

Pramesh Maharjan

# Development of IoT Educational Platform

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Thesis

11 May 2018

Author Title	Pramesh Maharjan Development of IoT Educational Platform
Number of Pages Date	38 pages + 1 appendixes 11 May 2018
Degree	Bachelor of Engineering
Degree Programme	Electronics
Professional Major	
Instructors	Anssi Ikonen, Senior Lecturer
<p>The main goal of the project was to design and implement a sensor platform by designing and building the PCB board and interfacing Particle Photon device with sensors and the XBee.</p> <p>At first, the prototype board was developed, then sensors were embedded, followed by programming the board that supports all the connected sensors. The functional test of board was verified. As the board worked properly, then the platform could be used for testing varieties of other sensors and its further development. The platform was also suited for including ZigBee to support Personal Area Network (PAN). The platform was based on ARM based CPU board with integrated IoT connectivity.</p> <p>Sometimes, we find very frustrating when a lot of jumper wires and sensors are required in a breadboard for various projects. The platform board could be used to get rid of breadboard and jumper wires for various sensor projects. The board consists of analog, digital and I2C ports for connecting the sensor modules conveniently. The platform replaces the frustrating problem of large number of chaotic jumper wires. The board uses Particle Photon as a CPU where all the sensors are interfaced. The particle Photon is connected to the cloud and helps to get access for the projects that we are working, from anywhere. The XBee will be used as a RF module for the platform enabling the device or sensors for wireless communications.</p> <p>The aim of the project was achieved. However, some issues were discovered while soldering the PCB board and hence, it was directed for testing the prototype design on a breadboard.</p>	
Keywords	Sensors, Photon particle, IoT, Temperature, CPU, PCB

## Contents

### List of Abbreviations

1	Introduction	1
2	Theoretical Study	2
2.1	Sensors	2
2.1.1	CO <sub>2</sub> Sensor Module	2
2.1.2	Air Quality Sensor	5
2.1.3	Barometer Sensor Module	7
3	Introduction to the Hardware	8
3.1	Particle Industries	8
3.2	Photon	9
3.2.1	Features	9
3.2.2	Powering Options	10
3.2.3	RF Section	11
3.2.4	Peripherals and PINS	11
3.2.5	Connecting Photon to the cloud	13
3.3	ZigBee	14
3.3.1	ZigBee Architecture	15
3.4	XBEE	16
3.5	Sensor Interfacing Methods	17
3.5.1	Sensor Interface	18
3.5.2	Grove	18
3.5.3	Grove Connectors	19
3.5.4	IIC	20
3.5.5	Data Protocol of IIC	21
4	Development of PCB	22
4.1	Introduction to PCB	22
4.2	Common PCB Issues	23
4.3	Eagle Autodesk	24
4.4	System Design	25
4.5	Schematic Design	26

4.6	PCB Layout	27
4.7	CircuitCAM and BoardMaster	28
5	Development of the Software	29
5.1	Particle Build	30
5.2	Libraries/Functions	30
5.3	General Description	31
6	Results	35
6.1	Design Issues	37
7	Conclusion	38
8	References	39

## Appendices

Appendix 1. Code used for the project

## List of Abbreviations

ADC	Analog to Digital Converter
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DC	Direct Current
DFM	Design for Manufacturability
EMI	Electromagnetic Interference
GND	Ground
GPIO	General purpose Input Output
HVAC	Home Ventilation and Air conditioning
IDE	Internet Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IIC	Inter Integrated Circuit
IoT	Internet of Things
LTE	Long Term Evolution
PCB	Printed Circuit Board
PIR	Passive Infrared
PPM	Parts Per Million
PWM	Pulse Width Modulation

RAM	Random Access Memory
RF	Radio Frequency
RGB	Red Green Blue
RTOS	Real Time Operating System
SCL	Serial Clock
SDA	Serial Data
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

## 1 Introduction

In a simple sense, IoT covers everything that is connected to the internet. The IoT is made up of devices comprising from a simple sensor to the smartphones and wearables which are connected. The IoT is a concept encompassing different things and methods for exchanging information. The concept of IoT has become fashionable that is influencing on how we live and how we work.

Kevin Ashton, in 1999, was the foremost to mention the IoT during his presentation to Procter & Gamble. Most of the traditional fields such as wireless networks, embedded systems, control systems, automation systems and the internet has contributed to the rise of the IoT. By 2020, it is believed that more than half of the new evolving businesses will be conducted on the IoT and various experts predict that the number of IoT devices might range over 30 billion by that time [1]. With the advancement in Digital technology, the IoT is taking a vital place in the life of the generation. The progress in the IoT world has made our life easier, faster and efficient. The IoT world is a giant network for businesses and others, offering us various opportunities for making our life prosperous and productive. Hence, development of IoT has also been a great part of modern education. This need has resulted in the development of educational platform in IoT.

The goal of this project was to design and implement a sensor platform board for interfacing the new Particle Photon device with sensors and the XBee. The first part was designing a Printed Circuit Board(PCB) and interfacing the sensor modules and the CPU on the same PCB board by soldering. Then, the prototype was programmed making the CPU board compatible with the sensors provided. The final part included testing and verifying the functionality of the sensors on the board.

In this project, the theoretical background and the working principles of the devices used will be covered. The CPU board provided by the degree program was Particle Photon and the sensor modules provided were Barometer sensor, CO<sub>2</sub> sensor, and Air quality sensor.

## 2 Theoretical Study

### 2.1 Sensors

Although the present generation of cost efficient and smart sensors are being expanded and verdict their mode and methods into applications, the interface circuits have progressed to comprehend signal amplifying circuits, ADCs (Analog to Digital Converter), bus interface systems and data communication or even information and sensor networking facilities. This has further supplemented to the sensor specific design practices which is being developed for improved accuracy, low power consumption and the intellectual integrated smart behavior capabilities foreseen in the industrialized sensors.

There are many definitions to what a sensor is but it could be defined as a device which detects the input and provides an output with respect to a physical environment. The inputs might be heat, light, sound, temperature, pressure, humidity or any other phenomena from the environment. Thus, the sensors are designed with a view to sense those environmental phenomena, converting them in the form of electronic or optical signals.

Today, we live in a sensor world. Sensors are used almost everywhere. There are various kinds of Sensors at our homes, offices, schools etc. making our lives easier. Speed sensors are used to detect the speed of a car or an object. Smoke sensors are used for controlling the temperature in the industries and used in smoke detector Alarm, Ultrasonic sensors are used for the measurement of distance. Similarly, PIR (Passive Infrared) sensors are used for automatic door opening systems and so on.

The demand for the sensors is growing very rapidly. In the age of science and technology, sensors are making our life safe, easier, faster and smoother. For the Hardware platform, three different sensors were given for the project. The required sensors for the project are discussed in this section.

#### 2.1.1 CO<sub>2</sub> Sensor Module

A carbon dioxide sensor is a type of sensor used for the measurement of Carbon dioxide gas. It works on the principle of Infrared gas sensors and chemical gas sensors. CO<sub>2</sub>



sensors are used in HVAC (Home Automation and Air Conditioning) systems, capnograph devices, industrial and control processes and for monitoring the air quality.

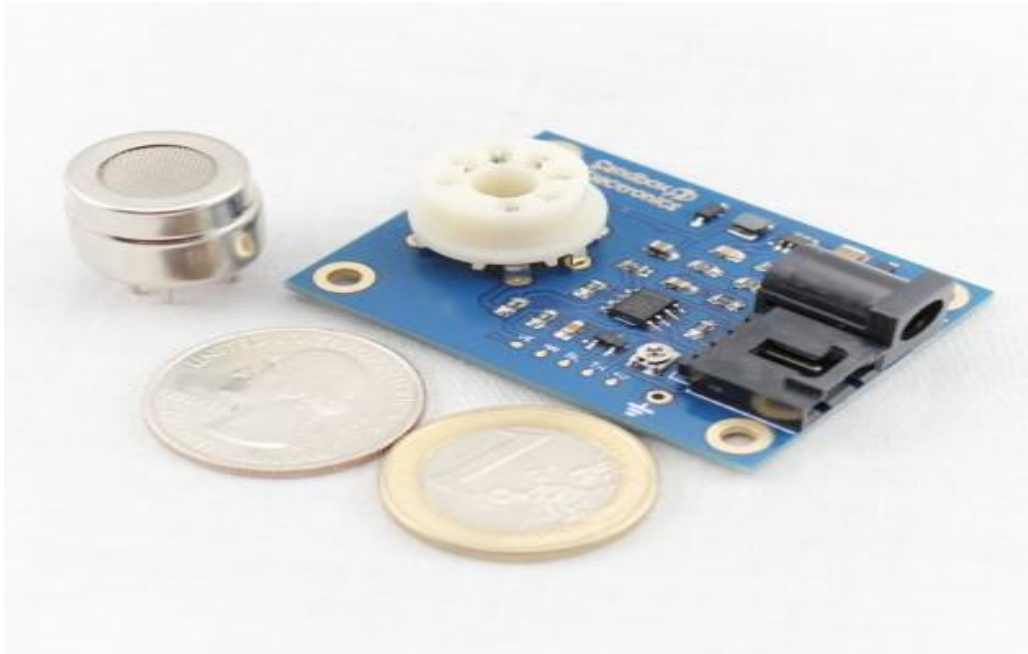


Figure 1. CO<sub>2</sub> Sensor module [2]

The sensor that was used in the project was CO<sub>2</sub> sensor Module produced by Sandbox electronics whose compact size is illustrated in Figure 1. With a MG-811 on board as the sensor component, the sensor module is used in air quality measurement, ferment process and control applications. The MG-811 component senses the amount of CO<sub>2</sub> present in the air and is very sensitive to CO<sub>2</sub> (Carbon di-oxide) gas and less responsive to organic gases like CO (Carbon Mono-oxide). The module has Analog outputs and digital outputs with signal amplifying circuit and heating control circuit as well. This sensor is used for detecting CO<sub>2</sub> concentration at room temperature. [2].

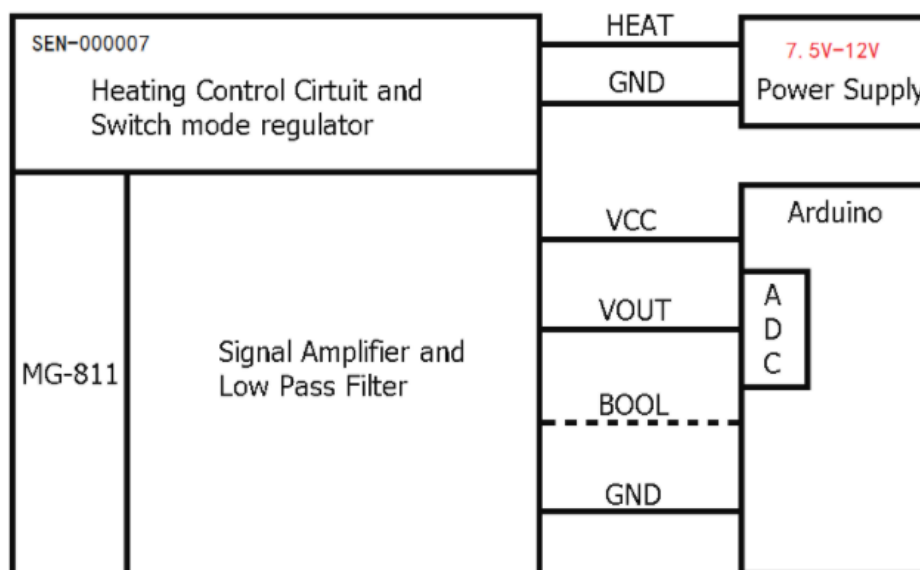


Figure 2. Application schematic for the use of sensor with Arduino [3]

As shown in figure 2, the sensor module has separate places for three components. It has Heating control circuit and switch mode regulator as a first component, Signal amplifier and low pass filter as second component and a MG-811 sensor as a third component. The sensor module looks like a cell with 6 pins with an output signal range of 100-600mV (400-1000ppm CO<sub>2</sub>). Because of this very small output signal, it needs to be amplified for having a clear reading which has resulted in the signal conditioning circuit and the heating circuit cells. The signal conditioning circuit helps in amplifying the output signal, whereas the heating circuit heats the sensor for clear readings. [3].

The module needs a heating power supply(HEAT) of around 7.5V to 12V. The supply voltage(VCC) should be 5V for signal conditioning but should also be less than 5.5V. The module works with a supply current of around 200mA and a heating power of around 1200mW. The operational temperature of the sensor ranges from -20°C to 50°C. The sensor's typical output at 440ppm CO<sub>2</sub> in fresh air ranges from 200mV to 600mV, termed as Zero Point Voltage(V<sub>0</sub>), and also the baseline voltage. As the concentration of the CO<sub>2</sub> increases, the output voltage drops down and vice versa. The concentration of CO<sub>2</sub> can be calculated by:

$$V_s = V_0 + \frac{\Delta V_s}{(\log_{10} 400 - \log_{10} 1000)} * (\log_{10} C_{CO_2} - \log_{10} 400) \quad (1)$$

Where  $\Delta V$  = sensor voltage at 400ppm- sensor voltage at 1000ppm.

$\Delta V$ s, Reaction Voltage is drop voltage of CO<sub>2</sub> concentration seen in between 400ppm to 1000ppm and might be different according to the type of sensor used. The typical Reaction voltage value is about 30mV to 90mV. Proper calibration is required for accurate result of the CO<sub>2</sub> concentration. The signal conditioning circuit has a DC gain of 8.5 which ranges the output voltage(VOUT) to about 0.85V-5V. [3].

### 2.1.2 Air Quality Sensor

It is also known as Air pollution sensor. As the name suggests, the sensor detects the presence of harmful gases such as Carbon monoxide, nitrous oxide, sulfur dioxide, acetone, formaldehyde etc. in the surrounding environment. Air pollution is a serious threat to our health. The measurement of pollution in the air that we breathe is very essential for social awareness with a purpose for creating a clean environment. These sensors can be used for both indoor and outdoor environments. Nowadays, the sensors are even used for Ozone monitoring and control. These sensors are used everywhere, and most widely used in those areas where there is high air pollution.

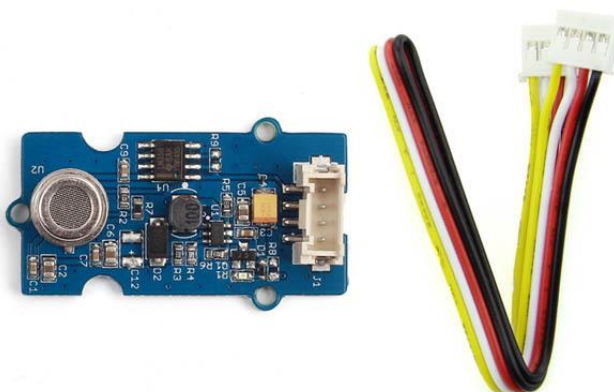


Figure 3. Air quality sensor [4]

One of the most commonly used Air quality sensors on the market, Grove Air Quality Sensor v1.3 is shown in figure 3, which is manufactured by Seedstudio. The sensor has better functioning in describing qualitative results over harmful gases.

Grove air quality sensor module was provided by the degree program for this project. The sensor has compatibility with both 5V and 3.3V and has MP503 gas sensor as a sensing element. The module design is intended for monitoring the indoor air quality and is responsive to many organic gases and highly sensitive to alcohol and smoke. It is used for homes and offices for the detection of harmful gases, automatic exhaust device, air cleaner and so on. The sensor can't specify the output data to describe the concentration of gases quantitatively but can describe the qualitative results and used in applications like car refresher sprayers and car air cycling systems. The sensor features good stability and consumes low power.

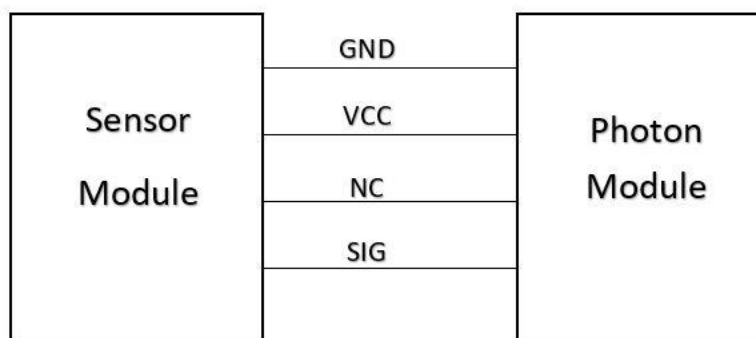


Figure 4. Application Schematic

The general application Schematic of the Air quality sensor is shown in figure 4. It illustrates how the sensor pins get connected to the Photon.

The physical interface module has 4 pin terminal sockets. It has a detection range of about 10-1000ppm for alcohol. The supply voltage needed for the module is either 3.3V or 5V. [4]. 3.3V is used as a power supply from the Particle Photon to the sensor module. The output signal pin is connected to the Analog pin of the photon. There is a pin named NC which is not connected to the photon. The sensor provides qualitative results over the targeted harmful gases. It shows fresh air indicating the environment is without the harmful gases, shows low pollution indicating the low concentration of the harmful gases

and high pollution indicating the high concentration of the harmful gases in that environment.

### 2.1.3 Barometer Sensor Module

A device that detects the pressure in the atmosphere is known as Barometer. The change in atmospheric pressure helps to forecast the changes in the weather. Barometer is used widely for meteorological studies. The barometer sensor is a type of sensor which measures the pressure of liquids or gases. It is even called as pressure sensor as the name itself carries the property of the sensor. Usually, it works as a transducer generating signals in terms of pressure function.



Figure 5. High Accuracy Barometer Sensor [5]

The type of barometer sensor that was given for the project was Grove Barometer(High-Accuracy) from Seedstudio which is shown in figure 5. With a HP206C chip as a sensing component, the sensor has a very wide atmospheric pressure range and used for the detection of atmospheric pressure and temperature. The sensor also detects the altitude so it is even used for measuring high precision altitude above sea level. The sensor has quite wide areas of its application. The sensor is used for Automotive Systems, Weather Station Equipment, Medical Gas Control Systems, Indoor Navigation, Map Assist, Adventure and Sports Watches, Industrial Pressure and Temperature Sensor Systems, Ventilation and Air Conditioning as well.

The sensor used for the project is compatible with two wire I2C interface where 24-bit ADC digitize the Pressure and Temperature sensing outputs resulting the precise data of Pressure and Temperature, whereas there is a patented algorithm to measure the Altitude value which is calculated according to the data output by the pressure and temperature data [6]. Data acquisition is interfaced according to the easy command reading and programmable service interrupt and event control is also available. Figure 6 shows the typical block diagram of the device.

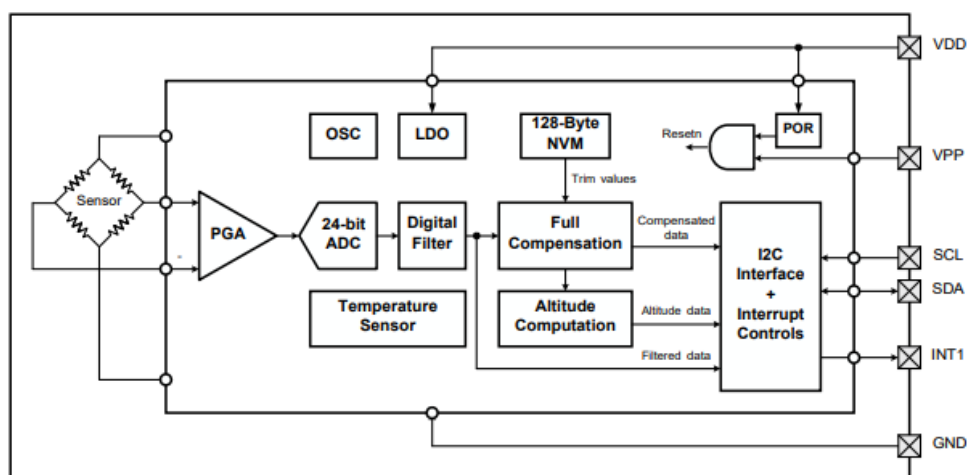


Figure 6. Block diagram of the sensor [6]

The sensor has a very wide range of measuring barometric pressure. The operating pressure ranges from about 300mbar-1200mbar even with high accuracy of about 0.01mbar for ultra-high mode. The operating range of temperature is from -40°C to 85°C. The module is compatible with a supply voltage of either 3.3V or 5V. The high speed I2C bus interface directly connects the microcontroller.

### 3 Introduction to the Hardware

#### 3.1 Particle Industries

Particle is an IoT company formerly known as Spark. Zach Supalla is the founder and CEO of the company. The company is based on both hardware and software for the development of IoT products. In Science, Particle is an extremely small unit within a

larger system. The name of the company Particle is quite familiar with the meaning, resulting in the name for the company and its objective. Hence, Particle is the company providing development kits and services as a very small unit to build the greater whole of IoT. The Particle hardware devices available are Core, Photon, Electron, Argon, The Boron LTE and The Xenon. [7].

## 3.2 Photon

Photon is an open source hardware development kit from Particle Industries with Wi-Fi for prototyping connected products. The module contains an ARM Cortex M3 microcontroller which is very powerful and it is combined along a Wi-Fi chip, which is a very small sized unit. The small sized unit is termed PØ (P-Zero). It is one of the easiest, cheapest and most powerful IoT devices which is very easy to connect to the cloud.

The photon is available in two physical form. The form is either with headers or without the headers.

### 3.2.1 Features

The features of the photon can be listed as follows [8]:

- Particle PØ (P-Zero) module.
  - Broadcom BCM43362 Wi-Fi chip
  - 802.11b/g/n Wi-Fi
  - STM32F205Rgy6 120Mhz ARM Cortex M3
  - 1MB flash, 128KB RAM
- On-Board RGB status LED provided with external drive.
- 18 Mixed-signal GPIO and advanced peripherals.
- Open source design.
- Free RTOS.
- Soft AP setup.

- FCC, CE and IC certified.

Figure 7 shows the overview of the Photon Hardware Module. It is the second board from Particle after Core.

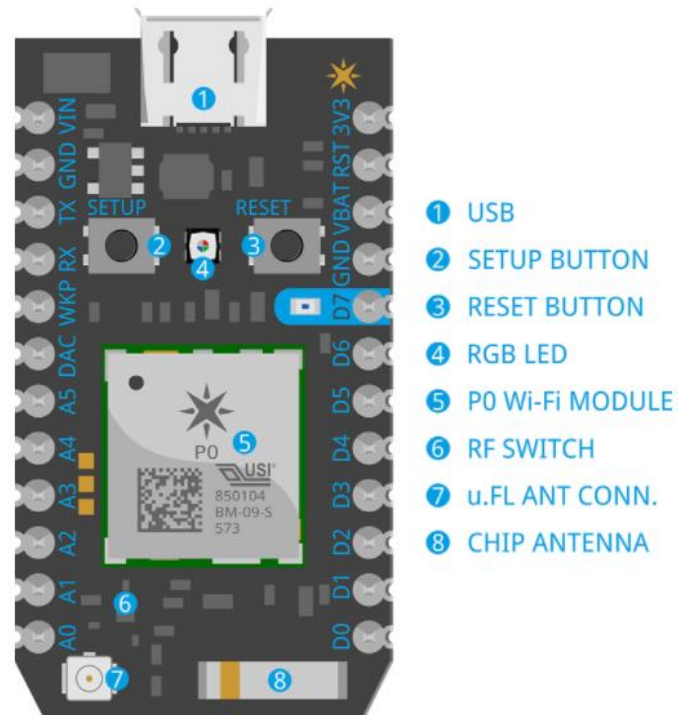


Figure 7. Particle photon device [8]

### 3.2.2 Powering Options

The photon can be powered via two different ways. There is an on-board USB (Universal Serial Bus) Micro B connector for connecting from computer or power source to the micro USB terminal and a VIN pin for the power supply through batteries. The either of the two powering methods can be used. The voltage for powering the device in both methods need to be considered while supplying the power. The voltage should be controlled if the power is directly supplied via the VIN pin. In this method, the regulated voltage should be in between 3.6V DC and 5.5V DC. On the other hand, if the photon is given power through the USB port, VIN outputs approximately a voltage of 4.8V DC. This is because of a Schottky diode (in between VIN and V+ of USB) that protects the reverse polarity. The maximum load of VIN becomes 1A if used as an output. There is another pin named



3V3, which can be applied as the output. However, it has inadequate supply output current (100mA). [8].

At 5V VIN pin, when the Wi-Fi is being used, the device consumes about 80mA of current, whereas the Deep sleep current consumes about 80uA. If the USB connector is used for powering the device, it is advised to practice a good quality cable which helps to limit the Voltage drops in the wire, whereas high resistance cable peaks the currents strained from the Photon while receiving and transmitting, that results input voltage sag. The power supply needs to be quite enough to output 1A current for tolerable overhead current. When the Photon is supplied power via long wires to VIN and USB, great caution should be taken for the protection. It is wise to use the short wirings for the input supply. [8].

### 3.2.3 RF Section

The Photon has a controlled network with finely tuned impedance called RF section. This section helps in optimizing the sensitivity and efficiency for the Wi-Fi communications. The RF section lies on the bottom left side of the Photon. The RF feed line is connected to a SPDT RF-switch from the PØ module and contains several ports in the RF switch and numerous Logic level controls inside it. The Logic level selects connecting ports in RF switch that helps to connect with RF feed line. Each control line consists of 100pF decoupling capacitor. One of the ports is connected to the chip antenna whereas, other one to a u.FL (connector) that helps in adapting external antenna.

### 3.2.4 Peripherals and PINS

The photon device is compatible with numerous interfaces. It has compatibility with Analog, digital and communication interfaces. The interfaces can be listed as shown in figure 8.

Peripheral Type	Qty	Input(I) / Output(O)	FT <sup>[1]</sup> / 3V3 <sup>[2]</sup>
Digital	18	I/O	FT/3V3
Analog (ADC)	8	I	3V3
Analog (DAC)	2	O	3V3
SPI	2	I/O	3V3
I2S	1	I/O	3V3
I2C	1	I/O	FT
CAN	1	I/O	3V3 <sup>[4]</sup>
USB	1	I/O	3V3
PWM	9 <sup>[3]</sup>	O	3V3

Figure 8. Peripherals and GPIO [8]

As illustrated in figure 8, FT is the tolerant voltage pin which is equal to 5V. When the pins are not in analog mode, all the pins are 5V tolerant except A3 and DAC pins. 3V3 pin denotes a 3.3V.

From the figure 7 and 8, it could be seen that the pins from A0 to A7 are the Analog to Digital 12-bit inputs which might also be used as digital GPIOs. The pins from D0 to D7 are only the Digital GPIOs but pins D0, D1, D2 and D3 can be used as PWM output as well. The VIN can be practiced as input pin or output pin. If practiced as input pin, the Photon needs a supply voltage of 3.6-5.5VDC. The maximum load becomes 1A if the VIN is used as an output. RST defines Active low reset input. The device has pull up resistor (1K ohm) that is sandwiched along RST and 3V3, with a capacitor (0.1uF) sandwiched along RST and GND. The pin VBAT denotes the voltage supplied to the internal registers, SRAM and RTC. 3V3 denotes the output voltage of the regulator that is connected internally with the VDD (Wi-Fi module). RX denotes the pin used primarily as UART RX whereas TX is used primarily as UART TX. However, both pins could be practiced as PWM or digital GPIOs. WKP denotes the wakeup which awakes the Photon from sleep. This pin could be practiced as PWM, digital GPIO or ADC input whenever

the pin is not used for Wakeup purpose. DAC denotes the Digital to Analog 12-bit output. [8].

Photon Pin	JTAG	SWD	STM32F205RGY6 Pin	PØ Pin #	PØ Pin Name	Default Internal <sup>[1]</sup>
D7	JTAG_TMS	SWD/SWDIO	PA13	44	MICRO_JTAG_TMS	~40k pull-up
D6	JTAG_TCK	CLK/SWCLK	PA14	40	MICRO_JTAG_TCK	~40k pull-down
D5	JTAG_TDI		PA15	43	MICRO_JTAG_TDI	~40k pull-up
D4	JTAG_TDO		PB3	41	MICRO_JTAG_TDO	Floating
D3	JTAG_TRST		PB4	42	MICRO_JTAG_TRSTN	~40k pull-up
3V3	Power	Power				
GND	Ground	Ground				
RST	Reset	Reset				

Figure 9. JTAG and SWD [8].

As shown in figure 9, the Pin D3 through D7 are JTAG interfaces which are used if re-programming is required for boot loading the Photon or user firmware along the standard JTAG tools. Whenever there is shortage of available pins, the SWD mode may also be used. The SWD mode requires less connections. [8].

### 3.2.5 Connecting Photon to the cloud

Connecting the Particle photon to the cloud is very quick and simple. The photon can be connected to the cloud either by using the web application or by using the smartphone. The method used for connecting the Photon to the internet for the project was done using a smartphone. [9]

First step includes downloading the particle application for the smartphone. Power source to the Photon was given by plugging the USB cable from the laptop. The RGB LED begins blinking blue immediately after connecting the device via USB cable. If not, then the SETUP button should be hold down for about 3 seconds. The next step includes

opening the particle application and either logging in or if, account haven't been made then, signing up is applicable. A plus icon could be seen on the front page of the app and pressing the icon gives the option of adding the Photon device. Then the app will pop up a list of instructions for connecting the device to Wi-Fi. Just following the instructions is better. The photon Wi-Fi will pop up on the Wi-Fi menu of the smartphone settings. This process includes connecting the smartphone to the Photon's Wi-Fi network. The notification indicating that Your Phone has been connected to Particle device will pop up. Tapping the notification will continue Setup. Then after, the photon is connected to the home Wi-Fi network by selecting the using network's name and password.

Once it is connected, giving name to the photon is advisable. At this moment, the RGB LED of the Photon will start breathing blue. That's all, connecting the photon to the internet process is completed and online status with green light indicator could be seen in the particle app. It will take little time to connect but if the instructions have been followed properly as indicated by the app, then there is nothing to worry about. If the Photon is being connected for the first time, RGB Led will blink purple for a while, signifying that it is downloading the updates.

### 3.3 ZigBee

Today, there are a number of available data communication protocols. Some protocols have high data rate whereas some have low data rate. However, they do not meet the communication standard for sensors and control devices. These data communication standards need to have low latency and the energy consumption should also be consistently low, even at the lower bandwidths. To meet the present need for the wireless networks and such devices and systems, ZigBee technology was introduced.

ZigBee is a communication protocol developed by ZigBee Alliance for the invention of communication channel with low cost and low power devices. It is an IEEE 802.15.4 based specification for Wireless Personal Area Network (WPAN) built specially for control and sensor networks for a suit of high level communication. It has a data rate of 250 kbps and operates at the frequencies 868 MHz, 902-928 MHz and 2.4 GHz [10]. It is a transitional two-way data communication of sensors and controllers and best appropriate

for periodic as well. It is deployed as a mesh network with low power for the applications at a range of 10-100 meters.

ZigBee technology is used widely in the present world in various fields such as Home automation, Industrial Automation, Smart Grid Automation and so on.

### 3.3.1 ZigBee Architecture

The structure of ZigBee Network consists of following:

- ZigBee Coordinator

Each ZigBee network contains one coordinator that functions as a bridge and source. The coordinator is in charge for handling and storage of the data/information every time it needs to receive and transmit the operations. It is the most capable device in the system.

- ZigBee router

The router acts as an intermediary device that is responsible for the authorization of data passage over other devices.

- ZigBee End Device

The end devices have restricted functionality for the communication within the parent nodes which helps to save the power of battery.

The quantity of coordinators, routers and end devices in a ZigBee system depends on the type of networks such as cluster, mesh and star network.

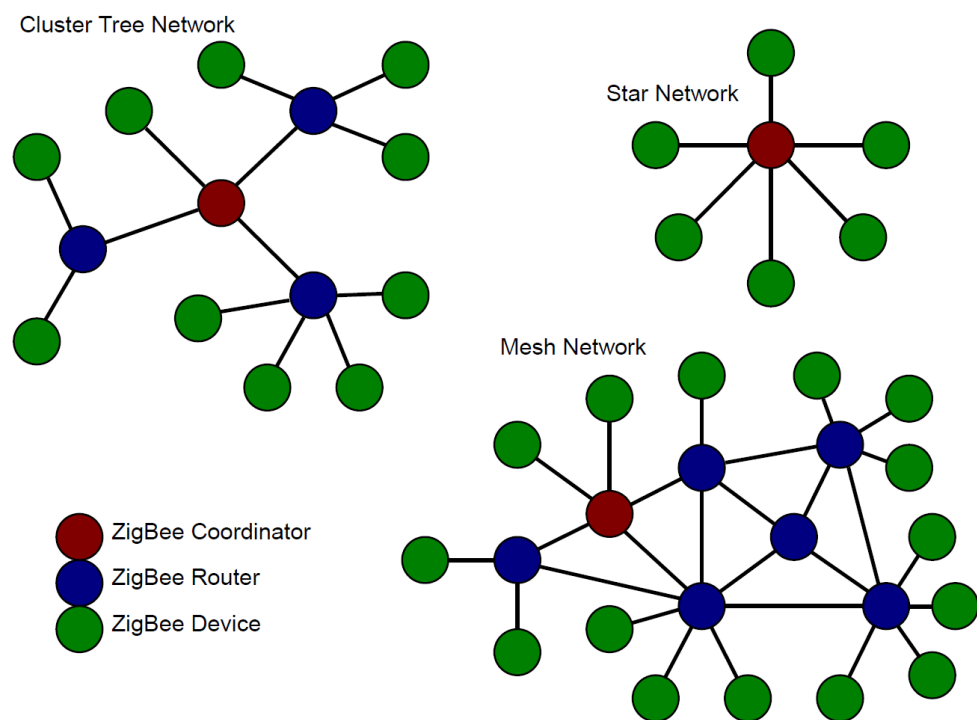


Figure 10. Architecture of ZigBee Network [11]

Figure 10 illustrates the general architecture and topologies of ZigBee Network. The most widely used configurations are Cluster tree, Star and Mesh Network topologies. ZigBee operates in different modes which results for the longer battery life of the system. The networks can be extended with the use of routers and nodes resulting in the connection between each other which creates a wider area network.

### 3.4 XBEE

XBee is a wireless microcontroller radio module which has wireless capability for the connection between devices to communicate with each other. The XBee radios are built and manufactured by Digi International and are engineered encompassing the IEEE 802.15.4 standards [12]. It utilizes ZigBee 802.15.4 protocol for communication between the radios wirelessly. The XBee modules are simple to use and cost efficient with low power. The XBee RF modules are used for Wireless Sensor Networks.

The XBee has a total of 20 pins and an antenna which depends on the version. The modules have various antenna options. It consists of 11 digital pins and 4 analog pins

and requires 3.3V to operate and uses 2.4 GHz of frequency [12]. The XBee radios are available in two forms in general- through hole and surface mount. A simple 1mW XBee transceiver device with wire antennae is shown in figure 11 which is manufactured by Digi International.

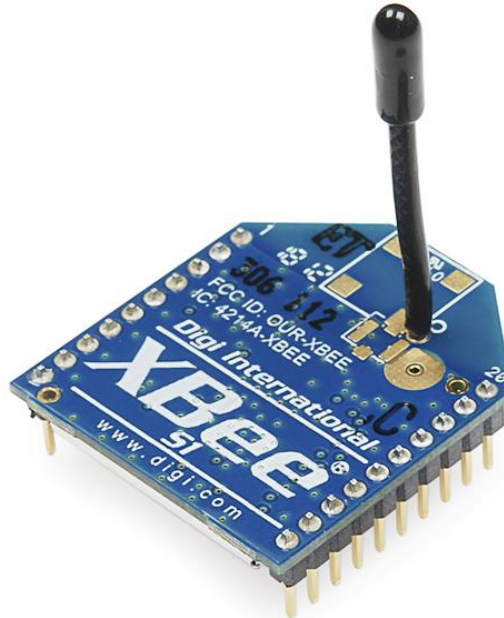


Figure 11. A XBee device [13]

### 3.5 Sensor Interfacing Methods

Interfaces are predefined ways to communicate between the electronic devices. Interfacing is a way to transfer data from one device to another. The data passed through interface may be very simple (open/closed, high/low) or complex (serial/parallel). A simple example can be using a microcontroller at one place to read the data with a temperature sensor. The process needs a communicating medium which is the interfaces. The temperature sensor will have an interface such as SPI which helps to communicate between the devices. The common interfaces used these days are I2C, SPI, RS-232. Microcontrollers have limited I/O. Even the larger ones have the limit for the I/O. To minimize the problems for I/O, interfaces give us a way. Also, most external devices require an interface. For example, EEPROM is not going to hook up with a bunch of individual parallel wires, we need I2C interface to connect it.

### 3.5.1 Sensor Interface

Contradicting to living creatures, systems created by humans do not have brain and are executed through algorithms only. However, in most of the cases, these non-intellectual systems might somehow behave a smart behavior. For instance, most automobiles today have smart systems which might stimulate safety strategies like air bag deploying, collision warning, anti-braking systems, tire pressure monitoring and so on, whenever obligatory. We denote such non-intelligent systems designed by the humans and somehow act intelligent, as smart systems. For an intelligent device, the system must always first, attain information from the surroundings. Then, should determine the proper action to be taken and last of all, act consequently. These obligatory process is clearly a mock of the normal behavior of living creatures.

The essence of interfaces for intelligent systems and in electronic devices could possibly be valued now. Sensors transport the information/data from a non-electrical form into electrical; though, in practice, it is suitable to attain information with digital electronic systems and hence electronic interface in sensors is vital for converting the electrical signals into convenient digital signal responses.

### 3.5.2 Grove

Grove is an interfacing system from Seedstudio which is modular and easy to use. The system has standardized connector for project prototyping allowing us to minimize the effort. Nowadays, there are several choices available for the hardware development environments, but the grove system is safe, easy to use and compatible with most systems available today. This system is preferable over messy jumper cables, which is time consuming and a headache. The system has two parts, the base system also known as stem and the various modules also known as twigs with the standardized connectors. The base system can connect any input or output from the modules. Every grove module performs only one function like a simple alarm button or a more complex heart rate sensor. The available categories for Grove modules manufactured nowadays are Grove modules for Sensors, Actuators, Display, Communication and others.



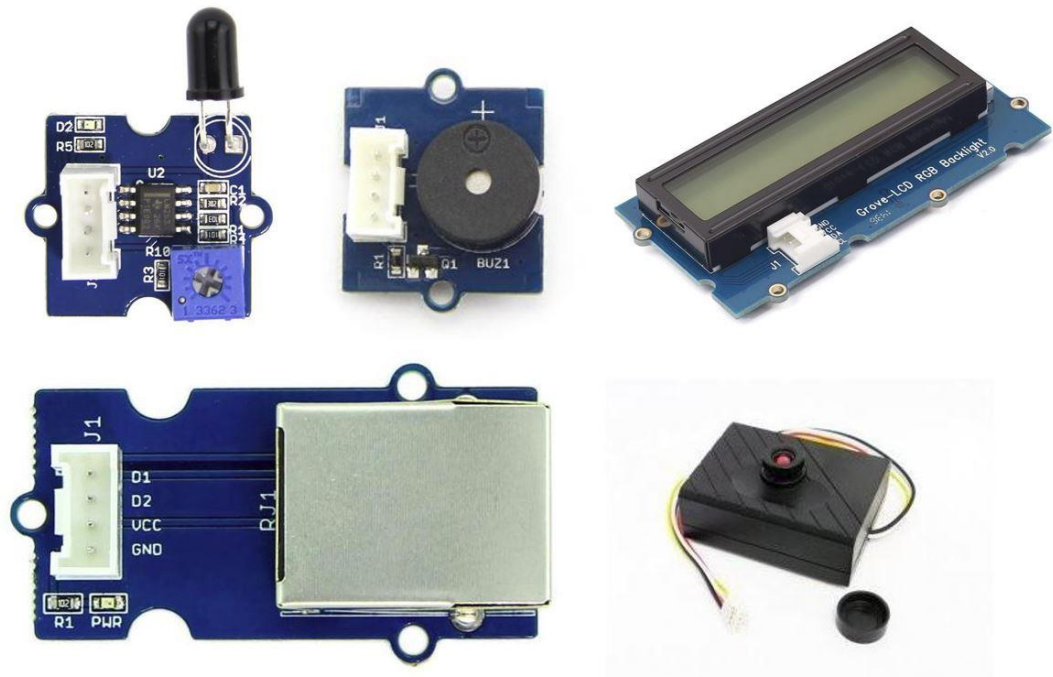


Figure 12. Grove Modules

In the Figure 12, there are some Grove modules manufactured from Seedstudio. The first picture on the first row is a Grove Flame sensor module. It denotes the sensor category of Grove modules available today. The second picture on the first row is a Buzzer denoting the Actuator category of Grove Modules available in the market. The third picture is a Grove LCD with RGB Backlight. It is the grove module for Display Category. The fourth one, which is on the left side of the second row, is a Grove RJ45 Adapter that is used for the conversion of a standard connector into a RJ45 connector. This is a Grove module with Communication category. The last one is a Grove-Serial color camera which is categorized into others Grove module category.

For the connection of the Grove Modules, a base unit is not necessary. We can simply use grove to pin header cable to connect on the platform boards to the Grove connectors.

### 3.5.3 Grove Connectors

It is a four-pin connection plug standardized for the connection between base system and the modules. The connectors are of the exact dimensions as the manufacturer Seedstudio's specifications and are common to all of the Grove modules. The connectors are available in flat 90degree versions and vertical versions. The universal 4-pin buckled

standard connector and the universal 4 pin cable for the grove modules can be seen in figure 13.

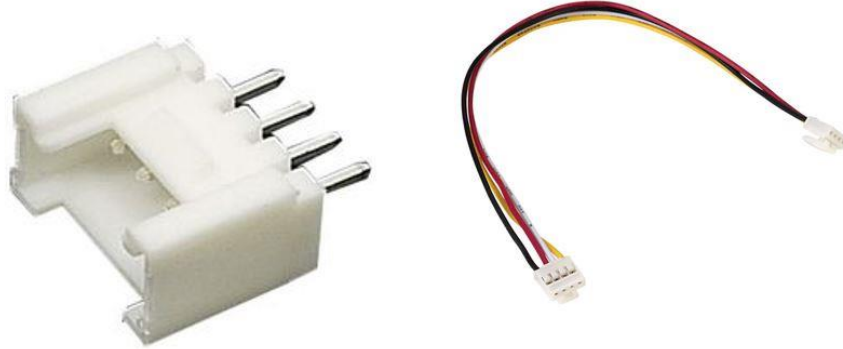


Figure 13. Universal 4-pin Grove Connectors [14]

#### 3.5.4 IIC

IIC, an acronym for Inter Integrated Circuit communications, is a serial bus interface protocol used for the communication between the components residing on the same circuit board. Also abbreviated as I2C, the bus is a short distance protocol invented by Philips Semiconductors in the early '80s for the enhancement of communication between the component devices or the microcontroller devices. The component devices may be Analog-to-digital Converters(ADC), serial EEPROMs, display drivers and so on.

IIC is one of the most popular serial bus communication protocols used today as it could be applied easily in numerous electronic designs that support data exchange between master and slave devices. This implementation requires only two wires or bus lines for the communication between nearly 128 devices while using 7-bit addressing and up to 1024 devices while using 10-bit addressing. This is possible as each device has a unique ID or address which can be found from the data sheet of the device and the master will choose the unique ID or address for communicating with the device. The two signal lines are termed as Serial Clock (SCL) and Serial Data (SDA). The SCL signal is generated by the master device responsible for synchronizing the transfer of data among the devices on the IIC bus. The second one, SDA signal that transports the data.

### 3.5.5 Data Protocol of IIC

The Data protocol for IIC is shown in figure 14. The data signal transfer is carried out in the 8 bits sequence. This first 8 bits sequence takes place as soon as a special start condition occurs. This sequence provides the slave address for sending the data. There is a following process for a bit after an interval of 8 bits pattern/sequence. This bit sequence is known as Acknowledge. In most cases, the addressing sequence occurs after the first acknowledge bit, but now, the acknowledge occurs in slave device for its internal registers. Subsequently, the data sequence takes place, followed by the addressing sequence till the data sending process is complete and finally, ending the process. The ending process occurs on a condition known as special stop condition. If we take a close look at these events, the start condition takes place as soon as the data signal gets low whereas, the clock signal gets still high. Afterwards, the process gets started again and the individual data bit gets transferred to clock pulse separately. [15].

The addressing sequence for the device starts first along the most significant and ends along the least significant bits. These bits are of 7 bits. But, there is 8<sup>th</sup> bit, used for indication purpose. It means the 8<sup>th</sup> bit will check if the master device will write or read to the slave. The states are termed as logic low and logic high respectively. The slave device uses the next bit Acknowledge to specify if the previous bit sequence has been received successfully. During the process, the master device approaches to the slave device for controlling the SDA line and checks if the previous sequences has been received successfully by the slave device, pulling down the SDA line to the condition called Acknowledge. The slave device might not pull the SDA line down and this condition is called Not Acknowledge which means it didn't successfully receive the previous sequence. This might be the consequences of some reasons. To mention few, the slave device might not know the received signal or command, the slave is unable to collect any more signals or the busy slave. If so, the master device is responsible to decide for proceeding. [15].

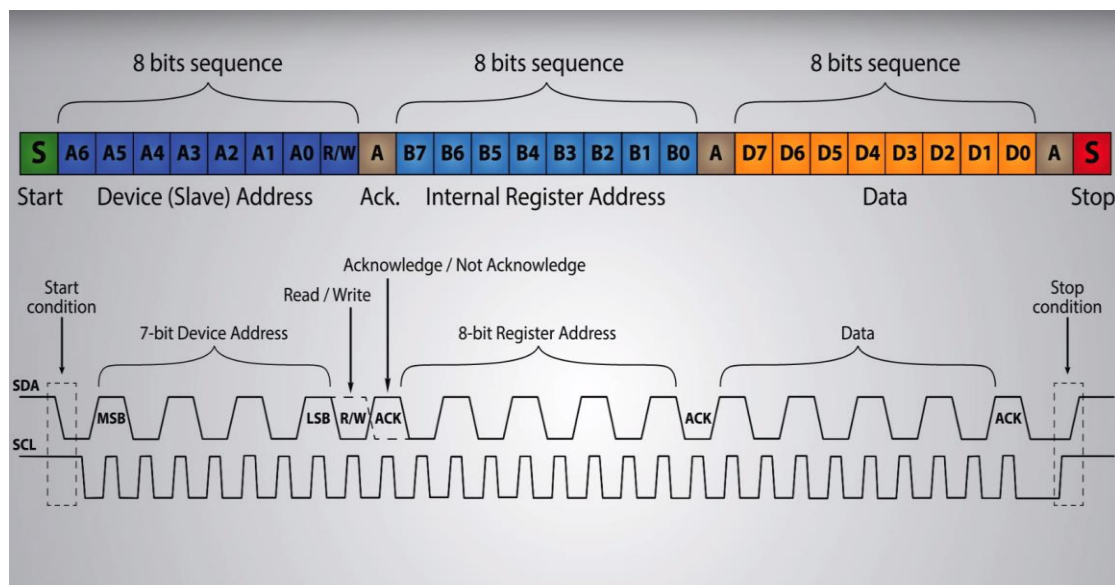


Figure 14. IIC Protocol [16]

The next process is addressing the internal register. The internal registers contain data and information which is located to the memory slave. After this process, the transfer of data sequence begins from the master device or the slave device that depends on the mode selected at Read/Write. When data sending process is complete, the transfer ends following a stop condition that takes place once the SDA signal drives from low to high although the SCL signal is still high. [15]. This is how the data protocol works for IIC which is easily summarized in figure 14.

## 4 Development of PCB

### 4.1 Introduction to PCB

Printed Circuit Board often called as Printed Wiring board, is the foundation of most electronic devices and some electrical devices as well. It is a thin base board made of pads and lines for electrical connection among various electronic components together. The board consist of lines, pads, tracks and etched sheet of copper laminated layers onto or between the flat sheet non-conductive substrate. Generally, the electronic components are then soldered onto the board for electrical connection. The available PCBs are single sided, double sided or multi-layer.

In the past years, electronic circuits were constructed by the process of point-to-point wiring. This process was very laborious and even resulted in the failure on the proper design of the electronic circuits. This traditional process even led to failures at the wire junction, bad insulation, short circuits, bulky, heavy and relatively fragile circuits and so on. As the development of electronics moved from vacuum tubes to silicon and ICs, the electronics industry took a boom resulting to the invention of modern electronic components and the consumer market. This resulted the manufacturers for seeking out the better solutions for the electronic circuits and PCB was born. Thus, development of the modern PCBs started at the early 1900's.

In the modern electronics, PCBs are the most common technique of lumping the circuits. Initially, PCBs were manually designed but not anymore. Today, there are a number of layout software for the design of the modern PCBs. These software uses various automation tools for the electronic designs. Among them, Eagle Autodesk software was used for the design of the PCB in this project.

#### 4.2 Common PCB Issues

Even smart people make stupid mistakes. It is said that doing mistake is a good thing that inspires people to learn from those mistakes. There are a surprising number of mistakes that can arise during PCB design. Some of the known issues that might occur during PCB design are discussed here. [17].

- Electromagnetic interference is one of the flaws that need to be taken into consideration while designing circuit boards which might be reduced by increasing the GND area of the circuit board. Prohibition of 90° angles for the components helps in reducing the EMI interference. Additionally, using shielded cable is sensible for reducing Interference.
- Starved thermals are a hinderance during soldering or when the circuit gets heat stress resulting overheating and heat damage to the PCB. Therefore, proper thermal connections or replacing the thermal is recommended.
- Acute angles should be prohibited in a circuit board as they might make the circuit malfunctioning and root serious problems in future.

- Copper is the most active conductive component used in PCBs whereas, is soft and susceptible to corrosion as well. Insufficient Copper to Edge Clearance causes many problems regarding to the functionality of the PCB board. The exposed copper planes can easily contact another by touching a conductive substantial and might cause short. However, proper spacing between the edges of copper and the board helps to avoid such issues.
- The solder mask in some PCBs might be absent partially or completely between pads. This exposes more copper than necessary and consequences in solder bridges between pins during assembly resulting a short and problems regarding to the functionality of the PCB. This can be avoided by DFM check protocol or double checking the design.

Besides these, Proper DFM checking is a must for minimizing the problems that might occur during Assembly and fabrication process.

#### 4.3 Eagle Autodesk

Eagle is a software tool for Electronic Design Automation developed by the company called Cadsoft computer. It was initially released 30 years ago with numerous features like schematic design, printed circuit board layout, auto router and computer aided manufacturing. The abbreviation of EAGLE is Easily Applicable Graphical Layout Editor. [18]. The software is one of the most popular application for designing the circuit diagrams, creating schematics and extending the schematic to the board and creating Gerber files for the desired PCB designs and for producing the designs into small-scale or large-scale PCB manufacturing industry. Since the release of the software, it has various versions with the newer updates to the software on the newer versions. The latest Eagle version format in the market is 8.6.0 and is compatible on all operating systems like Windows, Mac OS, and Linux. The software is available for both free and premium versions and is compatible for students, individuals, commercial and non-commercials.

#### 4.4 System Design

Initially, the test and design of the prototype was carried out on a breadboard with Arduino and the necessary sensor components with wires. After building the circuit on a breadboard with all the required components, the prototype design was programmed through Arduino compilation. As the design was working properly with the Arduino, the second step was to design the prototype on a breadboard with the Particle Photon. In this process, the PCB design was also taken into consideration as the prototype design was working fine with the Arduino Board which looks similar for the design with Particle Photon in order to complete the project on time.

In the process of designing the prototype on breadboard with Particle Photon, the schematic design of the Prototype was done with Eagles EDA. As soon as the schematic part was finished, the Board Layout of the design was carried out and the necessary Gerber files were created. Then, the milling for the prototype was also carried out. After this, the next step involved finding the required components for the board and to solder them onto the milled PCB Board. For this purpose, the degree program was responsible for providing the necessary components from the designated Manufacturers. While this process was going on, the task of programming the prototype design on the bread board was also going on.

The milled board was then required to be soldered with the required components. Finally, the prototype board was ready but still in need to be tested, connecting all the components and the last step was to test the Board by flashing the code into the Particle Photon. But, the milled board had some design issues which was not applicable for soldering process. Thus, the soldering process for milled board was skipped and the prototype design was tested on a breadboard instead of milled board.

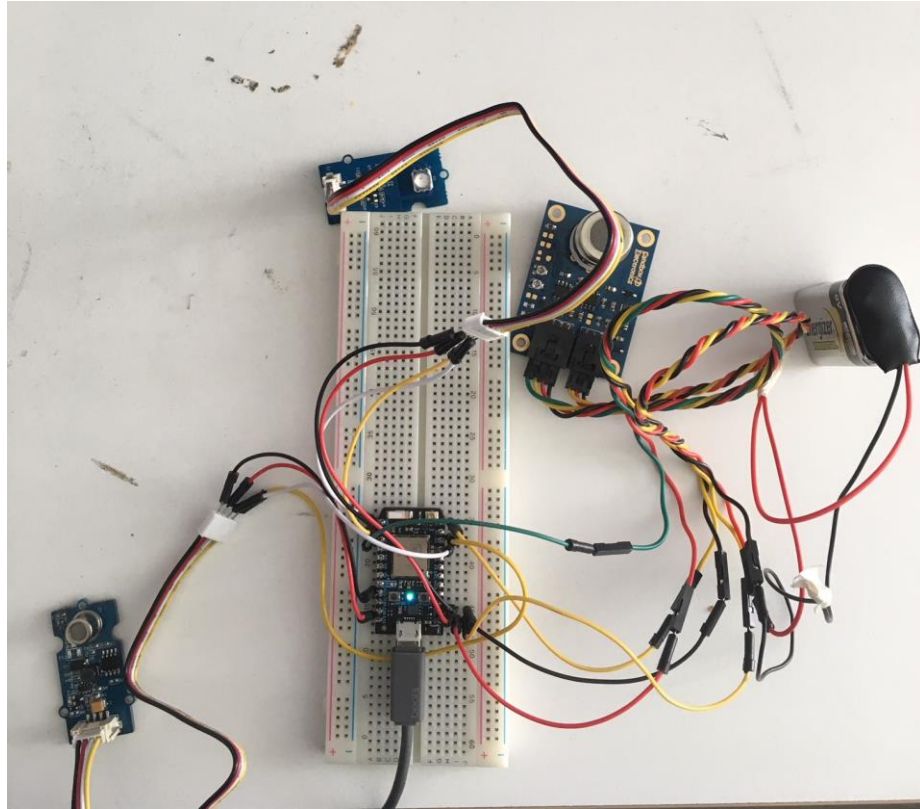


Figure 15. Prototype design on a breadboard

The prototype of the design with the sensor modules and an external source of power supply for CO<sub>2</sub> sensor module tested on a breadboard is shown in figure 15. For the milling purpose, the board was milled in the Milling Machine Room at Metropolia University of Applied Sciences.

#### 4.5 Schematic Design

For the Schematic Design, Eagle Autodesk Software was used as a EDA tool. The software was quite easier and simpler to design the schematic and layout files. The design was quite simple as it includes one Particle Photon, three analog connectors, two IIC connectors, one XBee and the additional female header connectors for XBee. There were many symbols and packages for the various electronic components in the Eagle library. But, the symbol and the package for some components had to be built from the Eagle library editor. The Particle Photon and XBee were built and modified through library editor in Eagle. The Grove connectors package for Eagle was imported from Seedstudio library whereas the two 1x10 pin connectors for XBee was found in the Eagle



library. As the schematic file was ready, the next step was to generate the layout file for the Board.

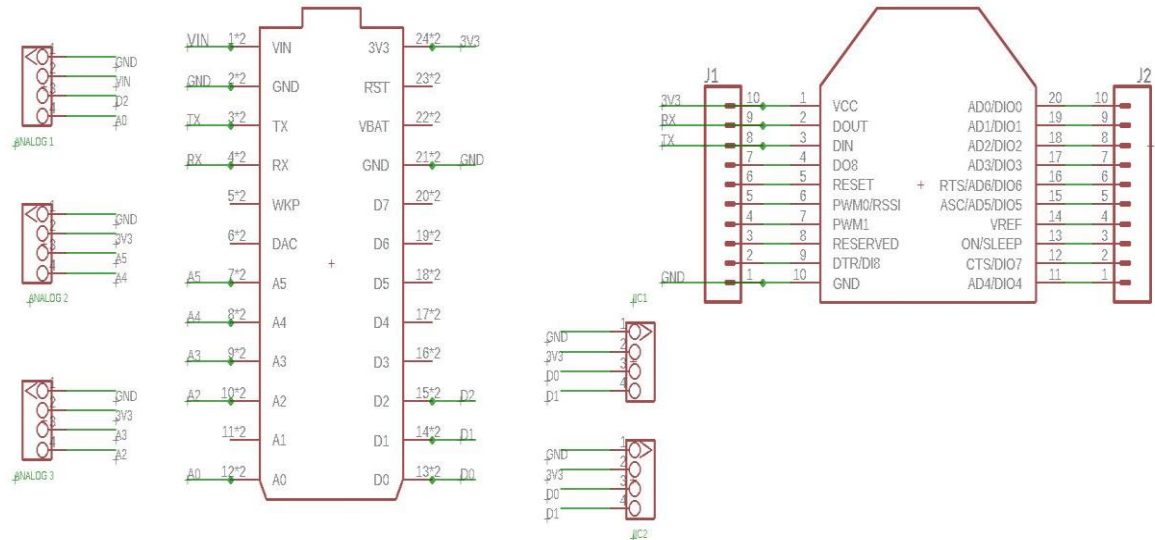


Figure 16. Schematic Design for the PCB Board

Figure 15 illustrates the general schematic of the prototype board for the project created by Eagle Autodesk.

#### 4.6 PCB Layout

As soon as the schematic file design was finished, the layout file was also created by turning the schematic into a board via the Eagle Software. Hereafter, we can begin to layout so that we can eventually get it fabricated and solder on the components and have a working circuit. The required components are spread outside the dimensional area and we should arrange all the components inside that area. Basically, the dimensional area is the size of the board for milling. The processes such as Layout, Automatic and Manual routing, Ratsnest, Net classes, Ground ports, layer and grid controls everything includes in this part. Then, the necessary Gerber files for PCB manufacturing were created via CAM processor.

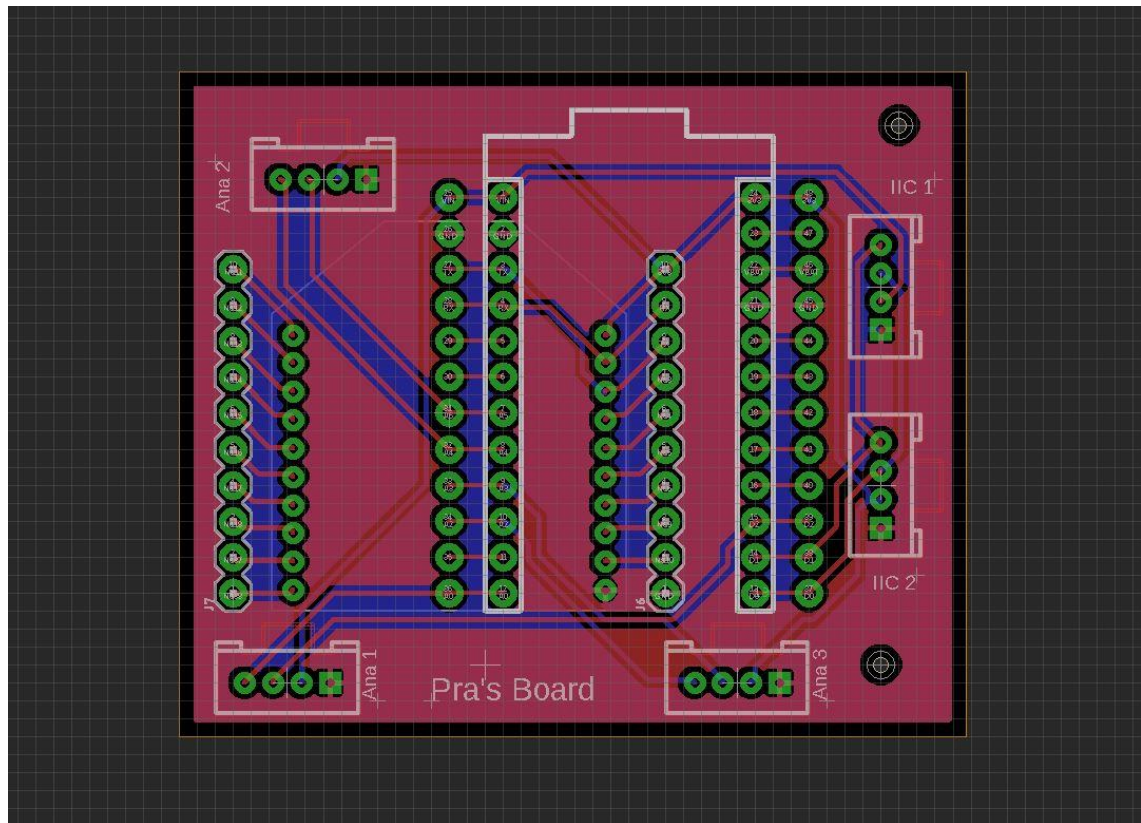


Figure 17. Board file for the PCB

Figure 16 describes the prototype layout board designed using Eagle Software. It shows all the required footprints for the components. The board is a two layered PCB board and both layers are connected to the ground with the intention that, no traces were required for the ground pins. The traces and via for the wires are drawn according to the schematic.

#### 4.7 CircuitCAM and BoardMaster

As soon as the Gerber files were created through Eagle EDA tool, the Gerber files were then required to be converted into CAM and LMD files to mill the PCB board. For this purpose, the Gerber files were imported to CircuitCAM software to create LMD file. The manuals for CircuitCAM and BoardMaster were available at Metropolia UAS and were used for creating the files and producing the board from BoardMaster software which was connected to the milling machine. All the instructions and tutorials found in the manuals were followed exactly as stated. Then, the board was milled with LMD file using BoardMaster software which was connected to the LPKF milling machine. The LPKF

milling machine was available at Metropolia UAS and was used to mill the Board. As soon as the board had been milled, the next step was soldering the board with required components.

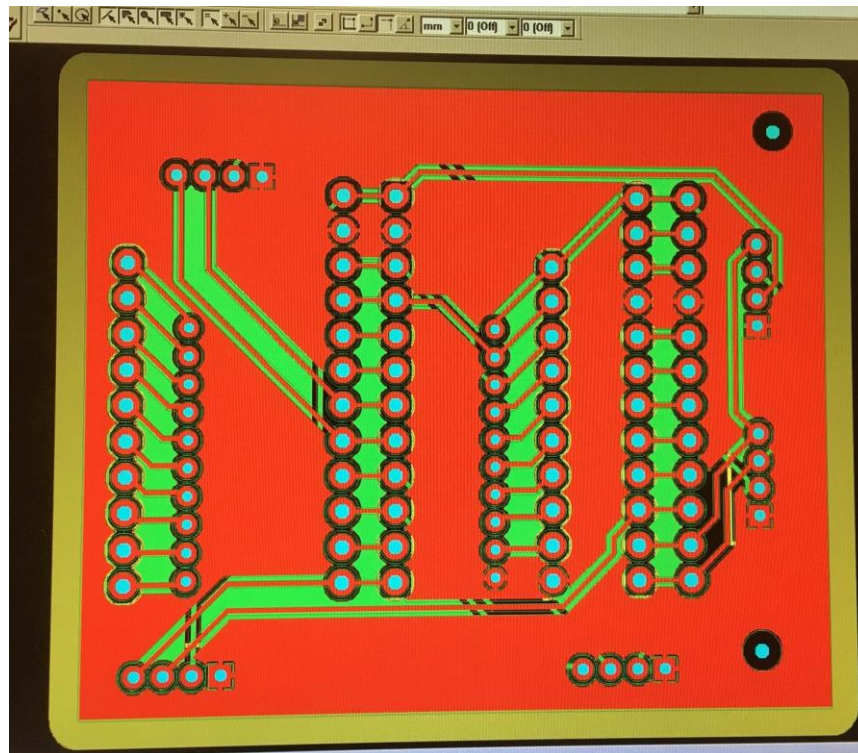


Figure 18. LMD file for Milling

Figure 17 shows the LMD file created from CAM processor at Metropolia UAS which is then imported for milling purpose.

## 5 Development of the Software

Particle has distributed free software infrastructure for its users called Particle Cloud which is highly secured platform. The Photon can be connected easily to the cloud by using a Particle account and once it is connected, cloud helps to manage the device in a desired way. A web compiler or IDE is accessible for flashing the code to the Photon over air. The Development Environments available are Particle Build- an Web IDE and Particle Dev- Desktop IDE. The former IDE, Particle Build was used for the project. The Particle Photon has a built-in function known as publish that transmits data to the Particle Cloud from the Photon, which could also be forwarded to other cloud platforms via various integrations.

## 5.1 Particle Build

Particle Build is a Web Integrated Development Environment for the easy software development used in a web browser. The codes are written in Particle Build and then flashed into the Photon device for building various applications and projects. The applications and projects that had been successfully created, can also be shared to the others via Share this revision on Particle Build. After that, the revision becomes public and can be accessible to anyone.

## 5.2 Libraries/Functions

Libraries are very important for the easy project development. Particle has a number of reusable firmware libraries that could be included to various Particle projects. Firmware libraries are an essential tool for the connection of the Photon to the sensors or actuators. These libraries help in reusing and modifying the codes for various Particle projects. Including verified libraries help to build the applications and projects reducing the time, risk and cost, significantly. The libraries are accessible through all development tools inside Particle ecosystem and are also hosted on GitHub [19]. The required library for the sensors or actuators for the project could be found and could be included as a firmware library into the projects. This adds the library dependency to the file which is compiled with the projects when flashed.

In addition to these, most libraries compatible with Arduino are also accessible for Particle projects. The firmware for Particle has been developed in such a way that it contains mostly used functions and commands compatible for Arduino. This means submission of Arduino libraries into the Particle ecosystem is favorable without modification. Most of the popular libraries for Arduino are available in the Particle library ecosystem and many other libraries can also be modified or improved easily for compatibility. Furthermore, the particle libraries can be accessed and included for other particle libraries as well. Particle ecosystem has many reliable libraries that can be accessed to a maximum limit for building applications and projects.

### 5.3 General Description

The software uses IntervalTimer to call three functions namely *quality* (For Air quality sensor), *I2C\_com\_barometer* (For Barometer sensor) and *C02\_barometer* (For C02 sensor) that runs at 5000, 3000 and 5000 hmSec respectively. In order to use Interval timer of Particle Photon with ARMcortex MCU, *SparkIntervalTimer.h/SparkIntervalTimer.cpp* libraries are used. Three interval timers are declared as myTimer, myTimer1 and myTimer2.

Each of three function works to collect sensor data, processes them according to the library and produce the useful information that is printable in console. These data are saved in console variable.

Inside the main loop, console prints all these sensor data saved in console variable. There is delay of few seconds between each interval of printing the data in console. This timers with their allocated times run continuously calling respective functions which are assigned by them. These are set inside setup () of the software. For example

*IntervalTimer myTimer;* #myTimer is the instance of IntervalTimer class

*myTimer.begin(quality,5000,hmSec);* #quality is the function called at 5000hmSec

For air quality sensor,

AirQuality sensor Library (AirQuality.cpp, AirQuality.h) is used.

*airqualitysensor* is the instance of AirQuality.

*airqualitysensor.init(14);*

This would called init class function that would initialize initial settings of sensor reading. For example, comparing initial ADC(Analog to Digital Converter) readings of pin with connected sensor pin and unconnected sensor pin and checking if there occurs any error.

*current\_q=airqualitysensor.slope();*

This gives the current status of sensor readings. After comparing measured ADC values, it gives the corresponding alert messages. For example,

Inside *AirQuality.cpp* library file of Airquality sensor,

```
AirQuality ::slope(void)
{
/////
    if(first_vol-last_vol>1700||first_vol>1900)
    {
        Serial.println("High pollution! Force signal active.");
        Particle.publish("Airquality_sensor","High pollution! Force sig-
nal active.",60,PRIVATE);
        return 0;
//////////
    }
}
```

If the ADC reading from air is 1800, then it is polluted air. Every interval of 5000hmSec, quality function is called.

```
void quality(void)
{
    airqualitysensor.last_vol=airqualitysensor.first_vol;
    airqualitysensor.first_vol=analogRead(A2);
    airqualitysensor.counter=0;
    airqualitysensor.timer_index=1;
}
}
```

This would update the current readings of sensor values every time.

Similarly,

For barometer Sensor,

Barometer sensor communicates through I2C\_com\_barometer()

*Void I2C\_com\_barometer()*

This function would initiate I2C communication between sensor and Particle Photon. All the configuration for I2C communication is defined its library *spark\_wiring\_i2c.h*.

In addition to these, this function produces and processes barometer sensor data, they are Temperature in Celsius, Temperature in Fahrenheit, Pressure and Altitude.

Like other sensor's data, they are printed in console by the following lines of code in main loop

```
Particle.publish("altitude :", String(altitude));
Particle.publish(" pressure ", String( pressure ));
Particle.publish(" CTemperature", String( CTemperature ));
Particle.publish("FTemperature", String(FTemperature));
```

For MG811 CO<sub>2</sub> sensor,

The reading of CO<sub>2</sub> level in CO<sub>2</sub> sensor is expressed in ppm level (400-10000). As it measures its ppm level based on output voltage (100-600) mV which is actually amplified for signal conditioning.

All the reconfiguration is defined before the program runs. For example, the sensor output voltage is in the range of 200-600mV in fresh air (400ppm CO<sub>2</sub>).

```
#define ZERO_POINT_VOLTAGE
#define REACTION_VOLTAGE
```

Reaction voltage is the voltage difference when CO<sub>2</sub> ppm level of 400 is dropped to 1000 ppm. Here, for our program and this particular sensor, reaction voltage is 20mV.

```
double CO2Curve[3] = {2.602, ZERO_POINT_VOLTAGE, ( REACTION_VOLTAGE/(2.602 - 3))};
```

CO<sub>2</sub> Curve gives the approximation of original curve of targeted gas. Using this curve as a reference, a line is formed with the data format:{x,y,slope};

As mytimer2 is implemented in its fixed interval of 5000hmSec, it calls function CO2\_barometer()

```
void CO2_barometer(void)
{
    int i;    double v=0;

    CO2_ADC = analogRead(MG_PIN);
    v += analogRead(MG_PIN);
    for(i=0; i<READ_SAMPLE_TIMES; i++)
    {
v +=  analogRead(MG_PIN);

        delay(READ_SAMPLE_INTERVAL);

        v = (v)*3.3 / 4096 ;

        if ((v / DC_GAIN ) >= ZERO_POINT_VOLTAGE) {

            percentage = -1;
        }else{
            percentage = pow (10, ((v/DC_GAIN)-CO2Curve[1])/CO2Curve[2]+CO2Curve[0]);
        }
    }
}
```

This functions at first measures sensor output voltage and then uses same output voltage and pcurve of target gas to calculate ppm level using some Mathematics. Using this tangent of reference gas 's curve (CO2curve) and a point in it, X coordinate of line is calculated from available y coordinate value (which in this case is output voltage of sensor). X coordinate of line gives ppm level of CO<sub>2</sub>,

*Math.h* library has to be included in the program to support the math function.

```
percentage=pow(10, ((v/DC_GAIN)-CO2Curve[1])/CO2Curve[2]+CO2Curve[0]);
```



## 6 Results

All the above processes involved milling and design of the board. The board needed to be designed in a way that it gets minimum size possible. At first, the board that was milled was found to be larger in size. The design had all the components on the top of the PCB board and hence, resulted for bigger size. So, the board needed to be designed again in such a way that it gets minimum size. For this, the idea was made in such a way that the board could be designed with XBee lying on the top whereas, Particle Photon and the connectors lying on the bottom of the PCB board. The idea was good enough for making the PCB board as small as it could be done.

The Board that was milled, was then needed to be soldered with the connectors. But, some issues were found with the milled PCB board resulting prohibition of soldering the connectors. Then, the idea of testing the design on the milled PCB board was skipped and the test was carried out on a breadboard. The observations found during the test were satisfactory as shown in figure 19 and 20.

EVENT NAME	DATA
Airquality_sensor	Air fresh
CO2 Sensor's ADC	74.000000
CO2 Sensor's ppm	5733991.320861
CTemperature	26.340000
pressure	1021.420000
altitude :	10417.940000
Airquality_sensor 'ADC	75
CO2 Sensor's ADC	74.000000
Airquality_sensor:	Air fresh

Figure 19. Test results in Particle console

EVENT NAME	DATA
Airquality_sensor	High pollution! Force signal active.
CO2 Sensor's ADC	411.000000
CO2 Sensor's ppm	651604.950267
CTemperature	30.580000
pressure	1732.640000
altitude :	6177.660000
Airquality_sensor 'ADC	409
Airquality_sensor	Low pollution!
CO2 Sensor's ADC	221.000000
CO2 Sensor's ppm	2166588.168525
CTemperature	29.900000

Figure 20. Test results with some changes in sensor data

The figures 19 and 20 illustrate that the design had working results and the test was successful. Changes in data value can be seen in between those figures. In figure 19, as the device came online, Air quality sensor illustrated the data as Air Fresh which indicates the fresh air in the surrounding environment. The event, CO2 sensor ADC shows the ADC value of the CO2 sensor module and has a data value of 74. Likewise, the pressure and temperature in degree Celsius as shown by barometer sensor be 1021.42 and 26.34 respectively. The altitude value be 10417.94 which is permuted by special algorithm from pressure and temperature data. The ppm value and the ADC values of the CO2 sensor module are also displayed which is illustrated in figure 19.

As soon as the perfume is sprayed near the surroundings of air quality and CO2 sensor module, the change in data value can be observed. The data in figure 20, shows change in ADC value of Air quality sensor indicating low pollution and High Pollution. The fresh air environment data from figure 19 has been observed as polluted environment as the gases from perfume bottle gets sprayed over it, which can be seen in figure 20. The ADC value for CO2 sensor module in figure 20 is also seen changed indicating the change in concentration of CO2 gas in that environment because of CO2 gas, coming out from the perfume bottle.

When the barometer sensor is pressed with the finger, pressure gets exerted in the sensor and results the change in the data value of pressure which could be seen in the figure 20. As much as it exerts pressure, the value keeps increasing which could be compared from the figures 19 and 20. Correspondingly, the change in pressure value seen from figure 19 to figure 20 be from 1021.42 to 1732.

This design could be delivered to the other parties for the further development of the project design and making the milled board compatible and working.

## 6.1 Design Issues

The issues that were discovered while designing the PCB board for the project are briefly described here. Because of these issues, the milled PCB was not compatible for soldering purpose and henceforth, the project design was carried out on a breadboard and the design on the milled PCB board was not tested.

- Clearance issue with the circuit CAM milling was the foremost. The first PCB prototypes were milled using 10 mils of clearance which had tens of tiny pieces of copper creating short cuts and were manually difficult to remove. Later, it was found that using 18mils clearance was advisable.
- The default pad size for the components and the connectors were relatively small which results problem for soldering with the components manually. It is recommended to increase the default pad size for proper soldering manually.
- Connector components cover the pads and traces on the layer where they are installed. This also results problem for manual soldering. Thus, all the traces should be connected to the connector pads on the opposite layer. It means if the connector is on the TOP layer, all the traces should be connected on the BOTTOM layer and vice-versa.
- The milled PCB board was not texted properly and it results difficulty in identifying the layers and components of the board. Thus, adding text for the components for proper indication of the layers is favorable. It is recommended to add text on components to indicate whether the layer is top or bottom and also advantageous to indicate the pin numbers for connectors and components.

## 7 Conclusion

The aim of this thesis project was to design and simply develop the IoT platform for educational purpose. The primary goal was to design the platform for sensors so that it could be used by anyone to study or for projects or for developing the prototype into the advanced one. Although the goal was to design a milled PCB platform and run some test on it with the sensors provided, some design issues forbade soldering the connectors with the milled PCB and the test was performed on a breadboard. But, the complete prototype design should be based on milled PCB board and should also focus on XBee with the platform and requires good knowledge of ZigBee and XBee. Anyway, the prototype on the breadboard exhibited the tests working properly and I consider the project successfully accomplished.

This is the first stage of designing the IoT platform and this prototype will be used as a reference project for its further development. The platform board has good potential and can be used for many projects. The size of the PCB board is made very small which could be very handy. The sensor connectors used in the platform are also compatible with other sensors with grove systems. The sensors like air quality, barometer and CO<sub>2</sub> sensor used for the project could be an alternative equipment for studying environmental conditions. As, technology is expanding its space all around the globe, the platform could be very convenient for research and IoT purposes. However, IoT is a dynamic field, the change in development of smart systems is very essential and I believe the multiple uses of this prototype might be restricted in near future.

## 8 References

- [1] "wikipedia," [Online]. Available: [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things). [Accessed 13 February 2018].
- [2] S. Electronics, Sandbox EElectronics, [Online]. Available: <http://sandboxelectronics.com/?product=mg-811-co2-gas-sensor-module>. [Accessed 21 February 2018].
- [3] S. Electronics, "MG-811 Sensor module," 3 February 2014. [Online]. Available: <http://sandboxelectronics.com/?p=147>. [Accessed 2 March 2018].
- [4] Seedstudio, "Grove-Air Quality Sensor V1.3," Seedstudio, [Online]. Available: [http://wiki.seeedstudio.com/Grove-Air\\_Quality\\_Sensor\\_v1.3/](http://wiki.seeedstudio.com/Grove-Air_Quality_Sensor_v1.3/). [Accessed 17 February 2018].
- [5] Seedstudio, "Grove-Barometer(High-Accuracy)," Seedstudio, [Online]. Available: <http://wiki.seeedstudio.com/Grove-Barometer-High-Accuracy/>. [Accessed 17 February 2018].
- [6] H. Electronics, "HP206 datasheet," [Online]. Available: [file:///C:/Users/pRAMES/Downloads/HP206C\\_Datasheet%20\(3\).pdf](file:///C:/Users/pRAMES/Downloads/HP206C_Datasheet%20(3).pdf). [Accessed 19 February 2018].
- [7] P. P. Ltd, "Introducing IoT with Particle's Photon and Electron," 7 September 2016. [Online]. Available: <https://hub.packtpub.com/introducing-iot-particles-photon-and-electron/>. [Accessed 13 February 2018].
- [8] Particle, "Photon Document," [Online]. Available: <https://docs.particle.io/guide/getting-started/intro/photon/>. [Accessed 11 February 2018].
- [9] Particle, "Connecting Photon to internet," Particle, [Online]. Available: <https://docs.particle.io/guide/getting-started/start/photon/#step-1-power-on-your-device>. [Accessed 20 April 2018].
- [10] EL-PRO-CUS, "ZigBee Wireless Topology," 2015. [Online]. Available: <https://www.elprocus.com/what-is-zigbee-technology-architecture-and-its-applications/>. [Accessed 1 March 2018].

- [11] E. Technology, "ZigBee Wireless Networking Technology," [Online]. Available: <https://www.electricaltechnology.org/2017/09/zigbee-technology-wireless-networking-system.html>. [Accessed 1 March 2018].
- [12] Wikipedia, "XBee," [Online]. Available: <https://en.wikipedia.org/wiki/XBee>. [Accessed 7 March 2018].
- [13] Robotshop, "Digi XBee Transceiver Module(Wire Antennae)," [Online]. Available: <https://www.robotshop.com/en/digi-1mw-xbee-transceiver-module-wire-antennae.html>. [Accessed 12 March 2018].
- [14] Ebay, "Seed technology Universal 4 Pin Connector," [Online]. Available: <https://www.ebay.co.uk/itm/110990030-Seeed-Technology-Grove-Universal-4-Pin-Connector-Pk10-/132106294695>. [Accessed 19 March 2018].
- [15] H. t. Mechatronics, "How I2C communication works," [Online]. Available: <https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/>. [Accessed 17 March 2018].
- [16] H. t. Mechatronics, "How I2C Communication works," How to Mechatronics, 5 October 2015. [Online]. Available: <https://www.youtube.com/watch?v=6IAkYpmA1DQ&t=148s>. [Accessed 16 March 2018].
- [17] "Guide to Common PCB Issues," Millenium Circuits Limited, 13 10 2016. [Online]. Available: <https://www.mclpcb.com/guide-pcb-issues/>. [Accessed 18 April 2018].
- [18] Wikipedia, "EAGLE," [Online]. Available: [https://en.wikipedia.org/wiki/EAGLE\\_\(program\)](https://en.wikipedia.org/wiki/EAGLE_(program)). [Accessed 27 March 2018].
- [19] Particle, "Firmware Libraries," [Online]. Available: <https://docs.particle.io/guide/tools-and-features/libraries/>. [Accessed 24 April 2018].
- [20] M. Rouse, "Sensor," Whatis.com, July 2012. [Online]. Available: <https://whatis.techtarget.com/definition/sensor>. [Accessed 15 February 2018].

## Appendix 1.

### Code used for the Project

```

#include <SparkIntervalTimer.h>
#include <application.h>
#include <spark_wiring_i2c.h>
#include "math.h" // FOR CO2 sensor math's calculation

//For Airquality Sensor//////////
#include "AirQuality.h"
#include "Arduino.h"
//////////For Airquality Sensor//////////

//Analog pin 0 for CO2
//Analog pin 2 for Airquality sensor

//****Airquality Sensor****//////////

AirQuality airqualitysensor;
int current_q =-1;
IntervalTimer myTimer; // Timer for airquality sensor
void quality(void);

//const uint8_t ledPin = D7;
//****Air quality Sensor****//////////

//*****CO2*****//////////

#define MG_PIN (0) //define which analog input channel you are going to use
#define BOOL_PIN (2)
#define DC_GAIN (5.5) //define the DC gain of amplifier

//*****Software Related Macros*****/

#define READ_SAMPLE_INTERVAL (50) //define how many samples you are going
to take in normal operation

#define READ_SAMPLE_TIMES (15) //define the time interval(in milisecon-
d) between each samples in normal operation

//*****Application Related Macros*****/
//These two values differ from sensor to sensor. user should derermine this
value.

#define ZERO_POINT_VOLTAGE (0.220) //define the output of
the sensor in volts when the concentration of CO2 is 400PPM

#define REACTION_VOLTGAE (0.020) //define the voltage drop
of the sensor when move the sensor from air into 1000ppm CO2

Double CO2Curve[3] ={2.602,ZERO_POINT_VOLTAGE, (REACTION_VOLTGAE/(2.602-3))};
double percentage;
double volts;
double CO2_ADC=0;
IntervalTimer myTimer2; // Timer2 for collecting Co2 sensor data

//two points are taken from the curve.
//with these two points, a line is formed which is
//"approximately equivalent" to the original curve.
//data format:{ x, y, slope}; point1: (lg400, 0.324), point2: (lg4000, 0.280)
//slope = ( reaction voltage ) / (log400 -log1000)

```

```

////////////////////////////////////**CO2**////////////////////////////////////

// Set variables for BAROMETER
// HP206C I2C address is 0x76(118)

#define Addr 0x76
double CTemperature = 0.0, FTemperature = 0.0, pressure = 0.0;
double altitude = 0.0;
IntervalTimer myTimer1; // Timer1 configuration for Barometer
// Set variables for BAROMETER

void setup()
{
    // Air quality Sensor
    // pinMode(ledPin, OUTPUT);
    Particle.variable("current_q", current_q);

    delay(1000);
    // Particle.variable("cq", current_quality);

    airqualitysensor.init(14);

    myTimer.begin(quality, 5000, hmSec);
    // Air quality Sensor

    // Set variables for BAROMETER
    // Particle.variable("i2cdevice", "HP206C");
    Particle.variable("CTemperature", CTemperature);
    Particle.variable("pressure", pressure);
    Particle.variable("FTemperature", FTemperature); // Only pressure is used

    Particle.variable("altitude", altitude);
    // Particle.variable("altitude", altitude);
    // Initialise I2C communication as Master

    Wire.begin();
    myTimer1.begin(I2C_com_barometer, 3000, hmSec); // Interval of
    2000hmSec. If there is delay in i2x_barometer function, might need to remove
    timer
    //airqualitysensor.init(14);
    // Initialise serial communication, set baud rate = 9600
    // BAROMETER

    // CO2
    // Particle.variable("percentage", percentage);
    // Particle.variable("volts", volts);

    pinMode(BOOL_PIN, INPUT); //set pin to input
    digitalWrite(BOOL_PIN, HIGH); //turn on pullup resistors

    myTimer2.begin(CO2_barometer, 5000, hmSec);

    // Particle.publish("Event1", "MG-811 Demostration", 60);

    // CO2

    Serial.begin(9600);

```



```

    delay(300);
}

void loop()
{

    ////////////////****Air quality Sensor****////////////////////
    current_q=airqualitysensor.slope();

    delay(2000);

    // Particle.publish("current_quality",String(current_quality),60,PRIVATE);
    // Particle.publish("Airquality_sensor",String(current_q),120,PRIVATE);

    delay(2000);

    Particle.publish("Airquality_sensor 'ADC ",String(airqualitysen-
sor.first_vol=analogRead(A0)),120,PRIVATE);
    ////////////////****Air quality Sensor****////////////////////

    ////////////////Barometer

    // Output data to dashboard

    Particle.publish("altitude :", String(altitude));
    Particle.publish("pressure", String(pressure));
    Particle.publish("CTemperature", String(CTemperature));
    Particle.publish("FTemperature", String(FTemperature));

    delay(2000);

    ////////////////Co2 Senso////////////////////////////////////

    if (percentage == -1) {
        // Particle.publish("Event1","V ");

        Particle.publish("C02 Sensor's ppm", "<400", 60, PRIVATE);
        // Serial.print( "<400" );
    } else {

        Particle.publish("C02 Sensor's ppm", String(percentage), 60, PRIVATE);
        // Serial.print(percentage);
    }

    // Serial.print( "ppm" );
    // Particle.publish("C02 Sensor's ppm",String(percentage),60,PRIVATE);

    Particle.publish("C02 Sensor's ADC",String(C02_ADC),60,PRIVATE);

    ////////////////Co2 Sensor //////////////////////////////////////

}

```

```
void I2C_com_barometer(void )
{
    unsigned int data[6];

    // Start I2C transmission
    Wire.beginTransmission(Addr);
    // Send OSR and channel setting command
    Wire.write(0x44 | 0x00);
    // Stop I2C transmission
    Wire.endTransmission();

    // Start I2C transmission
    Wire.beginTransmission(Addr);
    // Select data register
    Wire.write(0x10);
    // Stop I2C transmission
    Wire.endTransmission();

    // Request 6 bytes of data
    Wire.requestFrom(Addr, 6);

    // Read 6 bytes of data
    // cTemp msb, cTemp csb, cTemp lsb, pressure msb, pressure csb, pressure lsb
    if (Wire.available() == 6)
    {
        data[0] = Wire.read();
        data[1] = Wire.read();
        data[2] = Wire.read();
        data[3] = Wire.read();
        data[4] = Wire.read();
        data[5] = Wire.read();
    }

    // Convert the data to 20-bits
    CTemperature = (((data[0] & 0x0F) * 65536) + (data[1] * 256) + data[2]) /
100.00;
    FTemperature = (CTemperature * 1.8) + 32;
    pressure = (((data[3] & 0x0F) * 65536) + (data[4] * 256) + data[5]) /
100.00;

    // Start I2C transmission
    Wire.beginTransmission(Addr);
    // Send OSR and channel setting command
    Wire.write(0x44 | 0x01);
    // Stop I2C transmission
    Wire.endTransmission();

    // Start I2C transmission
    Wire.beginTransmission(Addr);
    // Select data register
    Wire.write(0x31);
    // Stop I2C transmission
    Wire.endTransmission();

    // Request 3 bytes of data
    Wire.requestFrom(Addr, 3);

    // Read 3 bytes of data
    // altitude msb, altitude csb, altitude lsb
    if (Wire.available() == 3)
    {
        data[0] = Wire.read();
```

```

    data[1] = Wire.read();
    data[2] = Wire.read();
}

// Convert the data to 20-bits
altitude = (((data[0] & 0x0F) * 65536) + (data[1] * 256) + data[2]) /
100.00;
}

//////////****CO2****////////////////////////////////////
void CO2_barometer(void)
{
    int i;
    double v=0;

    CO2_ADC = analogRead(MG_PIN);
    v += analogRead(MG_PIN);

    //Particle.publish("ADC",String(v),60,PRIVATE); Debugging for purpose only

    v = (v)*3.3/4096 ;

    // Particle.publish("log",String(v),60,PRIVATE); Debug-
    ging for purpose only

    if ((v/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
        percentage = -1;
    } else {
        percentage = pow(10, ((v/DC_GAIN)-CO2Curve[1])/CO2Curve[2]+CO2Curve[0]);
    }
}

//////////****CO2****////////////////////////////////////

//////////Airquality////////////////////////////////////

void quality(void)
{
    // digitalWrite(ledPin,!digitalRead(ledPin));

    airqualitysensor.last_vol=airqualitysensor.first_vol;
    airqualitysensor.first_vol=analogRead(A2);
    airqualitysensor.counter=0;
    airqualitysensor.timer_index=1;
    //PORTB=PORTB^0x20;
}

//////////Airquality////////////////////////////////////

```