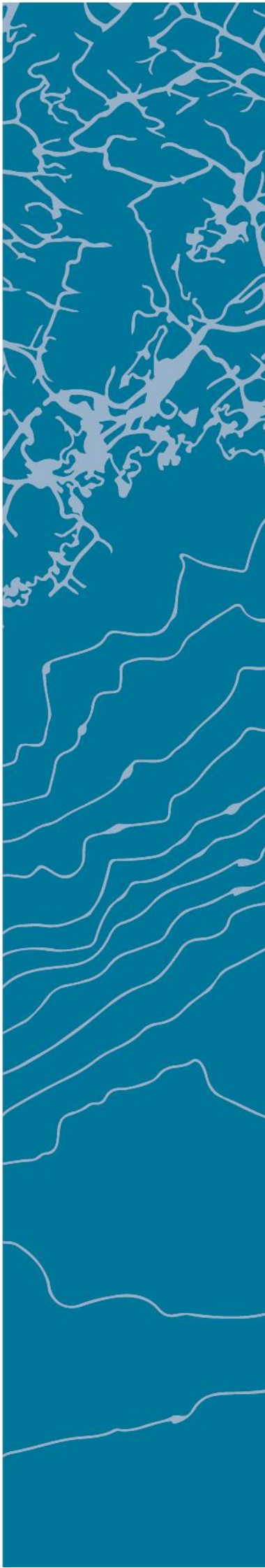
UNIVERSITETET I AGDER

# Design and Implementation of Self-Powered Long Range IoT Device Based on LoRaWAN

HARI BHUSAL

SUPERVISOR
Linga Reddy Cenkeramaddi
Ajit Jha

**University of Agder, 2020**
Faculty of Engineering and science
Department of Information and Communication Technology



# Preface

First of all, I would like to express my heartfelt gratitude to my supervisors, Associate Professor **Linga Reddy Cenkeramaddi** and Associate Professor **Ajit Jha,** for the continuous help in my research for their tolerance, inspiration, enthusiasm, and immense knowledge. Their guidance helped a lot throughout the research and writing of this thesis. I could not have imagined having better advisors. Also, I would like to thank my parents without their prayers and support, achieving this goal was impossible.

Grimstad
28.05.2020
Hari Bhusal

# Abstract

This project proposes a self-powered long-range wireless sensor node based on Long Range Wide Area Network (LoRaWAN) with various sensing capabilities. The nodes have been designed in such a way that these are self-sustainable throughout the components' lifespan. These are completely powered by ambient solar energy harvesting. Also, these nodes can be deployed on a large scale and are maintenance-free. In addition, these nodes can be deployed in remote places where the accessibility is limited, and maintenance is difficult.

The wireless sensor nodes can be deployed both in indoor and outdoor environments with sufficient light levels for the solar panel, such as indoor lights in the indoor environment and ambient sunlight in the outdoor environment. The fully functional wireless sensor node is designed and tested. The developed long-range node has a power consumption that is lower in comparison to the amount of power harvested in the ambient environment. The designed node senses the air quality parameters: the level of carbon dioxide, amount of humidity, pressure levels, temperature, and the total organic volatile compounds and gases. It is possible to add several other sensors without modifying the base hardware.

# Table of Contents

Preface ii
Abstract iii
Table of Contents iv
List of Figures vii
List of Tables viii
Abbreviation ix
Chapter 1: Introduction 1
1.1 Statement of Problem 1
1.2 Sources of Indoor Air Pollution 2
1.2.1 Human Emissions: 2
1.2.2 Equipment: 2
1.2.3 Cleaning Products: 2
1.2.4 Combustion: 2
1.2.5 Outdoor Sources for Air Pollution: 2
1.2.6 Humidity: 2
1.2.7 Temperature: 2
1.2.8 Carbon Dioxide ($CO_2$): 2
1.3 Project Goals 3
1.3.1 ATmega32u4 WSN 3
1.3.2 Energy Harvesting 3
1.3.3 Sensors 3
1.3.4 Android Application Development 3
1.3.5 Testing 3
1.4 Report Outline 3
Chapter 2: Overview of Technology and Processes 5
2.1 Power Harvesting 5
2.2 Solar Panels 5
2.3 Energy Storage Battery 6
2.4 Solar Li-Po Charger 6
2.4.1 Solar Charging: 7
2.4.2 USB & DC Charging: 7
2.4.3 Indicator LEDs: 7
2.5 ATmega32u4 (Arduino Micro) 8

2.5.1 Power ..... 9
2.5.2 Memory ..... 9
2.5.3 I/O Pins ..... 9
2.5.4 Communication ..... 10
2.5.5 Programming ..... 10
2.5.6 USB Overcurrent Protection ..... 10
2.6 Transceiver Module - RN2483 ..... 10
2.7 Low Power Gas, Pressure, Temperature & Humidity Sensor ..... 12
2.7.1 Sensor Modes: ..... 12
2.7.2 Indoor-air-quality: ..... 13
2.8 Total Volatile Organic Compounds (TVOCs) Sensor ..... 13
2.9 LoRa Gateway ..... 14
2.10 Communication Protocol ..... 15
2.10.1 LoRaWAN: ..... 15
2.10.2 Universal Asynchronous Receiver-Transmitter: ..... 17
2.10.3 Inter-Integrated Circuit: ..... 18
2.10.4 MQTT: ..... 18
2.11 Software and Tools ..... 19
2.11.1 Software: ..... 19
2.11.2 Tools: ..... 19
Chapter 3: System Designing ..... 20
3.1 Brief Description of System ..... 20
3.2 Energy Harvesting ..... 21
3.3 Real Time Clock(DS3231) ..... 24
3.4 Sensors ..... 24
3.4.1 BME680 ..... 24
3.4.2 CCS811 ..... 25
3.5 Communication ..... 25
3.6 The Things Node Using ATmega32u4 and RN2483 Transceiver ..... 27
3.7 LoRa Setup ..... 28
3.8 Modulation and Data Rate ..... 29
3.9 LoRa Antenna ..... 31
3.10 Expected energy consumption ..... 31
3.11 Data extraction, Manipulation, and Visualization ..... 32
3.11.1 Gateway ..... 32
3.11.2 The Things Network (TTN) ..... 32
3.11.3 The Things Network to Android Application ..... 33

3.11.4 Python Integration....34
Chapter 4: System Testing....36
4.1 Energy Harvesting Test....36
4.1.1 Battery and Solar Panel Inside Room Light....37
4.1.2 Battery and Solar Panel in Full Sunlight....37
4.1.3 Power consumption for designed wireless IoT LoRa WAN node....38
4.2 Range....39
4.3 Air Quality Monitoring of Wisenet Lab....41
4.4 Visualization of Sensor Data Through Android Application....45
Chapter 5: Conclusion....47
References....48
Appendix....51

# List of Figures:

Figure 2.1: Solar Panel....5
Figure 2.2: A 2000mAh Li-Po battery....6
Figure 2..3: Solar Li-Po Charger....6
Figure 2.4: Led indicator on the solar charger....7
Figure 2.5: Different led indication on solar charger [14]....8
Figure 2.6: ATmega32u4 (Arduino Micro)....8
Figure 2.7: Transceiver module - RN2483....11
Figure 2.8: Pinouts of RN2483 [19]....11
Figure 2.9: BME680 sensor....13
Figure 2.10: CCS811 Gas sensor....14
Figure 2.11 Gateway Board....14
Figure 2.12 LoRaWAN Class-A Data Transmission....15
Figure 2.13 LoRaWAN Architecture [28]....16
Figure 3.14 Block Diagram of Overall System....20
Figure3.15: Block diagram of energy harvesting system. ....21
Figure3.16: Current-Voltage and Power-Voltage curve of the solar panel [14]....22
Figure 3.17: Solar cell I-V curve [14]....22
Figure3.18: Behavior of solar charger as light in solar panel varies [14]. ....23
Figure3.19: System load sharing diagram [14]....23
Figure 3.20: I2C and UART connection....26
Figure 3.21: Connection between ATmega32u4(Arduino) and RN2483....27
Figure 3.22: The Things Network(TTN) Console....29
Figure 3.23 Uplink Transmission configuration....30
Figure 3.24 Gateway overview on The Things Network....32
Figure 3.25: Device setting in The Things Network(TTN) ....33
Figure3.26: Data logging in .csv file....35
Figure 4.27: charging current/voltage Figure 4.28: Inside test location ....37
Figure 4.29: Charging power vs. voltage in indoor condition....37
Figure 4.30: Outdoor location....38
Figure 4.31: Charging power vs. voltage in outdoor condition....38
Figure 4.32: ATmega32u4 sleep current with BME680, CCS811, and RN2483....39
Figure 4.33: Serial monitor of sensor node while transmitting data and interrupt firing every 1 minute....39
Figure 4.34: Sensor node placed on the longest distance from the sensor....40
Figure 4.35: Gateway location....40
Figure 4.36: Range covered in google maps....41
Figure 4.37: Monitoring location....41
Figure 4.38: Line chart with normal sample data collected from sensors....42
Figure 4.39: Line chart of $CO_2$ concentration....42
Figure 4.40: Line chart of the total concentration of gas....43
Figure 4.41: Line chart of humidity concentration....43
Figure 4.42: Line chart showing atmospheric pressure....44
Figure 4.43: Line chart showing temperature....44
Figure 4.44: Line chart of concentration of total volatile organic compound....45
Figure 4.45: Android application screenshot....46

# List of Tables:

Table 2.1: RN2483 specification [19].....11
Table 2.2: Mode differences [23].....12
Table 2.3 : LoRaWAN specification for Europe [30].....17
Table 4: Important features of BME680 [23] .....25
Table 5: The pin connections of Arduino micro and RN2483.....27
Table 6: EU863-870 Data Rate [45] .....29
Table 7: EU863-870 TXPower [45].....30
Table 8: Expected worst case energy consumption of all components .....31
Table 9: Expected worst case sleep energy consumption of all components .....31
Table 10: Units of measurement of data.....36

# Abbreviation

| | |
|---|---|
| ABP | Activation By Personalization |
| ACK | Acknowledgement |
| ADC | Analog to Digital Converter |
| ADDR | Address |
| AES | Advanced Encryption Standard |
| ALM | Alarm |
| AM | Ante meridiem |
| APPEUI | Application EUI |
| AppKey | Application Key |
| AppSKey | Application Session Key |
| API | application program interface |
| AQI | Air Quality Index |
| AVR | Advanced Virtual RISC |
| AWS | Amazon Web Service |
| BLE | Bluetooth Low Energy |
| BW | Bandwidth |
| $CO_2$ | carbon dioxide |
| COM | Communication Port |
| CPU | Central Processing Unit |
| DC | Direct Current |
| DevEUI | Device EUI |
| DevAddr | Device Address |
| $eCO_2$ | Equivalent Calculated Carbon-dioxide |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| EH | Energy Harvesting |
| ETSI | European Telecommunications Standards Institute |
| EUI | Extended Unique Identifier-A globally unique ID |
| FSK | Frequency Shift Keying modulation technique |
| GFSK | Gaussian Frequency Shift Keying |
| GND | Ground |
| GPIO | General Purpose Input Output |
| GPS | Global Positioning System |
| HPA | High Pressure Air |
| HTTP | Hypertext Transfer Protocol |
| IAQ | Indoor Air Quality |
| ICSP | In Circuit Serial Programming |
| ICT | information communications technology |
| ID | Identification |
| IEEE | Institute of Electrical and Electronics Engineers |
| IOT | Internet of Things |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| KB | Kilobyte |
| LoRa | Long Range modulation technique |
| Lora WAN | Long Range Wide Area Network |
| LPWAN | Low-Power Wide-Area Network |

| | |
|---|---|
| MAC | Medium Access Control |
| MCP | Master Control Panel |
| MCU | Microcontroller Unit |
| MHZ | Megahertz |
| MIC | Message Integrity Code |
| MISO | Master in Slave Out |
| MOSI | Master out Slave in |
| MOX | Mixed Oxide |
| MPPT | Maximum power point tracking |
| MQTT | Message Queuing Telemetry Transport) |
| NwkSKey | Network Session Key |
| OS | Operating System |
| OTAA | On The Air Activation |
| PC | Personal Computer |
| PCB | Printed Circuit Board |
| PM | Post Meridiem |
| PV | Photovoltaic |
| PWM | Pulse Width Modulation |
| PWR | Power |
| RF | Radio Frequency |
| RH | Relative Humidity |
| RST | Reset |
| RTC | Real Time Clock |
| RX | Receiver |
| SCK | Serial Clock |
| SCL | Serial Clock Technology |
| SD | Secure Digital |
| SDI | Serial Digital Interface |
| SF | Spreading Factor |
| SPI | Serial Peripheral Interface |
| SQW | Square Wave |
| SRAM | Static Random Access Memory |
| TCP | Transmission Control Protocol |
| TTN | The Things Network |
| TVOC | Total Volatile Organic Compound |
| TX | Transmitter |
| USB | Universal Serial Bus |
| UV | Ultra-Violet |
| VOC | Volatile Organic Compound |
| WAN | wide area network |
| WSN | Wireless Sensor Network |
| XML | Extensible Markup Language |

# Chapter 1 : Introduction

This project is assigned by Associate professor Linga Reddy Cenkeramaddi and Associate professor Ajit Jha as part of the Master's degree at the University of Agder and specialization in ICT / Embedded Systems and designed/developed by Hari Bhusal.
The project's primary objective is to design and test a self-powered long-range wireless IoT device based on LoRa WAN protocol to sense the quality of indoor air (extended to outdoor also) by mitigating lower power techniques. The designed node is self-powered by solar energy harvesting from both indoor and outdoor environments.

## 1.1 Statement of Problem

Fresh and unpolluted air is one of the most important factors for people's health. People exposed to polluted environments, both indoor and outdoor, which results in health deterioration to a large extent by the condition of the polluted environment. This master thesis deals with the design, development, and testing of a real-time low-powered air quality monitoring wireless IoT device for indoors and outdoors based on LoRaWAN. In indoor environments, the administrators in charge of the building can monitor and adjust and/or control the quality of air. With numerous achievements being made in technology, including rapid advancements in the efficiency of sensors and other embedded devices, communication systems have evolved to a great extent. By integrating all these advancements in the technology together can be utilized to monitor and control the indoor environments remotely in an efficient manner.
Research indicates that people's health and other living creatures are directly impacted by the quality of air. People living in a cold climate spend more than 90% of their time indoors [1]. Thus the need to improve the quality of indoor air is imminent by enforcing ways of using appropriate measures. The productivity of indoor workers and their health is also significantly affected by the quality of air. The impact of ambient air pollution representatives includes temperature, humidity, and $CO_2$ levels, and the places which are more prone to such environments include nurseries/playschools, workplaces, schools, and homes [2]. Therefore, an improvement in the quality of air directly correlates to an increase in the health of the population. Some of the major consequences of the poor quality of air indoors are:

- Issues when concentrating
- Headache
- Nausea
- Respiratory problems (dyspnea)
- Nasal irritation
- Throat dryness

Self-powered long-range IoT device with indoor air quality (IAQ) sensors enables real-time analysis of the six main parameters of measuring the level of air quality in the indoor environment:

- Temperature
- Level of Carbon Dioxide
- Total volatile organic compound

- Gas
- Humidity
- Pressure

## 1.2 Sources of Indoor Air Pollution

Various sources, including materials of the buildings, pollute the indoor air. Apart from these materials used during the process of construction, even the ones used for the interior of the buildings can also be the reason for air pollution. One such example is Formaldehyde, a volatile organic compound that is used in the process of renovating. It is harmful to human health and is used for thermal insulation in new buildings [3].

### 1.2.1 Human Emissions:

Breathing pollutes the air as it produces $CO_2$. It is only one of the many human emissions that causes pollution. Other factors include intestinal gases, perfumes, dead skin cells, and the bacteria emitted during sneezing or coughing [3].

### 1.2.2 Equipment:

Board markers, papers, computers, photocopiers, printers (including 3D printers), and other electronic equipment produce many volatile components like degreasing agents and release agents that cause indoor pollution [4].

### 1.2.3 Cleaning Products:

The products used for cleaning contain disinfectants, solvents, preservatives, organic acids, and other acidic agents. These can be a major cause of air pollution [3].

### 1.2.4 Combustion:

Kitchens, tobacco, open fires, anti-mosquitos, etc.

### 1.2.5 Outdoor Sources for Air Pollution:

Some of the outdoor sources for air pollution include exhaust gases of vehicles, smoke produced by factories, pollination from plants, or gases in the mud. The $CO_2$ levels, temperature, pressure conditions, relative humid conditions, and total volatile organic compound in the air, If these parameters have a process of being estimated regularly and automatically, it will help a lot in the limitation of their harmful effects on the health and efficiency of the general population. On average, 2 million people have ailments due to the below-par quality of air, according to a report by the Actions for Indoor Healthy Air [5].

### 1.2.6 Humidity:

Dry leads to a lot of problems such as dry skin, mucous, irritation, and chapped skin. Another cause of discomfort is static electricity that can be produced when the air is dry. However, humid air leads to the growth of germs in the building and spread of fungi, which is the result of increased condensation. This results in the spread of infectious diseases.

### 1.2.7 Temperature:

Humidity in the air influences the temperature being experienced indoors [3]. This results in the temperature and humid conditions being felt to be different in the different seasons.

### 1.2.8 Carbon Dioxide ($CO_2$):

Carbon Dioxide concentration in the outside air is normally varying between 250 to 400ppm (parts per million); however, for air that is indoors, this value of $CO_2$ levels increases. According to an estimation, the indoor air quality is considered correct if the concentration of $CO_2$ is up to 1000ppm. Its prolonged exposure has an impact on human well-being and health.

$CO_2$ is one of the measures for the quality of the dwellings. Therefore, if the carbon dioxide level is below 1000ppm, the quality of the air inside is considered to be good. The negative effects are experienced in health if its concentration is above 1000ppm [6]. It is not only just carbon dioxide but the presence of the before mentioned volatile compounds in the air such as formaldehyde, solvents, pollen, and bacteria.

## 1.3 Project Goals

The main goals related to this project are given below.

### 1.3.1 ATmega32u4 WSN

This part includes the development of a WSN based on ATmega32u4 and the RN2483 transceiver for the wireless communication in the 868MHz band. The RN2483 transceiver provides LoRaWAN protocol connectivity by using a simple UART interface with ATmega32u4. The Evaluation Board is used during prototype development.

### 1.3.2 Energy Harvesting

This part focuses on the development of an energy harvesting (EH) circuit for the designed wireless IoT nodes. Among all possible energy harvesting resources, solar seems to be the most practical one, and solar energy is the most feasible source of energy for both indoor and outdoor use. A rechargeable battery is integrated along with a battery charging circuit and solar panel.

### 1.3.3 Sensors

The most suitable sensors for the wireless IoT nodes must be Ultra-Low Power (ULP) and require as little on-time as possible to further reduce the energy consumption. The environmental values of interest are:

- Temperature
- Relative Humidity (RH)
- $CO_2$
- Atmospheric Pressure
- Indoor Air Quality (IAQ)
- Total Volatile Organic Compound

### 1.3.4 Android Application Development

For the transfer and visualization of real-time data from the wireless sensor node, MQTT protocol based server, and Android application, Publish-Subscribe Architecture is used.

### 1.3.5 Testing

The designed wireless IoT nodes are tested extensively to determine the indoor and outdoor energy harvesting, power consumption, and range. A line of sight range test is conducted in several scenarios. For air quality monitoring, nodes are deployed in the Lab located at WISENET center, UiA.

## 1.4 Report Outline

- Chapter 1 – Introduction
  Chapter 1 includes the problem statement and lays the background for the project. The goals are stated and explained with brief descriptions.

- Chapter 2 – Overview of Technology and Process
  Chapter 2 describes the concepts used during the project. The theory on the photovoltaic harvesting, energy storage, wireless IoT node, and communication protocols and technologies are described.

- Chapter 3 - System Design
  Chapter 3 presents the detailed and technical description of the hardware components, design of circuits, as well as the firmware for the nodes and gateways.

- Chapter 4 – Testing
  Chapter 4 covers the overall testing at the system level.

- Chapter 5 – Conclusion
  Chapter 5 concludes the design, development, and testing process and the performance evaluation of the wireless IoT nodes designed based on LoRa WAN.

# Chapter 2 : Overview of Technology and Processes

## 2.1 Power Harvesting

Main energy harvesting sources include wind, solar, RF and thermal energy, etc. Of all these, solar is the most practical one and prevalent in both indoor and outdoor environments. The devices known as energy or power harvesters convert the natural/physical energy into the electrical energy (for example, solar cells convert ambient light energy into electrical energy). The typical harvesting systems comprise a source of energy, a harvester, a storage element, and a power management system [7].
Photovoltaic cells (also popularly known as solar cells) have been very promising in their performance for giving a higher output of energy than the other sources, despite it is being weather dependent source. The entire harvesting architecture includes an integrated power management system containing regulators for stepping up or stepping down the voltage, maximum power point tracking (MPPT), battery storage, supercapacitors [8].

## 2.2 Solar Panels

We have selected 6V, 3.5 W rated solar panels as an energy harvesting module for these nodes. It is not only resistant to scratches and waterproof, but it can resist UV rays too. Solar panels have a high-efficiency level, being a monocrystalline cell. They can give a 6V output at 530 mA. It is designed to be robust and lightweight owing to its substrate is a composite of aluminum and plastic [9]. They are made to be used easily outdoors, being able to withstand extreme weather conditions, and can even be leaned and can survive a drop. A Solar panel used in the development of the prototype is shown in Figure 2.1.

**Figure 2.1: Solar Panel**

The solar panel consists of several cells made using photovoltaic elements, which convert light into electrical energy [9].
Solar cells are of two types: thin-film and crystalline. Photovoltaic modules are the commercial product which is formed by an interconnected network of photovoltaic cells. These vital parts

of the power system are encapsulated to form the mounting structure for the array or the module, and several such modules, when connected, can produce the amount of current and power that is desired [10].

## 2.3 Energy Storage Battery

As shown in Figure 2.2, a lithium-ion polymer (also known as 'Li-Po / Li-Poly') battery is used as the storage element. The capacity of this battery is 2000mAh. The protection mechanism of the circuit is efficient in the battery. It does not overcharge, or the voltage does not go too high, and it does not get overused, or the voltage does not go too low [11].


**Figure 2.2: A 2000mAh Li-Po battery**

## 2.4 Solar Li-Po Charger

The integration of a solar panel, charger, and wireless IoT node is shown in Figure 2.3. A solar charger module controls the amount of the current that is used to charge the battery. The safety feature protects from overcharging and provides a good quality of the charged device while preventing additional charge and discharging in conditions where there is low light. In most of the devices, to prevent the additional discharge of current, blocking diodes are already integrated into the solar panels [12].

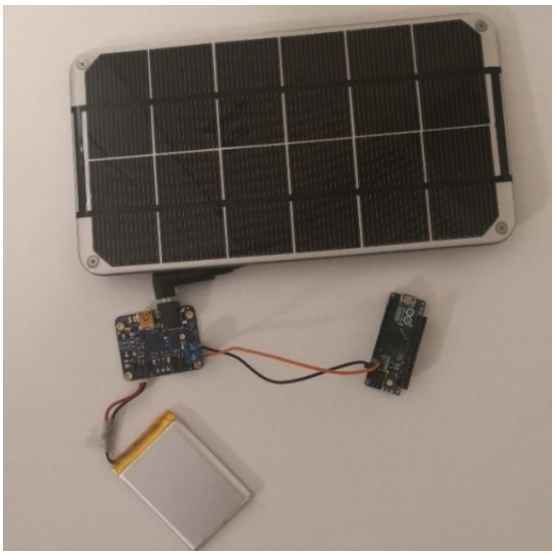


Figure 23: Solar Li-Po Charger

As shown in Figure 2.4, There are three color indicator LEDs that can be adjusted for various levels of the flow of current. They are Power good, Charging, and Done. Low Battery Indicator (fixed at 3.1V) with LED output on (labeled CHRG) adjusted at 500mA level of charging. The maximum possible voltage should be drawn through a battery until the maximum allowed charge rate [13].

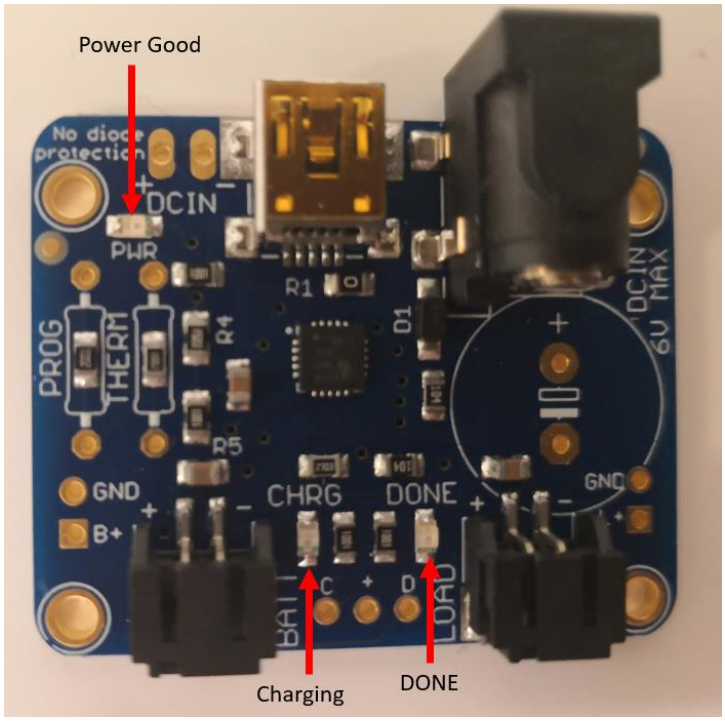



**Figure 2.4: Led indicator on the solar charger**

## 2.4.1 Solar Charging:

The solar charging is easy for our solar charger board. The solar panel can simply be plugged into the DC jack. The battery is then plugged into the BATT slot. The battery which is used is a 3.7V Lithium-Ion/Polymer battery. The PWR GOOD LED indicates that the solar panel is providing power.

CHRG charging light glow indicated that the battery is being charged. Great care is taken in assuring that the panel is facing direct sunlight, not shaded, and not behind any glass or plastic. When the battery is fully charged, the green DONE LED lits up [14].

## 2.4.2 USB & DC Charging:

To facilitate charging at night or in the absence of the sun, there is a USB port on the board as well. Any mini-B cable can be used to plug in and charge. If something is connected to the DC jack, it will mechanically disconnect the mini USB connector. Therefore, it is important to unplug the solar panel when USB charging

The DC wall plug adapter can directly be plugged into the jack. For easiness, the inner diameter connector is 5.5mm/2.1mm, which is very common.

If the DC power is needed for something purposes, there is a facility available to connect DC input via the 0.1" breakout. As the power can be fed into the breakout pins as well, it is not polarity protected. Therefore, a Schottky Diode is used for safety purposes [14].

## 2.4.3 Indicator LEDs:

The status of the charger is represented through three types of LEDs. Good power is shown through the red PWR LED. The charging status is indicated by the orange CHRG LED. The charger charges the battery completely when switched on. When there is no power source,

acting also in the manner of an indication of low battery (fixed at 3.1V).[13] So, upon the voltage of the battery goes down under 3.1V, the LED colored orange is turned on, given that no other solar panel is wired up. After completion of the charging process, the green DONE LED gets lit up. “STATUS 0.1" break-out (at the lower side of the PCB) is another thing that can be used to attach microcontrollers and larger LEDs to the system. The system is an open connection and leads to being shorted to the earth when 'active' and float when 'inactive.' A pull-up resistor is used for the digital signals/LEDs, as shown in Figure 2.5 [14].

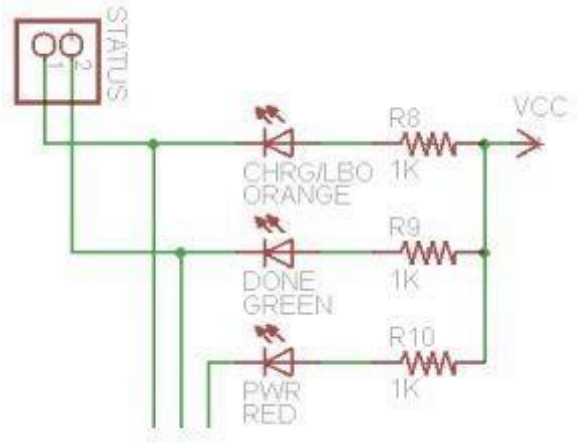



**Figure 2.5: Different led indication on solar charger [14]**

## 2.5 ATmega32u4 (Arduino Micro)

The Arduino Micro based on ATmega32u4 is a microcontroller device. This board contains 20 digital input/output pins (7 digital and 12 as analog inputs), a micro USB connection, a 16 MHz crystal oscillator, indication LEDs, an ICSP header, and a reset button[14] The Micro board resemblance to the Arduino Leonardo such a way as the ATmega32u4 has built-in USB communication. Hence, the requirement of an additional processor is eliminated. The Micro thus looks like a PC in the end, having a keyboard and mouse. It also contains a virtual serial / COM port [15].

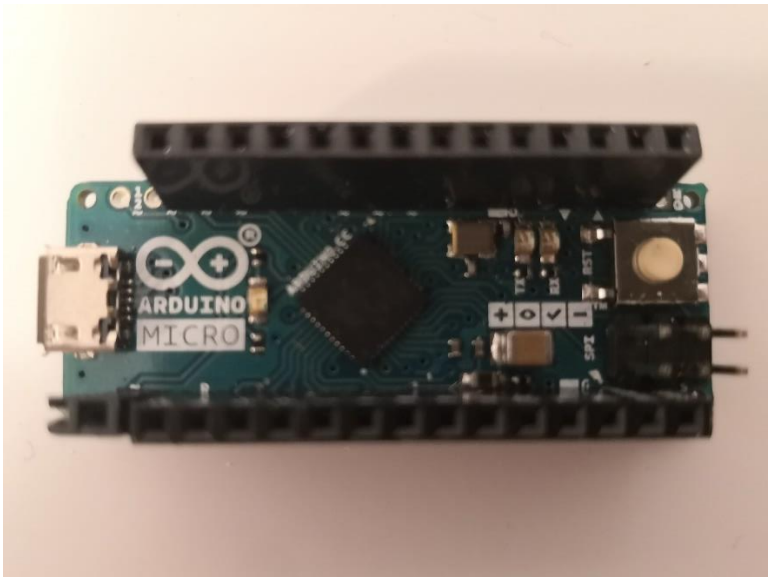



**Figure 2.6: ATmega32u4 (Arduino Micro)**

### 2.5.1 Power

The Arduino Micro has two options for input power, i.e., an external source of power or a micro USB connection. The Source of power gets automatically put on. An external or non-USB source of power has the option of being supplied from a direct DC supply or from a cell, which has the ability to both be connected to the GND and VIN pins. The supply of 6-20V externally can be used to operate the board. Recommendations are to maintain values between 7 - 12 volts [16]. Power pins are as following:

- Ground pins.
- **VI**: It can be used for providing the current via an exterior source to the board.
- 3V: On-board, regulator supplies the generation of a 3.3V 50mA.
- 5V: The mini controller, along with additional parts are being powered up on the board using a supply of power that is monitored, which could be the result of using a regulator on the board and VIN/USB of 5V.

### 2.5.2 Memory

With a total memory of 32KB, the board has a boot-loader used by 4 KB. The full name electrically erasable programmable read only memmory (EEPROM) and full name (SRAM) are1kB and 2.5 kB, respectively [15].

### 2.5.3 I/O Pins

By applying digitalWrite(), digitalRead() and pinMode() applications, pins on the micro board are utilized in the form of either output or input pins, operating around 5V.[18] The pins have a capacity to give or take up-to 40 mA current, with the option of a resistor in the internal of 20-50 kOhms. Besides, the following pins provide some additional utility:

- **TWI 2 (SDA) and 3 (SCL):** Through the Wire repository, it can support TWI.
- **Serial 0 (RX) and 1 (TX):** These pins transmit (TX) and receive (RX) serial data by using a micro board's serial capability. On the Micro board, USB communication is Serial communication.
- **PWM 3, 5, 6, 9, 10, 11 and 13:** These pins use analogWrite() function to support an 8-bit PWM result.
- **External Interrupts 0(RX), 1(TX), 2, and 3:** The pins provide an interruption on a small result and a difference in value.
- **RX_LED/SS:** It is an extra pin concerning Leonardo, with a connection to the RX_LED, displaying the performance of data reception in the process of communicating via USB. In SPI communication, it acts as an SS or a slave select.
- **SPI on the ICSP Header:** It supports the SPI protocol taking use of the SPI repository. They do not have any connection to the digital input/output ones because of a connection to the Arduino Uno. So, it is found only through ICSP connections and through pins, which have been marked MOSI, SCK, and MISO.
- **Analog Inputs:** A0-A5 and A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12). There are a total of 12 analog inputs. A0 to A5 have been marked on top of the pins as given on board, while the remaining pins can be accessed through coding using the constants from A6. Each can also be found as digital I/O and has a provision for a resolution of 10 bits or 1024 different values.
- **LED 13:** Digital pin 13 is connected via LED. At the value HIGH, the LED is marked as ON, and upon the pin being LOW, it is marked off.

### 2.5.4 Communication

The ATmega32u4 board can establish communication with another microcontroller, or with computers. It can also be used alongside the ATmega32U4 pins 1 (TX), and 0 (RX) provides UART communicational methods, additionally offered on digital too. It is visible like a virtual port to the software. It works like in a manner of a USB 2.0 device at high speed, with the usage of USB COM drivers. A Windows computer has a requirement of a .inf file. It has TX and RX LEDs that blink upon the transmission of data between the USB connection and the computer. Serial communications are made possible via a SoftwareSerial library through the digital pins collection of Micro. ATmega32U4 also supports SPI and I2C (TWI) communication, following which there is a simplification in the process owing to the Arduino software in the usage process of I2C bus protocol with the Wire repository. Using the SPI library is recommended for SPI communication, and input devices can be programmed by the Micro to control the Keyboard and Mouse classes [17].

### 2.5.5 Programming

Arduino software assists in burning the program onto the Atmega32u4 board. No external hardware programmer is required since ATmega32U4 on the Arduino Micro is equipped beforehand with a pre-burned bootloader, which can communicate with the help of the AVR109 protocol. ICSP (In-Circuit Serial Programming) header can also be used for Microcontroller programming [18].

### 2.5.6 USB Overcurrent Protection

To protect USB ports from shorts and overcurrent of the computer system, the Micro board has a resettable polyfuse. Even though many of the computers have an internal protection system, this polyfuse supports additional protection. The fuse will be burnt out when high current, i.e., higher to 500 mA, has an application to the USB port [15].

## 2.6 Transceiver Module - RN2483

RN2483 transceiver module, as shown in Figure 2.7, is based on Low-Power Long Range LoRa Technology, which is a low-power device that is used for wireless data transmission over longer ranges. This device's specifications comply with the Class A protocol of LoRaWAN. It is a complete long-range solution as it integrates RF, command Application Programming Interface (API) processor, a baseband controller. The RN2483 transceiver module, along with external host MCU is perfect for simple applications involving the use of long-range sensors [19].

Moreover, this transceiver module has LoRa Technology RF modulation. Because of RF modulation, it can provide a broad spectrum suitable for long-range communication without the interference of external factors. Additionally, it can provide for a greater receiver sensitivity of -146 dBm, which, along with the integrated +14 dBm power amplifier, provides an economical budget. This makes it usable for long-range requirements and robust nature of tasks [20].


**Figure 2.7: Transceiver module - RN2483**

LoRa Technology modulation also provides various advantages as compared to the conventional modulation techniques in both blocking and selectivity, and therefore it provides a long-range, no interference of external factors, along with lower consumption of power. Apart from that, this device is able to deliver an advanced phase noise, linearity, and the selectivity of the receiver and IIP3 for a noticeable fall in the consumption of power [19].

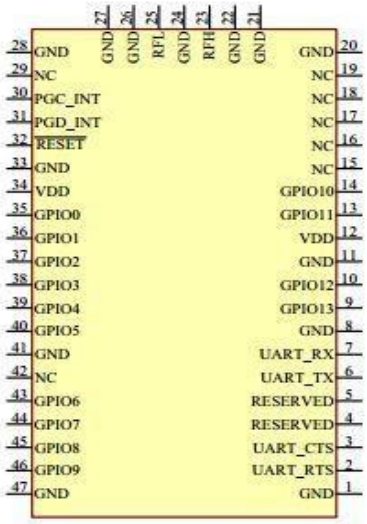



**Figure 2.8: Pinouts of RN2483 [19]**

**GPIO Pins:** The module has 14 GPIO pins, as shown in figure 2.8, which are indicated as GPIO0 to GPIO13. These I/O pins have the feature of being connected to either an LED, a switch, and relay outputs for various functions. The pins are used as either logic outputs or inputs. Most have the ability of an accessible analog input using the device firmware. Though their sink is limited, they can provide a few functions with small values.

**Power Pins:** Power pins (Pin 12 and 34) must be connected with a stable supply voltage with sufficient current. Additional filtering capacitors are not required for normal operation, but for noisy environments, these should be used to ensure stable supply voltage.

**Table 2.1: RN2483 specification [19]**

| **Specifications** | **Description** |
|---|---|
| Frequency Band | 863.00 MHz to 870.00 MHz<br>433.05 MHz to 434.79 MHz |
| Modulation Method | GFSK, FSK, and LoRa Technology modulation |

| | |
|---|---|
| RF Connection | Board edge connection |
| Interface | UART |
| Operation Range | Up to 15 km at suburban; coverage up to 5 km at urban area |
| RF TX Power | Maximum 10 dBm (Adjustable) on 433 MHz band and maximum 14 dBm on 868 MHz band |
| Temperature (operating) | -40°C to +85°C |
| Temperature (storage) | -40°C to +115°C |

## 2.7 Low Power Gas, Pressure, Temperature & Humidity Sensor

Being a digital sensor, the BME680 has the capability to measure Gas, temperature, pressure, and humidity. Additionally, the sensor is a compact device and housed in a metal-lid LGA package. Because the device has a small size and requires a very low level of consumption of electricity, it can be integrated with devices that are run by batteries and need to be coupled with a frequency [21]. Some of the devices include:

- Home automation and control
- Quality of the air inside
- IoT
- Advancement of GPS technology
- Weather estimations
- Outdoor navigation, application in sports and leisure, etc.
- Indoor navigation such as changing and floor detection, elevator detection, etc.

### 2.7.1 Sensor Modes:

There are two modes of power that the sensor can detect, namely, sleep and forced mode. The mode control register is used to select these modes. On being switched on, it gets started by default on the sleep mode. If the device is currently taking the measurements, the processing of switching between commands on the mode is postponed until the running measurement period completed. Also, other modes that bring any change to the commands to the registers will not be paid any attention until the mode executes the command to change. So, all control registers should be set to values that are needed prior to the command being written [22]. The major differences between the modes are given in table 2.

**Table 2.2: Mode differences [23]**

| Operation mode | mode<1:0> | Key features |
|---|---|---|
| Sleep | 00 | • No measurements are performed<br>• Minimal power consumption |
| Forced mode | 01 | • Single TPHG cycle is performed<br>• Sensor automatically returns to sleep mode afterwards<br>• Gas sensor heater only operates during gas measurement |

In forced mode, measurements are taken in a sequence, i.e., temperature, pressure, humidity, and gas conversion. This mode of sequencing is known as TPHG (Temperature, Pressure, Humidity, and Gas). To store set points in the sensors, up to 10 temperature set-points and heating durations can be done so for the gas sensor [23].

### 2.7.2 Indoor-air-quality:

BME680, as shown in Figure 2.9, is used to detect IAQ. It is a metal oxide-based sensor that can adsorb on its sensitive layer. Therefore, this sensor is receptive to some of the compounds which are very volatile such as carbon dioxide, causing air pollution inside homes. Instead of measuring for one specific component, this sensor can measure the sum of VOCs/contaminants in the air. Therefore, of the process, it can provide for the detection of exhaust from furniture, paint, and from litter. Also, from greater levels of VOC resulting from cooking and eating food, breath that is exhaled out and/or sweating, etc. The signal being rough, this sensor will provide the value of resistance, which changes as VOC concentrations change (lower is the concentration of reduction of VOCs, the higher the resistance). The rough signal also depends upon other factors (e.g., humidity level), this signal is transformed into an indoor air quality (IAQ) index using some algorithms, with values from 0 (clean air) to 500 (heavily polluted air) [24]. This algorithm can automatically adapt itself to calibrate to the surroundings where the sensor is kept for operations. The calibration process does the calculations depending upon the recent measurement history (four days).


**Figure 2.9: BME680 sensor**

## 2.8 Total Volatile Organic Compounds (TVOCs) Sensor

Being a digital sensor, it can measure a large number of Total Volatile Organic Compounds (TVOCs), such as equivalent carbon dioxide ($eCO_2$) gas and metal oxide (MOX) levels. VOCs may generate from a large number of places like materials used in construction (wood polish, carpet, paints, etc.), machines (copiers, processors, printers), including even humans through the acts of smoking and breathing, and are categorized as pollutants and sensory irritants. This sensor is designed for keeping track of the quality of air indoors in personal devices, but because of the breakout board, it can be used as a regular I2C device. Idle mode of this sensor extends battery life in portable applications, whereas it supports multiple measurement modes during an active sensor measurement on low power consumption [25].

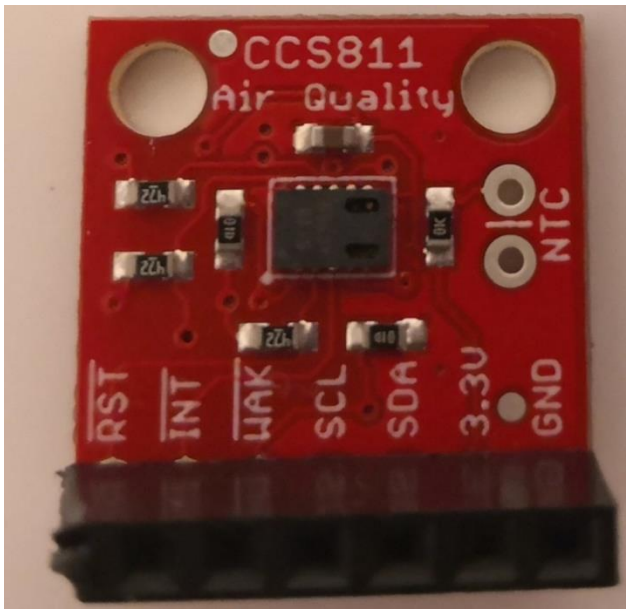



**Figure 2.10: CCS811 Gas sensor**

This gas sensor is an ultra-low-power device consist of metal oxide (MOX) gas sensor having an Analog-to-Digital Converter (ADC), a microcontroller unit (MCU), and an I²C interface. It has an application in the detection of a large variety of Volatile Organic Compounds (VOCs) to measure air pollution [25]. Moreover, hardware and software designs are simple for this sensor because of the I²C digital interface.

## 2.9 LoRa Gateway

This Gateway is used along with a selected part of the LoRa Technology Transceiver RN modules for development in various applications and products. Also, a provision of excellent communication by it with the Microchip that it supports, e.g., LoRa network and application server. Microchip's Gateway receives and forwards uplink packets transmitted according to the LoRaWAN specification [26]. Microchip Technology provides multiple Gateway Radio board devices, to support the available Microchip RN modules with different frequency bands. The particular server can also be supplied with smooth communication through TCP/IP protocol.

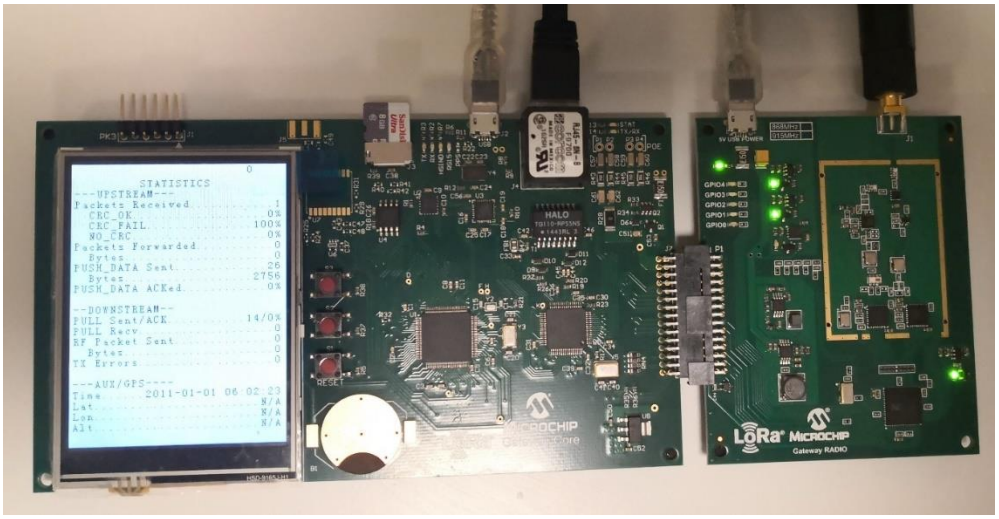



**Figure 2.11 Gateway Board**

The LoRa Gateway board collects data information received by the Radio board. The on-board microprocessor is then responsible for forwarding that data through the encoder device. This encoder device converts the processed data into a TCP/IP packet structure and outputs at the Ethernet connector. There can be an exchange in information as the LoRa Gateway and the network server can do so through Ethernet communication. The given information gets forwarded to the specified application server. USB IC (USB2412) is used for debugging the information, and basic commands are sent through the micro-USB between the host PC and microcontroller unit. The configured settings of this device can be stored onto a microSD card. Communication with this card is granted through the I2C protocol [27].
A GPS unit provides an accurate timestamp and navigational information related to Gateway world location. The coin cell connector can be used for the processing of satellite information related to the GPS.

## 2.10 Communication Protocol

### 2.10.1 LoRaWAN:

The abbreviation of LoRaWAN is the "Long Range Wide Area Network". LoRaWAN™ deals with the (network layer) communication protocol. On the other hand, it is the LoRa physical layer that enables the long-range communication link. The network architecture and the protocol are the two key components in determining the battery lifetime of a node, the network capacity, the quality of service, the security, and the variety of applications served by the network.

**Technical specification:**

LoRaWAN™ supports the three different class of devices: Class A, Class B, and Class C.
The short description of these are given below:

- Class A devices: Class A devices are battery-powered. When an uplink is sent to the server, the device opens two short downlink windows for eventual commands. If the server cannot send a downlink communication in these two short windows(Situation 1 in Figure 2.12), it will have to wait for the next uplink message.

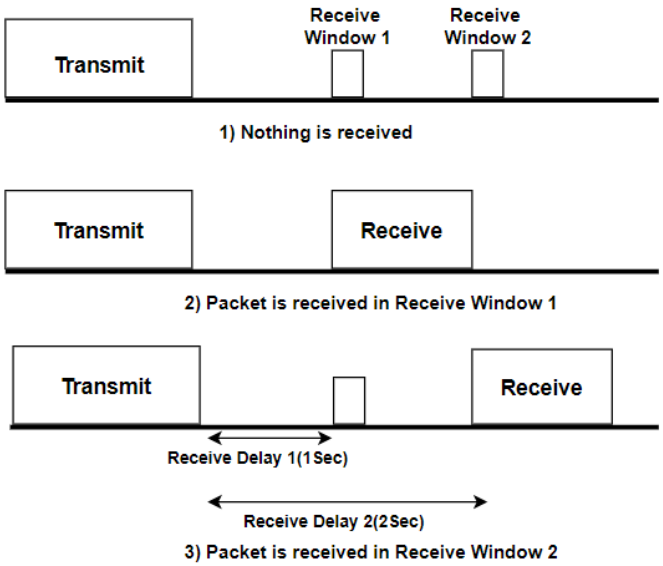



**Figure 2.12 LoRaWAN Class-A Data Transmission**

- Class B devices: Class B devices are battery-powered. Also, the two short windows of Class A and Class B have extra downlink windows which are opened at scheduled times. The windows are synchronized with the server using a Beacon from the gateway and respond to the server when the end device is listening.
- Class C devices: Class C devices are electrically powered. The receive windows are, most of the time, continuously open and close only in the process of transmission.

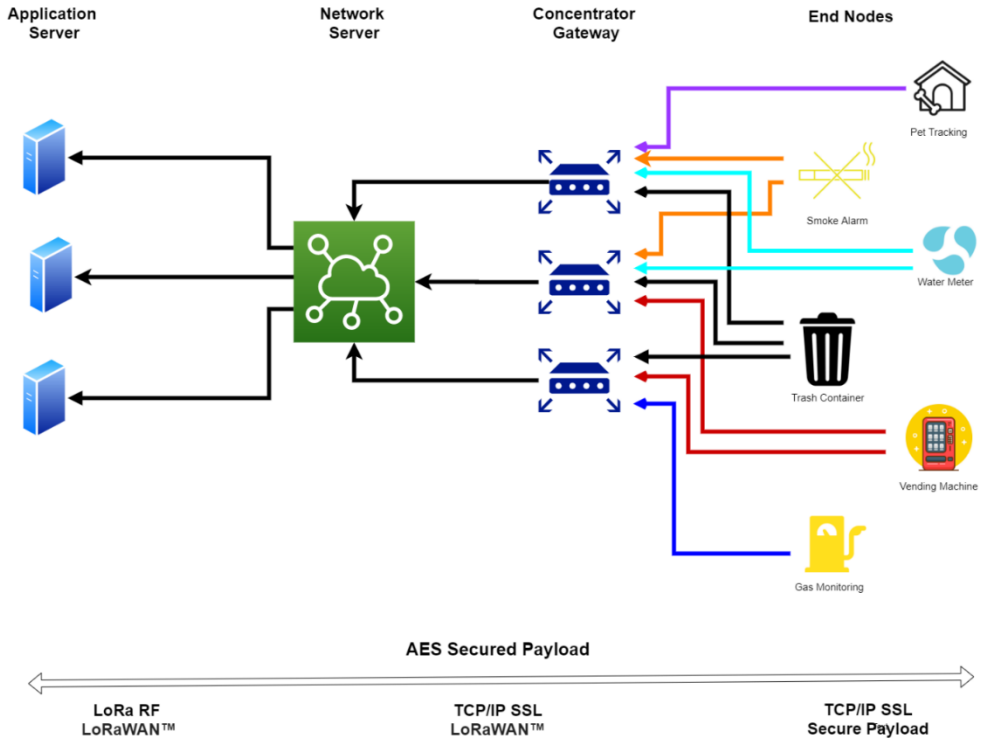



**Figure 2.13 LoRaWAN Architecture [28]**

LoRaWAN architecture has the following parts:

1. End Nodes: The end devices have different classes (Class A, B, C) depending upon different capabilities and characteristics. Also, each device class is a trade-off between network downlink communication latency versus battery-life. The End Nodes are LoRa embedded sensors. Like the Class A Nodes, which typically have sensors used to detect the changing parameter, for example, temperature, humidity, accelerometer, or GPS. LoRa transponder to transmit signals over LoRa patented radio transmission method, and optionally a micro-controller (with onboard memory) [29].
2. Concentrator/Gateway: Forward and receive the message from end nodes (or from a network server).
3. Network Server: Remove duplicate messages, manages ACKs, manages radio link parameters, etc.

Here in this project, the application server is also needed, which gets data from the network server, analyze data, display in real-time, and realize some function. In the LoRaWAN, the end node aggregates the collected data to the gateway through an uplink, and the gateway performs

simple data processing and transferring, the network server receives the data sent by the gateway through the TCP/IP . Each end-device is needed to be personalized and activated to join into a LoRaWAN network.
Activation of an end-device can be achieved via.

- Over-The-Air Activation (OTAA): End-devices follows a join procedure first before participating in the data exchange with the network server in case of over-the-air activation. An end-device has to pass through a new join procedure whenever the session context information is lost. The joining procedure requires the end-device to be personalized with the following information before it starts the join procedure: a globally unique end-device identifier (DevEUI), the application identifier (AppEUI), and an AES-128 key (AppKey).
- Activation By Personalization (ABP): In ABP (Activation by Personalization), The two-session keys, NwkSKey, AppSKey, and DevAddr, are preprogrammed into the device, and thus it is pre-registered on the network. For communicating purposes, the device uses the session keys bypassing the initial join procedure.

In this Project, the device uses the ABP to participate in LoRaWAN network
**LoRaWAN for Europe:**
The LoRaWAN specification varies differently depending upon the different regional spectrum allocations and regulatory requirements. LoRaWAN defines ten channels. The eight of these are multi-data rate from 250bps to 5.5 kbps, a single high data rate LoRa® channel at 11kbps, and a single FSK channel at 50kbps. The maximum output power allowed by ETSI in Europe is +14dbm. The only exception is for the G3 band, which allows +27dBm. Although there are duty cycle restrictions under ETSI, there are no such max-transmission or channel dwell time limitations [30].
The LoRaWAN specification for Europe is shown in Table 2.3,

Table 2.3: LoRaWAN specification for Europe [30]

| | |
|---|---|
| Frequency band | 867-869MHz |
| Channels | 10 |
| Channel BW Up | 124/250kHz |
| Channel BW Dn | 125kHz |
| TX Power Up | +14dBm |
| TX Power Dn | +14dBm |
| SF Up | 7-12 |
| Data rate | 250bps-50kbps |
| Link Budget Up | 155dB |
| Link Budget Dn | 155dB |

## 2.10.2 Universal Asynchronous Receiver-Transmitter:

UART (Universal Asynchronous Receiver/Transmitter) is a physical circuit in a microcontroller, unlike SPI and I2C that are communication protocols. A UART receives and transmits serial data between devices using only two wires. UART communication enables two UARTs to communicate directly with each other. One UART collects parallel data from a controlling device like CPU, converts it in serial data, and transmits it to the other UART. The receiving UART receives the serial data and converts it back into parallel data for the receiving device. The flow of data is from the transmitting UART's Tx pin to the receiving UART's Rx

pin. When a start bit is detected by the receiving UART, it starts to read the incoming bits at a specific frequency known as the baud rate. The Baud rate measures the speed of data transfer in bits per second (bps). Both the receiving and transmitting UARTs are required to operate at about the same baud rate with the maximum difference of 10%. Otherwise, the timing of the bits will get too far off [31].

### 2.10.3 Inter-Integrated Circuit:

It is a bidirectional synchronous serial bus tailored for serial communication. It consists of only two wires called SDA (Serial data line) and SCL (Serial clock line) and a few pull-up resistors. The SDA line connects the SDA pins of all devices and sends the data. The SCL line connects SCL pins of all devices and sends the clock signal for proper communication timing. The pull-up resistors keep both the lines on the HIGH state.
Start Condition: When the master device switches the SDA line from high voltage level to low voltage level and then the SCL line from high to low, the transmission starts. This acts as a signal for a slave device that a transmission is about to happen. If two master devices send a start condition simultaneously, whoever pulls the SDA low first takes ownership of the bus [32].
Stop Condition: After all the data frames have been sent, the stop condition will be transmitted. The SCL line will return to a high voltage level from a low voltage level, and after that, the SDA line will also return to high voltage from a low voltage. The change in the value of SDA when SCL is high during normal data writing operation can cause a false stop condition. Therefore, it should be avoided [32].

### 2.10.4 MQTT:

Message Queuing Telemetry Transport (MQTT) is a straightforward and lightweight "publish and subscribe" protocol for messaging. It is catered for high-latency, low-bandwidth, and for Internet of Things applications it is the perfect solution. The basic designing principles is to use minimum bandwidth, device resources by keeping in view the assurance and dependability of delivery. It is due to these fundamentals that this protocol is perfect for the emerging "Machine to Machines (M2M)" or "Internet of things (IoT)" world of interconnected devices. It is also ideal for low-bandwidth low-battery mobile apps. The establishment of communication between multiple devices becomes simple [39].
Following are the few primary concepts in MQTT

- Publish/Subscribe
- Topics
- Messages
- Broker

**Publish/Subscribe:** In this system, a device can either be able to publish a message on any topic, or it can be subscribed to a receive messages based on a specific topic. For instance: If "Device 1" publishes a message on a particular topic and "Device 2" happens to be subscribed to that exact topic, then "Device 2" will receive the message.
**Topics:** Topic is a general area of interest for which you can specify how you would like to publish a message. Conversely, you can register for incoming messages on various topics as well. Their representation is denoted using strings separated by a forward slash. Each slash indicates a topic level.
**Message:** Message can be any data or command that is required as information to exchange between the chosen devices.

**Broker:** The broker's responsibility involves firstly receiving the messages, then filtering them and subsequently deciding who interested by those messages. Finally, the broker then publishes the message to whatever clients have subscribed to it[39].

## 2.11 Software and Tools

The numerous software and tools used throughout the development and testing during the project are listed below.

### 2.11.1 Software:

- Arduino Software 1.8.12 - The Arduino Software is an open-source IDE written in Java, which is used to write and upload code to the hardware.
- PyCharm 2018.3.5 - PyCharm is an integrated development environment used in computer programming, specifically for Python.
- Android studio 3.4 - Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software, designed explicitly for Android development.
- LoRa Development Utility v 1.0.1 - The Utility is used for demonstration of the LoRa network evaluation, easy configuration, and management of connected Gateway/RN Module devices.

### 2.11.2 Tools:

- UNI-T UT33C Digital multimeter: The UNI-T multimeter is an electronic measurement device that measures voltage, current, and resistance.
- Mooshimeter: Mooshimeter is a smartphone multimeter which combines with BLE technology. It is mainly used to log electrical power parameters.
- Huawei Mate 10 Pro Smartphone: This android smartphone is used for USB debugging and to run the Air quality monitor app.

# Chapter 3 : System Designing

The fundamental consideration for the system designing was the compact size, the lower consumption of power, communication over a broader spectrum, measurement of the level of $CO_2$, humidity, pressure, temperature, the volatile organic compound. Based on these essential points, this project may be implemented on platforms such as installation on drones, etc. by the developers in the future.

## 3.1 Brief Description of System

The hardware infrastructure encompasses the use of two environmental sensors, BME 680 and CCS811, opted to collect AQI data. The sensors received data values are then processed by the ATmega328u4, being an 8bit AVR based microcontroller that encompasses 32KB flash memory with write plus, the read operation. The values obtained are translated to digital float values and transmitted by the RN2483 through the ATmega 328u4 UART interface. The RN2483 has a lower power consumption, providing transmissions over longer ranges and gives the option of economic power supply for long-range data transmission. It runs on a physical layer modulation scheme and compiles with class-A LoRaWAN protocol. It integrates with the Application Programming Interface(API) processor, which makes provides for a low power wide spectrum solution. In this project, RN2483 modules transmit the float values processed from the IAQ sensor by the ATmega328u4 through the UART interface.
For the energy harvester, battery, solar charger circuit, and solar panel are connected. So, it recharges whenever light appears on the solar panel and charger circuit power for ATmega328u4 microcontroller.
On the user interface part, an android based mobile application is developed, which processes the information via the internet cloud, after which it displays it to the user. Moreover, the local server is being used to store the data.

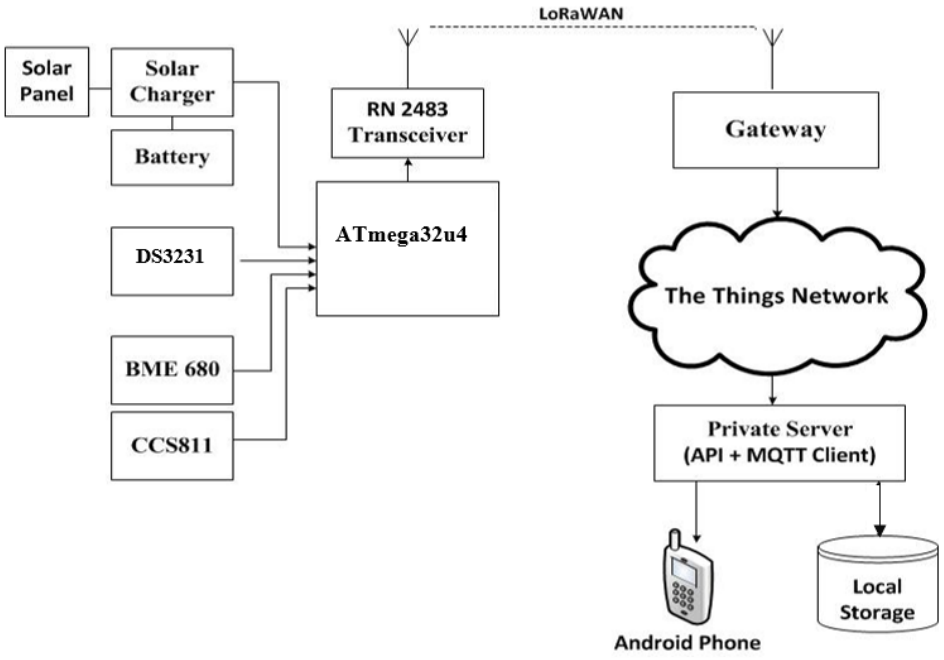



**Figure 3.14 Block Diagram of Overall System**

## 3.2 Energy Harvesting

It comprises a solar panel, solar charger, and ATmega32u4 controller. The solar charger consists of 6V and provides 530 mA current with a peak power of 3.69 Watts, which is enough to run the system. The physical dimension of solar panel is 210mm x 113mm x 5mm / 8.3" x 4.4" x 0.2". The solar panel is connected with the solar charger through a 3.5mm x 1.1mm DC jack connector [33].

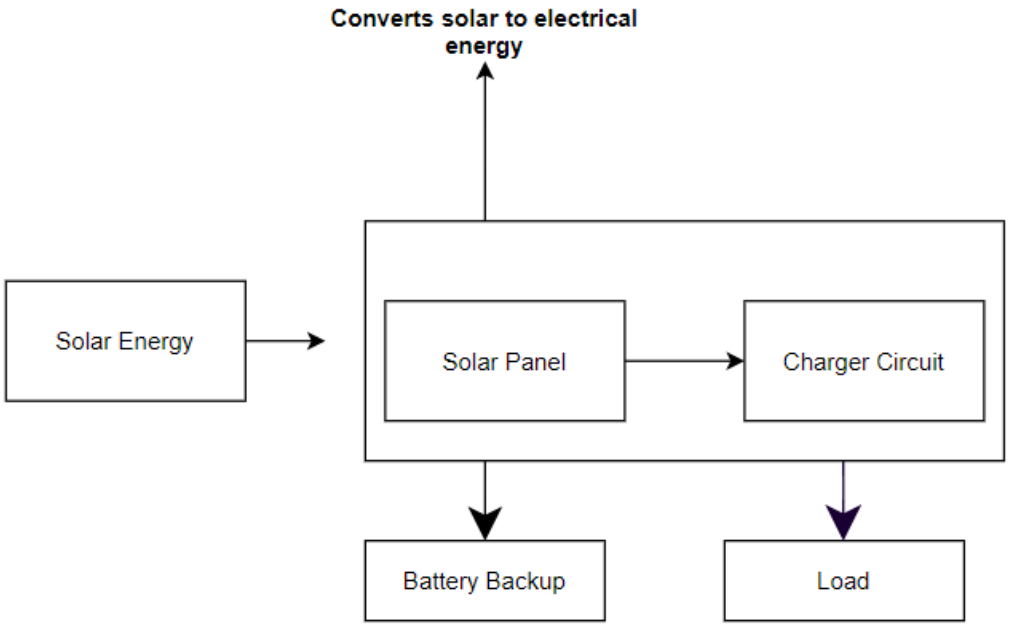



**Figure3.15: Block diagram of energy harvesting system.**

As shown in Figure 3.15, the Solar charger is connected with 3.7V Lithium-Ion and 6V Solar Panel, and further with ATmega32u4 board. It gets the energy via the solar panel and supplies the required power into the system and also charges the battery at the same time. The Lithium-Ion battery has an output between 4.2V to 3.7V depending upon the charged level. The cell of the battery has a limit of 2000mAh, which is as per the specification of the system to power it up. Solar charger works on the basic principle of MPPT, but the structure of the charger does not represent actually 'true' MPPT in the name. This is because the trackers of the Max power point analyze the 'tracking' of the current and voltage curve of the solar panel to get the maximum total energy [14]. Meaning, with a change in density of emission of light, the tracking of current and voltage is done carefully. Generally, controllers perform MPPT with a DC/DC converter. For example, the charging 6V battery with an approximately 12V panel needs voltage to be variable between the values of 9V and 14V. This is dependent upon the visible light and current draw. The buck converter does the most that it can to maintain the current draw to maximize the total power that the output can provide.

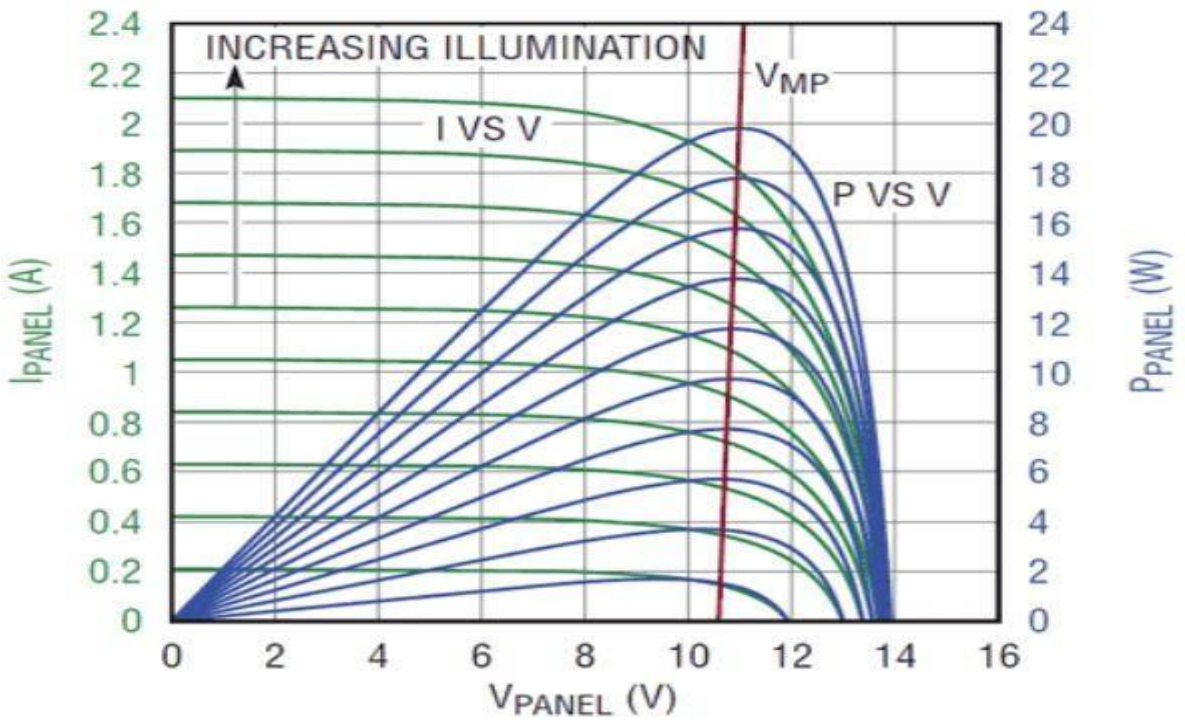



**Figure3.16: Current-Voltage and Power-Voltage curve of the solar panel [14]**

The green line in the panel represents an I-V curve, and it shows the condition for a given light source. With an increment in the light power, the voltages stay constant, but it leads to an increase in the amount of current, which is drawn. DC/DC converter will be getting maximum power when it operates on the red line [14].
In Solar panels, the current and level of voltage changes regularly with a dependence on the amount of light, as shown in Figure 3.16 and Figure 3.17.

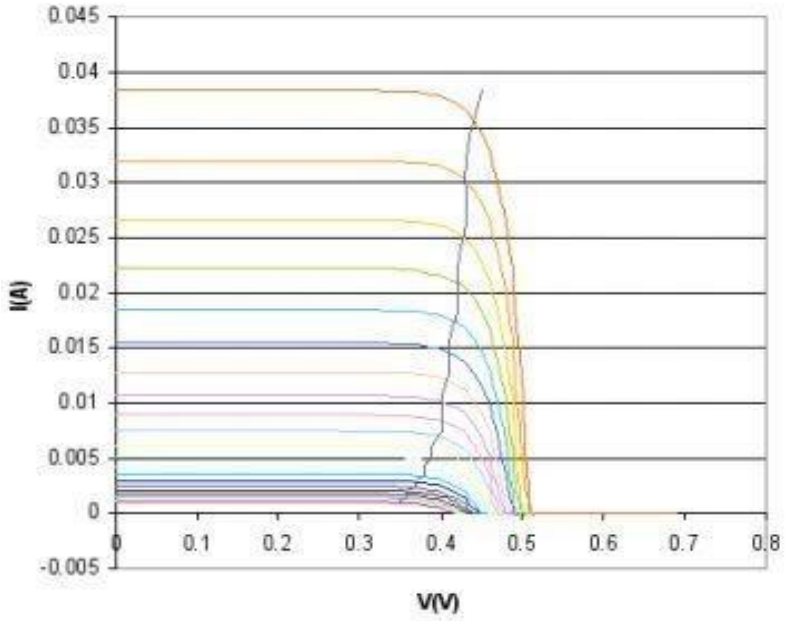



**Figure 3.17: Solar cell I-V curve [14].**

Consider the red line at the top, which shows the level of most light starts to the right of the point where the line meets the horizontal scale. That represents the point of the current (I) = 0.

The cell has a voltage of 0.5V, showing that it is the open-circuit voltage. When drawn current witnesses an increase, there is a drop in voltage by a small value until the level where it starts drawing 38mA, the voltage is around 0.4V at this point. Subject to the conditions of light, current within the short circuit is 38mA i.e. the most amount which can lie from 38mA (red) to 32 mA (orange) down to 5mA (yellow) or even lower. Batteries of solar cells depending upon their types may vary but the open circuit has 0.5V voltage. Collection of cell circuits is connected as a series circuit to sum it. There are 12 cells in a 6V panel (12 * 0.5V = 6V) [14]. Given a condition of light, when the current being taken in through the charger is lower than short circuit current of the panel, it will function at a satisfactory level. If the level of light fluctuates even a bit, it leads to an unstable nature in the charger. If it draws a lot of current, it can cause the voltage to completely collapse, and that can turn off the charger. As a result, it causes a reduction in the amount of current that can be drawn, and it causes the voltage of the panel to recover itself. This can turn back the charger to an on position. If again a lot of current is taken in by the battery, the cycle will repeat itself. Figure 3.18 shows the behavior of solar charger as light in solar panel varies [14].

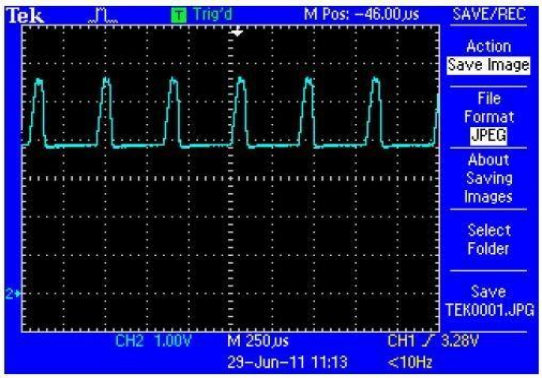



**Figure3.18: Behavior of solar charger as light in solar panel varies [14].**

**Load Sharing:** The MCP73871 chip located within the charger is equipped with the ability of 'load sharing'. This translates to the usage of the battery during the process of being charged and connects the load directly with the battery. Thus, the charger is doing both, i.e., charging the cells and providing energy to the load at the same time. The process of charging and discharging takes place inside the battery at the same time continuously [34]. The diagram displaying load sharing within the system is displayed in the Figure 3.19.

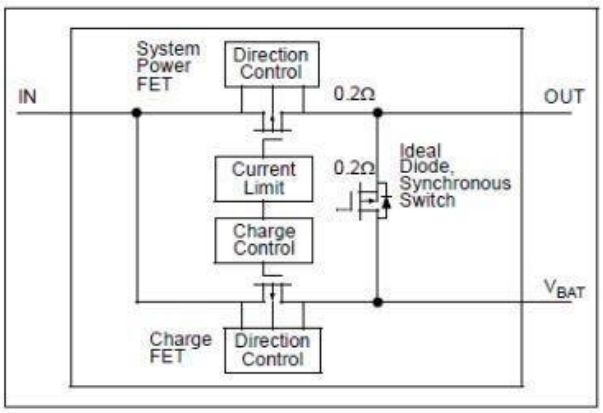



**Figure3.19: System load sharing diagram [14]**

This solar charger contains inside the chip a transistor, which has connections to the output load from the input voltage to maintain an efficient supply from battery charging. The lipo battery provides current up to 1.8A in an instance where the panel cannot provide the amount of current that is required.

## 3.3 Real Time Clock (DS3231)

The DS3231 is an economical, highly efficient in accuracy, I2C real-time clock (RTC), which has the integration of a temperature-compensated crystal oscillator (TCXO) and crystal. It comprises an input for cells along with maintenance of accuracy in estimation when the supply of power is not interfered with in any way. The presence of the crystal resonator increases the accurate nature over a long time period and causes a reduction in the count of pieces while manufacturing. The DS3231 device can be readily available for all within industrial and commercial values of temperature. Information on all time units is maintained by the RTC. Different months have different numbers of days, which is adjusted automatically, as is the case of leap years. The nature of AM and PM in time formats, depending on the 12 or 24 hours nature of displaying time is also adjusted. There is a serial transfer of data and addresses through an I2C bidirectional bus [35].
Since the interface between ATmega32u4 and DS3231 is I2C. Respective SDA and SCL pins are connected with each device. VCC and GND pins from DS3231 is connected to the 3.3V and GND from ATmega32u4, respectively. And, SQW pin is connected to D7 interrupt pin from ATmega32u4. Interfacing the DS3231 RTC on the ATmega32u4 is done with the help of *DS3232RTC.h* library [36].

```
1.  time_t t; //create a temporary time variable so we can set the time and read the ti
    me from the RTC
2.  t = RTC.get(); //Gets the current time of the RTC
3.   // set Alarm  to occur once per minute
4.  RTC.setAlarm(ALM2_EVERY_MINUTE, 0, 0, 0, 0);
```

We used the *RTC.setAlarm* function to set the alarm for every 1 minute. So, the interrupts fire to wakeup ATmega32u4 every 1 minute. When it wakes for 100ms, it just carries the data from sensors to the network server via LoRaWAN. Sleep mode can allow the program to lower the amount of consumption of power, and the AVR library was used for controlling sleep modes [37]. To reduce the power consumption by board, power indication led was removed.

## 3.4 Sensors

Various sensors were considered for this project, but finalized nodes mainly use the BME680 and CCS811, because of their features of temperature, RH, IAQ, TVOC, $eCO_2$, and atmospheric pressure sensing.

### 3.4.1 BME680

The BME680 is an integrated environmental sensor developed by Bosch Sensortec. It is used to measure temperature, RH, atmospheric pressure and IAQ, and supports communication using both protocols, i.e., I2C and SPI. [23]. Interfacing the BME680 sensor on the ATmega32u4 is done with the help of *Adafruit_BME680.h* library. [38]

**Connection**

The board can be hooked up as follows:

- VCC Pin: 3.3V from the Arduino board is connected to it.
- GND Pin: It is connected to the Gnd pin of the Arduino board.
- SDI Pin: SDI pin is tethered to the SDA line of the Arduino board.
- SCK Pin: SCK pin is tethered to the SCL line Arduino board.

Important features of BME630 are shown in Table 3.

**Table 4: Important features of BME680 [23]**

| Parameter / Symbol | Typical Value | Maximum Value |
|---|---|---|
| Supply Voltage / $V_{DD}$ | $1.8V$ | $3.6V$ |
| Standby Current / $I_{DDSB}$ | $0.29\mu A$ | $0.8\mu A$ |
| Start-up Time / $t_{startup}$ | | $2ms$ |
| Sleep Current / $I_{DDSL}$ | $0.15\mu A$ | $1\mu A$ S |
| Slave address | 0x77 | |
| Dimensions / $B \cdot D \cdot H$ | $3.0 \cdot 3.0 \cdot 0.93mm$ | |
| Pressure / P | $300hPa$ | $1100hPa$ |
| Temperature / $T_A$ | $-40^{\circ}C$ | $85^{\circ}C$ |
| Air Quality / $IAQ_{rg}$ | 0 | 500 |

### 3.4.2 CCS811

Air Quality Digital Sensor CCS811 is a gas sensor which has a ultra-low power composition. To identify broad types of VOCs (Volatile Organic Compounds), it incorporates a MOX gas (metal oxide) sensor. Therefore, using an integrated MCU (Micro-controller Unit), indoor air quality can be observed. MCU consists of an I2C interface and ADC (Analog-to-Digital Converter) [39]. Interfacing the CCS811 sensor on the ATmega32u4 is done with the help of *SparkFunCCS811.h* library [40]. The connected pin configuration of a CCS811 sensor is given below.

**Connection**

- 3.3 V pin – The 3.3V pin from the Arduino board is connected to it.
- Gnd – Gnd pin from the Arduino board is connected.
- SDA – Data pin is connected to the SDA pin of the Arduino board.
- SCL – Clock pin is connected to the SCL pin of the Arduino board.

## 3.5 Communication

The BME680 and CCS811 board are used, which consists of 4 and 2 different sensors, respectively. Also, the DS3231 is used for triggering an interrupt. RTC, as well as other sensors, have unique device addresses to facilitate ATmega32u4 in choosing the required device for communication. The Serial Clock (or SCL) and Serial Data (or SDA) are two wires/ lines. SDA line carries the data, whereas the SCL line is the clock signal which synchronizes the data transfer between the devices on the I2C bus, and it is generated by the ATmega32u4 controller.

As shown in Figure 3.20, The Serial Clock pin of the Arduino Board is connected to the Serial Clock pins of the three breakout boards. Similar is the case with the Serial Data pins. The boards are powered with the Gnd and the 3.3V pin from the Arduino Board. The pull-up resistors are not used as the breakout boards already have these.
Similarly, for UART data transfer Tx pin of ATmega32u4 is connected to the Rx pin of RN2483 Transceiver, and the Rx pin of ATmega32u4 is connected to the Tx pin of RN2483.
In order to communicate with these sensors, their unique addresses are required, which can be found from the datasheets of the sensors. Further, the addresses of the internal registers of sensors are also required in order to read the data from them. Communication with the device is performed by reading and writing to registers. Registers have a width of 8 bits so 8-bit addressing is utilized.

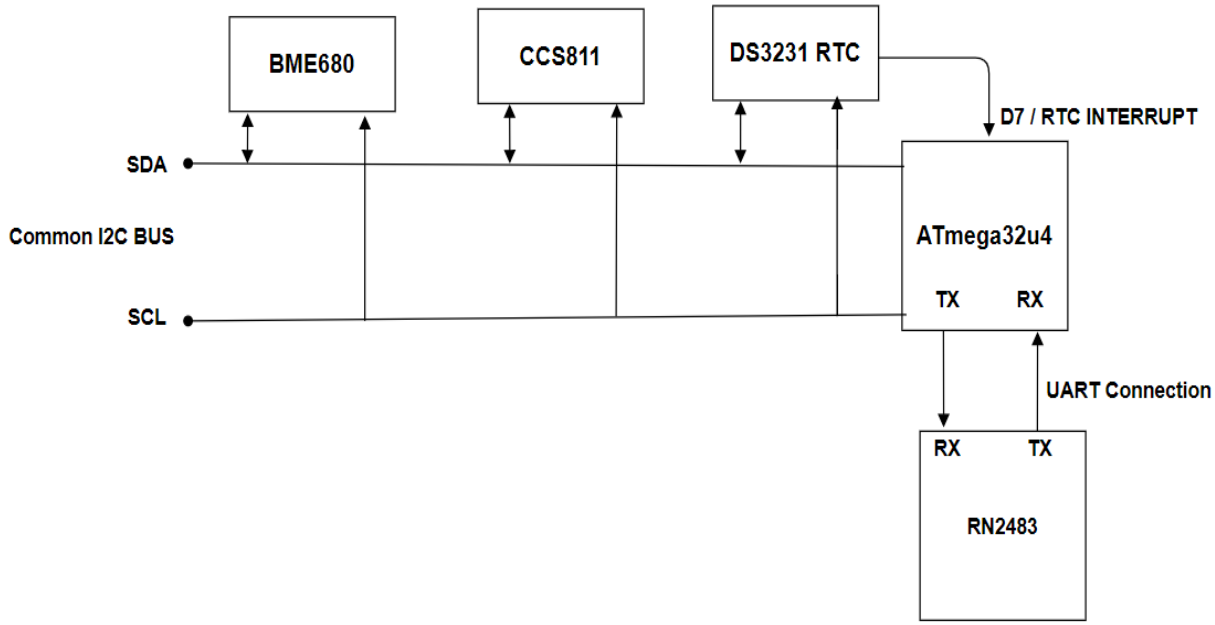



**Figure 3.20: I2C and UART connection**

## I2C Data Transmission

1. The master ATmega32u4 sends the starting condition to every connected slave by switching the SDA line from high to low voltage level before switching the SCL line from high to low.
2. The ATmega32u4 sends each slave a 7-bit address in order to communicate with, along with the read/write bit.
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the SDA line high.
4. The master sends or receives the data frame.
5. After each the data frame is transferred, the receiving device returns another ACK bit to the sender to acknowledge the successful receipt of the frame.
6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high.

**UART Communication:**
In UART communication, ATmega32u4 and RN2483 communicates directly with each other. ATmega32u4 coverts data collected from the sensor into serial form and transmit it in serial with RN2483. The transmitting UART(ATmega32u4) adds start and stop bits to the transmitting packets so that Receiving UART(RN2483) will know when to read the bits. When RN2483 UART gets start bit, it starts reading incoming bits at 9600bps.

## 3.6 The Things Node Using ATmega32u4 and RN2483 Transceiver

The LoRaWAN network protocol is initiated by the RN2483 microchip module. The LoRa Alliance certified this as the first one being in line with the specifications of LoRaWAN 1.0. The UART interface is adapted to communicate with the RN2483 by the ATmega32u4. UART's default configuration consists of no flow control, 1 stop bit, no parity, 8 bits, and 57600 bps [19]. Interfacing the RN2483 transceiver with ATmega32u4 is done with the help of *rn2xx3.h* library [41].The pins of the ATmega32u4 board and RN2483 are connected, as shown in Table 5 and Figure 3.21.

Table 5: The pin connections of Arduino micro and RN2483

| RN2483 pin name | Arduino pin number |
|---|---|
| UART_TX (6) | RX |
| UART_RX (7) | TX |
| RESET (32) | RST |
| VDD (34) | 3.3V |
| GND (33) | Gnd |

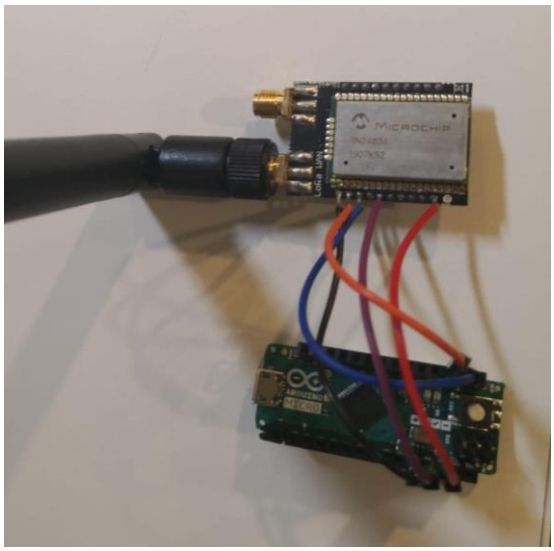


**Figure 3.21: Connection between ATmega32u4(Arduino) and RN2483**

## 3.7 LoRa Setup

The following keys were used to configure the RN2483 LoRa setup:

- DevUEI (mac address): DevUEI(MAC address) is the physical address of the RN2483 transceiver. The address is build up with 16 hexadecimal characters. The MAC-address is set by the manufacturer and is a unique value. Wait for 1 second and listen to the RX serial bus to receive the MAC-address [41] It is picked up from the RN2483 by writing the syntax:

```
String hweui = myLora.hweui();
 while (hweui.length() != 16)
 {
   Serial.println("Communication with RN2xx3 unsuccessful. Power cycle the board.")
;
   Serial.println(hweui);
   delay(1000);
   hweui = myLora.hweui();
 }
```

  If the length of obtained MAC-address is not of 16 hexadecimal characters, it waits for 1 second and listens to the RX serial bus.
- DevAddr (Device Address): The assigned device address specifies a unique number in the Lora network. This address is assigned to the RN2483. “DevAddr” is also the network address for LoRa network. After processing the data, the RN2483 sends feedback if the communication process was successful. There can then be listened to the RX of the ATmega32u4 to recover the status value [41].
- NwkSKey (Network Session Key): A network session key that is device-specific and is utilized to evaluate MIC code’s (Market Identifier Codes) data integrity; and encryption/decryption of payload field between the Backend and the device [41].
- AppSKey (Application Session Key): The Application Session Key is the device-specific key which is used for encryption of the payload field. If this AppSKey is not provided to the Backend, then the network cannot access the payload information [33].

```
const char *devAddr = "26011659";
const char *nwkSKey = "FA6BF38E66DFFE78477EFC000B659073";
const char *appSKey = "E7E91F531FB5742E9944A4EFC7EF4FE2";
join_result = myLora.initABP(devAddr, appSKey, nwkSKey);
while (!join_result)
{
  Serial.println("Unable to join TTN");
  delay(60000);
  join_result = myLora.init();
}
Serial.println("Successfully joined TTN");
```

A new device as well as a new application is registered first in the TTN dashboard. We have used ABP method to connect RN2483 to the network. So, there will be no handshaking between RN2483 and network [42]. Since node is registered as ABP on the TTN Dashboard, the correct keys are copied into the sketch from the TTN dashboard.

The node sends the join request, which will be received by the gateway in the TTN console. The Things Network validates the *devAddr, appSKey and nwkSKey* and establishes the communication between the node and TTN. If it is transmitting packets to TTN, messages arriving on the TTN dashboard can be seen, which is shown in Figure 3.22.

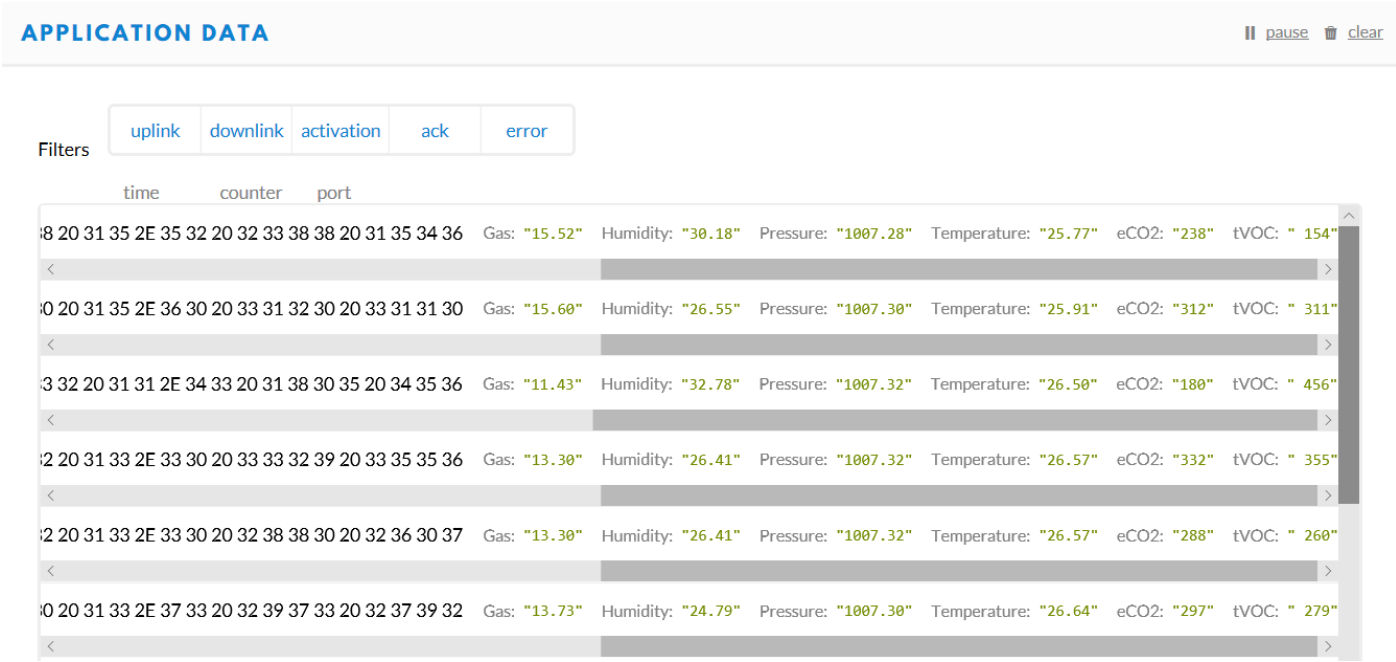



**Figure 3.22: The Things Network(TTN) Console**

## 3.8 Modulation and Data Rate

LoRaWAN uses LoRa modulation technique which is used for long-range communication link. Wireless system consume low power by using Frequency Shifting Keying (FSK) modulation. LoRa is based on Chirp Spread Spectrum (CSS) modulation which maintains the same low power characteristic as of FSK modulation with a significant jump in the communication range [43].Table 6 and Table 7 shows the encoding used for Data Rate(DR) and TXpower in the EU863-870 band.

| Data Rate(DR) | Configuration | Indicative Physical Bit rate(bit/s) |
|---|---|---|
| 0 | LoRa SF12/125kHz | 250 |
| 1 | LoRa SF11/125kHz | 440 |
| 2 | LoRa SF10/125kHz | 980 |
| 3 | LoRa SF9/125kHz | 1760 |
| 4 | LoRa SF8/125kHz | 3125 |
| 5 | LoRa SF7/125kHz | 5470 |
| 6 | LoRa SF7/125kHz | 11000 |
| 7 | FSK: 50kbps | 50000 |
| 8…..15 | RFU | |

Table 6: EU863-870 Data Rate [45]

| TXPower | Configuration |
|---|---|
| 0 | 20 dBm(If supported) |
| 1 | 14 dBm |
| 2 | 11 dBm |
| 3 | 8 dBm |
| 4 | 5 dBm |
| 5 | 2 dBm |
| 6….15 | RFU |

Table 7: EU863-870 TXPower [45]

```
if (_moduleType == RN2903)
{
  setTXoutputPower(5);
}
else
{
  setTXoutputPower(1);  //TXPower
}
sendMacSet(F("dr"), String(5)); //0= min, 7=max  DATA RATE

_serial.setTimeout(60000);
sendRawCommand(F("mac save"));
sendRawCommand(F("mac join abp"));
receivedData = _serial.readStringUntil('\n');

_serial.setTimeout(2000);
delay(1000);
```

End device transmission power(TXPower) was set to default value i.e 14dBm. EU frequency plan was used for the communication. Data rate was set to dr(5) as shown in snippet code above(5470 bit/s). Different configuration parameter used in our system is shown in Figure 3.23.


```
"gw_id": "eui-3256897412547810",
"payload": "QFkWASYAAAABOuQ8eQqUpU9WLhAj+2l0KNlq7OAwaW8XH46FSUd3vDF1",
"lora": {
  "spreading_factor": 7,
  "bandwidth": 125,
  "air_time": 87296000
},
"coding_rate": "4/5",
"timestamp": "2020-05-11T14:42:01.671Z",
"rssi": -75,
"snr": 9,
"dev_addr": "26011659",
"frequency": 868300000
```


**Figure 3.23 Uplink Transmission configuration**

## 3.9 LoRa Antenna

In order to expand the range it is possible to connect an external antenna on LoRa shield. A 900 MHZ LoRa antenna is connected to RN2483 transceiver board. It is good for any 900MHZ or less radio including our LoRa breakout board [43].

## 3.10 Expected energy consumption

Table 8 represents the worst case power and consumption of all components used in ATmega32u4 node.

**Table 8: Expected worst case energy consumption of all components [44] [23] [25] [19] [35]**

| Component | Current | Power |
|---|---|---|
| ATmega32u4 | 15mA | 75mW |
| BME680 IAQ<br>BME680 Temperature<br>BME680 RH<br>BME680 Pressure | 15mA<br>350µA<br>450µA<br>849µA | 49.5mW<br>1.2mW<br>1.5mW<br>2.8mW |
| CCS811 | 26mA | 46.8mW |
| RN2483 | 38.9mA | 128.37mW |
| DS3231 | 200µA | 726 µW |

Table 9 presents the total worst case sleep current and power consumption of each components.

**Table 9: Expected worst case sleep energy consumption of all components [44] [23] [25] [19] [35]**

| Component | Current | Power |
|---|---|---|
| ATmega32u4 | 12µA | 60µW |
| BME680 | 1µA | 3.3µW |
| CCS811 | 19µA | 62.7µW |
| RN2483 | 1.6µA | 5.28µW |
| DS3231 | 110µA | 363 µW |
| Total | 143.6µA | 494.28µW |

Removing extra hardware can save a bit of power. Each of the hardware comes up with voltage regulators, power LEDs, USB ports, etc. All the unnecessary parts were removed from the boards.

## 3.11 Data extraction, Manipulation, and Visualization

After designing the hardware system, the collection of ambient temperature, emission of $CO_2$, humidity, IAQ, etc. is done and processed by the ATmega32u4 as per the designed algorithm. After processing, this data is uploaded through the LoRa protocol and store in the local server. This section explains the extraction of data from the local network server, processing the extracted data into human-readable form, store, and visualization of the processed data to make it more intuitive. Brief introduction of the techniques is as follows:

### 3.11.1 Gateway

The Atmega32u4 sensor node transmits data to LoRa gateway, which then connects to the internet using the standard IP protocol. Finally, the data received from the LoRa embedded sensors is transmitted to the Internet, i.e., a network server [37]. Embedded EUI is used to connect the gateway on the Things Network. Also, A power source and Ethernet are permanently connected to the gateway. The Gateway overview in Things console is shown in Figure 3.24.

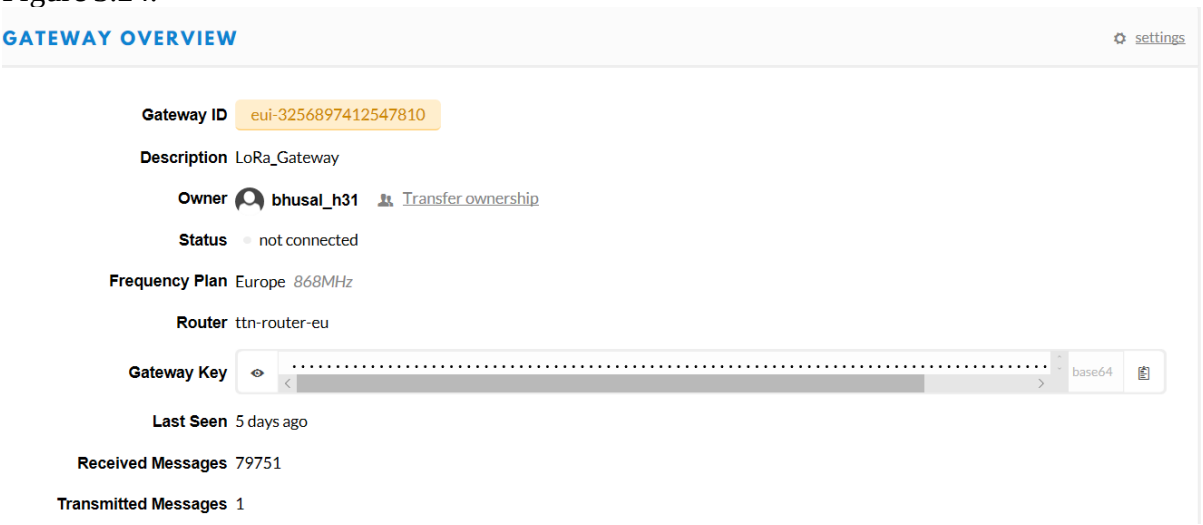



**Figure 3.24 Gateway overview on The Things Network**

### 3.11.2 The Things Network (TTN)

TTN is a decentralized network that allows the exchange of data from low powered devices using long-range gateways. TTN uses LoRaWAN technology to make communication and transfer data between devices and the platform. The TTN system allows for deployment options flexibility. Connecting to the community public network, which is hosted by the TTN Foundation or its partners, is considered the most favorable option. By utilizing the MQTT API, the application can connect to a Public Community Network Handler [45].

To communicate with the sensor node, an application is registered in The Things Network(TTN). The ABP method is used to establish a connection between the ATmega32u4 node and The Things Network. Figure 3.25 from the TTN console shows the parameter set to connect it with the ATmega32u4 node.

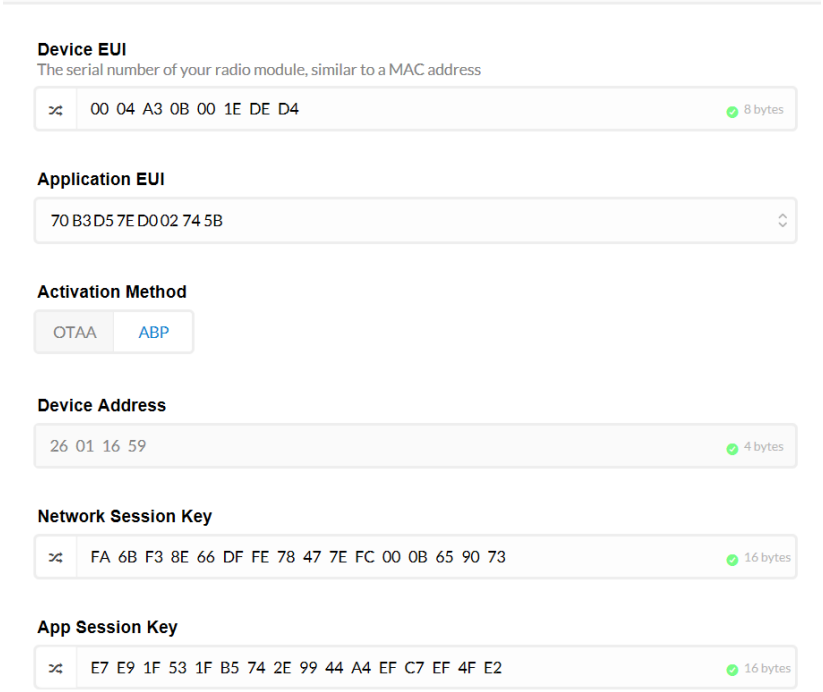



**Figure 3.25: Device setting in The Things Network(TTN)**

### 3.11.3 The Things Network to Android Application

TTN network provides multiple options to connect the network with other devices and platforms such as databases and AWS etc. To be able to receive the data from TTN networks to android apps, a medium is needed between them, which will transfer the data. The APIs provided by the TTN networks are:

1. HTTP Web Server
2. Data API (MQTT)

HTTP server or webserver API is useful for creating a project that involves or runs on the web. But for the android app, the continuous stream of data is needed, and hence Data API (MQTT) is used to access MQTT service on android. The paho MQTT library is installed by putting these implementations in build.gradle file [46].

```
1.  implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
2.  implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
```

After doing that, some permissions are set in an android manifest file such as permission for using the internet, etc.

```
1.  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
2.  <uses-permission android:name="android.permission.WAKE_LOCK" />
3.  <uses-permission android:name="android.permission.INTERNET" />
4.  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

The credentials were then extracted from the TTN network website and fed in the android app to create the connection. Here are the brief details of the credentials provided by the TTN network.

*Host/broker : <Region>.thethings.network*
*Topic : <Region>.thethings.network*
*Port: 1883*
*Username: '<AppID>'*
*Password: '<AppKey>'*
These credentials were used in android code to communicate with the TTN network using MQTT and start the connection.

```
client =new MqttAndroidClient(this.getApplicationContext(), "tcp://eu.thethings.network:1883",
                clientId);
//setting up mqttclient object, we need to give mqtt broker ip
MqttConnectOptions options = new MqttConnectOptions();
options.setUserName("bhusalloramote");
options.setPassword("ttn-account-v2.5WYkZSOz5d2fyixaNvueihH1A0Qy8Owfz3pRt-FF1iw".toCharArray());
options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);
```

This snippet of code shown above establishes a connection with the MQTT broker and thus the host, port, username, and password is provided. After making the connection, the topic is subscribed to get data from the TTN network. The following code does that.

```
final String topic = "bhusalloramote/devices/lora_mote/up";
int qos = 1;
try {
    IMqttToken subToken = client.subscribe(topic, qos);
}
```

After subscribing to the topic, a function was created that will be called whenever a message will be received from the MQTT broker. The application parses the data from JSON to string and shows it on the android app every time a message received.

```
            <TextView
                android:id="@+id/humidity"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="0"
                android:textAlignment="center"

                android:textSize="30sp" />
```

The short XML code for the android application layout is shown above. Different ids are used for the text view of different sensor data. So that data will be displayed in the text view.

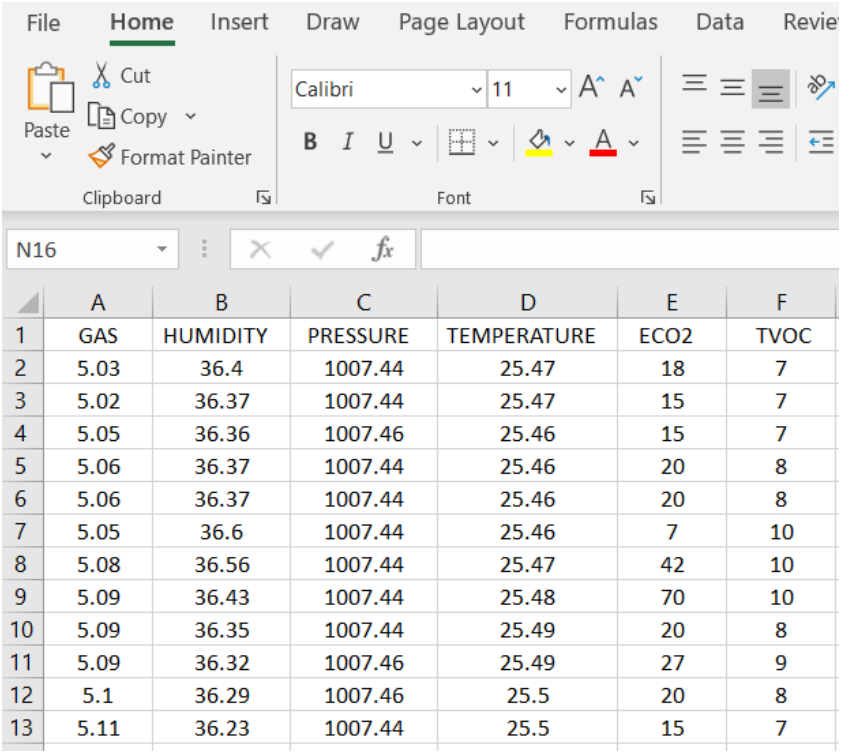
### 3.11.4 Python Integration

For python integration, the same credentials are used, which were used for the android app. Like android, the library is needed to be installed. The following command is used to install the library [47].

**Pip3 install paho-mqtt**

After the installation of the library, a connection is made with the broker by using the functions of that library and thus subscribed to the topic. The MQTT client is running in a loop, and a callback function will be executed every time a message is received. The received data will be in JSON format, and we have used the JSON library to parse JSON data and convert it to string

and then to write it to an excel file, as shown in Figure 3.26. For saving in excel file, the essential file functions of python programming language are used, which is already built-in python language.




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | GAS | HUMIDITY | PRESSURE | TEMPERATURE | ECO2 | TVOC |
| 2 | 5.03 | 36.4 | 1007.44 | 25.47 | 18 | 7 |
| 3 | 5.02 | 36.37 | 1007.44 | 25.47 | 15 | 7 |
| 4 | 5.05 | 36.36 | 1007.46 | 25.46 | 15 | 7 |
| 5 | 5.06 | 36.37 | 1007.44 | 25.46 | 20 | 8 |
| 6 | 5.06 | 36.37 | 1007.44 | 25.46 | 20 | 8 |
| 7 | 5.05 | 36.6 | 1007.44 | 25.46 | 7 | 10 |
| 8 | 5.08 | 36.56 | 1007.44 | 25.47 | 42 | 10 |
| 9 | 5.09 | 36.43 | 1007.44 | 25.48 | 70 | 10 |
| 10 | 5.09 | 36.35 | 1007.44 | 25.49 | 20 | 8 |
| 11 | 5.09 | 36.32 | 1007.46 | 25.49 | 27 | 9 |
| 12 | 5.1 | 36.29 | 1007.46 | 25.5 | 20 | 8 |
| 13 | 5.11 | 36.23 | 1007.44 | 25.5 | 15 | 7 |

**Figure3.26: Data logging in .csv file**

There are some functions in python code which are defined as:

- Write data: This function reads data from MQTT and writes it in an excel file.
- On connect: This function is executed when the server is connected with the MQTT broker.
- On Message: This function is executed when the message is received, this function will call write data function on receiving a message.

# Chapter 4 : System Testing

This chapter focuses on the extensive testing of the entire system and evaluates the performance. This test is categorized into four subtests:

- Energy harvesting testing
- LoRa range testing
- Power performance evaluation testing
- Visualization of data on android application

The designed wireless IoT node senses gas, temperature, humidity, pressure, $CO_2$ level, and a total volatile organic compound. The data from these sensors is uploaded to the network server through the LoRa base station, then MQTT subscription extracts the data from the network server. After the pre-processing of data, it is stored in local storage using the python script, and then the same is displayed on the Android-based mobile app. During the analysis of specifically collected data, it is easy to export corresponding data from storage and then analyze it with related data analyzing software and represent using various charts, which will be presented in the following sections. The sensors of this project have been calibrated before the first test. The output data from both sensors is compared for evaluation purposes, i.e., if output data or measured data of both sensors are the same, then it means sensors are functioning optimally. If data collected is not the same from both the sensors, then the test needs to be terminated to find the cause of the malfunction. In this project, the units of measurement of data are shown in Table 10.

**Table 10: Units of measurement of data**

| Data | Unit |
|---|---|
| Temperature | ℃ |
| Humidity | % |
| Pressure | hPa |
| Gas | KΩ |
| $eCO_2$ | ppm |
| TVOC | ppb |

## 4.1 Energy Harvesting Test

The energy harvesting circuit, along with solar panel and 2000mAh rechargeable battery, is tested for various configurations. The system is tested in a room with multiple windows and a desk LED light ON all the time. A Mooshimeter is used to measure the current consumption, as shown in Figure 4.27. A Mooshimeter can measure and log the current and voltage. It is a very portable app-controlled multimeter with full name (BLE).

### 4.1.1 Battery and Solar Panel Inside Room Light

The 2000mAh Li-Po battery is tested along with the 3.5W rated solar panel. Figure 4.29 depicts the resulting values for approximately 7 hours, where the green line represents power in Watt with a running average of 400 measurements, and the orange line represents voltage. The average power while charging the battery is measured to be 26.5mW. At this charging rate it takes approximately 270 hours to fully charge the battery. while undertaking the operation was 26.5mW. At this charging rate it takes approximately 270 hours to fully charge the battery. The inside test location is shown in Figure 4.28.

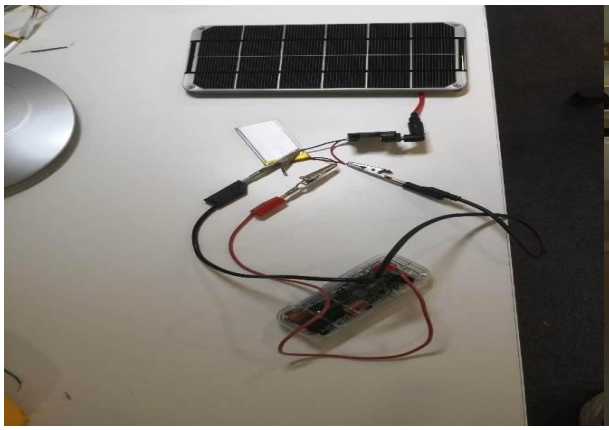


Figure 4.27: charging current/voltage measurement using mooshimeter

Figure 4.28: Inside test location

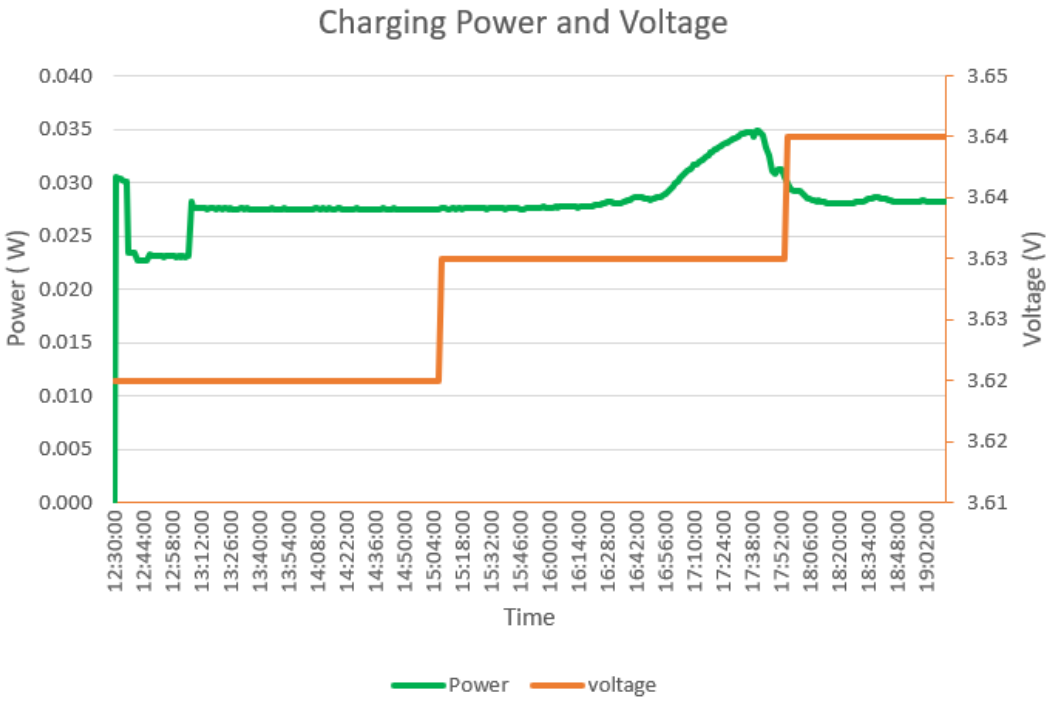



**Figure 4.29: Charging power vs. voltage in indoor condition**

### 4.1.2 Battery and Solar Panel in Full Sunlight

The 2000mAh Li-Po battery is tested along with the 3.5W rated solar panel. Figure 4.31 depicts the resulting values over a period of approximately 30 minutes, where the gray line represents power in Watt with a running average of 180 measurements, and the orange line represents voltage. The average power while charging the battery is measured to be 1.45W. At this charging rate, it takes about 5.5 hours to fully charge the battery. These measurements were taken in full sunlight, as shown in Figure 4.30. At this charging rate, it takes about 5.5 hours to fully charge the battery.

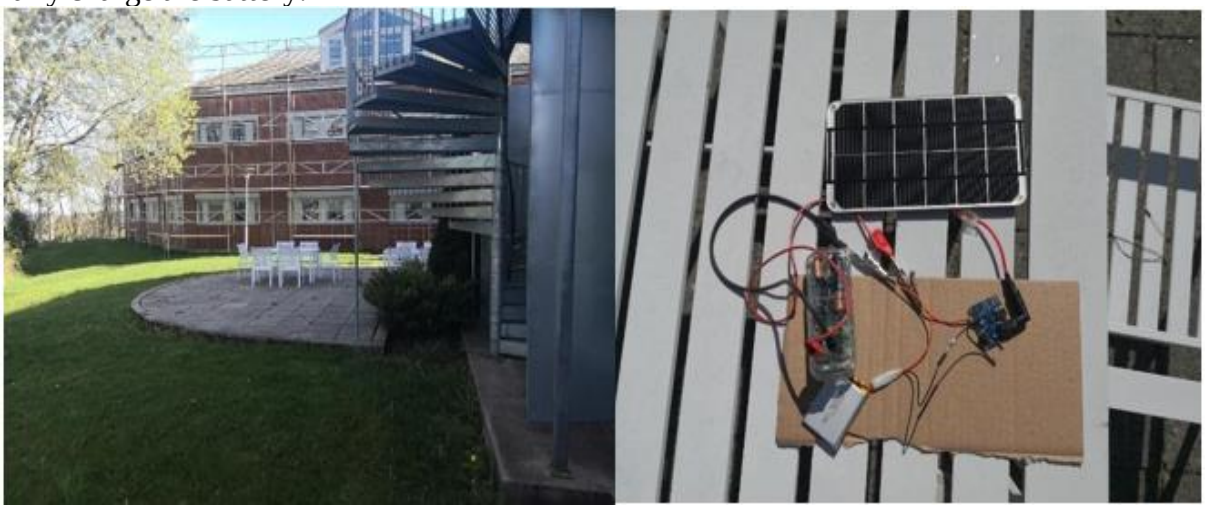


**Figure 4.30: Outdoor location**

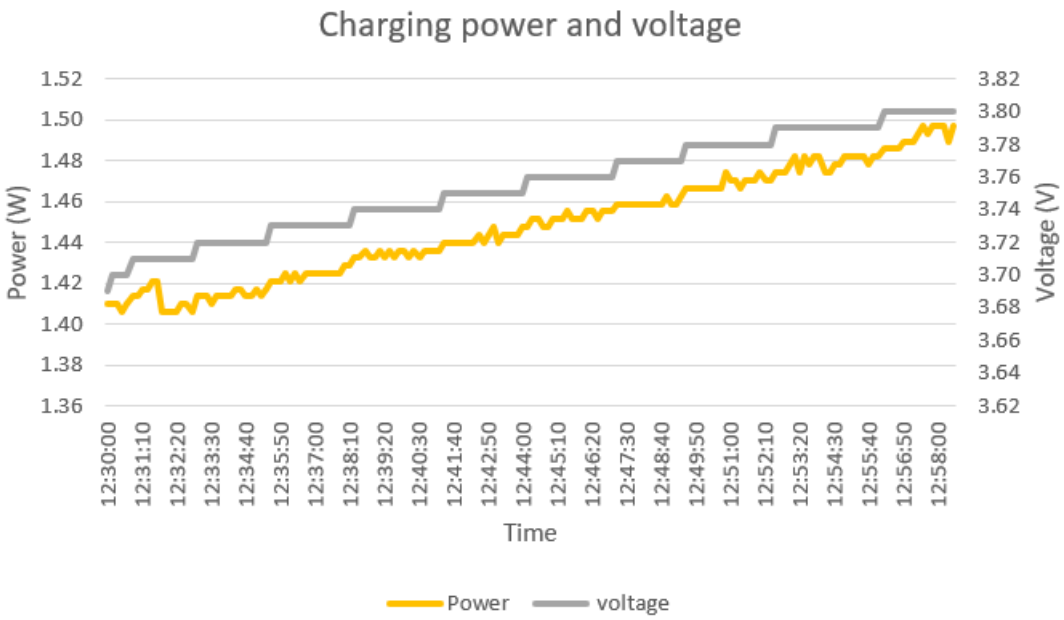



**Figure 4.31: Charging power vs. voltage in outdoor condition**

### 4.1.3 Power consumption for designed wireless IoT LoRa WAN node

The measurements of energy consumption in the ATmega328u4 nodes are conducted using the Mooshimeter [48]. The figure shows the sleep and wakeup current of the ATmega328u4 node with BME680, CCS811 sensors. The sensor node goes to sleep for one minute and wakes up for 100ms to transmit data and sleeps again.

The current consumption of the whole node is measured with the mooshimeter for 30 minutes, during which, the average current was 7mA and the average power was 25.9mW. At this transmission rate, a fully charged battery lasts for approximately 285 hours.

The current consumption of the whole node is measured with the mooshimeter for 30 minutes, during which the average current was 7mA, and the average power was 25.9mW. At this transmission rate, a fully charged battery lasts for approximately 285 hours. Figure 4.32 shows the line graph for the current consumption of the ATmega32u4 node for about 30 minutes.

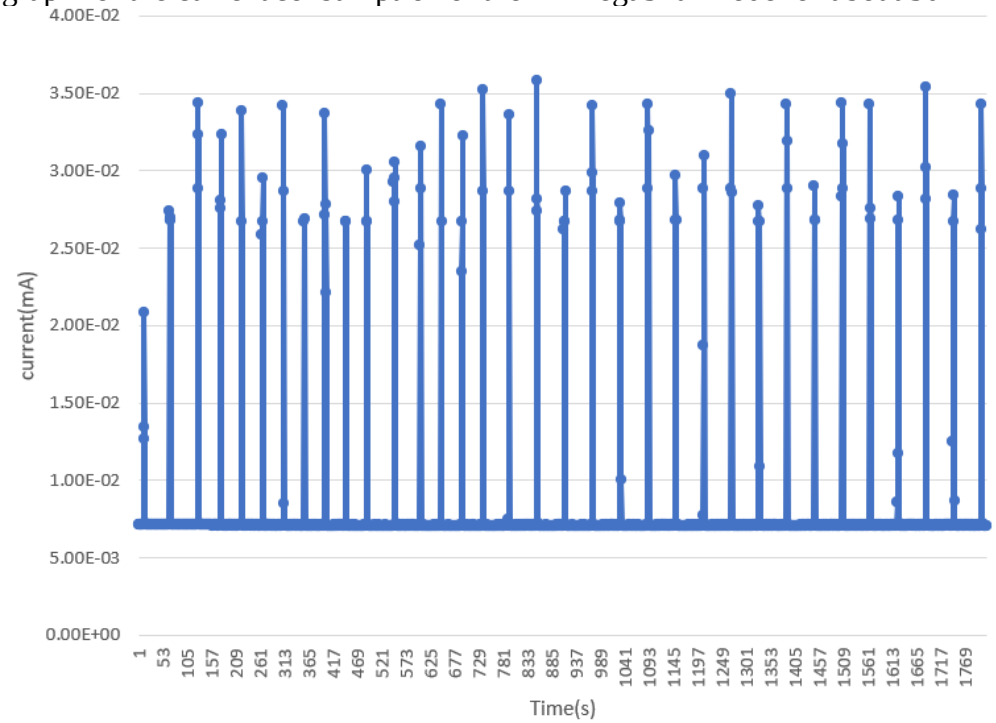



**Figure 4.32: ATmega32u4 sleep current with BME680, CCS811, and RN2483**

The serial monitor of final data transmission was captured and shown in Figure 4.33.

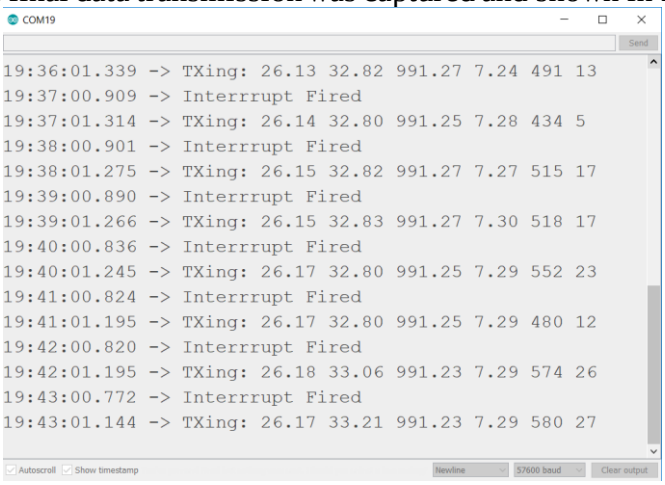



**Figure 4.33: Serial monitor of sensor node while transmitting data and interrupt firing every 1 minute**

## 4.2 Range

The RN2483 range is tested with the manual operation. Achieved range with transmission power set to 14dBm and bitrate to 5470bps, coded with the line of sight, was approximately 900 meters. The test was conducted near the University of Agder Campus Grimstad.

**Figure 4.34: Sensor node placed on the longest distance from the sensor**

**Figure 4.35: Gateway location**

The sensor node was moved manually from a different location to find out the actual range. Figure 4.35 shows the location of the gateway, and Figure 4.34 shows the sensor node location. Figure 4.36 shows the range covered in google maps.

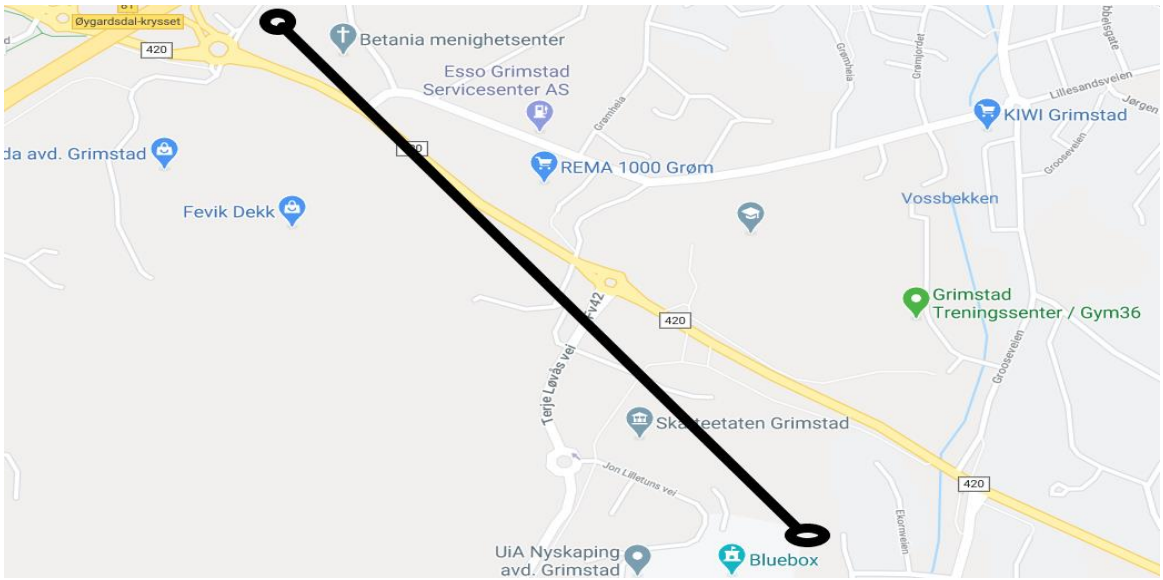



**Figure 4.36: Range covered in google maps**

## 4.3 Air Quality Monitoring of Wisenet Lab

Wisenet Lab is located in a region with one of the the best climate in Norway. It is situated inside the University of Agder, on the first floor of the building. Figure 4.37 shows the specific location of the lab.


**Figure 4.37: Monitoring location**

The lab is not adjacent to the main road. Therefore, the traffic is not dense, and the surrounding environment is also relatively empty because of the park and the parking. And the monitoring location is of an indoor scenario.

The system was sending the measurement sample every 1 minute; each sample includes temperature, humidity, atmospheric pressure, Gas, $CO_2$ level, and concentration of the total volatile organic compound.

Because the number of samples is significant, it is inconvenient to express them on the chart. Still, the variation trends of the concentration of different sensor data within 3 hours can be seen in Figure 4.38.

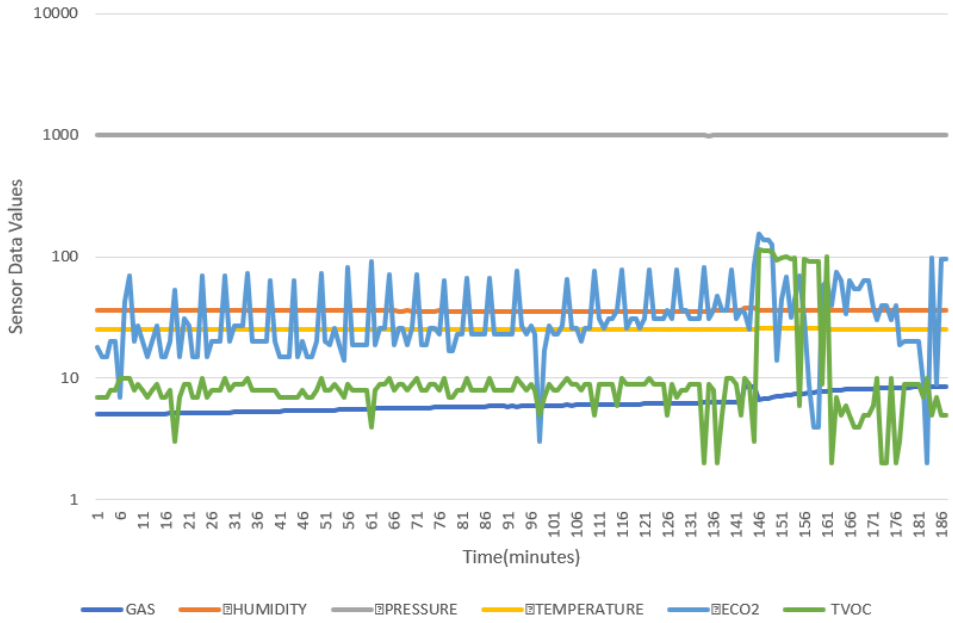



**Figure 4.38: Line chart with normal sample data collected from sensors**

It can be seen that the trends of the data collected from the sensor node are the same. To clearly show the changing trend of environmental data, this section uses the calculation of the average of the sample value for every hour: the maximum value, minimum value, and the average data. The variation trend of equivalent $CO_2$ is shown in Figure 4.39.

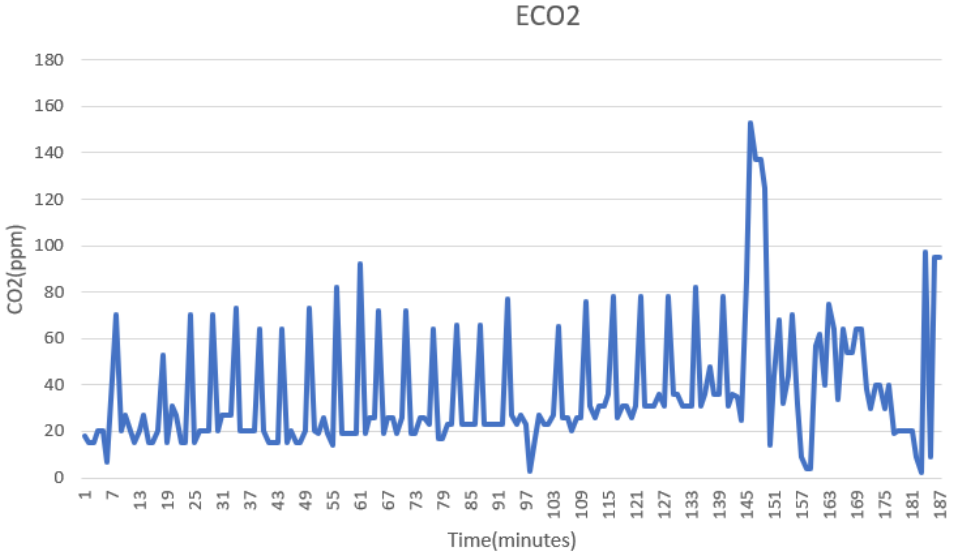



**Figure 4.39: Line chart of $CO_2$ concentration**

The maximum $CO_2$ concentration : 154ppm Minimum : 2ppm Average : 29ppm

The variation trend of total gas concentration is shown in Figure 4.40:

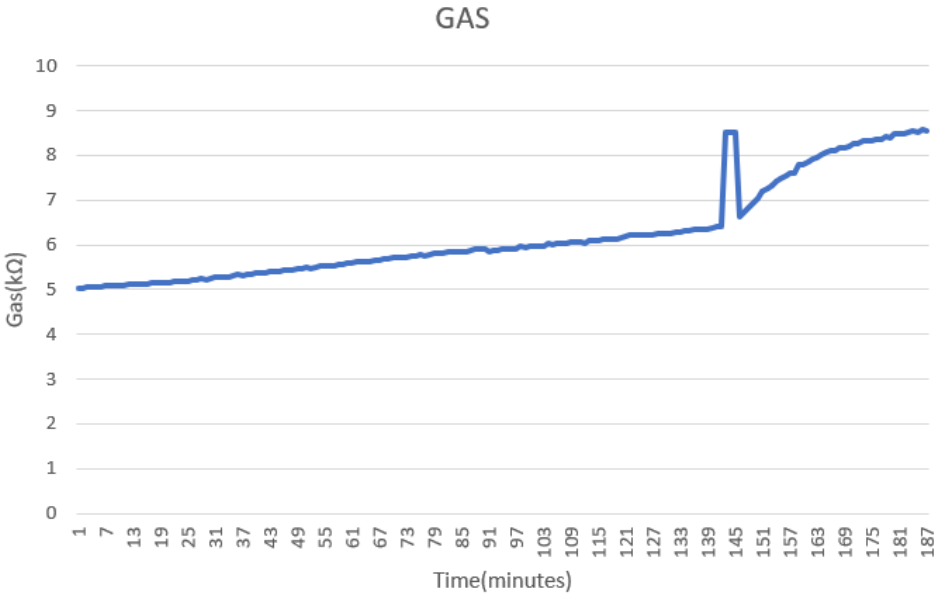



**Figure 4.40: Line chart of the total concentration of gas**

The maximum gas concentration: 8.77KΩ Minimum: 5.0KΩ Average: 7.1KΩ

The variation trend of humidity concentration is shown in Figure 4.41:

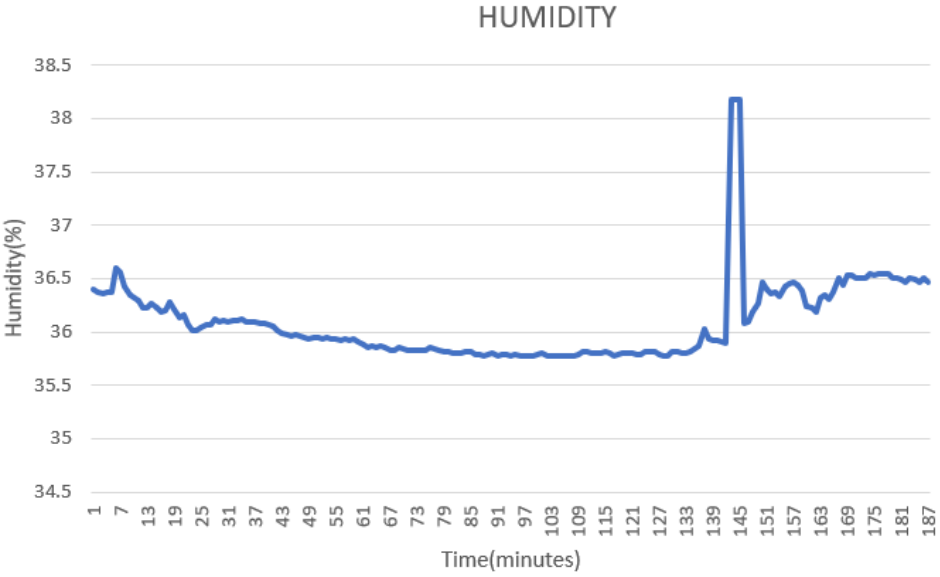



**Figure 4.41: Line chart of humidity concentration**

The maximum humidity concentration: 38.2% Minimum: 35.8% Average : 36.1%
The variation trend of obtained atmospheric pressure is shown in Figure 4.42:

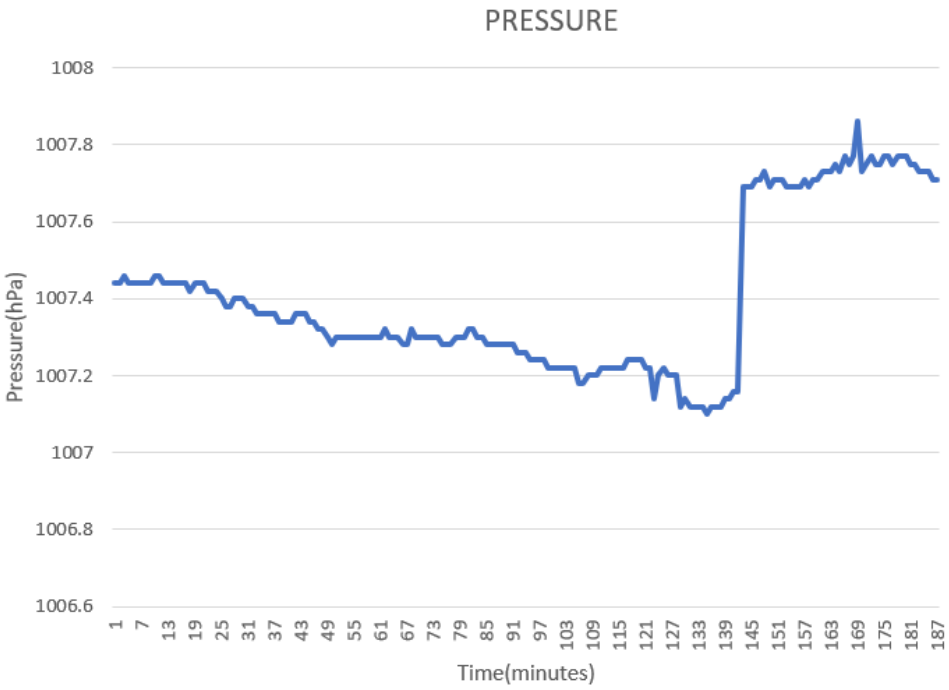



**Figure 4.42: Line chart showing atmospheric pressure**

The maximum atmospheric pressure concentration: 107.8hPa Minimum: 107.1hPa Average: 107.5hPa

The variation trend of temperature data is shown in Figure 4.43:

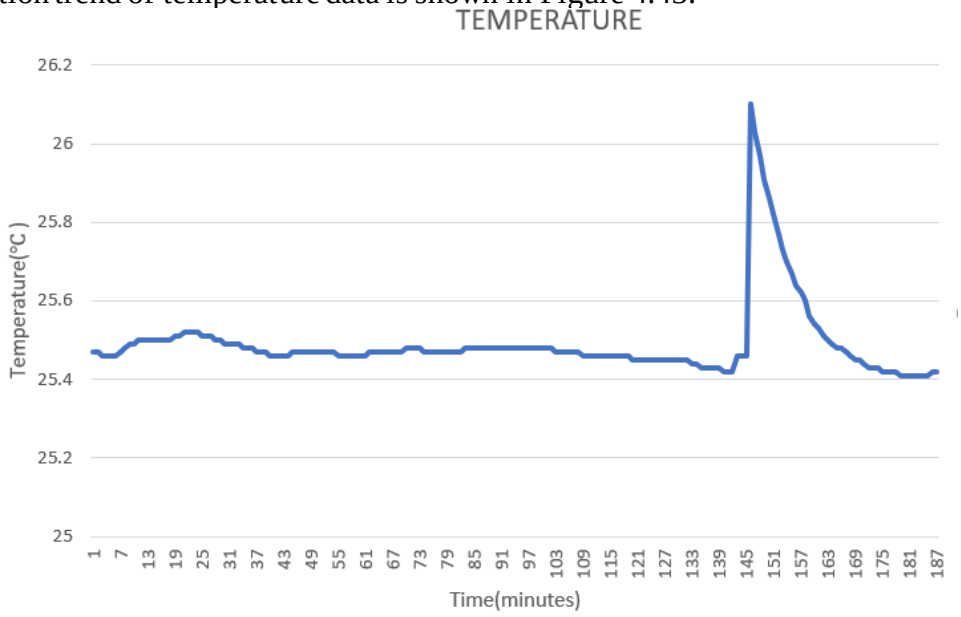



**Figure 4.43: Line chart showing temperature**

The maximum temperature obtained : 26.1 ℃ Minimum : 25.4 ℃ Average : 25.5 ℃

The variation trend of total volatile organic compound is shown in Figure 4.44:

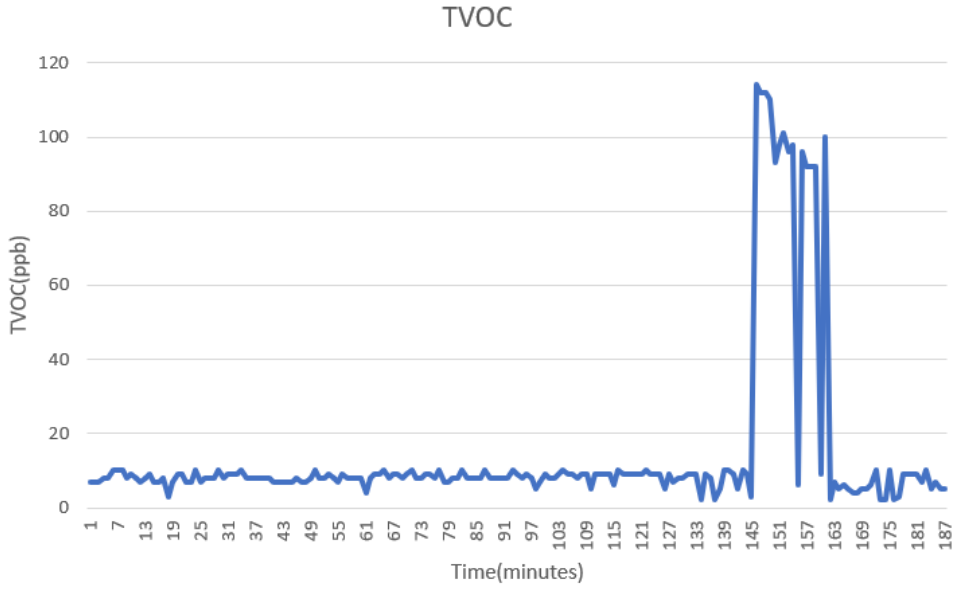



**Figure 4.44: Line chart of concentration of total volatile organic compound**

The maximum of Total volatile organic compound concentration: 116ppb Minimum: 2ppb Average: 10.34ppb

## 4.4 Visualization of Sensor Data Through Android Application

An android application is developed for the visualization of sensor data collected from the wireless sensor network. Figure 4.45 shows the android application where the user can see sensor data displayed.


**Figure 4.45: Android application screenshot**

The work described in this report resulted in a complete application for Android capable of visualizing real-time sensor data retrieved from an MQTT broker.

# Chapter 5 : Conclusion

Design, development, prototype implementation of Self-Powered Long-Range IoT Device based on LoRaWAN is presented. The designed IoT device is completely self-powered based on ambient solar energy harvesting. The developed sensor node is self-sustainable throughout the components' lifespan. For 3.5W rated solar panel used in an indoor room light environment, the developed nodes consume almost equal power with the obtained harvested energy. After making a node sleeping for 60 seconds and wakeup for 100ms to transfer data, the average current consumed was 7mA. This interval can be adjusted based on amount of energy harvested. It can be used for both indoor and outdoor operations. The range of sensor node measured to be approximately 900m while working in low power conditions. The designed node senses the air quality parameters: the level of carbon dioxide, amount of humidity, pressure levels, temperature, and the total organic volatile compounds and gases. It is possible to add several other sensors without modifying the base hardware. The nodes are deployed to measure the indoor air quality at University of Agder campus in Grimstad. Finally, an android application was developed and tested for user interface.

# References


[1] E. E. Agency, "environment, health and quality of life," may 2020. [Online]. Available: https://www.eea.europa.eu/soer/2010/synthesis/synthesis/chapter5.xhtml.

[2] "Dambruoso, P. & de gennaro, Gianluigi & Demarinis, Annamaria & Gilio, A. & Giungato, Pasquale & Marzocca, Annalisa & Mazzone, Antonio & Palmisani, Jolanda & Porcelli, Francesca & Tutino, Maria. (2013). School Air Quality: Pollutants, Monitoring and Toxici".

[3] "Jiang, Chuanjia & Li, Dandan & Li, Jinge & Wang, Juan & Yu, Jiaguo. (2017). Formaldehyde and volatile organic compound (VOC) emissions from particleboard: Identification of odorous compounds and effects of heat treatment. Building and Environment. 117. 10".

[4] "J. M. Samet, M. C. Marbury, and J. D. J. A. R. o. R. D. Spengler, "Health effects and sources of indoor air pollution. Part I," vol. 136, no. 6, pp. 1486-1508, 1987.".

[5] "Jantunen, Matti & De Oliveira Fernandes, Eduardo & Carrer, Paolo & Kephalopoulos, Stylianos. (2011). Promoting actions for healthy indoor air (IAIAQ). 10.2772/61352.".

[6] O. a. S, "Carbon Dioxide Detection and Indoor Air Quality Control," OH&S corporate, 1 april 2016. [Online]. Available: https://ohsonline.com/Articles/2016/04/01/Carbon-Dioxide-Detection-and-Indoor-Air-Quality-Control.aspx?Page=2. [Accessed 14 may 2020].

[7] "A. Erturk, D. J. J. S. m. Inman, and structures, "An experimentally validated bimorph cantilever model for piezoelectric energy harvesting from base excitations," vol. 18, no. 2, p. 025009, 2009.".

[8] "Nema, Pragya & Nema, Rajesh & Rangnekar, Saroj. (2009). A current and future state of art development of hybrid energy system using wind and PV-solar: A review. Renewable and Sustainable Energy Reviews. 13. 2096-2103. 10.1016/j.rser.2008.10.006.".

[9] "J. E. W. P. S. Du, "Solar panel fabrication," ed: Google Patents, 1971.".

[10] "H. J. J. S. Hovel, "Solar cells," vol. 76, p. 20650, 1975".

[11] "B. J. U. R. s. H. C. Schneider, "A guide to LiPo Batteries," 2012".

[12] "R. Ibrahim, T. D. Chung, S. M. Hassan, K. Bingi, and S. J. P. C. S. Salahuddin, "Solar energy harvester for industrial wireless sensor nodes," vol. 105, no. C, pp. 111-118, 2017.".

[13] Adafruit, "Solar charger," Adafruit, [Online]. Available: https://www.adafruit.com/product/390. [Accessed 15 may 2020].

[14] Adafruit, "USB, DC & Solar Lipoly Charger," Adafruit, [Online]. Available: https://learn.adafruit.com/usb-dc-and-solar-lipoly-charger/using-the-charger?view=all#downloads. [Accessed 15 may 2020].

[15] A. INC., "Arduino Micro," Arduino INC., [Online]. Available: https://store.arduino.cc/arduino-micro. [Accessed 15 may 2020].

[16] "J. M. Hughes, Arduino: a technical reference: a handbook for technicians, engineers, and makers. " O'Reilly Media, Inc.", 2016.".

[17] "G. Barbon, M. Margolis, F. Palumbo, F. Raimondi, and N. J. C. C. Weldin, "Taking Arduino to the Internet of Things: The ASIP programming model," vol. 89, pp. 128-140, 2016.".

[18] "S. J. M.-H. E. T. Monk, "Programming Arduino: Getting Started with Sketches, (Tab)," 2016.".

[19] M. INC., "Microchip," [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/50002346c.pdf. [Accessed 15 may 2020].

[20] "G. S. Ramachandran, F. Yang, P. Lawrence, S. Michiels, W. Joosen, and D. Hughes, "μPnP-WAN: Experiences with LoRa and its deployment in DR Congo," in 2017 9th International Conference on Communication Systems and Networks (COMSNETS), 2017, pp. 63-70: IEEE".

[21] "B. J. B. S. Sensortec, "BME680 Low Power Gas, Pressure, Temperature & Humidity Sensor," 2019.".

[22] "J. Jose and T. Sasipraba, "Indoor air quality monitors using IOT sensors and LPWAN," in 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 633-637: IEEE.".

[23] B. INC., "BME680," [Online]. Available: https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf. [Accessed 15 may 2020].

[24] "A. Schütze and T. Sauerwald, "Indoor air quality monitoring," in Advanced Nanomaterials for Inexpensive Gas Microsensors: Elsevier, 2020, pp. 209-234.".

[25] Sparkfun, "CCS811, ultra-low power digital gas sensor," [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf. [Accessed 15 may 2020].

[26] "A. I. Petrariu, A. Lavric, and E. Coca, "LoRaWAN Gateway: Design, Implementation and Testing in Real Environment," in 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2019, pp. 49-53: IEEE.".

[27] "J. de Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino, "LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities," in 2017 2nd International Multidisciplinary Conference on Computer and Energy Science".

[28] RESIoT, "What is LoRaWAN," ResIoT, [Online]. Available: www.resiot.io/en/what-is-lorawan/. [Accessed 23 MAY 2020].

[29] ecsxtal, "LoRaWAN," ecsxtal, [Online]. Available: https://ecsxtal.com/store/pdf/LoRaWAN.pdf. [Accessed 15 may 2020].

[30] lora-alliance, "What is LoRaWAN," lora-alliance, [Online]. Available: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf. [Accessed 23 may 2020].

[31] Circuitbasics, "basic-UART-communication," Circuitbasics, [Online]. Available: www.circuitbasic.com/basic-uart-communication/. [Accessed 23 may 2020].

[32] Circuitbasics, "Basics-of-the-i2c-communication-protocol," Circuitbasics, [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed 23 may 2020].

[33] "V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005., 2005, pp. 4".

[34] "B. Chu, "Li-Ion/Li-Poly Battery Charge and System Load Sharing Management Design Guide With MCP73871," 2009.".

[35] M. Integrated, "DS3231," [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf. [Accessed 15 may 2020].

[36] Jchristensen, "DS3232," GITHUB, [Online]. Available: https://github.com/JChristensen/DS3232RTC. [Accessed 24 May 2020].

[37] nongnu, "avr-libc," nongnu, [Online]. Available: https://www.nongnu.org/avr-libc/user-manual/group__avr__sleep.html. [Accessed 24 may 2020].

[38] Adafruit, "Adafruit_BME680," GITHUB, [Online]. Available: https://github.com/adafruit/Adafruit_BME680. [Accessed 24 may 2020].

[39] "D. M. Preethichandra, "Design of a smart indoor air quality monitoring wireless sensor network for assisted living," in 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2013, pp. 1306-1310: IEEE.".

[40] SparkFun, "SparkFunCCS811," GITHUB, [Online]. Available: https://github.com/sparkfun/SparkFun_CCS811_Arduino_Library. [Accessed 24 MAY 2020].

[41] RS-online, "LoRa Quick Start Guide," [Online]. Available: https://docs.rs-online.com/ae65/0900766b815bf8bf.pdf. [Accessed 15 may 2020].

[42] T. T. Network, "Library Usage," TTN INC., [Online]. Available: https://www.thethingsnetwork.org/docs/devices/arduino/usage.html. [Accessed 15 may 2020].

[43] Adafruit, "LoRa Antenna," Adafruit, [Online]. Available: https://www.adafruit.com/product/3340. [Accessed 15 may 2020].

[44] J. M. Hughes, Arduino: A Technical Reference: A Handbook for Technicians, Engineers and Makers, O'REILLY.

[45] T. T. Network, "MQTT," TTN, [Online]. Available: https://www.thethingsnetwork.org/docs/applications/mqtt/. [Accessed 15 MAY 2020].

[46] icraggs, "paho.mqtt.c," Github, [Online]. Available: https://github.com/eclipse/paho.mqtt.c. [Accessed 15 may 2020].

[47] "M. F. J. J. M. G. M. Sanner, "Python: a programming language for software integration and development," vol. 17, no. 1, pp. 57-61, 1999.".

[48] "R. A. Kjellby et al., "Design, Development and Deployment of Low-Cost Short-Range Self-Powered Wireless IoT Devices," in 2018 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS), 2018, pp. 104-107: IEEE.".

[49] "V. Varshney, P. Jha, M. N. Tiwari, D. J. P. o. t. t. I. N. D. B. V. s. I. o. C. A. Gupta, and Management, "Solar Powered Smart Bag," 2018".

[50] Libelium, "Waspmote LoRaWAN," [Online]. Available: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf. [Accessed 15 may 2020].

[51] T. T. Network, "Gateway connection to TTN," TTN, [Online]. Available: https://www.thethingsnetwork.org/docs/gateways/start/connection.html. [Accessed 15 may 2020].

[52] "U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks," in 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), 2008, pp. 791-798: IEE".

# Appendix

## A. Arduino Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <avr/sleep.h>//this AVR library contains the methods that controls the sleep modes
#include <rn2xx3.h>
#define ccs_811MAC 1
#define realTimeClock 1
#define softwareSerialRn 1


#if (ccs_811MAC)
#include "SparkFunCCS811.h"
#define CCS811_ADDR 0x5B //Default I2C Address
#if (realTimeClock)
#include <DS3232RTC.h>      // https://github.com/JChristensen/DS3232RTC
#define interruptPin 7 //Pin we are going to use to wake up the Arduino
bool sleep_status  = false;
#endif
CCS811 mySensor(CCS811_ADDR);
#endif
//#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME680 bme; // I2C
#if (softwareSerialRn)
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

//create an instance of the rn2xx3 library,
//giving the software serial as port to use
rn2xx3 myLora(mySerial);
#else
//create an instance of the rn2483 library, using the given Serial port
rn2xx3 myLora(Serial1);
#endif

char temp[6];
char humidity[6];
char pressure[8];
char gas_resistance[6];

uint16_t eco2;
uint16_t tvoc;
char buff[45];

// the setup routine runs once when you press reset:
void setup()
```

```
{

  // Open serial communications and wait for port to open:
  Serial.begin(57600); //serial port to computer
  mySerial1.begin(9600); //serial port to radio

  initialize_radio();
  #if (realTimeClock)
  pinMode(interruptPin, INPUT_PULLUP); //Set pin d7 to input using the buildin pull up resistor
  initializeRTC();
#endif
  if (!bme.begin()) {
    Serial.println("BME680 Failed");
  }
#if (ccs_811MAC)
  if (mySensor.begin() == false)
  {
    Serial.print("CCS811 error");
  }
#endif
  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
  //transmit a startup message
  myLora.tx("TTN Mapper on TTN node");

  //  led_off();
  delay(2000);
}
#if (softwareSerialRn)
void initialize_radio()
{
  //reset rn2483
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  delay(500);
  digitalWrite(12, HIGH);

  delay(100); //wait for the RN2xx3's startup message
  mySerial.flush();

  //Autobaud the rn2483 module to 9600. The default would otherwise be 57600.
  myLora.autobaud();

  //check communication with radio
        String hweui = myLora.hweui();
        while (hweui.length() != 16)
        {
          Serial.println("Communication with RN2xx3 unsuccessful. Power cycle the board.");
          Serial.println(hweui);
          delay(10000);
          hweui = myLora.hweui();
        }

        //print out the HWEUI so that we can register it via ttnctl
        Serial.println("When using OTAA, register this DevEUI: ");
        Serial.println(myLora.hweui());
        Serial.println("RN2xx3 firmware version:");
        Serial.println(myLora.sysver());

```

```
115.          //configure your keys and join the network
116.          Serial.println("Trying to join TTN");
117.          bool join_result = false;
118.          /*
119.             ABP: initABP(String addr, String AppSKey, String NwkSKey);
120.             Paste the example code from the TTN console here:
121.          */
122.          const char *devAddr = "26011659";
123.          const char *nwkSKey = "FA6BF38E66DFFE78477EFC000B659073";
124.          const char *appSKey = "E7E91F531FB5742E9944A4EFC7EF4FE2";
125.
126.          join_result = myLora.initABP(devAddr, appSKey, nwkSKey);
127.
128.          while (!join_result)
129.          {
130.            Serial.println("Unable to join TTN");
131.            delay(60000); //delay a minute before retry
132.            join_result = myLora.init();
133.          }
134.          Serial.println("Successfully joined TTN");
135.        }
136.
137.        #else
138.        void initialize_radio()
139.        {
140.          delay(100); //wait for the RN2xx3's startup message
141.          Serial1.flush();
142.
143.          //print out the HWEUI so that we can register it via ttnctl
144.          String hweui = myLora.hweui();
145.          while (hweui.length() != 16)
146.          {
147.            Serial.println("CRN2xx3 Fail");
148.            delay(10000);
149.            hweui = myLora.hweui();
150.          }
151.          Serial.println("DevEUI: ");
152.          Serial.println(hweui);
153.          Serial.println("RN2xx3 FW ver:");
154.          Serial.println(myLora.sysver());
155.
156.          //configure your keys and join the network
157.          Serial.println("Trying to join TTN");
158.          bool join_result = false;
159.
160.          //ABP: initABP(String addr, String AppSKey, String NwkSKey);
161.          join_result = myLora.initABP("26011659", "E7E91F531FB5742E9944A4EFC7EF4FE2
    ", "FA6BF38E66DFFE78477EFC000B659073");
162.
163.
164.          while (!join_result)
165.          {
166.
167.            Serial.println("Unable to join. Are your keys correct, and do you have T
    TN coverage?");
168.            delay(60000); //delay a minute before retry
169.            join_result = myLora.init();
170.          }
171.          Serial.println("Successfully joined TTN");
172.
173.        }
174.        #endif
175.
176.
177.        // the loop routine runs over and over again forever:
```

```
178.       void loop()
179.       {
180.         if (! bme.performReading()) {
181.           Serial.println("BMEFailed");
182.           //    return;
183.         }
184.         dtostrf(bme.temperature, 2, 2, temp);
185.       #if _debug
186.         Serial.print("Temperature = ");
187.         Serial.print(temp);
188.         Serial.println(" *C");
189.       #endif
190.         dtostrf(bme.pressure / 100, 4, 2, pressure);
191.       #if _debug
192.         Serial.print("Pressure = ");
193.         Serial.print(pressure);
194.         Serial.println(" hPa");
195.       #endif
196.         dtostrf(bme.humidity, 2, 2, humidity);
197.       #if _debug
198.         Serial.print("Humidity = ");
199.         Serial.print(humidity);
200.         Serial.println(" %RH");
201.       #endif
202.         dtostrf(bme.gas_resistance / 1000, 3, 2, gas_resistance);
203.       #if _debug
204.         Serial.print("Gas = ");
205.         Serial.print(gas_resistance);
206.         Serial.println(" KOhms");
207.       #endif
208.
209.       #if  (ccs_811MAC)
210.         if (mySensor.dataAvailable())
211.         {
212.           mySensor.readAlgorithmResults();
213.           eco2 = mySensor.getCO2();
214.           tvoc = mySensor.getTVOC();
215.       #if _debug
216.           Serial.print("CO2[");
217.           //Returns calculated CO2 reading
218.
219.           Serial.print(eco2);
220.           Serial.print("] tVOC[");
221.           //Returns calculated TVOC reading
222.
223.           Serial.print(tvoc);
224.           Serial.print("] millis[");
225.           //Display the time since program start
226.           Serial.print(millis());
227.           Serial.print("]");
228.           Serial.println();
229.       #endif
230.         }
231.       #endif
232.         sprintf(buff, "%s %s %s %s %d %d", temp, humidity, pressure, gas_resistanc
    e, eco2, tvoc);
233.         //  led_on();
234.         Serial.print("TXing: ");
235.         Serial.println(buff);
236.         myLora.tx(buff); //one byte, blocking function
237.         //  delay(2000);  //ok
238.         #if (realTimeClock)
239.       Going_To_Sleep();
240.       #endif
241.
```

```
242.        }
243.        #if (realTimeClock)
244.        void initializeRTC()
245.        {
246.          // initialize the alarms to known values, clear the alarm flags, clear the
    alarm interrupt flags
247.          RTC.setAlarm(ALM2_MATCH_DATE, 0, 0, 0, 1);
248.          RTC.alarm(ALARM_2);
249.          RTC.alarmInterrupt(ALARM_2, false);
250.          RTC.squareWave(SQWAVE_NONE);
251.          RTC.setAlarm(ALM2_EVERY_MINUTE, 0, 0, 0, 0);
252.          // clear the alarm flag
253.          RTC.alarm(ALARM_2);
254.          RTC.squareWave(SQWAVE_NONE);
255.          // enable interrupt output for Alarm 1
256.          RTC.alarmInterrupt(ALARM_2, true);
257.
258.        }
259.        void wakeUp() {
260.          sleep_status = false;
261.          Serial1.println("Interrrupt Fired");//Print message to serial monitor
262.          sleep_disable();//Disable sleep mode
263.          detachInterrupt(digitalPinToInterrupt(interruptPin)); //Removes the interr
    upt from pin 2;
264.        }
265.        void Going_To_Sleep() {
266.          sleep_status = true;
267.          sleep_enable();//Enabling sleep mode
268.          attachInterrupt(digitalPinToInterrupt(interruptPin), wakeUp, FALLING);//at
    taching a interrupt to pin d7
269.          set_sleep_mode(SLEEP_MODE_PWR_DOWN);//Setting the sleep mode, in our case
    full sleep
270.          sleep_cpu();//activating sleep mode
271.          RTC.setAlarm(ALM2_EVERY_MINUTE, 0, 0, 0, 0);
272.          // clear the alarm flag
273.          RTC.alarm(ALARM_2);
274.        }
275.        #endif
```

# B. Python Script

```
8.  import sys
9.  import json
10. import time
11.
12. import paho.mqtt.client as mqtt
13.
14. THE_BROKER = "eu.thethings.network"
15. THE_TOPIC = "bhusalloramote/devices/lora_mote/up"
16. filename = "./data.csv"
17. # SET HERE THE VALUES OF YOUR APP AND DEVICE:
18. # TTN_USERNAME is the Application ID
19. TTN_USERNAME = "bhusalloramote"
20. # TTN_PASSWORD is the Application Access Key, in the bottom part of the Overview se
    ction of the “Application” window.
21. TTN_PASSWORD = "ttn-account-v2.5WYkZSOz5d2fyixaNvueihH1A0Qy8Owfz3pRt-FF1iw"
22.
23. def writeData(filename, data):
24.
25.     fd = open(filename,'a')
26.     dataToWrite =  str(data)
27.     fd.write(dataToWrite)
28.     fd.close()
```

```
29.
30.
31. # The callback for when the client receives a CONNACK response from the server.
32. def on_connect(client, userdata, flags, rc):
33.     print("Connected to ", client._host, "port: ", client._port)
34.     print("Flags: ", flags, "return code: ", rc)
35.
36.     # Subscribing in on_connect() means that if we lose the connection and
37.     # reconnect then subscriptions will be renewed.
38.     client.subscribe(THE_TOPIC)
39.
40. # The callback for when a PUBLISH message is received from the server.
41. def on_message(client, userdata, msg):
42.
43.     themsg = json.loads(msg.payload.decode("utf-8"))
44.     print(themsg['payload_fields'])
45.     send = ""
46.     for val in themsg['payload_fields'].values():
47.         send+=val+","
48.     send+="\r"
49.     writeData(filename,send)
50. client = mqtt.Client()
51.
52. # Let's see if you inserted the required data
53. if TTN_USERNAME == 'VOID':
54.     print("You must set the values of your app and device first!!")
55.     sys.exit()
56. client.username_pw_set(TTN_USERNAME, password=TTN_PASSWORD)
57. client.on_connect = on_connect
58. client.on_message = on_message
59. client.connect(THE_BROKER, 1883, 60)
60. client.loop_forever()
```

# C. Android Code

```
61. package com.example.airqualitymonitor;
62.
63. import androidx.appcompat.app.AppCompatActivity;
64.
65. import android.os.Bundle;
66. import android.util.Log;
67. import android.widget.TextView;
68. import android.widget.Toast;
69.
70. import org.eclipse.paho.android.service.MqttAndroidClient;
71. import org.eclipse.paho.client.mqttv3.IMqttActionListener;
72. import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
73. import org.eclipse.paho.client.mqttv3.IMqttToken;
74. import org.eclipse.paho.client.mqttv3.MqttCallback;
75. import org.eclipse.paho.client.mqttv3.MqttClient;
76. import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
77. import org.eclipse.paho.client.mqttv3.MqttException;
78. import org.eclipse.paho.client.mqttv3.MqttMessage;
79. import org.json.JSONObject;
80.
81. public class MainActivity extends AppCompatActivity implements MqttCallback {
```

```
82.     public MqttAndroidClient client;
83.     @Override
84.     protected void onCreate(Bundle savedInstanceState) {
85.         super.onCreate(savedInstanceState);
86.         setContentView(R.layout.activity_main);
87.         final String clientId = MqttClient.generateClientId();
88.         client =
89.                 new MqttAndroidClient(this.getApplicationContext(), "tcp://eu.theth
    ings.network:1883",
90.                         clientId);
91.         //setting up mqttclient object, we need to give mqtt broker ip which is 192
    .168.0.59 (raspberry pi ip)
92.         //you can check ip on pi using ifconfig command and 1883 is port
93.         MqttConnectOptions options = new MqttConnectOptions();
94.         options.setUserName("bhusalloramote");
95.         options.setPassword("ttn-account-v2.5WYkZSOz5d2fyixaNvueihH1A0Qy8Owfz3pRt-
    FF1iw".toCharArray());
96.         options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);
97.
98.
99.         try {
100.                    IMqttToken token = client.connect(options);
101.
102.                    token.setActionCallback(new IMqttActionListener() {
103.                        @Override
104.                        public void onSuccess(IMqttToken asyncActionToken) {
105.                            // We are connected
106.                            Log.d("connection", "onSuccess");
107.                            client.setCallback(MainActivity.this);
108.                            final String topic = "bhusalloramote/devices/lora_mote/u
    p";
109.                            int qos = 1;
110.                            try {
111.                                IMqttToken subToken = client.subscribe(topic, qos);
    //subscribing hum for humidity
112.
113.                                subToken.setActionCallback(new IMqttActionListener()
     {
114.                                    @Override
115.                                    public void onSuccess(IMqttToken asyncActionToke
    n) {
116.                                        // successfully subscribed
117.                                        Toast.makeText(MainActivity.this, "Successfu
    lly subscribed to: " + topic, Toast.LENGTH_SHORT).show();
118.
119.                                    }
120.
121.                                    @Override
122.                                    public void onFailure(IMqttToken asyncActionToke
    n,
123.                                                          Throwable exception) {
124.                                        // The subscription could not be performed,
    maybe the user was not
125.                                        // authorized to subscribe on the specified
    topic e.g. using wildcards
126.                                        Toast.makeText(MainActivity.this, "Couldn't
    subscribe to: " + topic, Toast.LENGTH_SHORT).show();
127.
128.                                    }
129.                                });
130.                            } catch (MqttException e) {
131.                                e.printStackTrace();
132.                            }
133.
134.
```

```
135.                             catch (NullPointerException e) {
136.                                 e.printStackTrace();
137.                             }
138.                         }
139.
140.                         @Override
141.                         public void onFailure(IMqttToken asyncActionToken, Throwable
    exception) {
142.                             // Something went wrong e.g. connection timeout or firew
    all problems
143.                             Log.d("Connection", "onFailure");
144.
145.                         }
146.                     });
147.                 } catch (MqttException e) {
148.                     e.printStackTrace();
149.                 }
150.
151.             }
152.
153.             @Override
154.             public void connectionLost(Throwable cause) {
155.
156.             }
157.
158.             @Override
159.             public void messageArrived(String topic, MqttMessage message) throws Exc
    eption {
160.
161.                 /*
162.                  * To test ,publish  "open"/"close" at topic you subscibed app to in
     above .
163.                  * */
164.                 if(topic.equals("bhusalloramote/devices/lora_mote/up")) { //if temp
    value arrives
165.                     String vale = message.toString();
166.                     JSONObject reader = new JSONObject(vale);
167.                     JSONObject sys  = reader.getJSONObject("payload_fields");
168.
169.
170.                     TextView temp = (TextView) findViewById(R.id.temperature);
171.                     temp.setText(sys.getString("Temperature")); //put temp val on id
     t (text view)
172.                     TextView gas = (TextView) findViewById(R.id.gas);
173.                     gas.setText(sys.getString("Gas"));
174.                     TextView hum = (TextView) findViewById(R.id.humidity);
175.                     hum.setText(sys.getString("Humidity")); //put temp val on id t (
    text view)
176.                     TextView co2 = (TextView) findViewById(R.id.eco2);
177.                     co2.setText(sys.getString("eCO2"));
178.                     TextView pressure = (TextView) findViewById(R.id.pressure);
179.                     pressure.setText(sys.getString("Pressure")); //put temp val on i
    d t (text view)
180.                     TextView VOC = (TextView) findViewById(R.id.tvoc);
181.                     VOC.setText(sys.getString("tVOC"));
182.
183.
184.                 }
185.
186.
187.             }
188.
189.             @Override
190.             public void deliveryComplete(IMqttDeliveryToken token) {
191.
```

```
            }
        }
```

# D. Android Application Layout Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">


    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        app:srcCompat="@drawable/seclogo" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        app:srcCompat="@drawable/wisenetlogo" />
    <TextView

        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:textSize="35sp"
        android:textAlignment="center"
        />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_gravity="center_horizontal"
        android:weightSum="3">
        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"

            android:orientation="vertical"
            android:layout_weight="2">

            <TextView

                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Gas"
                android:textSize="30sp"
                android:paddingLeft="30dp" />

            <TextView

                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
58.                 android:text="Humidity "
59.                 android:textSize="30sp"
60.                 android:paddingLeft="30dp"/>
61.
62.             <TextView
63.
64.                 android:layout_width="match_parent"
65.                 android:layout_height="wrap_content"
66.                 android:text="Pressure"
67.                 android:textSize="30sp"
68.                 android:paddingLeft="30dp"/>
69.
70.
71.
72.             <TextView
73.                 android:layout_width="match_parent"
74.                 android:layout_height="wrap_content"
75.                 android:text="eCO2"
76.                 android:textSize="30sp"
77.                 android:paddingLeft="30dp"/>
78.
79.             <TextView
80.
81.                 android:layout_width="match_parent"
82.                 android:layout_height="wrap_content"
83.                 android:text="tVOC "
84.                 android:textSize="30sp"
85.                 android:paddingLeft="30dp"
86.                 />
87.             <TextView
88.                 android:layout_width="match_parent"
89.                 android:layout_height="wrap_content"
90.                 android:text="Temperature"
91.                 android:textSize="30sp"
92.                 android:paddingLeft="30dp"/>
93.         </LinearLayout>
94.
95.         <LinearLayout
96.             android:layout_width="0dp"
97.             android:layout_height="wrap_content"
98.             android:orientation="vertical"
99.
100.                 android:layout_weight="1">
101.
102.                 <TextView
103.                     android:id="@+id/gas"
104.                     android:layout_width="match_parent"
105.                     android:layout_height="wrap_content"
106.                     android:text="0"
107.                     android:textAlignment="center"
108.
109.                     android:textSize="30sp" />
110.
111.                 <TextView
112.                     android:id="@+id/humidity"
113.                     android:layout_width="match_parent"
114.                     android:layout_height="wrap_content"
115.                     android:text="0"
116.                     android:textAlignment="center"
117.
118.                     android:textSize="30sp" />
119.
120.                 <TextView
121.                     android:id="@+id/pressure"
122.                     android:layout_width="match_parent"
```

```
123.                         android:layout_height="wrap_content"
124.                         android:text="0"
125.                         android:textAlignment="center"
126.
127.                         android:textSize="30sp" />
128.
129.                     <TextView
130.                         android:id="@+id/temperature"
131.                         android:layout_width="match_parent"
132.                         android:layout_height="wrap_content"
133.                         android:text="0"
134.                         android:textAlignment="center"
135.
136.                         android:textSize="30sp" />
137.
138.                     <TextView
139.                         android:id="@+id/eco2"
140.                         android:layout_width="match_parent"
141.                         android:layout_height="wrap_content"
142.                         android:text="0"
143.                         android:textAlignment="center"
144.
145.                         android:textSize="30sp" />
146.
147.                     <TextView
148.                         android:id="@+id/tVOC"
149.                         android:layout_width="match_parent"
150.                         android:layout_height="wrap_content"
151.                         android:text="0"
152.                         android:textAlignment="center"
153.                         android:textSize="30sp" />
154.                 </LinearLayout>
155.             </LinearLayout>
156.         </LinearLayout>
```