

Vinh Loc Huynh

# Environmental Monitoring System

Metropolia University of Applied Sciences

Bachelor of Engineering

Environmental Engineering

Thesis

31.03.2018

Author Title	Vinh Loc Huynh Environmental Monitoring System
Number of Pages Date	39 pages + 6 appendices 31 March 2018
Degree	Bachelor of Engineering
Degree Program	Environmental Engineering
Professional Major	Water, Wastewater and Waste Treatment Technology
Instructors	Jari Olli, Lecturer, Clean Technology Minna Paananen-Porkka, Language Supervisor
<p>A sensor is utilized in almost every electronic application and yet, it has been neglected in the field of Environmental Engineering. Measurement is essential in data analysis and with the support of sensor technology, the work could be done effortlessly. However, there are plenty of models in the market and choosing the correct formula to work with is another issue.</p> <p>As a first step of the learning process, air quality monitoring was chosen as the objective to tackle utilizing sensors. Research and development was carried out continuously to decide the best combination of devices and sensors. Amid the process, there were some incompatibility issues and it required persistence to find the correct path.</p> <p>The result of the experiment shows that there are numerous conceivable outcomes to approach the sensor technology and combining them together could create more innovative solutions for air quality monitoring. Analyzing the data is a tedious process but the final result is reliable information and the addition of vehicles, it brings mobility to the sensors to act autonomously. Finally, further development is suggested and this application can be assigned to next group of students as an innovation project.</p>	
Keywords	IoT, Presnsr, environmental monitoring system, Arduino

## Contents

1	Introduction	7
2	Goals and Scope	8
3	Presnsr	9
4	Theory Review	10
4.1	Arduino and Single-board Microcontroller	10
4.1.1	History of Arduino	10
4.1.2	Arduino's Component	10
4.1.3	Single-board Microcontroller	10
4.2	Sensor	13
4.2.1	Introduction to Sensor Technology	13
4.2.2	List of Air Quality Sensors	13
4.3	ThingSpeak	15
4.3.1	Introduction	15
4.3.2	Benefits of ThingSpeak	16
4.3.3	Set-up	16
4.4	Internet of Things (IoT)	17
4.4.1	Introduction	17
4.4.2	Application	17
4.5	Calculation	18
4.5.1	Temperature and Humidity	18
4.5.2	Optical Dust Sensor	19
4.5.3	Digital Pressure Sensor	19
5	Experimentation	20
5.1	Environmental Monitoring Board (EMB)	21
5.1.1	Testing	21
5.1.1.1	Breadboard	21

5.1.1.2	Jumper wire	22
5.1.1.3	Testing process	23
5.1.2	Online Server	24
5.1.2.1	Ethernet Shield V1	24
5.1.2.2	Wireless	25
5.1.3	First Prototype	25
5.1.3.1	Hardware Setup	26
5.1.3.2	Software Setup	27
5.1.4	Second prototype	28
5.1.5	Final prototype	30
5.2	Vehicle	31
5.2.1	Motor Shield L293D	31
5.2.2	Ultrasonic sensor and 9g Servo	32
5.2.3	Bluetooth Module	33
6	Results	34
6.1	EMB	34
6.2	Vehicle	35
7	Conclusion	36
	References	37
	Appendices	38

**Appendix 1. Final prototype**

**Appendix 2. Bluetooth Module Diagram**

**Appendix 3. Ultrasonic Sensor Diagram**

**Appendix 4. Command for EMB Offline (Final version 20.09.17)**

**Appendix 5. Command for EMB Online (Final version 23.10.17)**

**Appendix 6. Command for the vehicle (Final version 25.02.18)**

## List of Figures

Figure 1 Presnr.....	9
Figure 2 Arduino Uno Rev3 .....	11
Figure 3 Wemos D1 Min (wemos.cc) .....	11
Figure 4 Arduino Primo (arduino.cc).....	12
Figure 5 Left - Optical dust sensor (sparkfun.com). Right - Internal schematic .....	14
Figure 6 Mq-135.....	14
Figure 7 BMP280 (sparkfun.com).....	14
Figure 8 HC-SR04.....	15
Figure 9 9g Servo (hobbyking.com) .....	15
Figure 10 HC-05 (Arduino-info.wikispaces.com .....	15
Figure 11 ThingSpeak Channel .....	17
Figure 12 Output-Density ( <a href="http://www.sharp-world.com">http://www.sharp-world.com</a> ) .....	19
Figure 13 830 Tie-points breadboard .....	22
Figure 14 Testing process .....	23
Figure 15 Ethernet Shield V1 .....	24
Figure 16 Wemos in the final setup .....	25
Figure 17 First prototype.....	26
Figure 18 Arduino IDE .....	27
Figure 19 Second prototype .....	29
Figure 20 Final prototype .....	31
Figure 21 First vehicle.....	32
Figure 22 Second vehicle .....	33
Figure 23 Temperature reading .....	34
Figure 24 Pressure reading .....	35
Figure 25 Carbon Dioxide reading.....	35

## List of Abbreviations

IoT	Internet of Things
IDE	Integrated development environment
NFC	Near-field Communication
LAN	Local Area Network
LCD	Liquid-Crystal Display
MATLAB	Matrix Laboratory
CO	Carbon Monoxide
CO <sub>2</sub>	Carbon Dioxide
LED	Light-emitting Diode
B2B	Business-to-Business
IEEE	Institute of Electrical and Electronic Engineers
I/O	Input/output
R&D	Research & Department
UI	User Interface
THT	Through-hole Technology
SMT	Surface-mount Technology
EMB	Environmental Monitoring Board
PCB	Printed Circuit Board

## 1 Introduction

Recent years have demonstrated an altogether expanding demand for electronic gadgets with the Internet-empowered component. It has aroused enthusiasm in manufactures and engineers, encouraging them to prioritize research and development (R&D) to respond to the growing demand. Individuals are getting more mindful of their surrounding conditions and the desire for more data has prompted the conception and rise Internet of Things (IoT). The most prevalent uses of IoT are, for example, smart farming where IoT is used for waste reduction and compost control and autonomous monitoring systems utilized in water industry and the renewable energy segment.

Sensor Technology is a division of microelectronics; it incorporates any module which has the capacity to read/write data and send it to a focal unit. The subject could have been an interesting course of its own, but it has, however, been overlooked and just instructed in one course: Physics for Sensor. Moreover, there are worries of how well it fits in a degree programme where the students lack in programming and imagination skills. The appropriate response is these prerequisites are discretionary in the circumstance of investigating new probability in learning. Anybody could turn on a LED and that is the initial step. With some energy and interest, turning on a LED transforms into a perusing of temperature, now it accompanies a remote network and in this manner, the entire framework is on wheels, gathering information naturally.

The application of this proposition is not advanced science and some of them have as of now a thing for a long time. Be that as it may, the reason here is to challenge the individual limit of pressure resistance in R&D since the measure of data is boundless with the possibility of contrariness in each and every trial being high. Since each part is built from scratch, this investigation requires persistence and versatile capacity in learning. Problems could happen randomly and solutions may just appear in dreams.

## 2 Goals and Scope

The target of this thesis project was to analyze the precision of sensor readings and the consistency of sensors in various conditions. At the beginning of the project, a large amount of input was gathered from pilot analyzers, which prompted different changes to the investigation stage. Moreover, a choice of cloud-based server is similarly vital in storing the information and holding the cost down. Sensors were tested with three applications: ThingSpeak, Blynk, and Particle. While Blynk contains a brilliant User Interface (UI) and it is perfect for smartphones, while Particle accompanies an extraordinarily solid ARM Cortex microcontroller in addition to a Mesh system and its plug-and-play stage is an incredible arrangement. However, ThingSpeak is the best decision thanks to its built-in MATLAB programming, which is the best alternative for breaking down and envisioning data logging. To put it plainly, this thesis focused for the most part on the ThingSpeak stage, , but future studies could investigate the others since every one of them is utilizing a similar language.

The purpose of this thesis was to present the universe of IoT in general and the Environmental monitoring system specifically as the foremost it could be connected with in the field of environmental engineering. Another aim was to increase awareness and make the study of IoT more prevalent in Metropolia.

The scope of thesis was to complete the sensor of the environmental monitoring system, and with the effectively settled startup, the items could at long last reach the go-to-market stage. This will be further explained in the following chapter.



### 3 Presnsr

Presnsr is a hardware-as-a-service startup thought established by Javier Hernandez, Product Development Manager of Cazza Inc – Construction Technologies and Services Company and Vinh Loc Huynh, R&D Engineer of SenzoIT Oy in June 2017. Presnsr began after a Startup Bootcamp in Kotka-Hamina Region, with the motivation from this claim proposal: Establishing a bridge between upkeep administrations and home security by having shrewd identifiers that alert at whatever point a high likelihood of hazard happens in light of a set algorithm. This is done all through numerous testing in various conditions and the principal market Presnsr targets is Nordic region. The client target is building segment, for example, shopping centers, real estates and office spaces. The plan of action is business-to-business (B2B) with two liberal collaborators already on board for pilot testing.

Presnsr works with a solitary guideline: simplicity. This applies to plans, equipment, programming and administrations. This acquires more potential investors while keeping the cost low. Since the undertaking is essentially self-subsidized, the beginning period endures huge problems in terms of both human resources and money assets.

The application used in this thesis will be used as a concept prototype of Presnsr for feedback and development. Figure 1 shows the rendering product of Presnsr. The team attended Kotka's LevelUp Accelerator, Slush 2017, Turbiini Accelerator and Cambridge Venture Camp 2018. The next target, could be the most imperative one is Y Combinator in California.

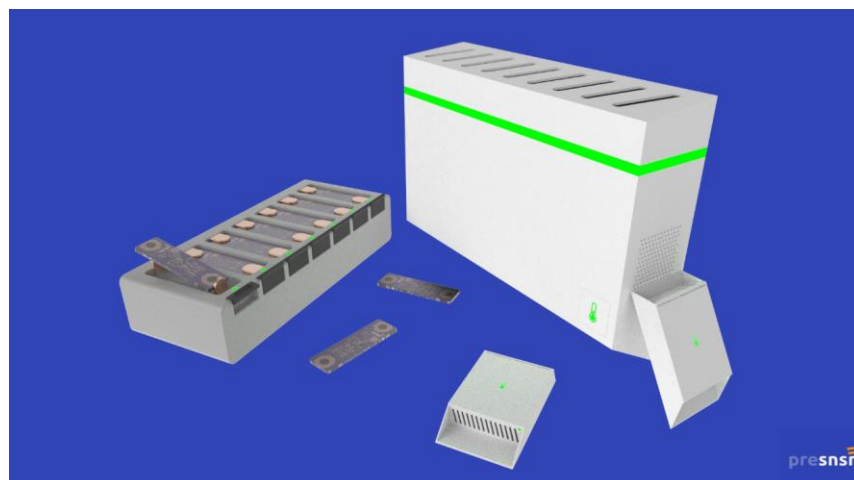


Figure 1 Presnsr

## 4 Theory Review

### 4.1 Arduino and Single-board Microcontroller

#### 4.1.1 History of Arduino

The historical backdrop of Arduino can be followed back to the time of 2003-2005 to the Master's proposition of Hernando Barragan on making an easier and less expensive programmable circuit load up called Wiring for students (Hernando, 2016). A gathering of analysts, including Hernando's superiors determined the undertaking and made their own particular platform. The platform was named Arduino. Hernando, after years battling for the privilege of unique thought, at long last recovered his equity and is now functioning as the Arduino Chief Design Architect.

According to Arduino company, the package comprises of a circuit board and the Integrated Development Environment (IDE) written in C++. The key purpose of Arduino which has made it the world's driving electronic platform is it was made to be open-sourced, which means everybody, paying little mind to informatics, could learn and utilize Arduino. Arduino has created different models relying upon the motivation behind tasks and including various features.

#### 4.1.2 Single-board Microcontroller

Since the thesis idea developed through time, new boards are added for testing. The change is needed for scaling the prototype: the size is cut down to few centimeters, the duration is increased while functionality is still the same or even better.

**Arduino Uno Rev3:** To begin with, Arduino Uno is the most widely recognized Arduino microcontroller board in the field and extraordinary compared to other boards. It works at 5V and could be associated by means of USB type B to PCs for uploading code and driving the board. It has 14 Digital pins for computerized signals, 6 Analog pins for Analog flags and separate 3.3V/5V Output control. To have the capacity to transfer the code from IDE, the board needs Flash Memory and in this model, it accompanies 32KB, which is extensively low for intermediate projects; however, this is appropriate for the testing performed in this thesis project. Figures 2 shows the Arduino Uno version and its pinouts.



*Figure 2 Arduino Uno Rev3*

**Wemos D1 Mini v2.3.0:** Properly the smallest development board in the market, with dimensions just barely 34 mm x 25 mm, Wemos D1 Mini is equipped with an ESP8266MOD module to utilize standard 802.11 protocol created by the Institute of Electrical and Electronic Engineers (IEEE), which is known as wireless connectivity. The module can transmit b/g/n and utilizes a solitary band of 2.4 GHz frequency; therefore, it is good to send information to most of all accessible wireless routers. The board cuts down the size while still being ready to have 11 Input/output (I/O) for logging. It has both 3.3V and 5V for power and pulse width modulation (PWM) for getting fading signal. Figure 3 shows the Wemos D1 mini layout.

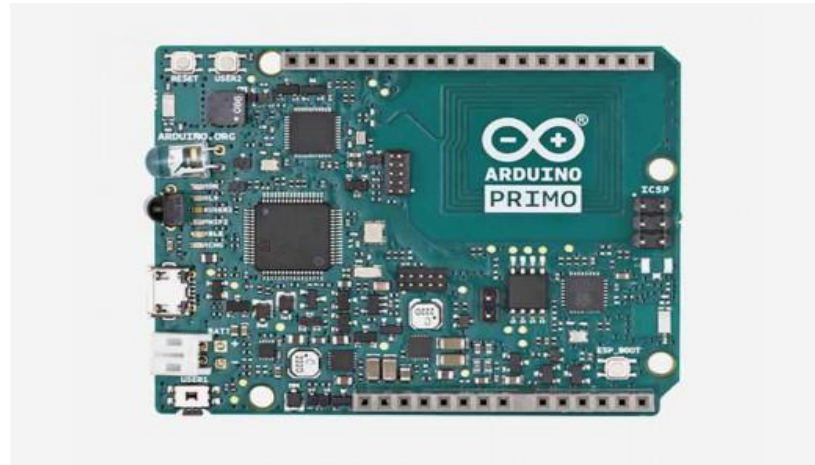


*Figure 3 Wemos D1 Min (wemos.cc)*

There are some different embellishments used to expand the utilization of boards.

**Arduino Primo:** In 2016, Arduino and Nordic Semiconductor cooperated to build Primo board. This board since then has become the primal core of microcontroller boards with some expressive features. Using the new processor from Nordic instead of ATmega, the boards come with built-in Wi-Fi (ESP8266 Chip), near field communication (NFC), Bluetooth Smart, infrared LED and a Li-ion/Li-Po battery charging circuit. It has up to 512KB of Flash Memory; therefore, it is a must-have microcontroller for large projects. This is a magnificent improvement since the size of Primo is identical to that of Uno Rev3. This board eliminated the use of ethernet shield for sending data online, granting a

remote control feature thanks to Bluetooth and with a built-in battery charging circuit, it could run independently everywhere. It costs 50\$ in the market at the moment. The only downside of this super board is its lack of developing support from Arduino and every infrastructure mainly coming from the open-source development. Furthermore, when it was advertised, Raspberry Pi 3 Model B was also released and since it was a complete computer, Primo lost the market and was forgotten. Figure 4 shows the Arduino Primo board.



*Figure 4 Arduino Primo (arduino.cc)*

**Liquid-Crystal Display (LCD):** LCD a popular technology used in, for example, computers and televisions to visualize images. In electronics, they are constructed with active matrix display grid like thin-film transistor (TFT) or passive matrix display grid. They are thin, flat and consume less power by blocking rather than emitting light. The common ones are 20x04 and 16x02 pixels, meaning 20 characters x 4 lines and 16 characters x 2 lines. It is simple to connect an LCD to Arduino by wires pins according to manual or using mediate Inter-Integrated Circuit (I2C).

**L293D Motor Shield:** The L293D Motor shield contains 4 DC Motor Input (M1, M2, M3 and M4) and 2 servos (Servo1, Servo2). It could be powered by an external source and easily connected to into any boards except Nano. This low-price shield is suitable for modelling cars, robots and motor controller. However, this shield could be replaced by a compact MX1508 dual-channel DC motor drive or an H-bridge. The setup will not be explained here since there is plenty of instructions online for research purposes.

## 4.2 Sensor

### 4.2.1 Introduction to Sensor Technology

Since its first appearance back in the nineteenth century, sensors have been a noteworthy part in everyday life as it shows in the greater part of electronic devices. The term is utilized to portray the capacity to distinguish physical values of items or atmosphere and response into digital values, which could be perused effectively. What makes the sensor becomes more important and common is its small size, sensitivity, and high accuracy. Sensors have various applications such as: temperature, pressure, gases, level and altitude (Moumita, 2016). What is more, it could be utilized as a part of the majority of the enterprises. Propelled innovation enables sensors to detect and record information without a moment's delay; the measurement precision of sensors has dropped from Nano to Pico meaning it could later be set anyplace undetected or traceable. The demand for electronics engineers is surely guaranteed if applicants are able to learn through-hole technology (THT) and surface-mount technology (SMT).

### 4.2.2 List of Air Quality Sensors

Throughout the practical work, plenty changes have been made to improve the model. The most significant change is identified with temperature reading: from DHT-22 to LM35 to BMP280. The accuracy is much higher while the surface area is decreased to one third. The following list presents all sensors that were used in the final model.

**Optical dust sensor GP2Y1010AU0F:** Dust particles are always the main factor in the air quality report. It is hard to detect whether or not there is dust in the area, which makes this optical dust sensor a mandatory component. The phrase "Optical" means it uses an infrared emitting diode to reflect light and the sensor is designed to let dust go through its hole and start counting the density. It consumes only 20mA while reading data with 0-7V input power. It could be operated in the temperature range of [-10; 65] C. Sensitivity depends on the conditions, but in overall, it varies from 0.35 to 0.65 V/ (0.1 mg/m<sup>3</sup>). Figures 5 explains the layout of the optical dust sensor.



■ Internal schematic

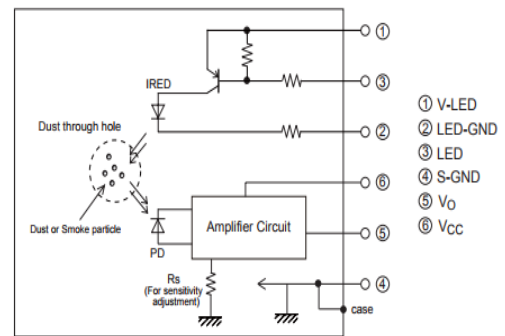


Figure 5 Left - Optical Dust Sensor (sparkfun.com). Right - Internal schematic

**Barometric pressure and temperature sensor BMP280:** This low-cost barometric pressure and temperature sensor is smaller than a cent; yet, it could do so many objectives (Lady, 2015). It could replace DHT-22 if humidity is not required and could also be used as an altimeter. Choosing the right combination depends entirely on the board compatibility to avoid duplicating pins and shortage. The BMP280 is a dedicated sensor and can be operated in three different modes: sleep, normal and forced. As the data is not measured continuously, sleep mode could enhance the battery's power and improve longevity.

**Air quality sensor MQ-135:** This air quality sensor is an extraordinary piece of equipment that could detect various factors such as ammonia ( $\text{NH}_3$ ),  $\text{NO}_x$ , alcohol, benzene and most notably, using for this particular project: carbon dioxide ( $\text{CO}_2$ ). Figure 6 shows the image of MQ-135, while Figure 7 shows the size of the BMP280.



Figure 6 Mq-135



Figure 7 BMP280 (sparkfun.com)

After identifying the primary sensors, there are some other notable ones which would benefit the usage of this thesis later on.



**Ultrasonic ranging sensor HC-SR04:** It's capable of sensing the non-contact obstacles in the distance from 2cm to 40cm. It functions similar to bats, using sonar principle for transmitting sounds and receiving echo back in order to calculate distance. This is a fascinating component used mainly in robotic, submarine, automation and in this thesis, it will act as the guidance for the third prototype, a mobile. Figure 8 shows the image of the HC-SR04.

**9g Micro Servo:** One of the best servo motor in the market for small projects, the 9g micro servo provides a 1.5kg of Torque and rotating 0.12 seconds per 60 degrees. The servo comes with arms and gear assortment. It works perfectly with the HC-SR04 sensor to provide an 180° range. Its lack of 360° continuous rotation is the only drawback of this servo. High torque model could be used in droids, bio-robotics and automation. Figure 9 shows the image of the 9g Micro Servo.

**Bluetooth HC-05:** The HC-05 is a Bluetooth 2.0 module which can be programmed by Arduino to work with Bluetooth-enabled devices. It transmits signals through serial communication and it can switch between Master mode and Slave mode for sending or receiving data. Due to that idea, on HC-05 there is TXD (sending) and RXD (receiving) pins which are used for above purpose. Figure 10 shows the image of the HC-05.

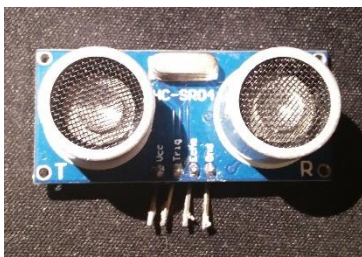


Figure 8 HC-SR04



Figure 9 9g Servo  
([hobbyking.com](http://hobbyking.com))



Figure 10 HC-05 ([Arduino-info.wikispaces.c](http://Arduino-info.wikispaces.c))

## 4.3 ThingSpeak

### 4.3.1 Introduction

ThingSpeak is an open-source IoT platform to transfer and store live information from devices remotely or through Local Area Connection (LAN). There are three fundamental

areas ThingSpeak focus on are gathering information from devices, dissecting and visualizing data with the help of integrated MATLAB and setting off a response modified by users. The information is stored online and could be downloaded for further research. Up until now, ThingSpeak is the most widely recognized decision for information logging. In any case, there are numerous more choices that give users more alternatives, for example, Xively by LogMeIn, ARTIK by Samsung, Blynk Mobile App, and so forth. Each service has its own particular key quality and users could pick base on projects' purpose.

#### 4.3.2 Benefits of ThingSpeak

According to ThingSpeak, making a free account will offer access to the platform with up to 3 million messages for every year and the information is logged every 15 seconds.

For higher classes, for example, Student, Home, Academic and Standard, ThingSpeak is accessible to be obtained with better features: 33 million messages for every year and the information is logged each second. ThingSpeak is appropriate for ventures, institutions, business and even government exercises.

#### 4.3.3 Set-up

ThingSpeak stage is clear and straightforward for clients. A new channel is made to store information and it incorporates fields for up to 8 unique sensors/reading at the same time. Each channel has a unique Application Programming Interface (API) key and Channel ID. These qualities will be utilized to impart amongst devices and ThingSpeak. Tweaking the channel by changing the settings: Name, Description, Field#, Metadata, Tags, Latitude, Longitude, Elevation, Link, and Video URL if necessary. The channel could likewise be set to be open or private and each field could be classified with unit, title and number of reading appeared. The most astonishing thing about ThingSpeak is its corporation with MATLAB to make Visualization and Analysis. These capacity will enable users to access to devices without owning MATLAB, which is genuinely costly. In the event that users are more into engineering mathematics and would prefer toward having just the information, the "Data Export" will download everything into .CSV format for additionally further research. Figure 11 shows the interface of ThingSpeak channels.



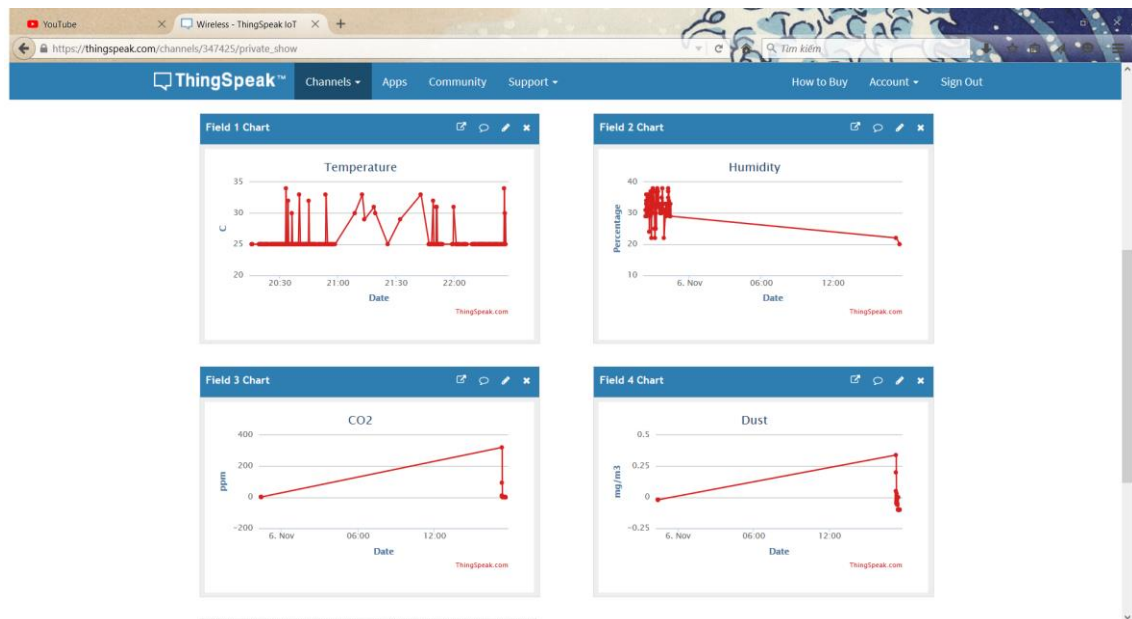


Figure 11 ThingSpeak Channel

## 4.4 Internet of Things (IoT)

### 4.4.1 Introduction

Internet of Things is everybody's imagination and creativity. The important thought is everything could be associated with the World Wide Web. The apparatuses as well as human, creatures and infrastructures. Furthermore, it gathers information and sends data to clients remotely, raising the entire new control level. IoT could save expenses and time since it acts independently and its application is anticipated to expanding up to 30 billion units by 2020. This is the glorious time of IoT and it will change the entire things.

### 4.4.2 Application

- **Smart configuration:** This definition includes any electronic-related objects that could be configured in order to ease the work of users. Smart home devices such as Air quality monitors, CCTV Cameras, Automatic lights, and thermostats are among top searches in the modern age. Smart-tech devices in the link of smartwatches, smartphones, and home assistants have been an irreplaceable piece of technology to everyone.
- **Industries:** IoT significantly improves the efficiency of industrial activities. With the ability to help administrators control and assess statistics from a remote

location, it gives the workload reduced and keeps the process operating smoothly. Assembly robots have since replaced man forces in heavy works with high precision and accuracy.

- **Transportation:** Since the rise of wheels and automobile, mankind has ambitiously thought about advanced technology in vehicles. With today's achievement, cars are run by electricity and tunnels are built for fast transportation. Top of that, all the movement will be controlled automatically. Predicting, preventing and sensing the surrounding area on the road, cars are smarter with the help of IoT. The projection of transportation is positive and people can finally dream of flying cars again.
- **Future application** – Possibilities are boundless and new inventions will give IoT a solid place in the age of advanced technology.

## 4.5 Calculation

### 4.5.1 Temperature and Humidity

Starting with the DHT-22, it uses a single-bus (1-wire) connection. Bus in this definition stands for a communication system that transferring data inside the computer (Aosong.com, n.d.). Single-bus means there is only one data line going through the DHT-22 and in this component, it forms as 8 bit binary and a simplified explanation for calculation according to the manual is presented below:

*Humidity = 8 bit integer data + 8 bit decimal data*  
*Temperature = 8 bit integer data + 8 bit fractional data*  
*Parity = 8 bit*  
*IF Humidity + Temperature = Parity => Release data*  
*IF Humidity + Temperature ≠ Parity => Re-calculate data*

When a pulse of information is collected, in total there are 40 data in which 32 are temperature and humidity details and 8 bit parity to compare. If the sum of 32 bit is equal to parity, then the value is accepted. If they are not equal, the calculation will be made again.

#### 4.5.2 Optical Dust Sensor

The linear regression was made by Chris Nafis to stimulate the characteristics of dust density based on Output Voltage. (Chris Nafis, 2012)

$$\text{Dust Density} = 0.17 \times \text{calcVoltage} - 0.1, \text{ with}$$

$$\text{calcVoltage} = \text{VoMeasured} \times \left(\frac{3.3V}{1024}\right)$$

- 3.3V is the input voltage for the sensor.
- 1024 is the maximum range of integer value reached at 3.3V.
- VoMeasured is the analog reading from pinout.

The formula is represented as a graph in Figure 12.

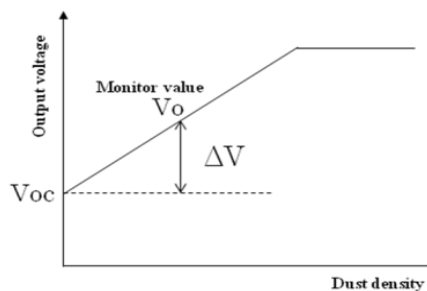


Figure 12 Output-Density (<http://www.sharp-world.com>)

#### 4.5.3 Digital Pressure Sensor

The compact sensor is utilized for many applications and it was executed in most smartphones. With the capacity to measure temperature, pressure and altitude, it has supplanted the first idea of the DHT-22 sensor and turns into the principal part of this project.

The datasheet released by **Bosch Sensortec GmbH** includes the calculation method and it was used in this application. (Appendix 1).

The digital signal from BMP280 is “t\_fine” which is the fine temperature value that could be doubled as the reference value to calculate pressure. (Bosch Sensortec,n.d. )

Following code is the calculation method for Temperature (T):

```
// Returns temperature in DegC, double precision. Output value of "51.23"
equals 51.23 DegC.
// t_fine carries fine temperature as global value
BMP280_S32_t t_fine;
double bmp280_compensate_T_double(BMP280_S32_t adc_T)
{
double var1, var2, T;
var1 = (((double)adc_T)/16384.0 - ((double)dig_T1)/1024.0) * ((double)dig_T2);
var2 = (((double)adc_T)/131072.0 - ((double)dig_T1)/8192.0) *
(((double)adc_T)/131072.0 - ((double)dig_T1)/8192.0) * ((double)dig_T3);
t_fine = (BMP280_S32_t)(var1 + var2);
T = (var1 + var2) / 5120.0;
return T;
}
```

Following is the calculation method for Pressure (p).

```
// Returns pressure in Pa as double. Output value of "96386.2" equals 96386.2
Pa = 963.862 hPa
double bmp280_compensate_P_double(BMP280_S32_t adc_P)
{
double var1, var2, p;
var1 = ((double)t_fine/2.0) - 64000.0;
var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
var2 = var2 + var1 * ((double)dig_P5) * 2.0;
var2 = (var2/4.0)+(((double)dig_P4) * 65536.0);
var1 = (((double)dig_P3) * var1 * var1 / 524288.0 + ((double)dig_P2) * var1) /
524288.0; var1 = (1.0 + var1 / 32768.0)*((double)dig_P1);
if (var1 == 0.0)
{
return 0; // avoid exception caused by division by zero
}
p = 1048576.0 - (double)adc_P;
p = (p - (var2 / 4096.0)) * 6250.0 / var1;
var1 = ((double)dig_P9) * p * p / 2147483648.0;
var2 = p * ((double)dig_P8) / 32768.0;
p = p + (var1 + var2 + ((double)dig_P7)) / 16.0;
return p;
}
```

Using **double** variable type means the value returns as a real number without shorten or rounded.

## 5 Experimentation

Everything starts with the basic concept and this thesis is not an exception. During the experimentation, specialized terms will be explained and carefully constructed from scratch to hatch!

## 5.1 Environmental Monitoring Board (EMB)

The submitted plan during May 2017 was to create an EMB at Myyrmäki Campus. However in the final decision, the testing was removed from university and was done in the author's house instead. Even though the environment was changed, the experiments were still continued normally. Planning and building the prototype was done throughout the summer and completed in August. Continuous development was carried out during September, October, and the final product was completed in November. Unfortunately, since there was not enough spare materials, previous prototypes were dismantled to build new ones. Because there were many steps, the whole process was divided into 3 stages: **testing, online server and prototype**. Each stage required different components and knowledge.

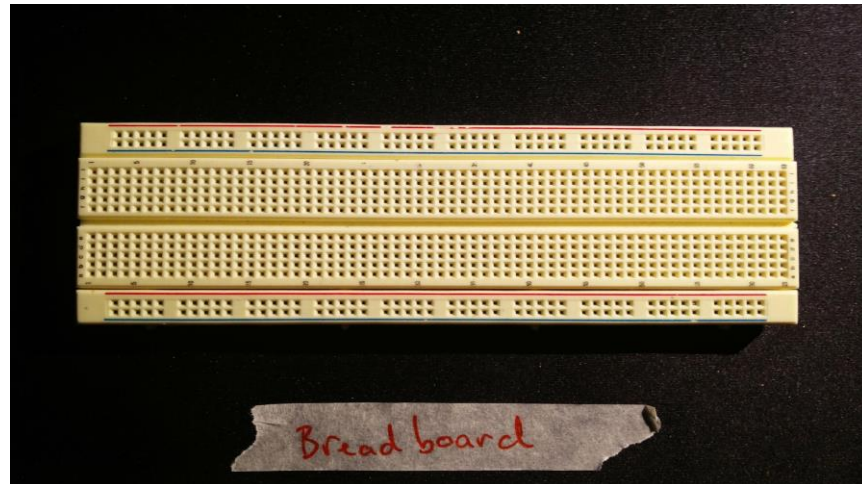
### 5.1.1 Testing

A testing stage is mandatory for every electronic-related project. By testing, accidents could be avoided and the schematic is planned beforehand. It is so important to test everything beforehand, bearing in mind that this is a complicated process and a small mistake could lead to failure of the system. During the thesis, a couple of components were destroyed and rendered unusable due to short circuit, power failure or intensive workload. The following sections describe the basic components used in this project.

#### 5.1.1.1 Breadboard

Breadboards are solderless electronic boards, making wiring components becomes effortless. They are great for building templates and temporary circuit boards. Using them could point out the problems in the system and power shortage if occurred. However, it is not suitable for the final product since its lack of professional look and loose connection since it is solderless. Using Printed circuit board (or PCB) is recommended. Breadboards are usually rectangular and consist of two power rails both side, DIP (Dual In-line Packages) support in the middle and terminal strips running vertically to the board.

Understand terminal strips is essential and the first-hand user could be confused by how it works. The simplest rule is: Vertical pins are the same + horizontal pins are different. Figure 13 shows the most common breadboard.



*Figure 13 830 Tie-points Breadboard*

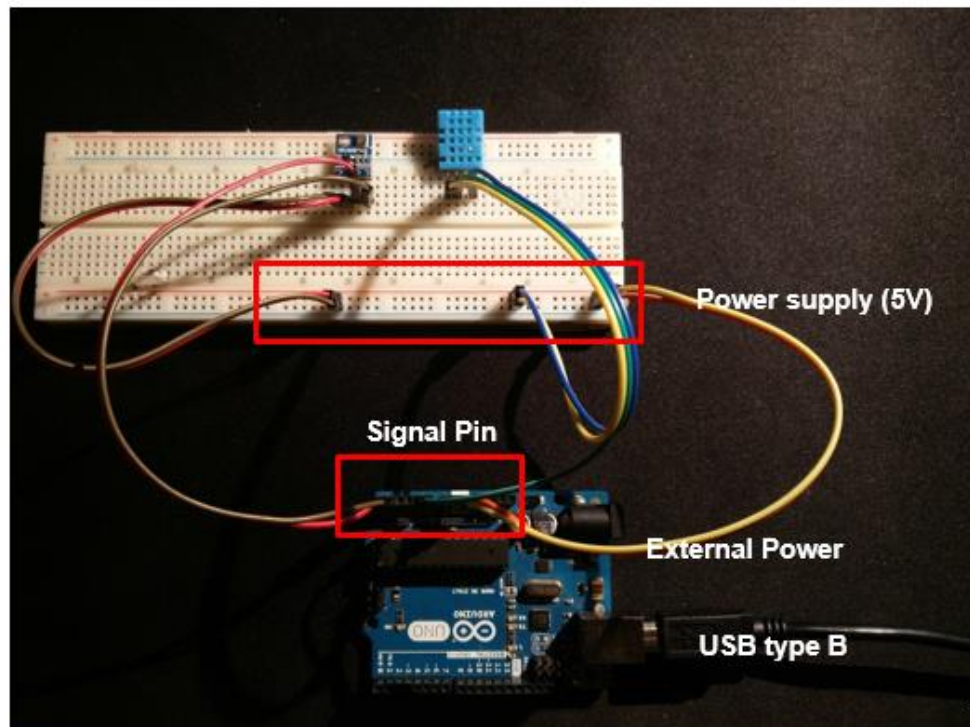
On power rails there is positive (+) and negative (-) connection, sensors are capable of 3-5V so input power is maximum 5V. Arduino is also powered by 6-12V through VIN thank to the voltage regulator on board.

#### 5.1.1.2 Jumper wire

These wires are the fastest way to work with both breadboards and PCB to run the connection. Having a couple hundred of them will speed things up and they are inexpensive. There are male and female wires to extend, hook up to pins or plug into the board. They come with different color, therefore, it helps to mark connection easier. In the final prototype, every jumper wires will be removed and soldered into a prototype Printed circuit board (PCB).

### 5.1.1.3 Testing process

On Arduino board, there is a USB type B connector and an external power supply. Both are used to power the board but the USB is also used to upload the codes to board and control sensors. Figure 14 shows how it works:



*Figure 14 Testing process*

- Connecting 5V & GND (Ground) pin from Arduino to power rail as the power source for sensors.
- Connecting VCC (Power) & GND pin from sensors to power rail to receive the power.
- Connecting signal pin from sensors to Arduino for transmitting data (Analog & Digital) and displaying on a computer.

Instruction manual for the sensor could be found online and since the Arduino is an open-source platform, it is widely used in many projects and the code could be easily accessed.

Some sensors require an additional library, which is a programmed command to guide them how to achieve the data. MQ sensors also require mandatory heating to determine the R-zero value (or the Calibrated point). This value is stabilized after 24



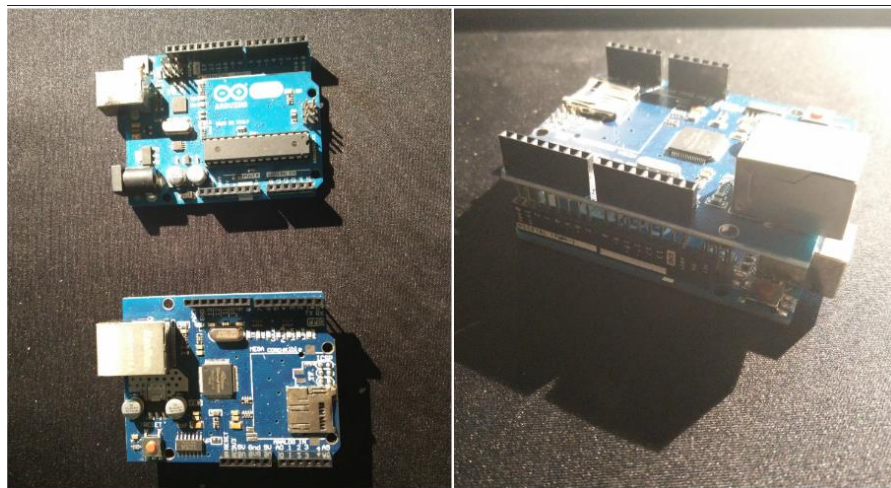
hours of heating and could be used as the base of the signal. When the testing is completed and every property is checked, it is time to build the station.

### 5.1.2 Online Server

In the previous point, the testing is shown offline data and could be read via computers only. In this step, everything will go online and data could be read via a cloud server, in this case: ThingSpeak and it is public to observers. There are two ways to make it online: Ethernet and Wireless. They come as an add-on for Arduino and increase independence to the system.

#### 5.1.2.1 Ethernet Shield V1

In Environmental Engineering courses, Arduino and Ethernet Shield were taught during the third-year program and it was the only IoT-related course of the whole curriculum. Many students have stopped studying about it and this is an opportunity to continue what had been left out. Ethernet Shield's size is as same as Arduino Uno (Left) and it could be snapped on easily (Right). With this shield, sensors' data could be uploaded to ThingSpeak through LAN cable. Figure 15 shows the size of Ethernet shield and how it connects to the Arduino Uno.



*Figure 15 Ethernet Shield V1*

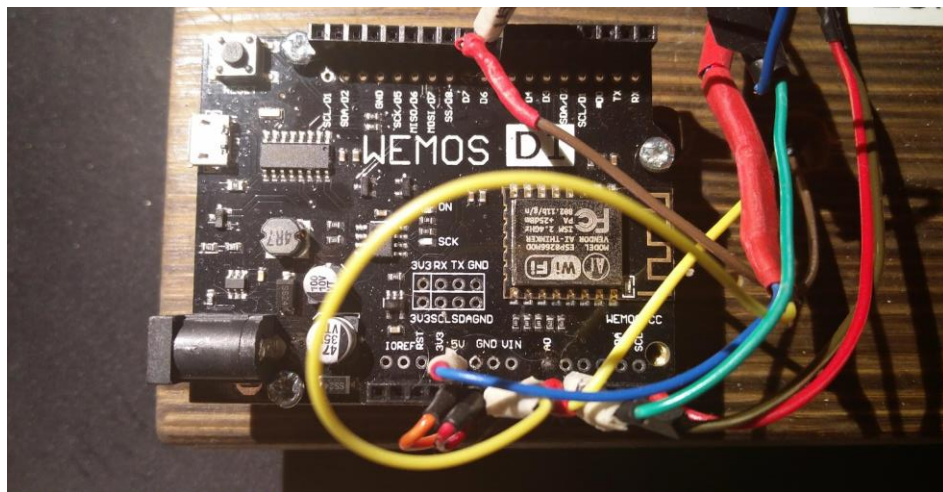
Setting up Ethernet Shield with ThingSpeak puts some more work, it requires API Key and Channel ID from ThingSpeak and a unique MAC address labeled on the back of Shield. Inserting this information into the code and it will create a gateway to



communicate between Server and EMB. Luckily, ThingSpeak has its own library and example code that suitable for both Ethernet and Wireless Shield so programming knowledge is at the intermediate level. The C++ Command is shown in Appendix 1 for further study.

### 5.1.2.2 Wireless

Going wireless is different and more difficult than Ethernet. It does not contain MAC address, which means this setup is manually from A to Z. In overall, It needs either Wireless Shield, Wireless Module or a Single-board with built-in Wireless. The most common one is called “ESP8266”, there are many different versions of it and it even could replace the single board and act as a standalone. However, the configuration for this chip is not simple as plug and play, therefore, in this thesis, the Wemos D1 R2 board is chosen for easy setup. Wemos acts similar to Arduino but since this is not an official board, firstly the board has to be recognized by the IDE and there is an instruction manual for this task. Figure 16 shows the Wemos in experiments. The C++ command is shown in Appendix 2 for further study.

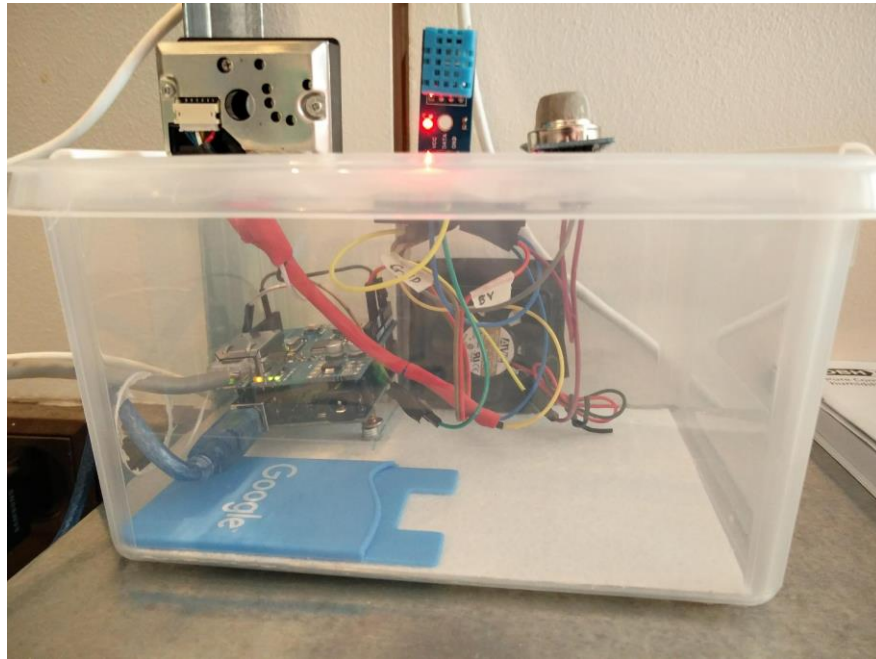


*Figure 16 Wemos in the final setup*

### 5.1.3 First Prototype

Initially, the most practical and common way to do this is making a transparent box to store every components. This would help keeping the device in place and convenience in showing students the inside point. The only thing is staying outside of the box are

sensors, because they need to be at the same atmosphere to sense air quality level. Figure 17 shows the first prototype.



*Figure 17 First prototype*

Two major works are hardware and software setup.

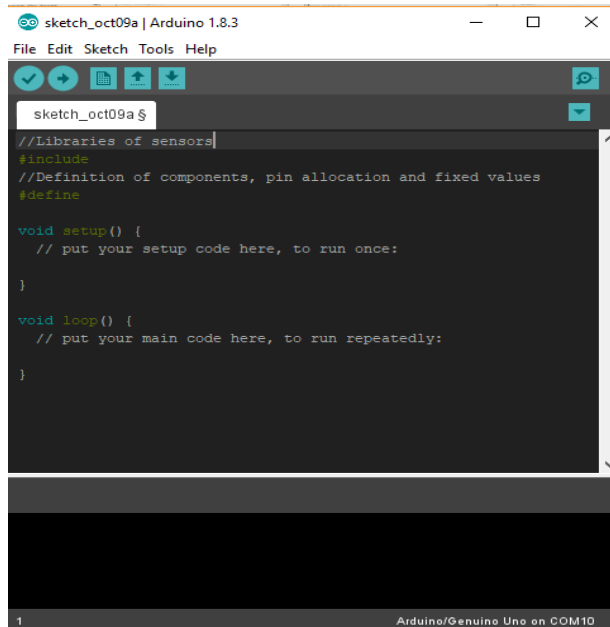
#### 5.1.3.1 Hardware Setup

Wiring from sensor pins to Arduino pins and calibrate the R-zero. Since there are more than two sensors and the number of sensors could be increased, using a small piece of breadboard as the middle point would be a good solution. All of the sensors run at 5V so Power and Ground pins could be connected to the same line on the breadboard, therefore, only two wires were needed to connect to Arduino. The Data pin from sensors was connected to the digital or analog pin depending on the schematic design. Bolts and nuts were used to stabilize the sensors on top of the box and to secure Arduino to the bottom.

### 5.1.3.2 Software Setup

After every wire was connected, it is time to use Arduino IDE to configure the data and send it online. This would be difficult for beginners but the more work is done, the easier it gets to understand the concept of IDE. It is just another one of the programming course.

Figure 18 shows the Arduino IDE in default settings.



```

sketch_oct09a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_oct09a$
//Libraries of sensors
#include
//Definition of components, pin allocation and fixed values
#define

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

1
Arduino/Genuino Uno on COM10

```

Figure 18 Arduino IDE

The default interface includes command “void setup() []” and “void loop() []” for setup code and repeated action.

Programmed libraries and definition for pinouts could be included in the beginning in order to help IDE identify the correct components.

In order to make the code works, the function has to be either from default or installed libraries.

Understanding the meaning of function would help in programming, and Table 1 shows some of the common lines in this thesis, while Table 2 shows the required library.

Table 1 Default C++ function

Default	
#include	Include required libraries
#define	Give name to a constant
//	Comments syntax to explain the code
Setup()	Uses to initialize or start. Only runs once
Loop()	Uses to control the components Run as a loop
digitalRead()	Read the value of digital pin
digitalWrite()	Write the value of digital pin
analogRead()	Read the value of analog pin
analogWrite()	Write the value of analog pin
Delay()	Pause the program for amount of time
Char()	Change a value to a character
Float()	Change a value to a floating-point
Int()	Change a value to a integer
String()	Create a string
Serial.print()	Print text on demand to display

Table 2 Libraries

Libraries		Author	Github
DHT.h	DHT sensors	Adafruit	/adafruit/DHT-sensor-library
ESP8266Wifi.h	Wifi support	Ivan Grokhotkov, 2014 Published by Free Software Foundation	/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/ESP8266WiFi.h
MQ135.h	MQ-135 sensor	G.Krocker Licensed by GNU GPLv3	/GeorgK/MQ135/blob/master/MQ135.h
LiquidCrystal_I2C.h	LCD	DFROBOT	/fdebrabander/Arduino-LiquidCrystal-I2C-library/blob/master/LiquidCrystal_I2C.h

The key command to active Ethernet shield and Internet lies in these codes, which MAC address is unique labeled under the shield:

```
// Local Network Settings
byte mac[] = { 0x**, 0x**, 0x**, 0x**, 0x**, 0x** }; // Must be unique
// ThingSpeak Setting
char thingSpeakAddress[] = "api.thingspeak.com"; // Connect to ThingSpeak

unsigned long myChannelNumber = *****; //found in channel setting
const char * myWriteAPIKey = "*****"; //found in channel setting
const int updateThingSpeakInterval = 15 * 1000; // 15 seconds every new data
```

#### 5.1.4 Second prototype

After one month running the first prototype, eventually, there were some suggestions and ideas to improve the function and the look of EMB. A next concept was selected by using a kitchen breadboard, which is a very popular item in both culinary and electronics. Before the invention of prototype breadboard, students and researchers had been using kitchen breadboard to run the wires since it has large surface and depth, making it ideal for projects. Being inspired by that, the final prototype is built on the breadboard and thus, combining both past and future technology together. One more improvement is the EMB is now connected wirelessly via WIFI. By using Wemos D1R2 board, it helps to eliminate the Ethernet Shield which took lots of space and requires LAN Cable. The EMB is also equipped with LCD Display, which shows all air quality data without looking up ThingSpeak or connecting to the computer. Figure 19 shows the second prototype functioning.

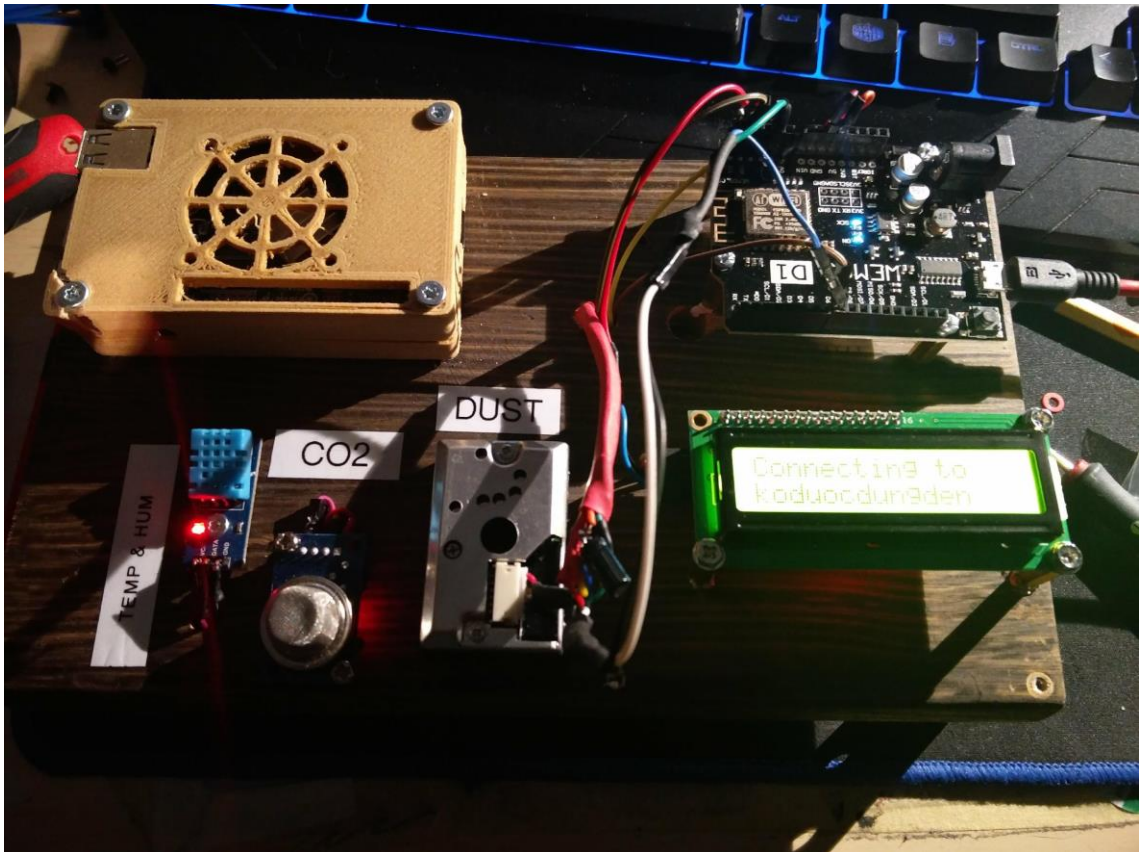


Figure 19 Second Prototype

The new board works perfectly in the setup even though there was no guarantee it would be compatible with sensors and LCD. Some lines of codes were changed to make it work. This is simple as connecting to WIFI and it takes 20-30 seconds to ready for uploading data.

```
// replace with your channel's thingspeak API key and your SSID and password
String apiKey = "apikey";
const char* ssid = "WIFI_NAME";
const char* password = "WIFI_PASSWORD";
const char* server = "api.thingspeak.com";
```

Continuously, the LCD also has to be recognized and set-up. In this thesis, the 16x02 LCD is used.

```
//LiquidCrystal Setup
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a 16
chars and 2 line display
```

Here is the code part to read the data from sensors:

```
//DHT Temp & Humid
float h = dht.readHumidity();
float t = dht.readTemperature();
```



```

// CO2 sensor
float ppm = gasSensor.getPPM();
//Dust sensor
digitalWrite(ledPower,LOW); // power on the LED
delayMicroseconds(samplingTime);

voMeasured = analogRead(measurePin); // read the dust value

delayMicroseconds(deltaTime);
digitalWrite(ledPower,HIGH); // turn the LED off
delayMicroseconds(sleepTime);

// 0 - 3.3V mapped to 0 - 1023 integer values
// recover voltage
calcVoltage = voMeasured * (3.3 / 1024);

// linear equation taken from http://www.howmuchsnow.com/arduino/airquality/
// Chris Nafis (c) 2012
dustDensity = 0.17 * calcVoltage - 0.1;

```

In `loop()` function, using these command to print sensors' data directly to the LCD. The function "`lcd.setCursor(x, y)`" represents the position of characters with x is the number of row and y is the number of column.

```

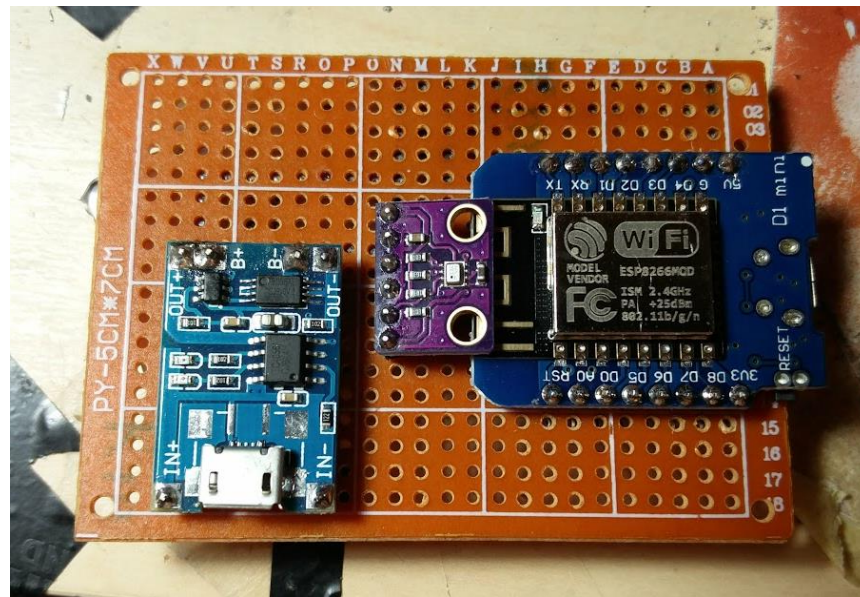
lcd.setCursor(0,0); // 1st row, position 0
lcd.print("Temp=      C  ");
lcd.setCursor(6, 0); // 1st row, position 6
lcd.print(t); //Print temperature
lcd.setCursor(0,1); // 2nd row, position 0
lcd.print("Humi=      %  ");
lcd.setCursor(6, 1); // 2nd row, position 6
lcd.print(h ); //Print humidity
delay(5000);
lcd.clear();
lcd.setCursor(0,0); // 1st row, position 0
lcd.print("Dust=      mg/m3");
lcd.setCursor(6, 0); // 1st row, position 6
lcd.print(dustDensity); //Print Dust density
lcd.setCursor(0,1); // 2nd row, position 0
lcd.print("CO2 =      ppm");
lcd.setCursor(6, 1); // 2nd row, position 6
lcd.print(ppm); //Print CO2 value

```

### 5.1.5 Final prototype

The final EMB is a new concept with every single components were replaced to reduce the size while enhancing the possibility of an Environmental Monitoring System.

For the final version, the Wemos D1 mini and BMP280 was used and everything worked perfectly. Signals were sent to ThingSpeak appeared normally and a closure was made to make the commercial version. The size of PCB is only 70 mm x 50 mm and it includes an 800 mAh Li-Po rechargeable battery. Figure 20 shows the final prototype on a PCB.



*Figure 20 Final prototype*

## 5.2 The vehicle

The EMB would have been a great conclusion for this thesis; yet, there is more concealed potential for this project to be finished. The board lacks portability and therefore, a more enhanced solution would be putting the board on wheels controlled either by Bluetooth or ultrasonic sensors.

The experiment was first arranged in August and the improvement was begun in September. There were a wide range of adaptations of the vehicle and changes were made to optimize the solution. In conclusion, an ultrasonic sensor, a Bluetooth module and an accelerometer were utilized to decide which one is the final version. The EMB is set on board and with a separate power source, the two devices could run autonomously for several hours.

### 5.2.1 Motor Shield L293D

The major contribution to this vehicle is the motor shield and it helps to reduce the workload tremendously. It consists of 2 L293D motor control chips and 4 H-bridges, meaning that the shield could power up to 4 DC Motor or 2 servos simultaneously. It could be powered up to 12V and 600mA for each DC gear motor, which is runnable at 6V and 200 mA. They come in handy and inexpensive; therefore, more than one

prototypes were made for demonstration. However, a major problem with this kind of application is its stability, for such a combined project, the power used not only for the motor but also for Arduino and sensors. Power distribution is a large problem in electronic devices, and a simple idea is using different power sources for them, which could reduce interference and noise that affects the reading of sensors.

The shield fits pleasantly to Arduino Uno and is controlled by Adafruit Motor Shield library. For smaller boards, an option would be L298 H-Bridge which controls 2 DC motor simultaneously.

### 5.2.2 Ultrasonic sensor and 9g Servo

The ultrasonic sensor is set at the front with a clear vision to transmit sound waves and decide the reaction to maintain a strategic distance from obstacles. Keeping in mind the end goal is to build instant response and high accuracy, 3 sensors could be set and the information is consolidated. Making the rule for the setup could be tedious. Rather than making more issues, by including the 9g Servo beneath of just a single sensor, the duo could fill in as a distance scanner while pivoting 180 degrees. This arrangement helps the vehicle to predict beforehand next movements simultaneously and senses the surrounding area. The DC motors' power range is 5-12V, therefore, using a single 9V battery is enough to handle the different speeds. Figure 21 shows the testing of the vehicle.

The ultrasonic sensors comes with 4 pins: VCC, GND, Trig and Echo. VCC is connected to 5V pin of Arduino Motor shield while Trigger and Echo are connected to A4 and A5 pins. These pins read the sound wave frequency and returns as analog signals.



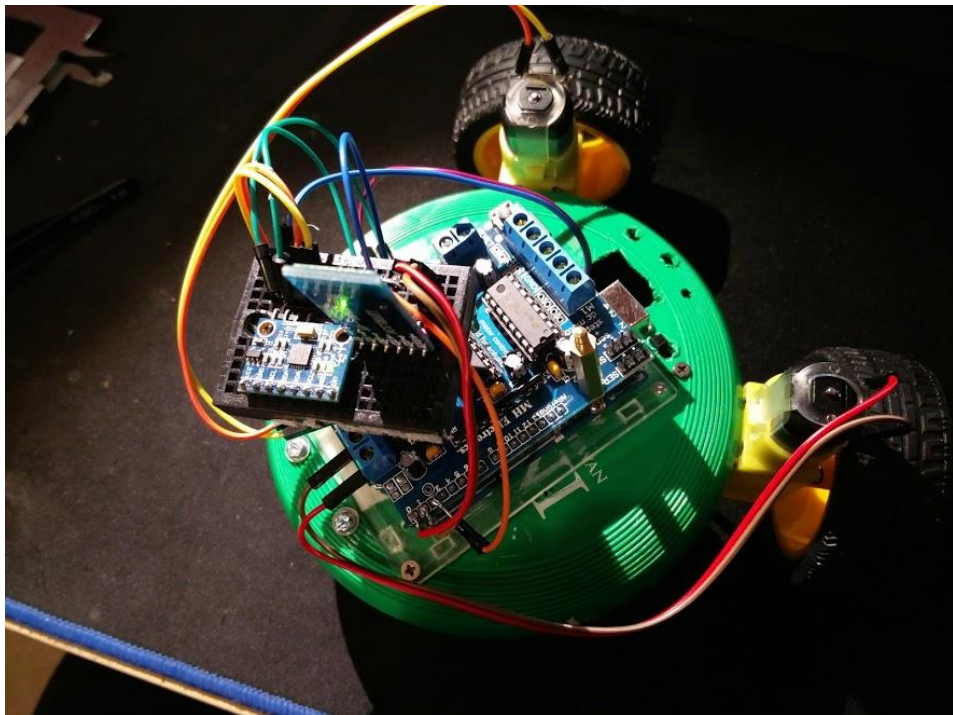
Figure 21 First vehicle



### 5.2.3 Bluetooth Module

Utilizing the HC-05 Bluetooth, the vehicle can run by means of serial communication from a smartphone through an application. The application characterizes the key for running forward, reverse, turning left and right. These keys are modified in the code to rule the activity of DC motors. These engines are bipolar, in this manner, it does not characterize positive and negative side. Figure 22 shows the second version of the vehicle.

Adafruit made the library called "AFMotor" and it was utilized as a part of this case for programming. Running forward and in reverse is just as setting the current to the abnormal state and switching the power pin. By utilizing that logic, when turning left or right, one engine is set to stop while the other one is set to the highest speed. Each conceivable circumstance should be possible by altering the speed and the course of engines.



*Figure 22 The second vehicle*

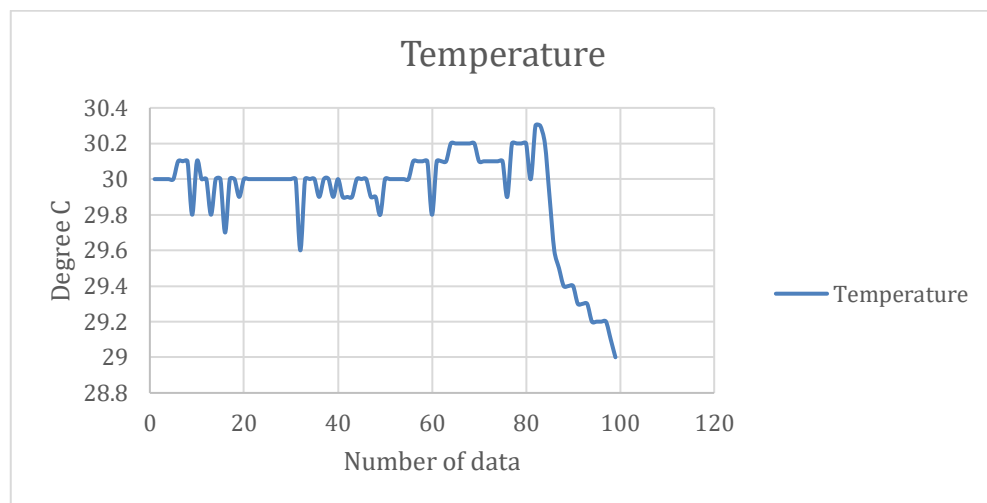
## 6 Results

### 6.1 EMB

From ThingSpeak, the data can be exported to .CSV file and plotted.

Figure 23 below shows the changing pattern of indoor temperature in degrees of Celsius during a period of time.

Overall, the temperature ranges from 29.8 to 30 and the median value is 30 degree Celsius. The curve shows in fact, the BMP280 was collecting new data every second. On the other hand, since ThingSpeak only takes input every 20 seconds, the curve's change is not dramatic. Without any major interrupts, the data is shown reliable and stable.



*Figure 23 Temperature reading*

The graph below shows in hPa the changing pattern of atmospheric pressure of indoor location during a period of time.

In overall, the pressure is always a stable value since the prototype has not been moved.

The pressure is computed based on the algorithm from Bosch

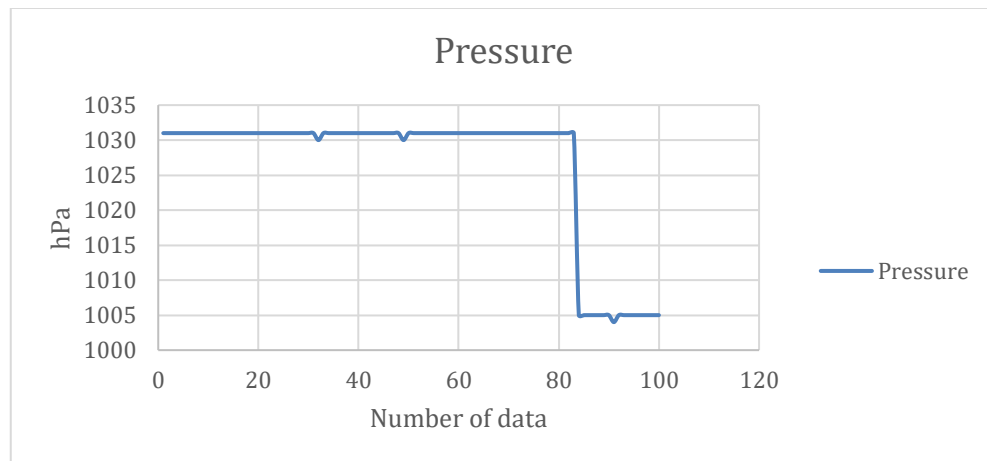


Figure 24 Pressure reading

The graph below shows in Part per millions (PPM) the changing pattern of Carbon Dioxide of indoor condition.

MQ-135 sensor is used to calculate the air quality level and it requires a 24-hour calibration. Once the setup is done, the reading is stable and ready for observation. The value ranges from 400 to 700 PPM, which is acceptable for indoor environment. When the testing is done, there is also a source of exhaust from soldering tool and direct sunlight in front of the test bench.

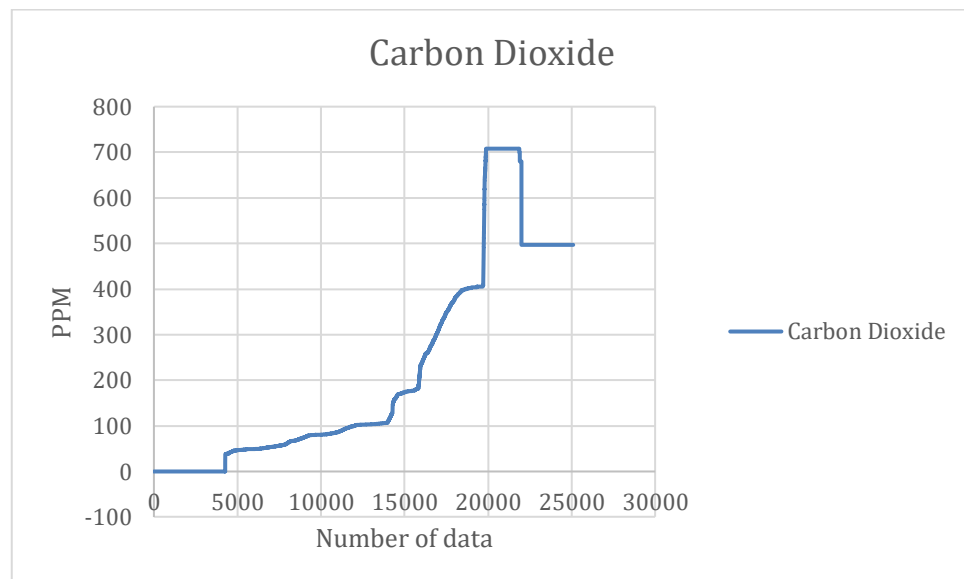


Figure 25 Carbon Dioxide reading

## 6.2 The vehicle

The vehicle is able to run with a Bluetooth-controlled device and autonomously. There is a slight difference in speed: the Bluetooth module carries less weight and the vehicle does not have to calculate the distance to decide the direction; therefore, it could run and react faster. On the other hand, using the ultrasonic sensor and a 9g Servo added

consuming power to the battery, which also increases the weight and lowers the duration. However, the sensor is a tool for showcase and explains the usage of mathematics in automation. Thus, it was used in the final version. The vehicle was ideal for a project, and it was an extra part from this thesis.

## 7 Conclusion

The project was done from April 2017 to March 2018 with a 3-month break in the middle. The final results show it is possible to make a compact environmental monitoring system and it is scalable to a commercial version. Even though in the end, there is not much of an innovative breakthrough, there is always room for improvement.

The finding from this project is that it does not require expensive equipment and sensors to gather reliable information. Furthermore, every single issue could be fixed through studying the datasheet or going the way around. Due to the fact that thesis was done in the field of environmental engineering, it was much harder to understand everything in the first try.

Finally, the study's limitation is acknowledged in terms of financial issues, materials and time management. Additionally, it was not possible to have a standard testing environment with this continuous change of thesis's plan. By missing this step, the comparison is done mostly by using the equipment from Vernier Software & Technology which is available in the laboratory. However, with the limited budget and enough brainstorming, the process was completed with 3 prototypes and 2 vehicles. This project is usable for future development and the setup is shared in Appendix 1-4 for further study. This is so far the greatest learning experiment and it was indeed not easy. The road map has seen some major changes due to the unexpected participation of accelerators and Slush. These events bring up new ideas and changes, but there was plenty of time lost to balance everything.

## References

Hernando Barragán, 2016. The Untold History of Arduino. [Online] Available at: <http://arduinohistory.github.io/> [Accessed 29 September 2017]

Moumita Das, 2016, Grasshopper.iics. Intel Edison Based Plant monitoring and Watering System for Agro-IoT with Matlab Driven Real Time Data Analytics over ThingSpeak Channel. [Online] Available at: <https://www.codeproject.com/Articles/1113749/WebControls/> [Accessed 20 Aug 2017]

Arduino. Introduction to the Arduino Board [Online] Available at: <https://www.arduino.cc/en/Reference/Board> [Accessed 29 September 2017]

Aosong.com, n.d. Temperature and humidity module DHT11 Product Manual. [Online] Available at: <https://akizukidenshi.com/download/ds/aosong/DHT11.pdf> [Accessed 10 Aug 2017]

ThingSpeak. License Options [Online] Available at: <https://thingspeak.com/prices> [Accessed 20 October 2017]

Optical Dust Sensor – GP2Y1010AU0F. [Online] Available at: <https://www.robot-r-us.com/vmchk/sensor-dust/optical-dust-sensor-gp2y1010au0f.html> [Accessed 15 October 2017]

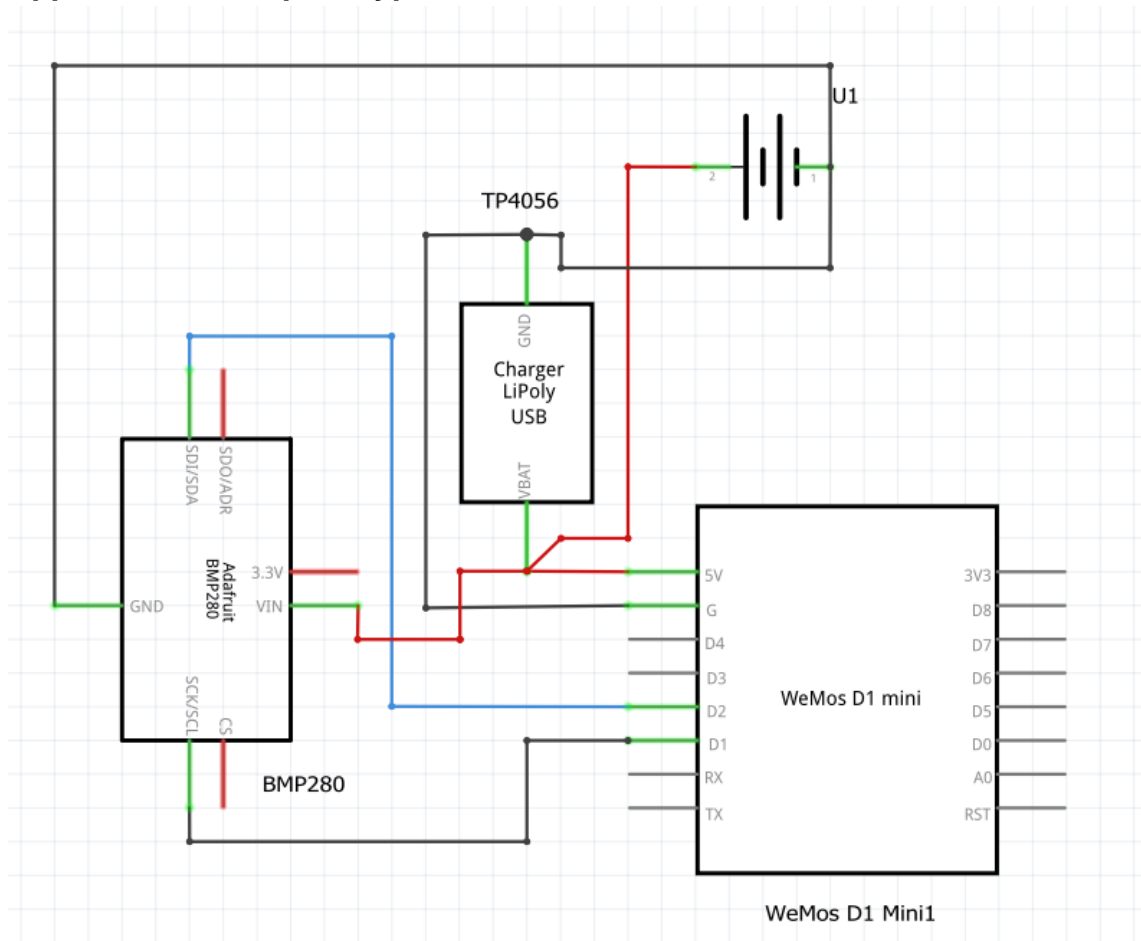
Lady ada, 2015. Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout. [Online] Available at: <https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout/overview> [Accessed 15 October 2017]

Chris Nafis, 2012. Air Quality Monitoring. [Online] Available at: <http://www.howmuchsnow.com/arduino/airquality/> [Accessed 24 November 2017]

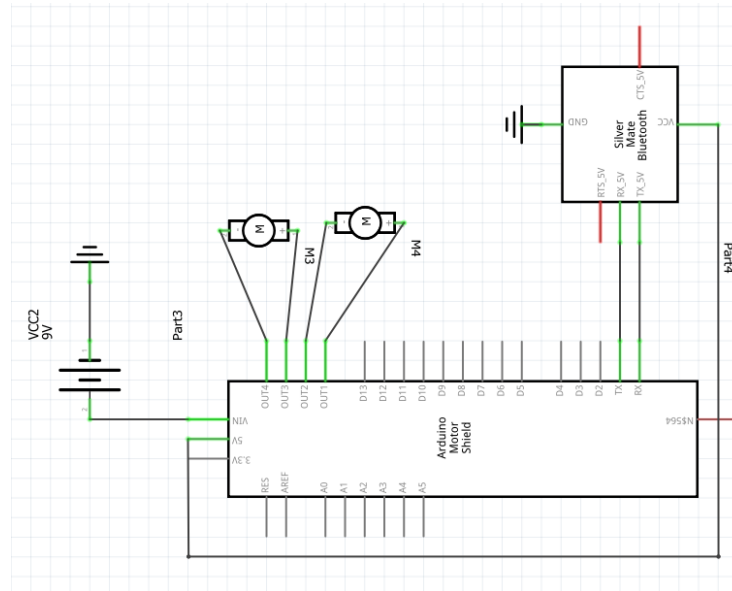
Bosch Sensortec,n.d. Datasheet BMP280 Digital Pressure Sensor. [Online] Available at: <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf> [Accessed by 5 January 2018]

## Appendices

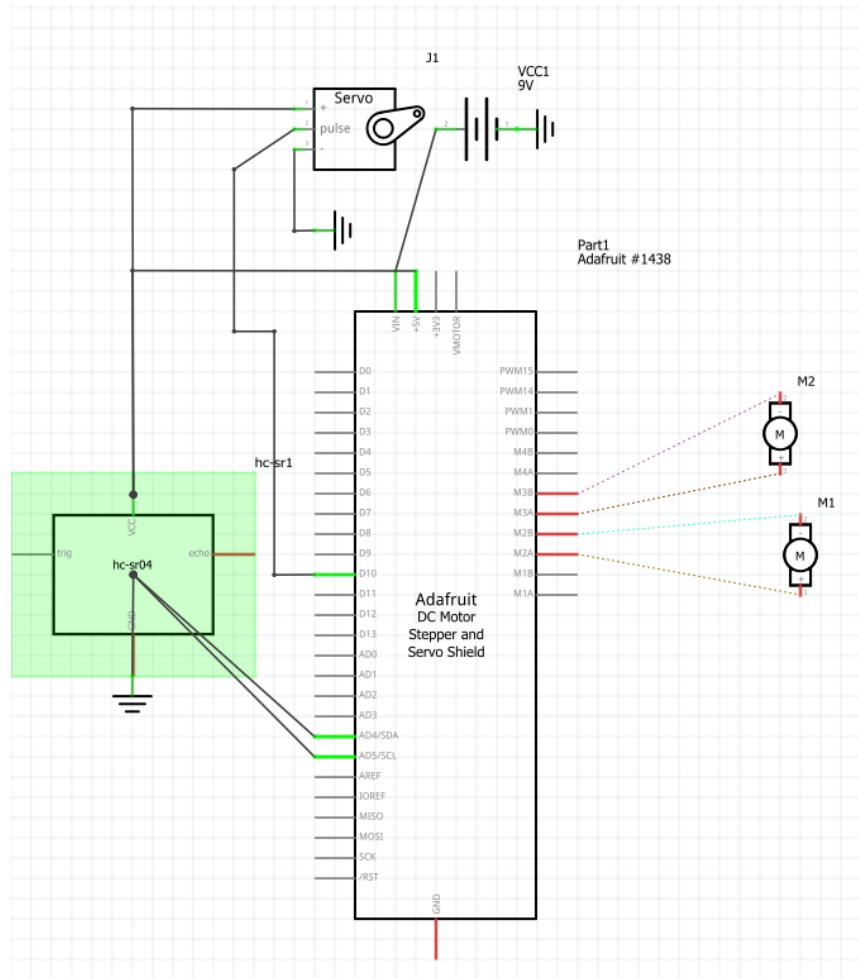
### Appendix 1. Final prototype



## Appendix 2. Bluetooth Module Diagram



Appendix 3. The ultrasonic sensor Diagram





## Appendix 4. Command for EMB Offline (Final version 20.09.17)

```

//LCD Display
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD
interface pin
// with the arduino pin number it is connected to
const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

/* MQ135
Library Repository : https://github.com/ckalpha/MQ135
Author : Damrongwit Nusuk
Email : jack@racksync.com
Website : http://www.racksync.com
*/

#include "MQ135.h"
#define ANALOGPIN A5 // Define Analog PIN on
Arduino Board
#define RZERO 206.85 // Define RZERO Calibration
Value
MQ135 gasSensor = MQ135(ANALOGPIN);
/* DUST SENSOR
int measurePin = A4;
int ledPower = 9;

unsigned int samplingTime = 280;
unsigned int deltaTime = 40;
unsigned int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

/* DHT_11
/*Weather Foreceasting using DHT-22 and Mediatek
LinkIt One*/

#include <DHT.h>;

//Constants
#define DHTPIN 8

ed = analogRead(measurePin);

delayMicroseconds(deltaTime);
digitalWrite(ledPower,HIGH);
delayMicroseconds(sleepTime);

calcVoltage = voMeasured*(5.0/1024);
dustDensity = 0.17*calcVoltage-0.1;

if ( dustDensity < 0)
{
dustDensity = 0.00;
}
Serial.print("Dust Density = ");
Serial.print(dustDensity);
Serial.println(" mg/m3");

//MQ135
float ppm = gasSensor.getPPM();

digitalWrite(13,HIGH);
Serial.print("CO2 = ");
Serial.print(ppm);
Serial.println(" ppm");
lcd.setCursor(0,0);
lcd.print("Temp= C");
lcd.setCursor(6, 0);
lcd.print(temp);
lcd.setCursor(0,1);
lcd.print("Humi= %");
lcd.setCursor(6, 1);
lcd.print(hum);
lcd.setCursor(0,2);
lcd.print("Dust= mg/m3");
lcd.setCursor(6, 2);
lcd.print(dustDensity);
lcd.setCursor(0,3);
lcd.print("CO2= ppm");
lcd.setCursor(6, 3);
lcd.print(ppm);
delay(1000);
}

```

## Appendix 5. Command for EMB Online (Final version 23.10.17)

```

//Libraries for DHT, ESP8266, MQ135 and LCD
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <MQ135.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//LiquidCrystal Setup
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD
address to 0x27 for a 16 chars and 2 line display

// replace with your channel's thingspeak API key and
your SSID and password
String apiKey = "*****";
const char* ssid = "*****";
const char* password = "*****";
const char* server = "api.thingspeak.com";

// Const
int measurePin = 0;
int ledPower = 8;

int samplingTime = 280;
int deltaTime = 40;
int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

// DHT_11
// *Weather Forecasting using DHT-22

#define DHTPIN D7
#define DHTTYPE DHT-22
DHT dht(DHTPIN, DHTTYPE);

#define ANALOGPIN 0 // Define Analog PIN on
Arduino Board

#define RZERO 206.85 // Define RZERO Calibration
Value

MQ135 gasSensor = MQ135(ANALOGPIN);

WiFiClient client;

void setup()
{
  lcd.init(); //initialize the lcd
  lcd.backlight(); //open the backlight
  Serial.begin(115200);
  delay(10);

  WiFi.begin(ssid, password);

  Serial.println();
  Serial.println();
  Serial.print("Connecting to: ");
  Serial.println(ssid);
  lcd.setCursor(0,0);
  lcd.print("Connecting to ");
  lcd.setCursor(0,1);
  lcd.print(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  delay(3000);
}

void loop()
{
  //DHT
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float ppm = gasSensor.getPPM();
  digitalWrite(ledPower,LOW); // power on the LED
  delayMicroseconds(samplingTime);

```

```

    voMeasured = analogRead(measurePin); // read the
dust value

    delayMicroseconds(deltaTime);
    digitalWrite(ledPower,HIGH); // turn the LED off
    delayMicroseconds(sleepTime);

    // 0 - 3.3V mapped to 0 - 1023 integer values
    // recover voltage
    calcVoltage = voMeasured * (3.3 / 1024);

    // linear equation taken from
http://www.howmuchsnow.com/arduino/airquality/
    // Chris Nafis (c) 2012
    dustDensity = 0.17 * calcVoltage - 0.1;

    delay(1000);
    if (isnan(h) || isnan(t))
    {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    if (client.connect(server,80)) {
        String postStr = apiKey;
        postStr += "&field1=";
        postStr += String(t);
        postStr += "&field2=";
        postStr += String(h);
        postStr += "&field3=";
        postStr += String(ppm);
        postStr += "&field4=";
        postStr += String(dustDensity);
        postStr += "\r\n\r\n";

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
        client.print("Content-Type: application/x-www-form-
urlencoded\n");
        client.print("Content-Length: ");
        client.print(postStr.length());
        client.print("\n\n");
        client.print(postStr);

```

```

        Serial.print("Temperature: ");
        Serial.print(t);
        Serial.println(" degrees Celsius");
        Serial.print("Humidity: ");
        Serial.print(h);
        Serial.println(" %");
        Serial.print("CO2 = ");
        Serial.print(ppm);
        Serial.println(" ppm");
        Serial.print("Raw Signal Value (0-1023): ");
        Serial.println(voMeasured);

        Serial.print(" - Voltage: ");
        Serial.println(calcVoltage);

        Serial.print(" - Dust Density: ");
        Serial.println(dustDensity);
        Serial.println("Sending data to Thingspeak");
        Serial.println();

        lcd.setCursor(0,0);
        lcd.print("Temp=   C  ");
        lcd.setCursor(6, 0);
        lcd.print(t);
        lcd.setCursor(0,1);
        lcd.print("Humi=   %  ");
        lcd.setCursor(6, 1);
        lcd.print(h );
        delay(5000);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Dust=   mg/m3");
        lcd.setCursor(6, 0);
        lcd.print(dustDensity);
        lcd.setCursor(0,1);
        lcd.print("CO2 =   ppm");
        lcd.setCursor(6, 1);
        lcd.print(ppm);
        //
    }
    client.stop();

    // thingspeak needs at least a 15 sec delay between
updates
    // 20 seconds to be safe
    //delay(15000); }

```

## Appendix 6. Command for the vehicle (Final version 25.02.18)

```

#include <AFMotor.h>
#include <Servo.h>
#include <NewPing.h>

#define TRIG_PIN A4
#define ECHO_PIN A5
#define MAX_DISTANCE_POSSIBLE 1000
#define MAX_SPEED 150 //
#define MOTORS_CALIBRATION_OFFSET 3
#define COLL_DIST 20
#define TURN_DIST COLL_DIST+10
NewPing      sonar(TRIG_PIN,      ECHO_PIN,
MAX_DISTANCE_POSSIBLE);

AF_DCMotor leftMotor(1, MOTOR12_64KHZ);
AF_DCMotor rightMotor(2,MOTOR12_64KHZ);
Servo neckControllerServoMotor;
int pos = 0;
int maxDist = 0;
int maxAngle = 0;
int maxRight = 0;
int maxLeft = 0;
int maxFront = 0;
int course = 0;
int curDist = 0;
String motorSet = "";
int speedSet = 0;

void setup() {
  neckControllerServoMotor.attach(10);
  neckControllerServoMotor.write(90);
  delay(2000);
  checkPath();
  motorSet = "FORWARD";
  neckControllerServoMotor.write(90);
  moveForward();
}

void loop() {
  checkForward();
  checkPath();
}

void checkPath() {
  int curLeft = 0;
  int curFront = 0;
  int curRight = 0;
  int curDist = 0;
  neckControllerServoMotor.write(144);
  delay(120);
  for(pos = 144; pos >= 36; pos--=18)
  {
    neckControllerServoMotor.write(pos);
    delay(90);
    checkForward();
    curDist = readPing();
    if (curDist < COLL_DIST) {
      checkCourse();
      break;
    }
    if (curDist < TURN_DIST) {
      changePath();
    }
    if (curDist > curDist) {maxAngle = pos;}
    if (pos > 90 && curDist > curLeft) { curLeft = curDist;}
    if (pos == 90 && curDist > curFront) {curFront =
curDist;}
    if (pos < 90 && curDist > curRight) {curRight =
curDist;}
  }
  maxLeft = curLeft;
  maxRight = curRight;
  maxFront = curFront;
}

void setCourse() {
  if (maxAngle < 90) {turnRight();}
  if (maxAngle > 90) {turnLeft();}
  maxLeft = 0;
  maxRight = 0;
  maxFront = 0;
}

void checkCourse() {
  moveBackward();
  delay(500);
  moveStop();
  setCourse();
}

```

```

void changePath() {
  if (pos < 90) {lookLeft();}
  if (pos > 90) {lookRight();}
}

int readPing() {
  delay(70);
  unsigned int uS = sonar.ping();
  int cm = uS/US_ROUNDTRIP_CM;
  return cm;
}

void checkForward() { if (motorSet=="FORWARD")
{leftMotor.run(FORWARD);
rightMotor.run(FORWARD); } }

void checkBackward() { if (motorSet=="BACKWARD")
{leftMotor.run(BACKWARD);
rightMotor.run(BACKWARD); } }

void moveStop() {leftMotor.run(RELEASE);
rightMotor.run(RELEASE);}

void moveForward() {
  motorSet = "FORWARD";
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
  for (speedSet = 0; speedSet < MAX_SPEED;
speedSet +=2)
  {

leftMotor.setSpeed(speedSet+MOTORS_CALIBRATI
ON_OFFSET);
  rightMotor.setSpeed(speedSet);
  delay(5);
  }
}

void moveBackward() {
  motorSet = "BACKWARD";
  leftMotor.run(BACKWARD);
  rightMotor.run(BACKWARD);
  for (speedSet = 0; speedSet < MAX_SPEED;
speedSet +=2)
  {

leftMotor.setSpeed(speedSet+MOTORS_CALIBRATI
ON_OFFSET);
  rightMotor.setSpeed(speedSet);
  delay(5);
  }
}

void turnRight() {
  motorSet = "RIGHT";
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
  delay(400);
  motorSet = "FORWARD";
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
}

void turnLeft() {
  motorSet = "LEFT";
  leftMotor.run(BACKWARD);
  rightMotor.run(FORWARD);
  delay(400);
  motorSet = "FORWARD";
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
}

void lookRight() {rightMotor.run(BACKWARD);
delay(400); rightMotor.run(FORWARD);}
void lookLeft() {leftMotor.run(BACKWARD); delay(400);
leftMotor.run(FORWARD);}

```