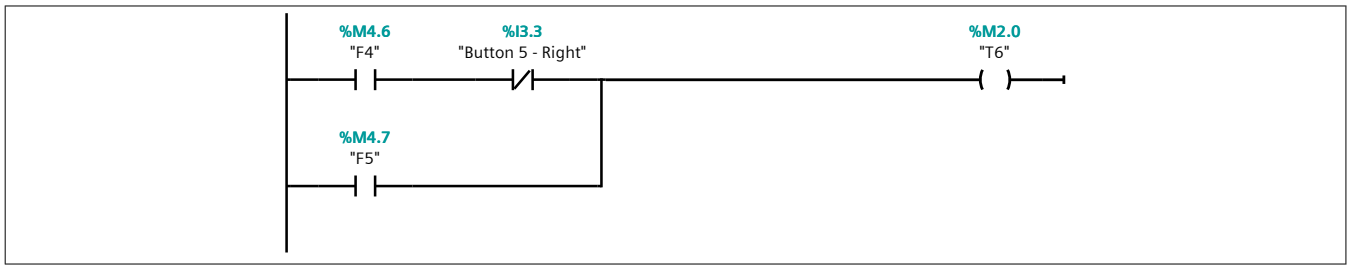
## Network 4:



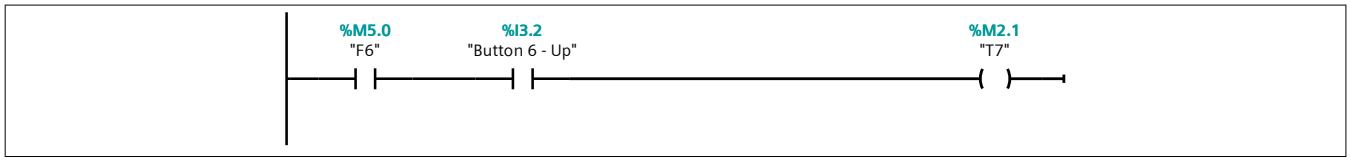

## Network 5:



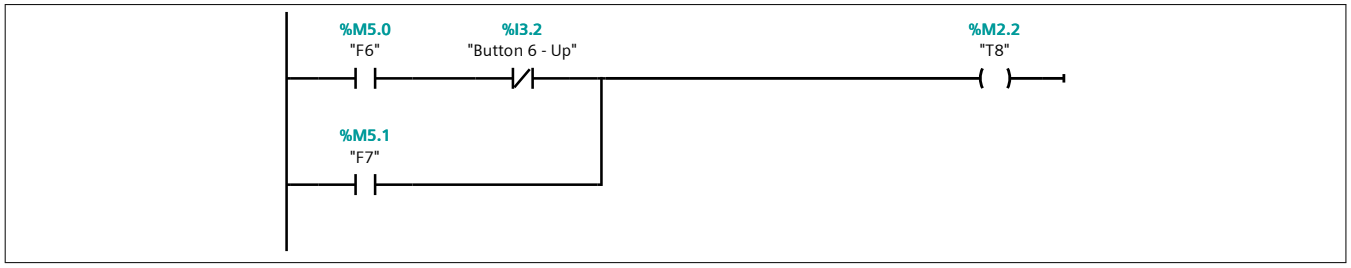

## Network 6:



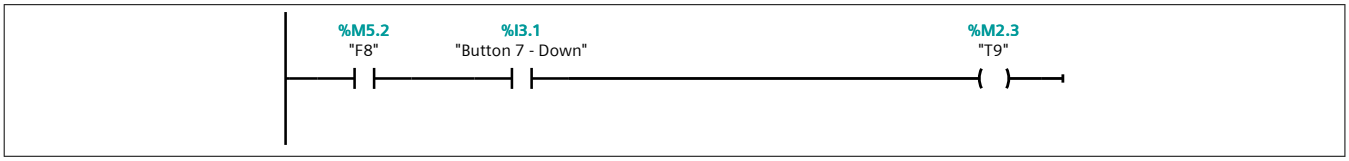

## Network 7:



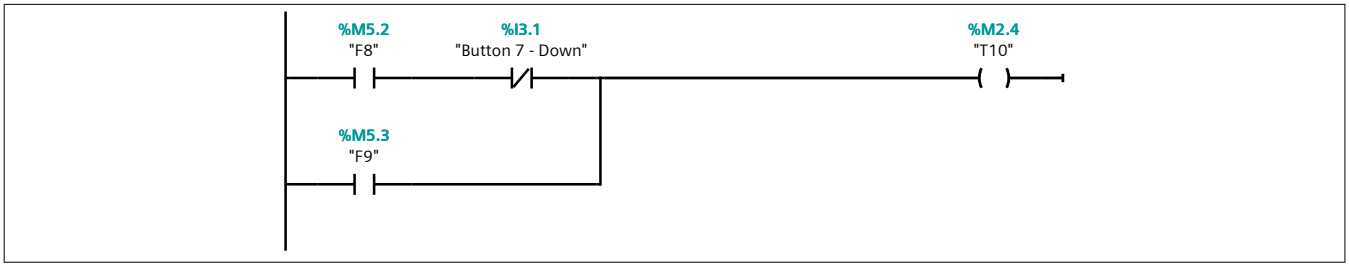

## Network 8:



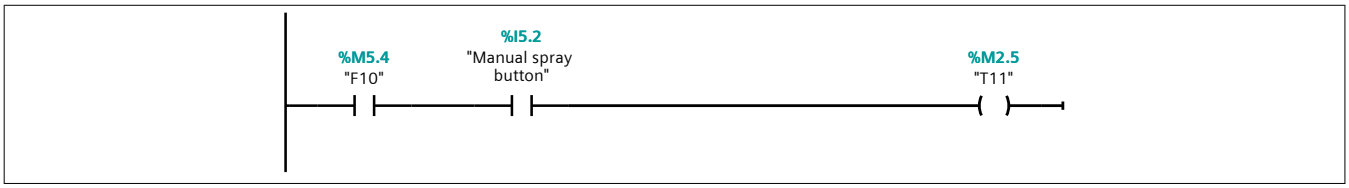

## Network 9:



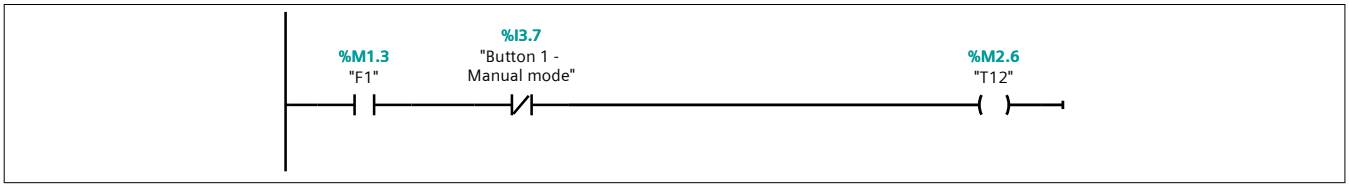

**Network 10:**



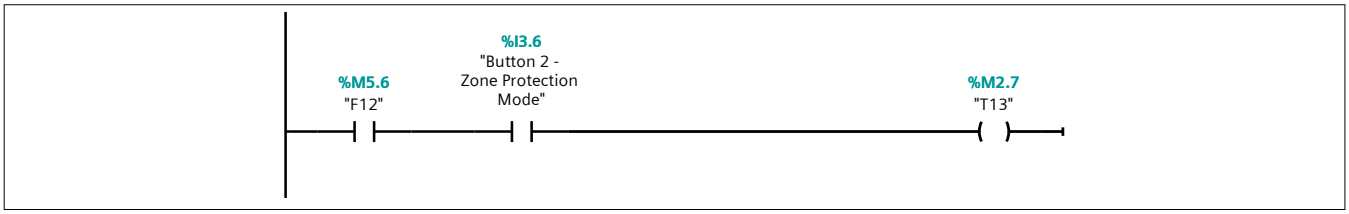

**Network 11:**




**Network 12:**

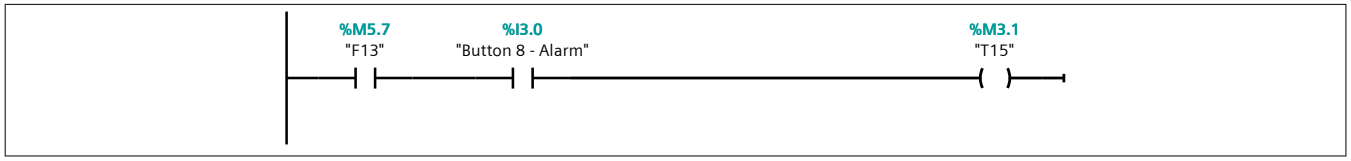



**Network 13:**

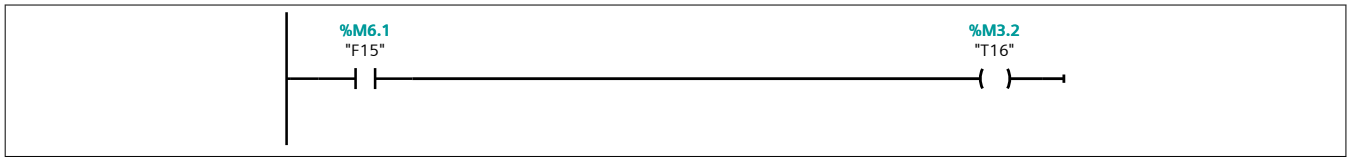



**Network 14:**

## Network 15:

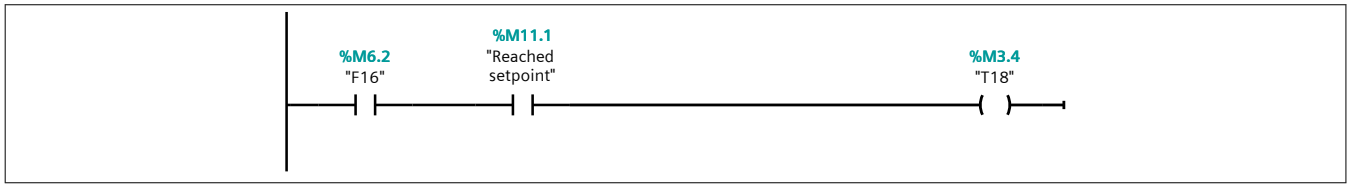



## Network 16:

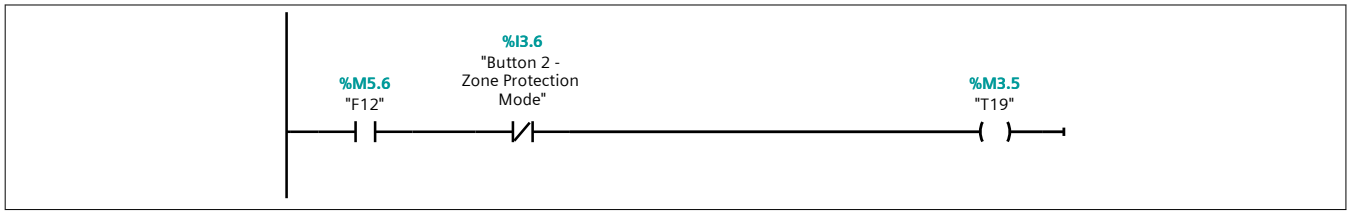



## Network 17:

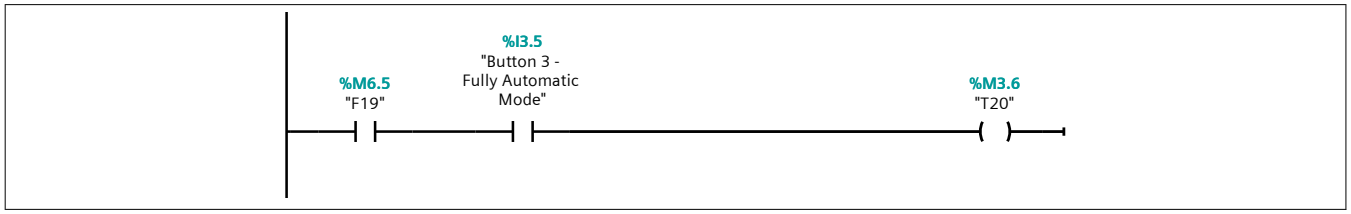

## Network 18:




## Network 19:

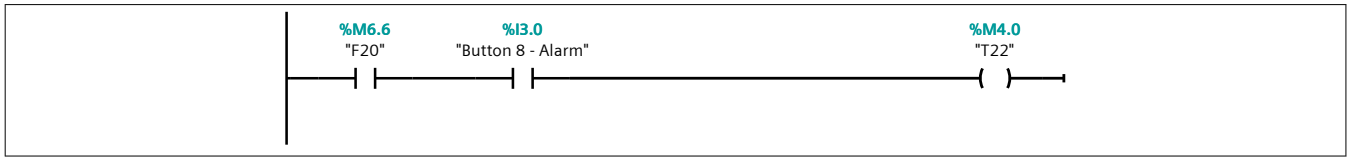



## Network 20:

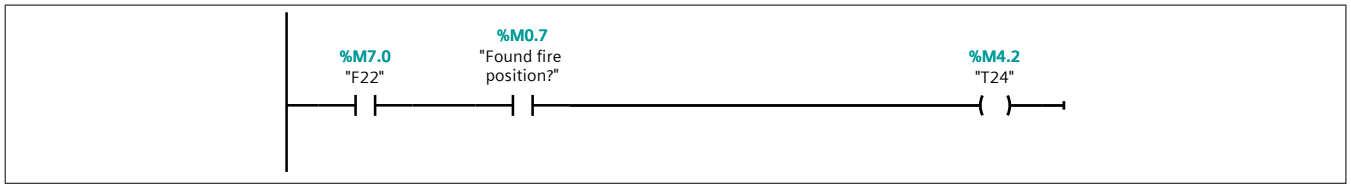


## Network 21:

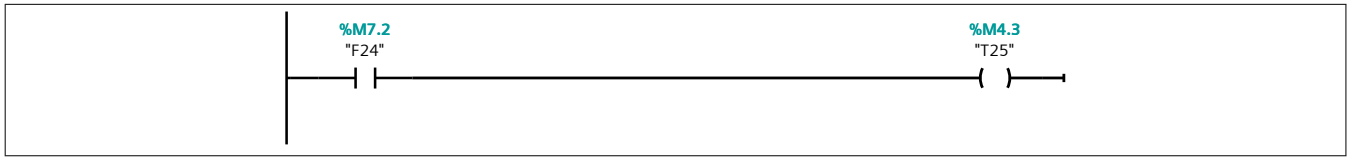

## Network 22:



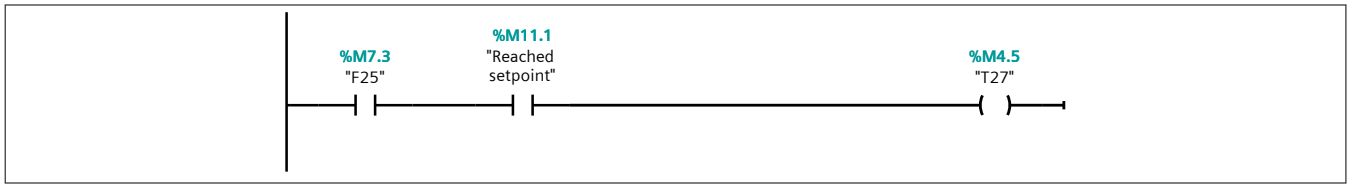

## Network 23:




## Network 24:




## Network 25:



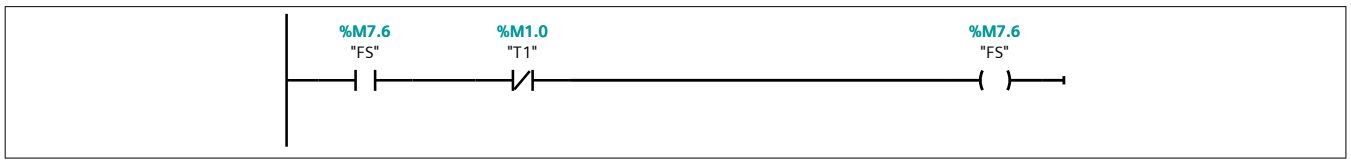
# Program blocks

## Functions [FC19]

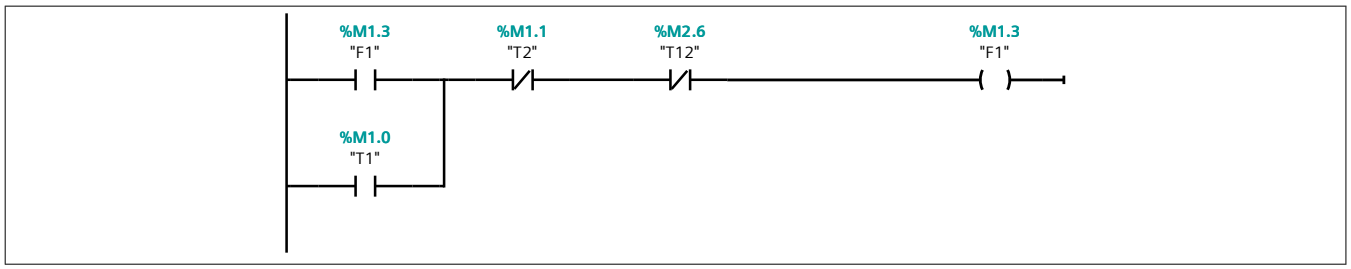
| Functions Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Functions | **Number** | 19 | **Type** | FC |
| **Language** | LAD | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| Output | | |
| InOut | | |
| Temp | | |
| ▼ Return | | |
| Functions | Void | |

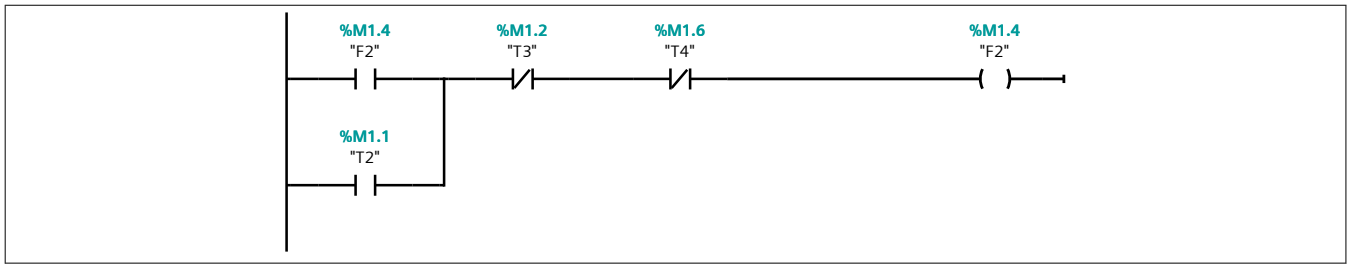
### Network 1:




### Network 2:

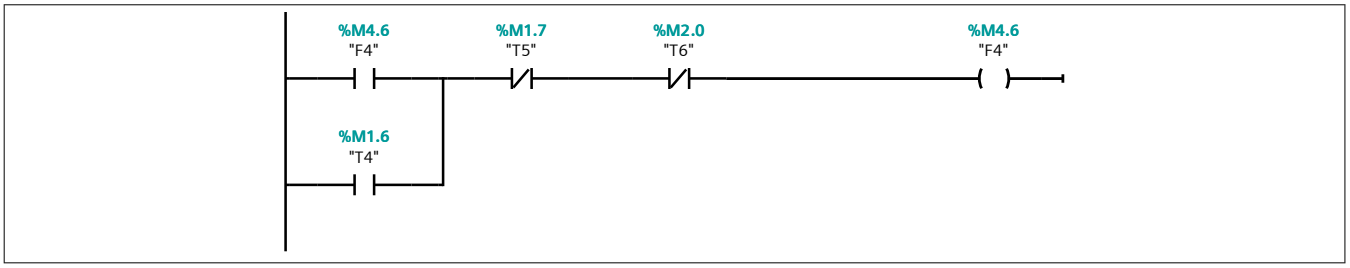



### Network 3:



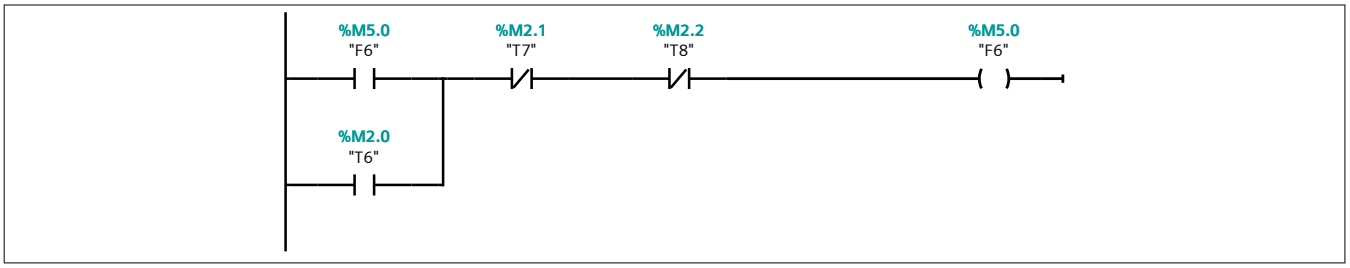

### Network 4:


**Network 5:**



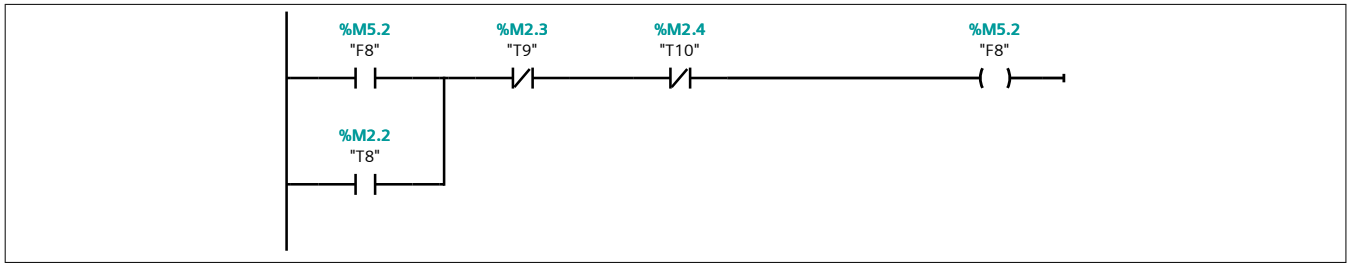

**Network 6:**


**Network 7:**



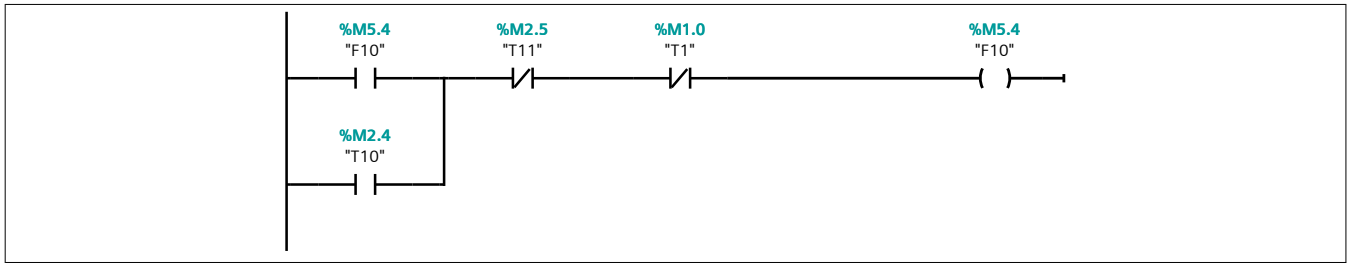

**Network 8:**

**Network 9:**



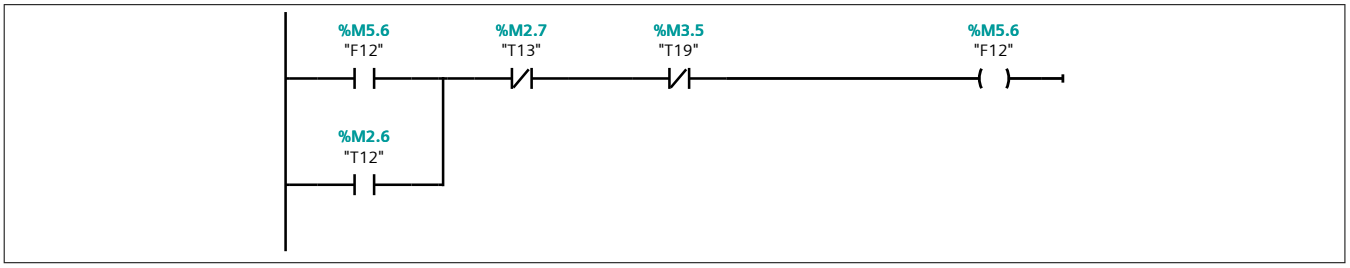

**Network 10:**


**Network 11:**

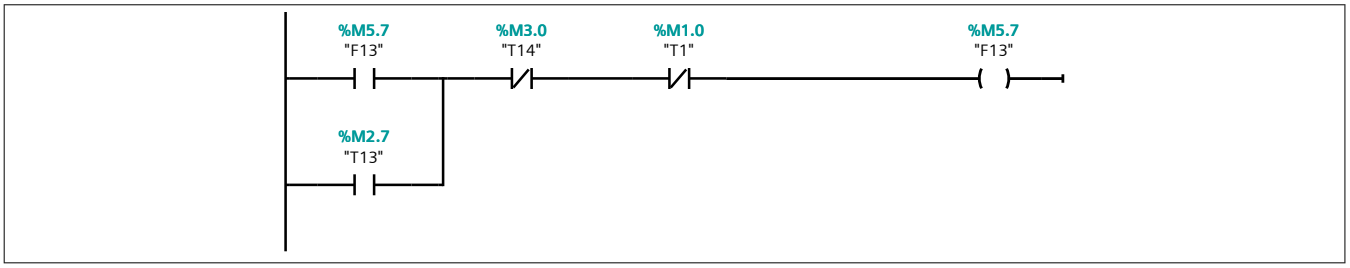



**Network 12:**


**Network 13:**

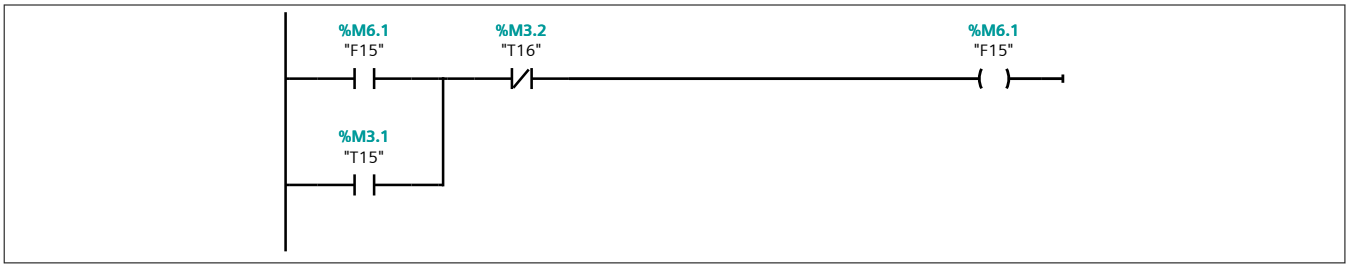



**Network 14:**

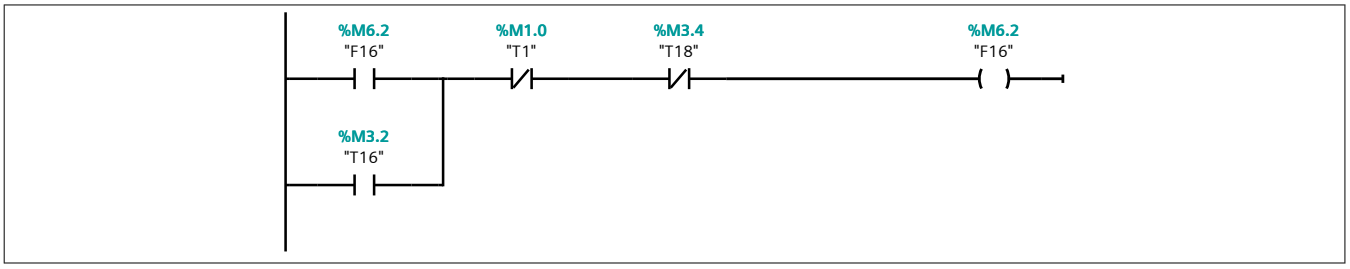



**Network 15:**


**Network 16:**

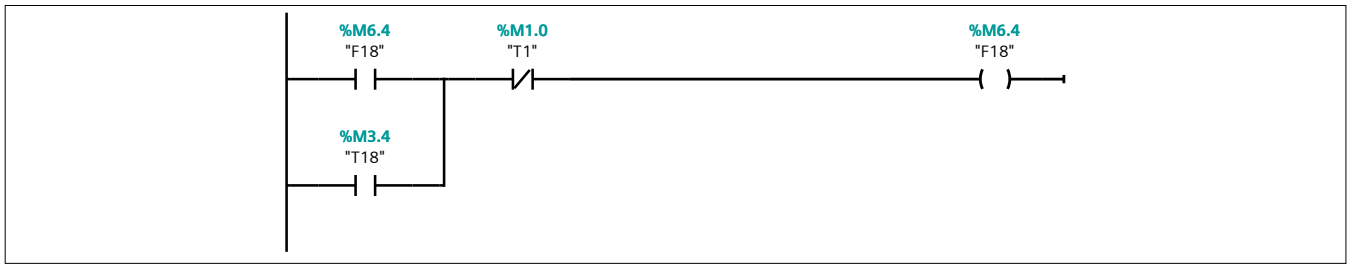



**Network 17:**

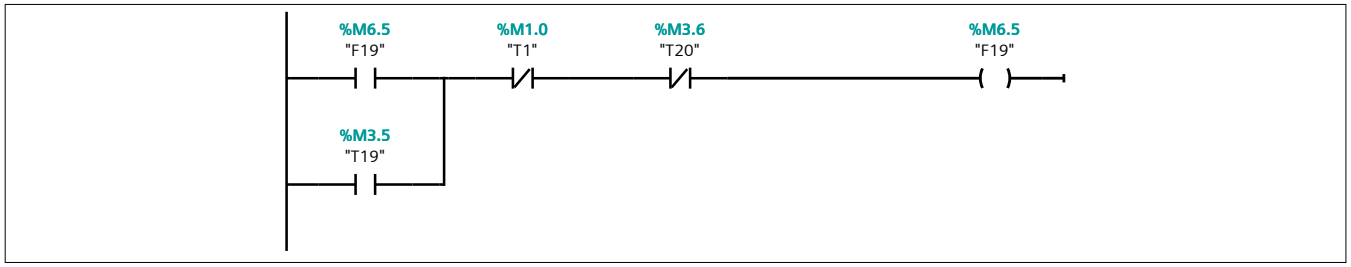


**Network 18:**

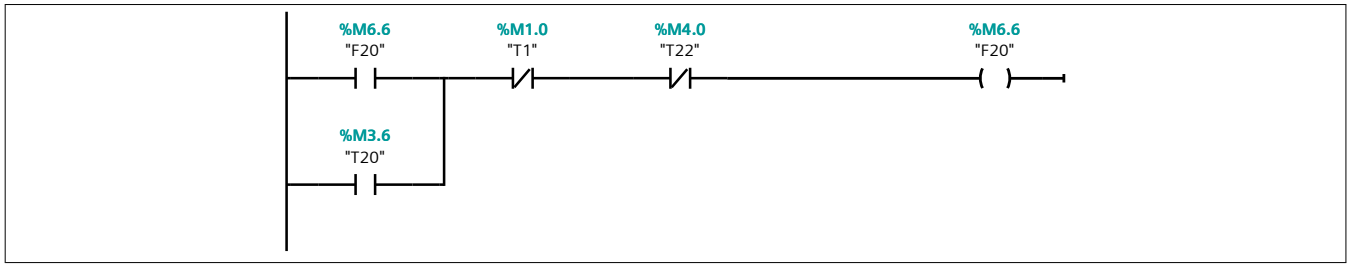

**Network 19:**




**Network 20:**

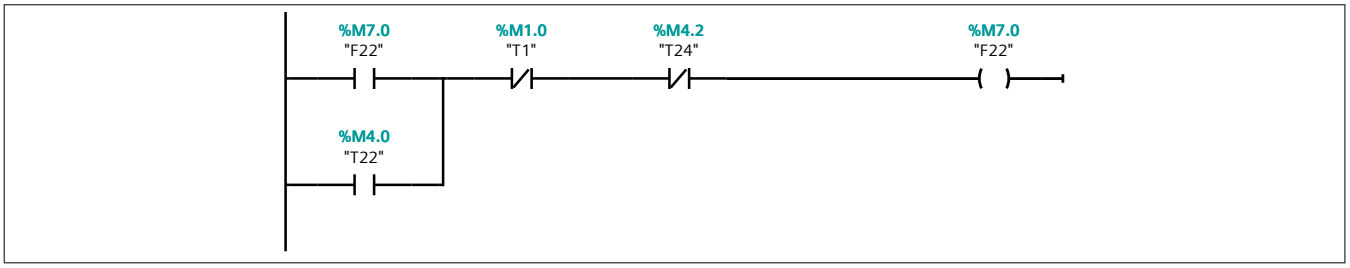



**Network 21:**



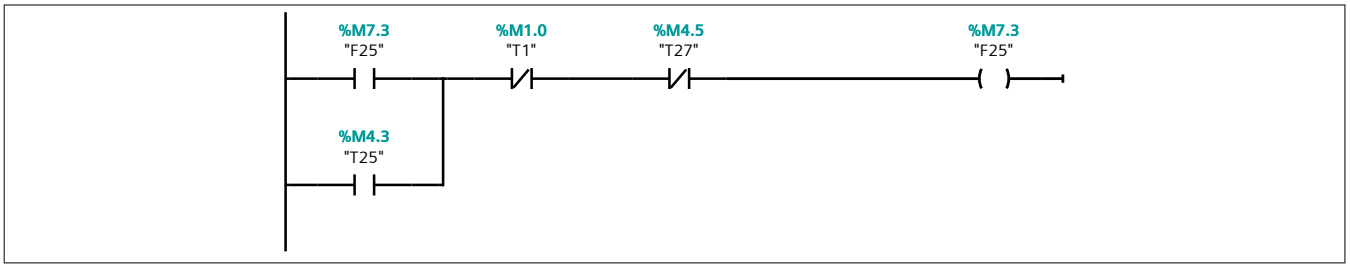

**Network 22:**


**Network 23:**




**Network 24:**


**Network 25:**




**Network 26:**

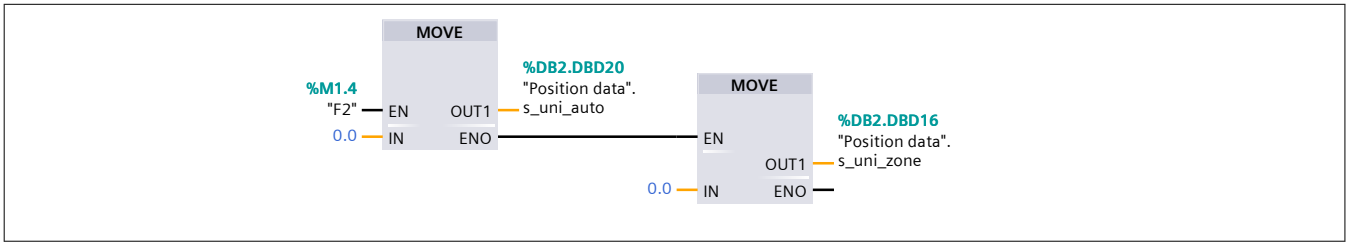
# Program blocks

## Outputs [FC20]

| Outputs Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Outputs | **Number** | 20 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| Input | | |
| Output | | |
| InOut | | |
| ▼ Temp | | |
| Temp_pressure | Real | 0.0 |
| ▼ Return | | |
| Outputs | Void | |

### Network 1:



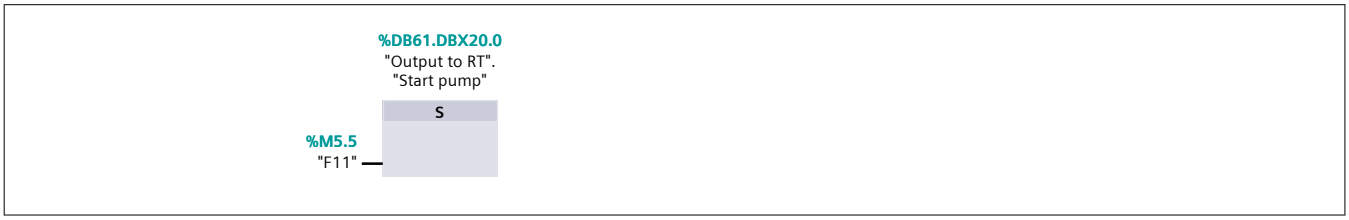

### Network 2:


### Network 3:

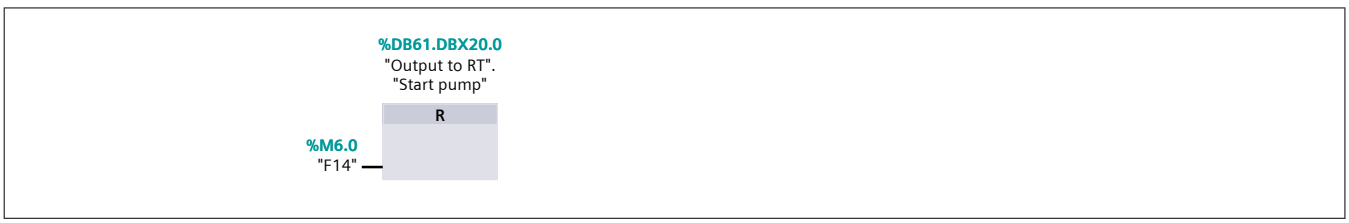

## Network 4:


## Network 5:

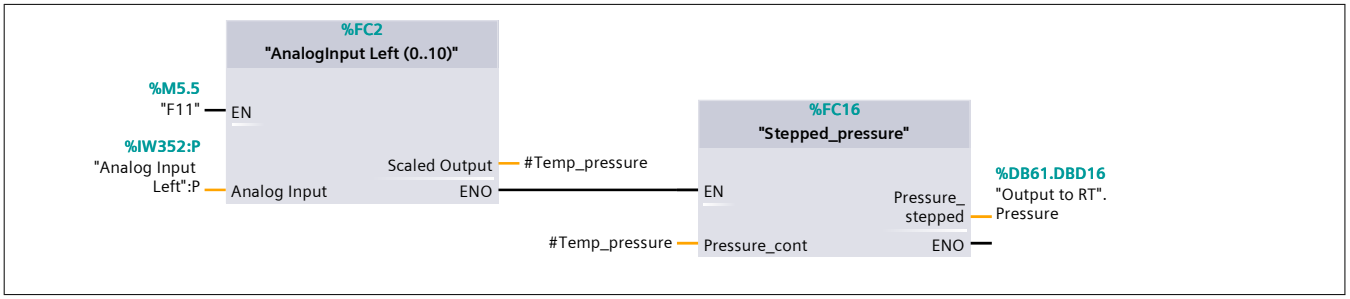

## Network 6:



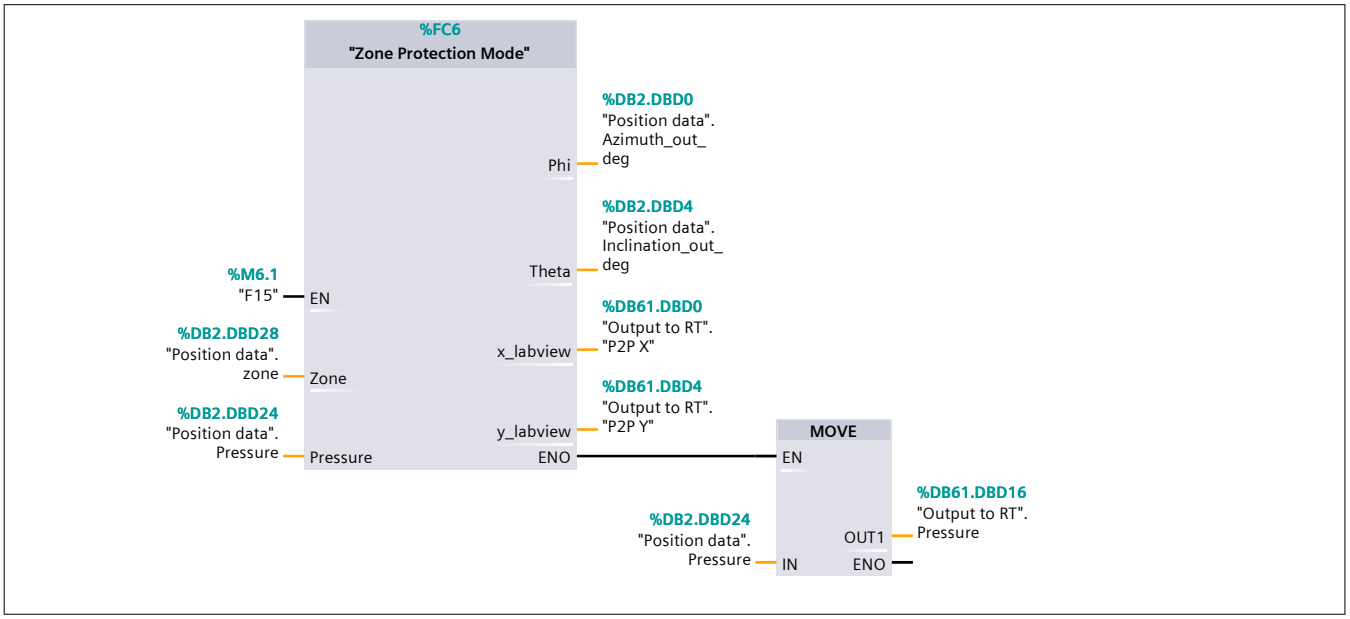

## Network 7:



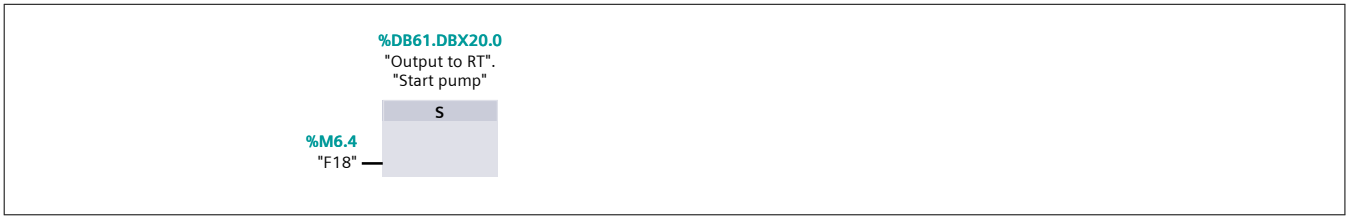
## Network 8:



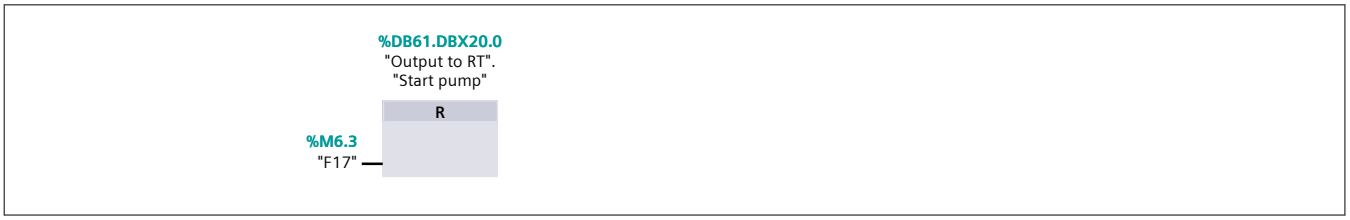

## Network 9:



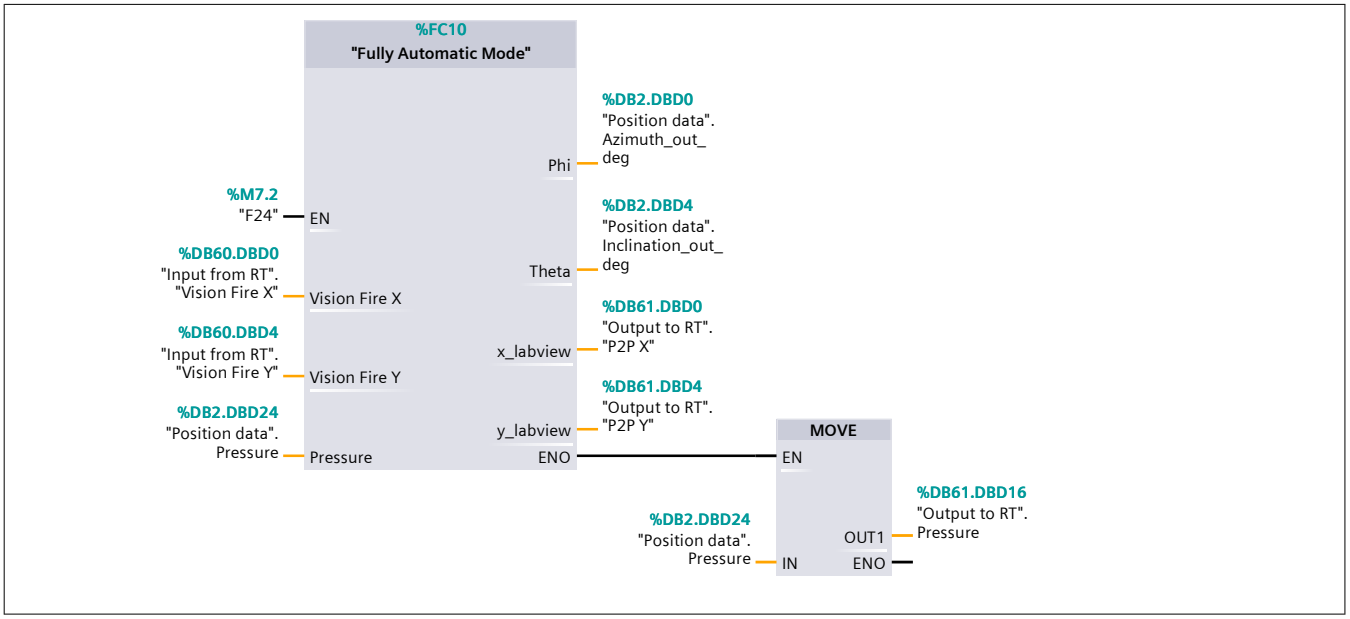

## Network 10:



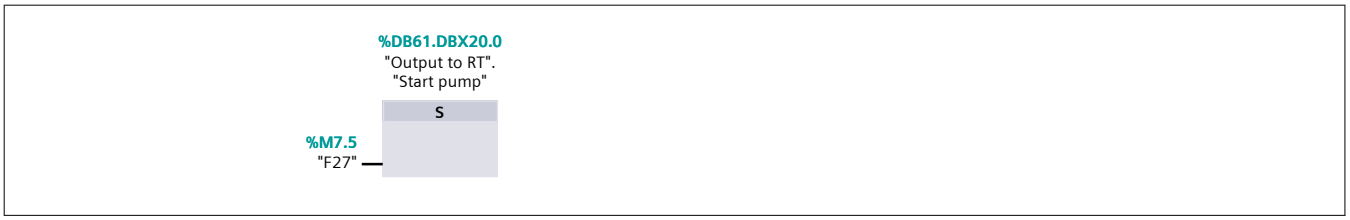

## Network 11:



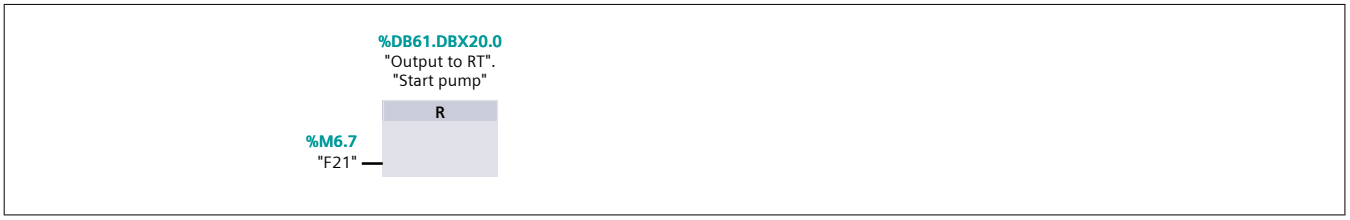
## Network 12:




## Network 13:




## Network 14:



# Program blocks

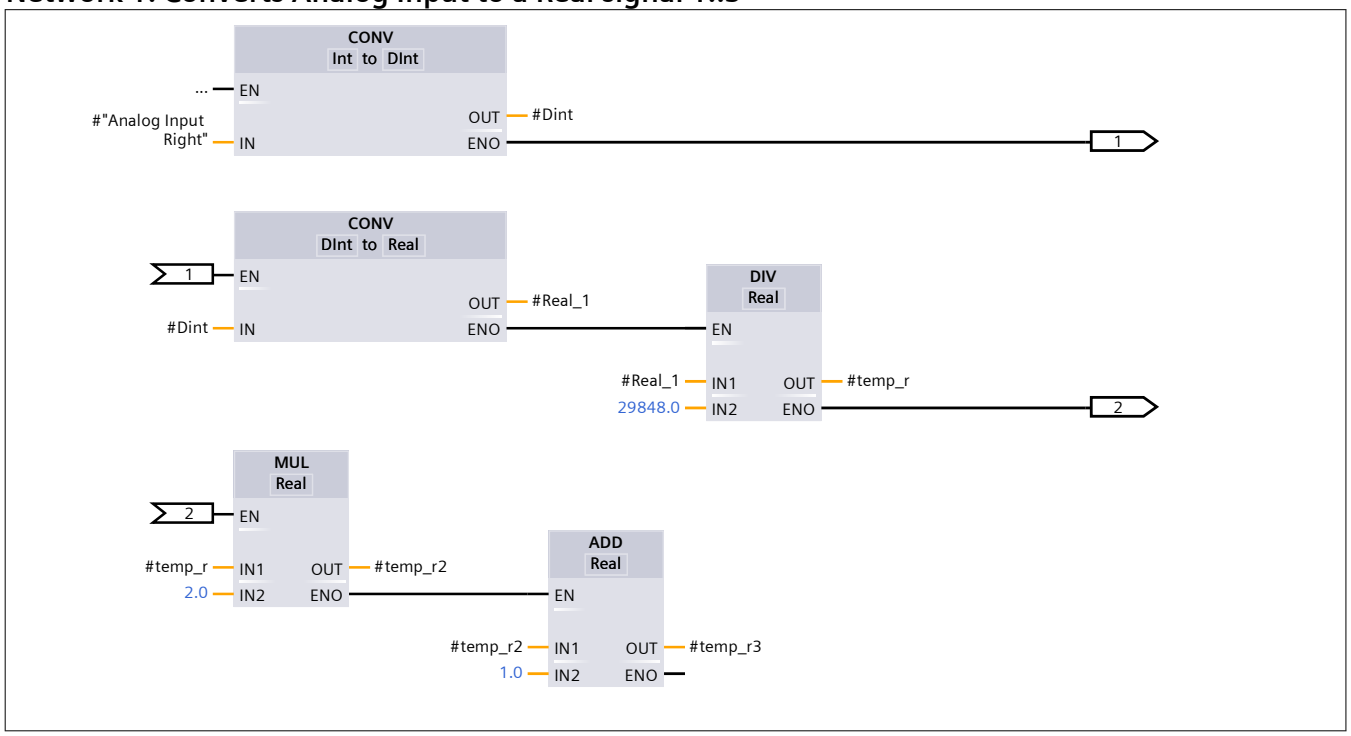
## AnalogInput Right (1..3) [FC4]

| AnalogInput Right (1..3) Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | AnalogInput Right (1..3) | **Number** | 4 | **Type** | FC |
| **Language** | FBD | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

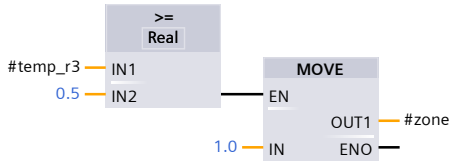
| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| Analog Input Right | Int | |
| ▼ Output | | |
| zone | Real | |
| InOut | | |
| ▼ Temp | | |
| Dint | DInt | 0.0 |
| Real_1 | Real | 4.0 |
| temp_r | Real | 8.0 |
| temp_r2 | Real | 12.0 |
| temp_r3 | Real | 16.0 |
| ▼ Return | | |
| AnalogInput Right (1..3) | Void | |

### Network 1: Converts Analog Input to a Real signal 1..3

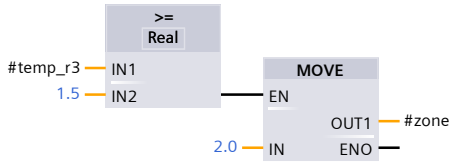
### Network 1: Converts Analog Input to a Real signal 1..3



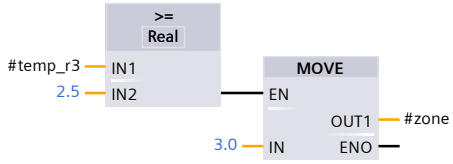
## Network 2: Step to 3




## Network 3: Step to 2




## Network 4: Step to 1



# Program blocks

## Point-To-Point Zone 2 [FC21]

| Point-To-Point Zone 2 Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Point-To-Point Zone 2 | **Number** | 21 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| At path start | Bool | |
| ▼ Output | | |
| x_out | Real | |
| y_out | Real | |
| ▼ InOut | | |
| s_uni | Real | |
| ▼ Temp | | |
| ▼ X_array | Array [1..21] of Real | 0.0 |
| X_array[1] | Real | 0.0 |
| X_array[2] | Real | 4.0 |
| X_array[3] | Real | 8.0 |
| X_array[4] | Real | 12.0 |
| X_array[5] | Real | 16.0 |
| X_array[6] | Real | 20.0 |
| X_array[7] | Real | 24.0 |
| X_array[8] | Real | 28.0 |
| X_array[9] | Real | 32.0 |
| X_array[10] | Real | 36.0 |
| X_array[11] | Real | 40.0 |
| X_array[12] | Real | 44.0 |
| X_array[13] | Real | 48.0 |
| X_array[14] | Real | 52.0 |
| X_array[15] | Real | 56.0 |
| X_array[16] | Real | 60.0 |
| X_array[17] | Real | 64.0 |
| X_array[18] | Real | 68.0 |
| X_array[19] | Real | 72.0 |
| X_array[20] | Real | 76.0 |
| X_array[21] | Real | 80.0 |
| ▼ Y_array | Array [1..21] of Real | 84.0 |
| Y_array[1] | Real | 0.0 |
| Y_array[2] | Real | 4.0 |
| Y_array[3] | Real | 8.0 |
| Y_array[4] | Real | 12.0 |
| Y_array[5] | Real | 16.0 |
| Y_array[6] | Real | 20.0 |
| Y_array[7] | Real | 24.0 |

| Name | Data type | Offset |
|---|---|---|
| Y_array[8] | Real | 28.0 |
| Y_array[9] | Real | 32.0 |
| Y_array[10] | Real | 36.0 |
| Y_array[11] | Real | 40.0 |
| Y_array[12] | Real | 44.0 |
| Y_array[13] | Real | 48.0 |
| Y_array[14] | Real | 52.0 |
| Y_array[15] | Real | 56.0 |
| Y_array[16] | Real | 60.0 |
| Y_array[17] | Real | 64.0 |
| Y_array[18] | Real | 68.0 |
| Y_array[19] | Real | 72.0 |
| Y_array[20] | Real | 76.0 |
| Y_array[21] | Real | 80.0 |
| ▼ s_dx | Array [1..21] of Real | 168.0 |
| s_dx[1] | Real | 0.0 |
| s_dx[2] | Real | 4.0 |
| s_dx[3] | Real | 8.0 |
| s_dx[4] | Real | 12.0 |
| s_dx[5] | Real | 16.0 |
| s_dx[6] | Real | 20.0 |
| s_dx[7] | Real | 24.0 |
| s_dx[8] | Real | 28.0 |
| s_dx[9] | Real | 32.0 |
| s_dx[10] | Real | 36.0 |
| s_dx[11] | Real | 40.0 |
| s_dx[12] | Real | 44.0 |
| s_dx[13] | Real | 48.0 |
| s_dx[14] | Real | 52.0 |
| s_dx[15] | Real | 56.0 |
| s_dx[16] | Real | 60.0 |
| s_dx[17] | Real | 64.0 |
| s_dx[18] | Real | 68.0 |
| s_dx[19] | Real | 72.0 |
| s_dx[20] | Real | 76.0 |
| s_dx[21] | Real | 80.0 |
| ▼ s_dy | Array [1..21] of Real | 252.0 |
| s_dy[1] | Real | 0.0 |
| s_dy[2] | Real | 4.0 |
| s_dy[3] | Real | 8.0 |
| s_dy[4] | Real | 12.0 |
| s_dy[5] | Real | 16.0 |
| s_dy[6] | Real | 20.0 |
| s_dy[7] | Real | 24.0 |
| s_dy[8] | Real | 28.0 |
| s_dy[9] | Real | 32.0 |
| s_dy[10] | Real | 36.0 |
| s_dy[11] | Real | 40.0 |
| s_dy[12] | Real | 44.0 |
| s_dy[13] | Real | 48.0 |
| s_dy[14] | Real | 52.0 |
| s_dy[15] | Real | 56.0 |

| Name | Data type | Offset |
|---|---|---|
| s_dy[16] | Real | 60.0 |
| s_dy[17] | Real | 64.0 |
| s_dy[18] | Real | 68.0 |
| s_dy[19] | Real | 72.0 |
| s_dy[20] | Real | 76.0 |
| s_dy[21] | Real | 80.0 |
| ▼ s_length | Array [1..21] of Real | 336.0 |
| s_length[1] | Real | 0.0 |
| s_length[2] | Real | 4.0 |
| s_length[3] | Real | 8.0 |
| s_length[4] | Real | 12.0 |
| s_length[5] | Real | 16.0 |
| s_length[6] | Real | 20.0 |
| s_length[7] | Real | 24.0 |
| s_length[8] | Real | 28.0 |
| s_length[9] | Real | 32.0 |
| s_length[10] | Real | 36.0 |
| s_length[11] | Real | 40.0 |
| s_length[12] | Real | 44.0 |
| s_length[13] | Real | 48.0 |
| s_length[14] | Real | 52.0 |
| s_length[15] | Real | 56.0 |
| s_length[16] | Real | 60.0 |
| s_length[17] | Real | 64.0 |
| s_length[18] | Real | 68.0 |
| s_length[19] | Real | 72.0 |
| s_length[20] | Real | 76.0 |
| s_length[21] | Real | 80.0 |
| i | Int | 420.0 |
| s | Real | 422.0 |
| v | Real | 426.0 |
| x | Real | 430.0 |
| y | Real | 434.0 |
| dt | Real | 438.0 |
| v_x | Real | 442.0 |
| v_y | Real | 446.0 |
| v_update | Real | 450.0 |
| ▼ Return | | |
| Point-To-Point Zone 2 | Void | |

```
0001 //Defines each corner in a square spiral shape around the calculates fire po-
     sition, X- and Y coordinate correspond to a point.
0002 //Defines x-points
0003 #X_array[1]:=21.2;
0004 #X_array[2]:=18.2;
0005 #X_array[3]:=24.2;
0006 #X_array[4]:=25.2;
0007 #X_array[5]:=17.2;
0008 #X_array[6]:=16.6;
0009 #X_array[7]:=25.8;
0010 #X_array[8]:=26.1;
0011 #X_array[9]:=16.3;
0012 #X_array[10]:=16.2;
0013 #X_array[11]:=26.2;
```

```
0014 #X_array[12]:=26.1;
0015 #X_array[13]:=16.3;
0016 #X_array[14]:=16.6;
0017 #X_array[15]:=25.8;
0018 #X_array[16]:=25.2;
0019 #X_array[17]:=17.2;
0020 #X_array[18]:=18.2;
0021 #X_array[19]:=24.2;
0022 #X_array[20]:=21.2;
0023 #X_array[21]:=21.2;
0024 //Defines y-points
0025 #Y_array[1]:=26.1;
0026 #Y_array[2]:=25.2;
0027 #Y_array[3]:=25.2;
0028 #Y_array[4]:=24.2;
0029 #Y_array[5]:=24.2;
0030 #Y_array[6]:=23.2;
0031 #Y_array[7]:=23.2;
0032 #Y_array[8]:=22.2;
0033 #Y_array[9]:=22.2;
0034 #Y_array[10]:=21.2;
0035 #Y_array[11]:=21.2;
0036 #Y_array[12]:=20.2;
0037 #Y_array[13]:=20.2;
0038 #Y_array[14]:=19.2;
0039 #Y_array[15]:=19.2;
0040 #Y_array[16]:=18.2;
0041 #Y_array[17]:=18.2;
0042 #Y_array[18]:=17.2;
0043 #Y_array[19]:=17.2;
0044 #Y_array[20]:=16.2;
0045 #Y_array[21]:=26.2;
0046
0047 //Path speed
0048 #v:=0.8;
0049 //Interupt time for OB35 in sec
0050 #dt:=0.010;
0051
0052 #x_out:=#X_array[1]+#x;
0053 #y_out:=#Y_array[1]+#y;
0054
0055 #s:=#s_uni;
0056 IF #"At path start"
0057 THEN
0058 //Calculate the vector orientation and length from a point to the next one
0059 FOR #i := 1 TO 21
0060  DO  #s_dx[#i]:= #X_array[(#i+1)]-#X_array[(#i)];
0061      #s_dy[#i]:= #Y_array[(#i+1)]-#Y_array[(#i)];
0062      #s_length[#i]:=SQRT(#s_dx[#i]**2+#s_dy[#i]**2);
0063 END_FOR;
0064
0065 //If on first strech, then go this direction and update the postion for each
     cycle
0066 //Do this for every strech..
0067   IF #s<#s_length[1]
0068 THEN
0069     #v_x:=(#s_dx[1]/#s_length[1])*#v;
0070     #v_y:=(#s_dy[1]/#s_length[1])*#v;
0071     #x:=#x+#v_x*#dt;
```

```
0072      #y:=#y+#v_y*#dt;
0073
0074  ELSIF  #s<(#s_length[1]+#s_length[2])
0075  THEN
0076      #v_x:=(#s_dx[2]/#s_length[2])*#v;
0077      #v_y:=(#s_dy[2]/#s_length[2])*#v;
0078      #x:=#x+#v_x*#dt;
0079      #y:=#y+#v_y*#dt;
0080
0081  ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3])
0082  THEN
0083      #v_x:=(#s_dx[3]/#s_length[3])*#v;
0084      #v_y:=(#s_dy[3]/#s_length[3])*#v;
0085      #x:=#x+#v_x*#dt;
0086      #y:=#y+#v_y*#dt;
0087  ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4])
0088  THEN
0089      #v_x:=(#s_dx[4]/#s_length[4])*#v;
0090      #v_y:=(#s_dy[4]/#s_length[4])*#v;
0091      #x:=#x+#v_x*#dt;
0092      #y:=#y+#v_y*#dt;
0093  ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5])
0094  THEN
0095      #v_x:=(#s_dx[5]/#s_length[5])*#v;
0096      #v_y:=(#s_dy[5]/#s_length[5])*#v;
0097      #x:=#x+#v_x*#dt;
0098      #y:=#y+#v_y*#dt;
0099  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6])
0100  THEN
0101      #v_x:=(#s_dx[6]/#s_length[6])*#v;
0102      #v_y:=(#s_dy[6]/#s_length[6])*#v;
0103      #x:=#x+#v_x*#dt;
0104      #y:=#y+#v_y*#dt;
0105  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7])
0106  THEN
0107      #v_x:=(#s_dx[7]/#s_length[7])*#v;
0108      #v_y:=(#s_dy[7]/#s_length[7])*#v;
0109      #x:=#x+#v_x*#dt;
0110      #y:=#y+#v_y*#dt;
0111  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8])
0112  THEN
0113      #v_x:=(#s_dx[8]/#s_length[8])*#v;
0114      #v_y:=(#s_dy[8]/#s_length[8])*#v;
0115      #x:=#x+#v_x*#dt;
0116      #y:=#y+#v_y*#dt;
0117  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9])
0118  THEN
0119      #v_x:=(#s_dx[9]/#s_length[9])*#v;
0120      #v_y:=(#s_dy[9]/#s_length[9])*#v;
0121      #x:=#x+#v_x*#dt;
0122      #y:=#y+#v_y*#dt;
```

```
0123 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10])
0124 THEN
0125     #v_x:=(#s_dx[10]/#s_length[10])*#v;
0126     #v_y:=(#s_dy[10]/#s_length[10])*#v;
0127     #x:=#x+#v_x*#dt;
0128     #y:=#y+#v_y*#dt;
0129 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11])
0130 THEN
0131     #v_x:=(#s_dx[11]/#s_length[11])*#v;
0132     #v_y:=(#s_dy[11]/#s_length[11])*#v;
0133     #x:=#x+#v_x*#dt;
0134     #y:=#y+#v_y*#dt;
0135 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12])
0136 THEN
0137     #v_x:=(#s_dx[12]/#s_length[12])*#v;
0138     #v_y:=(#s_dy[12]/#s_length[12])*#v;
0139     #x:=#x+#v_x*#dt;
0140     #y:=#y+#v_y*#dt;
0141 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13])
0142 THEN
0143     #v_x:=(#s_dx[13]/#s_length[13])*#v;
0144     #v_y:=(#s_dy[13]/#s_length[13])*#v;
0145     #x:=#x+#v_x*#dt;
0146     #y:=#y+#v_y*#dt;
0147 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14])
0148 THEN
0149     #v_x:=(#s_dx[14]/#s_length[14])*#v;
0150     #v_y:=(#s_dy[14]/#s_length[14])*#v;
0151     #x:=#x+#v_x*#dt;
0152     #y:=#y+#v_y*#dt;
0153 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15])
0154 THEN
0155     #v_x:=(#s_dx[15]/#s_length[15])*#v;
0156     #v_y:=(#s_dy[15]/#s_length[15])*#v;
0157     #x:=#x+#v_x*#dt;
0158     #y:=#y+#v_y*#dt;
0159 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16])
0160 THEN
0161     #v_x:=(#s_dx[16]/#s_length[16])*#v;
0162     #v_y:=(#s_dy[16]/#s_length[16])*#v;
```

```
0163      #x:=#x+#v_x*#dt;
0164      #y:=#y+#v_y*#dt;
0165  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
      th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17])
0166  THEN
0167      #v_x:=(#s_dx[17]/#s_length[17])*#v;
0168      #v_y:=(#s_dy[17]/#s_length[17])*#v;
0169      #x:=#x+#v_x*#dt;
0170      #y:=#y+#v_y*#dt;
0171  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
      th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
      s_length[18])
0172  THEN
0173      #v_x:=(#s_dx[18]/#s_length[18])*#v;
0174      #v_y:=(#s_dy[18]/#s_length[18])*#v;
0175      #x:=#x+#v_x*#dt;
0176      #y:=#y+#v_y*#dt;
0177  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
      th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
      s_length[18]+#s_length[19])
0178  THEN
0179      #v_x:=(#s_dx[19]/#s_length[19])*#v;
0180      #v_y:=(#s_dy[19]/#s_length[19])*#v;
0181      #x:=#x+#v_x*#dt;
0182      #y:=#y+#v_y*#dt;
0183  ELSIF
      #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
      th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
      s_length[18]+#s_length[19]+#s_length[20])
0184  THEN
0185      #v_x:=(#s_dx[20]/#s_length[20])*#v;
0186      #v_y:=(#s_dy[20]/#s_length[20])*#v;
0187      #x:=#x+#v_x*#dt;
0188      #y:=#y+#v_y*#dt;
0189  END_IF;
0190  //Reset when whole route is gone
0191    IF
      #s>(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
      [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
      th[12]+#s_length[13]+#s_length[14]+#s_length[15]+#s_length[16]+#s_length[17]+#
      s_length[18]+#s_length[19]+#s_length[20])
0192  THEN
0193    #s:=0;
0194    #x:=0;
0195    #y:=0;
0196  END_IF;
0197
0198  //Send out path position values
0199  //Update velocity and distance gone
0200  #x_out:=#X_array[1]+#x;
0201  #y_out:=#Y_array[1]+#y;
0202  #v_update:=SQRT(#v_x**2+#v_y**2);
```

```
0203 #s_uni:=#s+#v_update*#dt;
0204 END_IF;
```

# Program blocks

## Point-To-Point Zone 3 [FC22]

| Point-To-Point Zone 3 Properties | | | | | |
|---|---|---|---|---|---|
| **General** | | | | | |
| **Name** | Point-To-Point Zone 3 | **Number** | 22 | **Type** | FC |
| **Language** | SCL | | | | |
| **Information** | | | | | |
| **Title** | | **Author** | | **Comment** | |
| **Family** | | **Version** | 0.1 | **User-defined ID** | |

| Name | Data type | Offset |
|---|---|---|
| ▼ Input | | |
| At path start | Bool | |
| ▼ Output | | |
| x_out | Real | |
| y_out | Real | |
| ▼ InOut | | |
| s_uni | Real | |
| ▼ Temp | | |
| ▼ X_array | Array [1..15] of Real | 0.0 |
| X_array[1] | Real | 0.0 |
| X_array[2] | Real | 4.0 |
| X_array[3] | Real | 8.0 |
| X_array[4] | Real | 12.0 |
| X_array[5] | Real | 16.0 |
| X_array[6] | Real | 20.0 |
| X_array[7] | Real | 24.0 |
| X_array[8] | Real | 28.0 |
| X_array[9] | Real | 32.0 |
| X_array[10] | Real | 36.0 |
| X_array[11] | Real | 40.0 |
| X_array[12] | Real | 44.0 |
| X_array[13] | Real | 48.0 |
| X_array[14] | Real | 52.0 |
| X_array[15] | Real | 56.0 |
| ▼ Y_array | Array [1..15] of Real | 60.0 |
| Y_array[1] | Real | 0.0 |
| Y_array[2] | Real | 4.0 |
| Y_array[3] | Real | 8.0 |
| Y_array[4] | Real | 12.0 |
| Y_array[5] | Real | 16.0 |
| Y_array[6] | Real | 20.0 |
| Y_array[7] | Real | 24.0 |
| Y_array[8] | Real | 28.0 |
| Y_array[9] | Real | 32.0 |
| Y_array[10] | Real | 36.0 |
| Y_array[11] | Real | 40.0 |
| Y_array[12] | Real | 44.0 |
| Y_array[13] | Real | 48.0 |

| Name | Data type | Offset |
|---|---|---|
| Y_array[14] | Real | 52.0 |
| Y_array[15] | Real | 56.0 |
| s_dx | Array [1..15] of Real | 120.0 |
| s_dx[1] | Real | 0.0 |
| s_dx[2] | Real | 4.0 |
| s_dx[3] | Real | 8.0 |
| s_dx[4] | Real | 12.0 |
| s_dx[5] | Real | 16.0 |
| s_dx[6] | Real | 20.0 |
| s_dx[7] | Real | 24.0 |
| s_dx[8] | Real | 28.0 |
| s_dx[9] | Real | 32.0 |
| s_dx[10] | Real | 36.0 |
| s_dx[11] | Real | 40.0 |
| s_dx[12] | Real | 44.0 |
| s_dx[13] | Real | 48.0 |
| s_dx[14] | Real | 52.0 |
| s_dx[15] | Real | 56.0 |
| s_dy | Array [1..15] of Real | 180.0 |
| s_dy[1] | Real | 0.0 |
| s_dy[2] | Real | 4.0 |
| s_dy[3] | Real | 8.0 |
| s_dy[4] | Real | 12.0 |
| s_dy[5] | Real | 16.0 |
| s_dy[6] | Real | 20.0 |
| s_dy[7] | Real | 24.0 |
| s_dy[8] | Real | 28.0 |
| s_dy[9] | Real | 32.0 |
| s_dy[10] | Real | 36.0 |
| s_dy[11] | Real | 40.0 |
| s_dy[12] | Real | 44.0 |
| s_dy[13] | Real | 48.0 |
| s_dy[14] | Real | 52.0 |
| s_dy[15] | Real | 56.0 |
| s_length | Array [1..15] of Real | 240.0 |
| s_length[1] | Real | 0.0 |
| s_length[2] | Real | 4.0 |
| s_length[3] | Real | 8.0 |
| s_length[4] | Real | 12.0 |
| s_length[5] | Real | 16.0 |
| s_length[6] | Real | 20.0 |
| s_length[7] | Real | 24.0 |
| s_length[8] | Real | 28.0 |
| s_length[9] | Real | 32.0 |
| s_length[10] | Real | 36.0 |
| s_length[11] | Real | 40.0 |
| s_length[12] | Real | 44.0 |
| s_length[13] | Real | 48.0 |
| s_length[14] | Real | 52.0 |
| s_length[15] | Real | 56.0 |
| i | Int | 300.0 |
| s | Real | 302.0 |

| Name | Data type | Offset |
|---|---|---|
| v | Real | 306.0 |
| x | Real | 310.0 |
| y | Real | 314.0 |
| dt | Real | 318.0 |
| v_x | Real | 322.0 |
| v_y | Real | 326.0 |
| v_update | Real | 330.0 |
| ▼ Return | | |
| Point-To-Point Zone 3 | Void | |

```
0001 //Defines each corner in a square spiral shape around the calculates fire po-
     sition, X- and Y coordinate correspond to a point.
0002 //Defines x-points
0003 #X_array[1]:=-34.8;
0004 #X_array[2]:=-34.8;
0005 #X_array[3]:=-33.8;
0006 #X_array[4]:=-33.8;
0007 #X_array[5]:=-32.8;
0008 #X_array[6]:=-32.8;
0009 #X_array[7]:=-31.8;
0010 #X_array[8]:=-31.8;
0011 #X_array[9]:=-30.8;
0012 #X_array[10]:=-30.8;
0013 #X_array[11]:=-23.3;
0014 #X_array[12]:=-23.3;
0015 #X_array[13]:=-30.8;
0016 #X_array[14]:=-23.3;
0017 #X_array[15]:=-34.8;
0018
0019 //Defines y-points
0020 #Y_array[1]:=-35.3;
0021 #Y_array[2]:=-25.3;
0022 #Y_array[3]:=-25.3;
0023 #Y_array[4]:=-35.3;
0024 #Y_array[5]:=-35.3;
0025 #Y_array[6]:=-25.3;
0026 #Y_array[7]:=-25.3;
0027 #Y_array[8]:=-35.3;
0028 #Y_array[9]:=-25.3;
0029 #Y_array[10]:=-33.3;
0030 #Y_array[11]:=-33.3;
0031 #Y_array[12]:=-34.3;
0032 #Y_array[13]:=-34.3;
0033 #Y_array[14]:=-35.3;
0034 #Y_array[15]:=-35.3;
0035
0036
0037 //Path speed
0038 #v:=0.8;
0039 //Interupt time for OB35 in sec
0040 #dt:=0.010;
0041
0042 #x_out:=#X_array[1]+#x;
0043 #y_out:=#Y_array[1]+#y;
0044
0045 #s:=#s_uni;
0046 IF #"At path start"
```

```
0047 THEN
0048 //Calculate the vector orientation and length from a point to the next one
0049 FOR #i := 1 TO 15
0050  DO  #s_dx[#i]:= #X_array[(#i+1)]-#X_array[(#i)];
0051      #s_dy[#i]:= #Y_array[(#i+1)]-#Y_array[(#i)];
0052      #s_length[#i]:=SQRT(#s_dx[#i]**2+#s_dy[#i]**2);
0053 END_FOR;
0054
0055 //If on first strech, then go this direction and update the postion for each
     cycle
0056 //Do this for every strech..
0057   IF #s<#s_length[1]
0058 THEN
0059     #v_x:=(#s_dx[1]/#s_length[1])*#v;
0060     #v_y:=(#s_dy[1]/#s_length[1])*#v;
0061     #x:=#x+#v_x*#dt;
0062     #y:=#y+#v_y*#dt;
0063
0064 ELSIF  #s<(#s_length[1]+#s_length[2])
0065 THEN
0066     #v_x:=(#s_dx[2]/#s_length[2])*#v;
0067     #v_y:=(#s_dy[2]/#s_length[2])*#v;
0068     #x:=#x+#v_x*#dt;
0069     #y:=#y+#v_y*#dt;
0070
0071 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3])
0072 THEN
0073     #v_x:=(#s_dx[3]/#s_length[3])*#v;
0074     #v_y:=(#s_dy[3]/#s_length[3])*#v;
0075     #x:=#x+#v_x*#dt;
0076     #y:=#y+#v_y*#dt;
0077 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4])
0078 THEN
0079     #v_x:=(#s_dx[4]/#s_length[4])*#v;
0080     #v_y:=(#s_dy[4]/#s_length[4])*#v;
0081     #x:=#x+#v_x*#dt;
0082     #y:=#y+#v_y*#dt;
0083 ELSIF  #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5])
0084 THEN
0085     #v_x:=(#s_dx[5]/#s_length[5])*#v;
0086     #v_y:=(#s_dy[5]/#s_length[5])*#v;
0087     #x:=#x+#v_x*#dt;
0088     #y:=#y+#v_y*#dt;
0089 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6])
0090 THEN
0091     #v_x:=(#s_dx[6]/#s_length[6])*#v;
0092     #v_y:=(#s_dy[6]/#s_length[6])*#v;
0093     #x:=#x+#v_x*#dt;
0094     #y:=#y+#v_y*#dt;
0095 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7])
0096 THEN
0097     #v_x:=(#s_dx[7]/#s_length[7])*#v;
0098     #v_y:=(#s_dy[7]/#s_length[7])*#v;
0099     #x:=#x+#v_x*#dt;
0100     #y:=#y+#v_y*#dt;
```

```
0101 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8])
0102 THEN
0103     #v_x:=(#s_dx[8]/#s_length[8])*#v;
0104     #v_y:=(#s_dy[8]/#s_length[8])*#v;
0105     #x:=#x+#v_x*#dt;
0106     #y:=#y+#v_y*#dt;
0107 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9])
0108 THEN
0109     #v_x:=(#s_dx[9]/#s_length[9])*#v;
0110     #v_y:=(#s_dy[9]/#s_length[9])*#v;
0111     #x:=#x+#v_x*#dt;
0112     #y:=#y+#v_y*#dt;
0113 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10])
0114 THEN
0115     #v_x:=(#s_dx[10]/#s_length[10])*#v;
0116     #v_y:=(#s_dy[10]/#s_length[10])*#v;
0117     #x:=#x+#v_x*#dt;
0118     #y:=#y+#v_y*#dt;
0119 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11])
0120 THEN
0121     #v_x:=(#s_dx[11]/#s_length[11])*#v;
0122     #v_y:=(#s_dy[11]/#s_length[11])*#v;
0123     #x:=#x+#v_x*#dt;
0124     #y:=#y+#v_y*#dt;
0125 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12])
0126 THEN
0127     #v_x:=(#s_dx[12]/#s_length[12])*#v;
0128     #v_y:=(#s_dy[12]/#s_length[12])*#v;
0129     #x:=#x+#v_x*#dt;
0130     #y:=#y+#v_y*#dt;
0131 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13])
0132 THEN
0133     #v_x:=(#s_dx[13]/#s_length[13])*#v;
0134     #v_y:=(#s_dy[13]/#s_length[13])*#v;
0135     #x:=#x+#v_x*#dt;
0136     #y:=#y+#v_y*#dt;
0137 ELSIF
     #s<(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14])
0138 THEN
0139     #v_x:=(#s_dx[14]/#s_length[14])*#v;
0140     #v_y:=(#s_dy[14]/#s_length[14])*#v;
0141     #x:=#x+#v_x*#dt;
0142     #y:=#y+#v_y*#dt;
```

```
0143
0144 END_IF;
0145 //Reset when whole route is gone
0146   IF
     #s>(#s_length[1]+#s_length[2]+#s_length[3]+#s_length[4]+#s_length[5]+#s_length
     [6]+#s_length[7]+#s_length[8]+#s_length[9]+#s_length[10]+#s_length[11]+#s_leng
     th[12]+#s_length[13]+#s_length[14])
0147 THEN
0148   #s:=0;
0149   #x:=0;
0150   #y:=0;
0151 END_IF;
0152
0153 //Send out path position values
0154 //Update velocity and distance gone
0155 #x_out:=#X_array[1]+#x;
0156 #y_out:=#Y_array[1]+#y;
0157 #v_update:=SQRT(#v_x**2+#v_y**2);
0158 #s_uni:=#s+#v_update*#dt;
0159 END_IF;
```

# APPENDIX E

## Wind Measurements

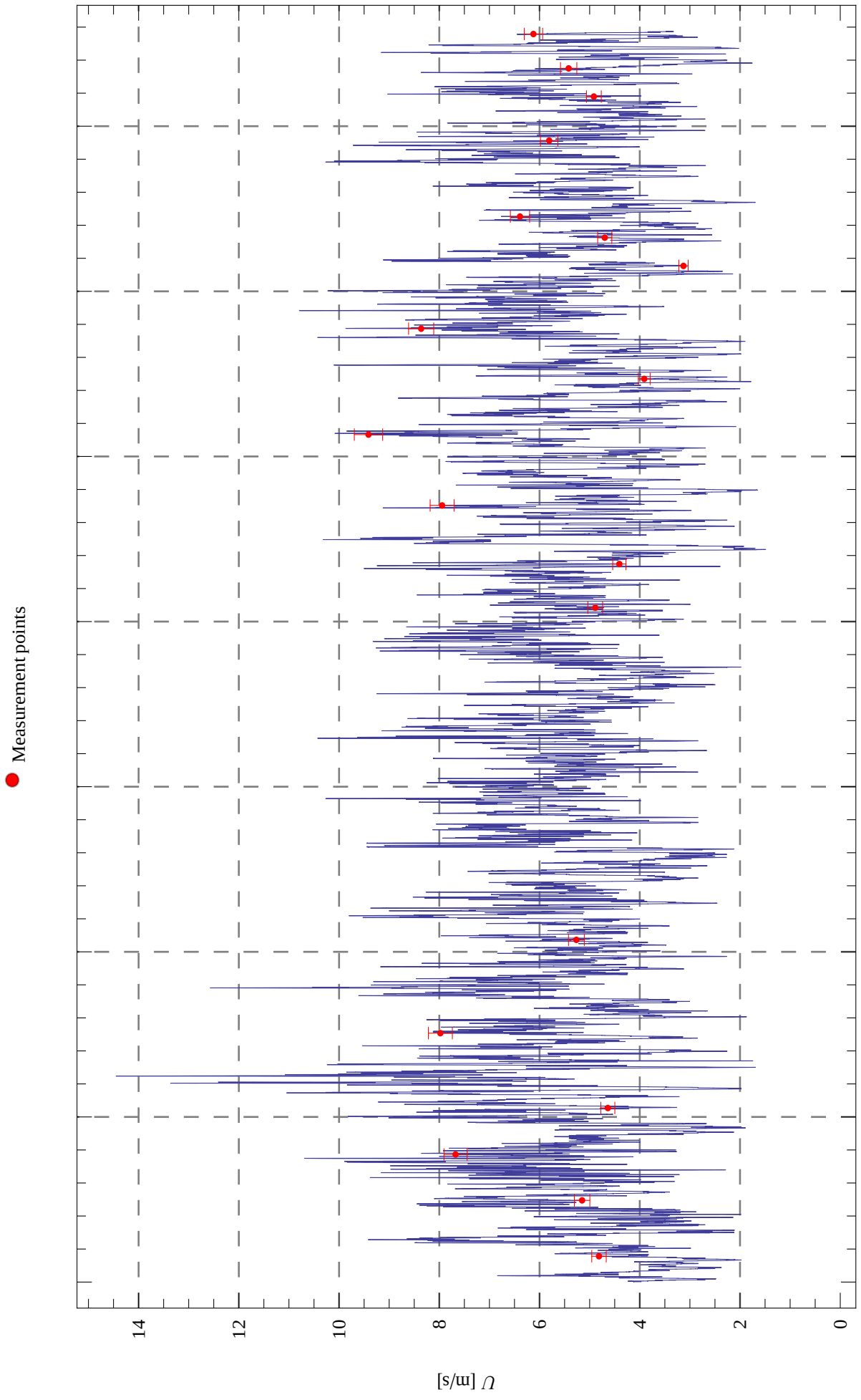



Figure E.1: Wind speed data and measurement points during experiment I.

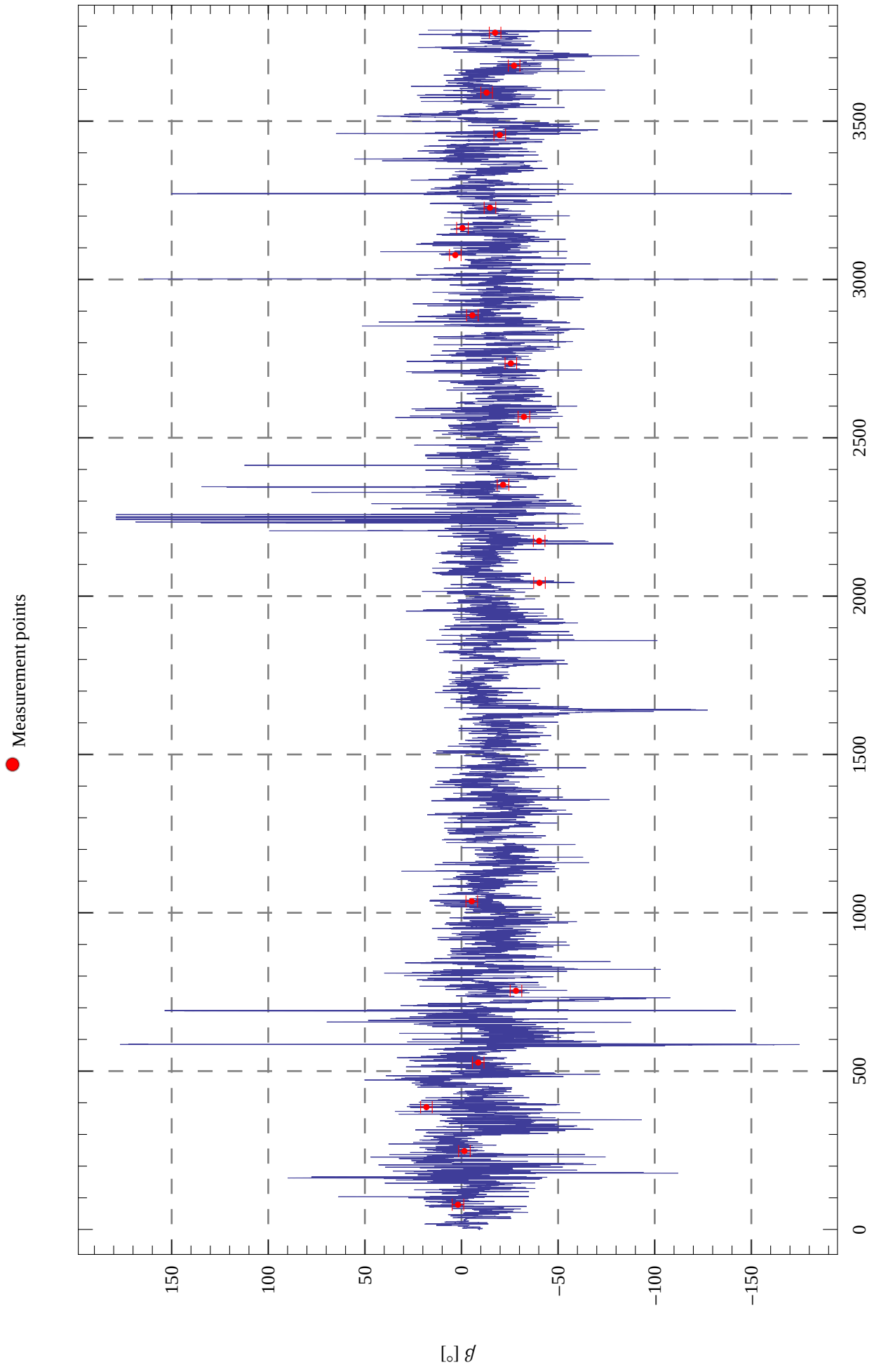



Figure E.2: Wind direction data and measurement points during experiment I.

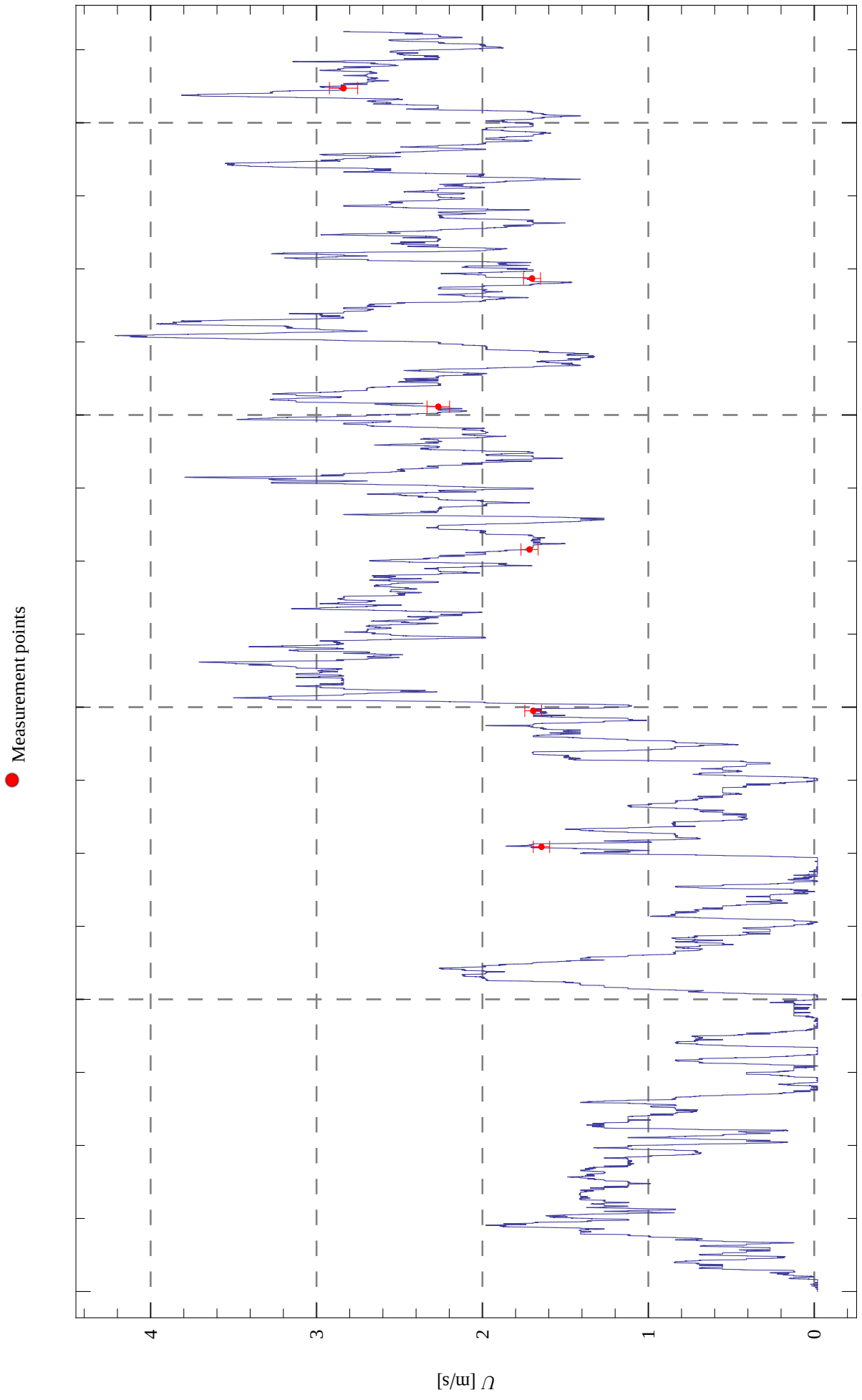



Figure E.3: Wind speeds during experiment II.

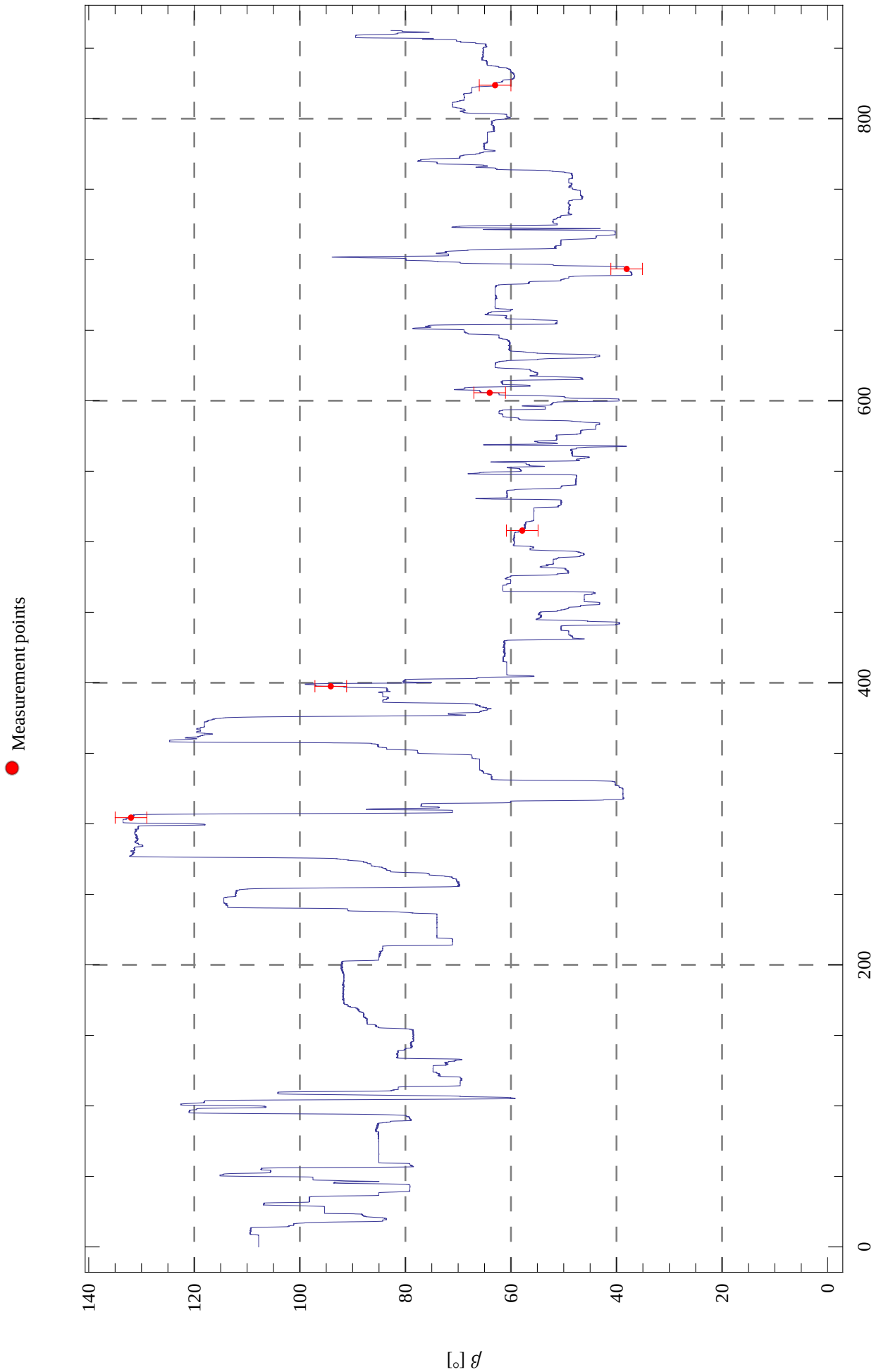



Figure E.4: Wind directions during experiment II.