

Design & Innovation Project (DIP)

Project Report

Smart Monitoring Electronic System for Environmental Noise and Dust

Project Group: E031

School of Electrical and Electronic Engineering

Academic Year 2024/25

Semester 1

Table of Contents

1. Purpose / Project Objectives
2. Project Summary
3. Scope
4. Schedule
5. Cost / Benefit
6. Outcome / Expected Outcomes
7. Project Management Review
8. Reflection
9. References

Appendix

Acknowledgement

We would like to express our gratitude to our main supervisor, Associate Professor Chan Pak Kwong, project lab technician, Mr Tay, for their support and assistance throughout the period of this Design and Innovation Project (DIP).

Table of Contents

[Chapter 1 Project Purpose 1](#_Toc182192193)

[Chapter 2 Project Summary 3](#_Toc182192194)

[Chapter 3 Project Scope 5](#_Toc182192195)

[Chapter 4 Environmental Noise and Dust 6](#_Toc182192196)

[Chapter 5 Overall System Design 11](#_Toc182192197)

[5.1 Hardware 11](#_Toc182192198)

[5.1.1 LoLin NodeMCU ESP8266 11](#_Toc182192199)

[5.1.2 SEN0232 14](#_Toc182192200)

[5.1.3 GY-BME280 16](#_Toc182192201)

[5.1.4 PMS7003 18](#_Toc182192202)

[5.1.5 Li-Po Battery 20](#_Toc182192203)

[5.1.6 DC-DC Step Up Booster Converter 2.5V - 5V (5W) 21](#_Toc182192204)

[5.1.7 2x Solar Panels – 6V, 1A 23](#_Toc182192205)

[5.1.8 Solar Charge Controller Module 6V 2A 25](#_Toc182192206)

[5.1.9 Custom PCB 27](#_Toc182192207)

[Chapter 6 Schedule 31](#_Toc182192208)

[Chapter 7 Cost 33](#_Toc182192209)

[Chapter 8 Outcomes / Benefits 35](#_Toc182192210)

[Chapter 9 Project Management Review 37](#_Toc182192211)

[Chapter 10 Reflection 38](#_Toc182192212)

[Chapter 11 Acknowledging/Declaring the Use of GAI 40](#_Toc182192213)

[Chapter 12 References 43](#_Toc182192214)

# Project Purpose

According to the World Health Organisation (WHO), noise and air pollution are significant environmental factors that can have detrimental impacts on our health. Air pollution is a complex mixture of particulate matter that can come from many sources such as cars, industrial plants and many more.

A screenshot of a medical report

Description automatically generated

Figure 1 - Correlation PM2.5 levels and ED admissions related to respiratory issues

When inhaled, the course PM10 could damage our lungs whereas the finer PM2.5 can enter our bloodstream and cause a multitude of chronic illnesses. A study conducted in Volos, Greece found a positive correlation between PM2.5 levels and the number of patients admitted to emergency departments (ED) for certain respiratory issues as shown in Figure 1. These respiratory issues include Upper Respiratory Infection (URI), Pneumonia and Chronic Obstructive Pulmonary Disease (COPD) exacerbation.

Noise on the other hand is an underestimated threat that can also cause many long and short term health issues and be damaging to our everyday lives. The WHO has also established that noise is a leading environmental nuisance in the European Region and there have been concerns from the public. Different groups of people react differently to noise exposure. For instance, children are more susceptible to night noises as they tend to sleep longer than adults. Additionally, chronically ill and elderly people are more sensitive to disturbances.

A screenshot of a screenshot of a medical report

Description automatically generated

Figure 2 - Relationship between probability risk of myocardial infarction and decibel levels from construction noise

One of the health risks associated with noise exposure is an increased likelihood of myocardial infarction. Figure 2 shows the increasing linear relationship between the probability risk of myocardial infarction and decibel levels from construction noise of individuals in varying age groups.

# Project Summary

The main function of this smart environmental monitoring system is to remotely monitor the dust and noise levels of the environment in real time, using machine learning to analyse, predict and alert the user to potentially hazardous environment conditions.

The core of this system consists of a microcontroller, sensors for detecting particulate matter (PM), decibel (dB) levels, temperature and humidity and Internet of Things (IoT) platform for data storage, visualization and predictive analysis through MATLAB. Additionally, a commonly used messaging service is used to send alerts and information to users through their mobile devices. For the system to function remotely, a lithium-ion polymer (LiPo) battery and solar panels are integrated, acting as a power source and harvesting element to allow the system to operate without an external power supply. A custom designed Printed Circuit Board (PCB) is used to connect all the components of the system together allowing the system to be more reliable, durable and compact.

Even though the system was able to achieve the majority of the project’s objectives by effectively monitoring the environmental conditions and alerting the user appropriately, the system is unable to sustain for long period of operation. This was mainly due to the lack of experience with working on new components and system design. There was also no secured enclosure created for this system, which would be ideal for making the system protected and convenient for deploying in any environment.

battery cannot last for long hours even in ideal sunny conditions. This is mainly due to the poor selection of the solar panels used. The output power is more than the input power from the solar panels causing the battery to eventually deplete. This can also cause some sensors to not work properly after a while as some sensors require a consistent Voltage Input.When the battery depletes over time, the voltage output of the battery will decrease below the required voltage of the sensos.

Another area where improvements could be made is to create a secure enclosure for the system using 3D printing. The reason for not making a secured enclosure is due to time taken to troubleshoot... TO CONTINUE

# Project Scope

This project aims to create and design a smart system to monitor environmental noise and dust by combining a microcontroller with an Internet of Things (IoT) platform. Equipped with sensors for noise, particulate matter (PM2.5 and PM10), temperature, and humidity, the system collects real-time data from the environment and sends it to the ThingSpeak IoT platform. On ThingSpeak, machine-learning models such as autoregressive (AR) and Autoregressive-Exogenous (ARX) modelling are used to analyse and predict future noise and dust levels. The processed data is displayed through interactive gauges and trend graphs, giving users a clear view of current conditions and forecasts.

Additionally, an application programming interface (API), Telegram bot, provides real-time notifications and alerts if noise or dust levels approach hazardous levels. This fully automated system allows users with proactive monitoring tools, enabling them to make informed choices to maintain a healthier living environment.

# Environmental Noise and Dust

Monitoring environmental noise and dust levels has become essential and crucial for public health. Long term exposure with high levels of noise and dust levels can impact health, contributing to several illnesses.

|  |  |
| --- | --- |
| Sound Level Description | Sound levels in dB |
| Normal | <70 |
| Elevated | 70-90 |
| High | 90-110 |
| Very High | >110 |

Table 1 - Threshold Level for Noise

Environmental noise levels are measured in decibels (dB), which indicate how intense a sound is. According to Singapore’s National Environment Agency (NEA) and health organisations worldwide, there are specific noise thresholds recommended to minimise risks to hearing and overall wellbeing. For example, noise at Normal or Safe levels (below 70 dB) are generally safe even over extended periods, while Very High or Hazardous levels (above 110 dB) can cause hearing damage if the exposure is prolonged.

In this project, we have adapted these threshold levels based on the general guidelines set by the NEA and other health organisations. These guidelines provide recommendations for safe noise exposure, which we have generalised to meet the specific needs of our project.

|  |  |
| --- | --- |
| PM2.5 Level Description | PM2.5 levels in ppm |
| Normal | <13 |
| Elevated | 13-55 |
| High | 55-150 |
| Very High | >150 |

Table 2 - Threshold Levels for PM2.5

| PM10 Level Description | PM10 levels in ppm |
| --- | --- |
| Normal | <50 |
| Elevated | 50-150 |
| High | 150-350 |
| Very High | >350 |

Table 3 - Threshold Levels for PM10

Environmental dust levels in the air can be quantified by measuring particulate matter (PM) concentrations. Particles small enough to be inhaled, specifically those 2.5 micrometres (PM2.5) and 10 micrometres (PM10) or smaller, are small enough to be inhaled and potentially lead to various health issues. The NEA uses PM2.5 and PM10 readings to inform the public about air quality, offering guidance on what levels are safe or risky. According to their standards, PM2.5 levels above 55 µg/m³ and PM10 levels over 150 µg/m³ are considered unhealthy.

This system of monitoring helps people make informed choices about outdoor activities and understand when preventive measures are needed, creating safer environments for everyone.

A group of colorful bars

Description automatically generated

Figure 3 - Correlation between temperature and air quality

A collage of graphs and charts

Description automatically generated

Figure 4 - Effects of humidity on the Temperature - PM2.5 [correlation](https://link.springer.com/article/10.1007/s12647-023-00656-8)

Temperature and Humidity could also have an impact on these levels of particles. A study in Delhi, India shows that higher temperatures could result in lower PM2.5 levels. Figure A illustrates that there are more days with satisfactory air quality at higher temperatures, while lower temperatures are associated with a higher number of days with very poor air quality. Additionally, a higher relative humidity could cause this relationship to have a stronger negative correlation as shown by Figure B. At higher relative humidity, PM2.5 levels tend to increase more significantly in lower temperatures, while they tend to decrease more in warmer temperatures. PM10 levels also contribute to the overall air quality and is also another form of particulate matter like PM2.5. Therefore, humidity and temperature would be key players in evaluating PM2.5 and PM10 levels.

# Overall System Design

## Hardware

The hardware side of the project consists of the microcontroller - ESP8266, Sensors - SEN0232, BME280, PMS7003 and Power Supply System – DC step up, Solar Charger controller, Solar Panels, LiPo Battery. All hardware modules used are connected to a custom PCB. A more detailed Specification of each module can be found in Appendix (X).

### LoLin NodeMCU ESP8266

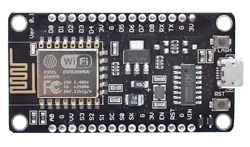


Figure 5 - ESP8266 [1]

#### Function

This microcontroller is widely used for various IoT applications due to its surface-mountable WiFi module with an embedded ESP8266 system on chip (SoC). It is capable for onboard data processing and integration with sensors through general-purpose input/output (GPIO) pins and power supply pins. ESP8266 is programmable with the Arduino integrated development environment (IDE) that has a wide range of libraries that provides extra functionality to the microcontroller. [2]

#### Specification

|  |  |
| --- | --- |
| Core | 1 |
| Architecture | 32 bits |
| Clock Speed | 80Mhz |
| WiFi | IEEE802.11 b/g/n support for WPA and WPA2 |
| RAM | 160KB - 64KB Instruction - 96KB Data |
| Flash | 4MB |
| Digital I/O Pins | 11 |
| Analog I/O Pins | 1 |
| Interfaces | Inter-Intergrated Circuit (I2C) - Serial Peripheral Interface (SPI) - Universal asynchronous receiver / transmitter (UART) - Inter-Integrated Circuit Sound (I2S) |
| Operating Voltage | 3.3V |
| Module Size | 58mm\*32mm |

Table 4 - Specification of ESP8266 [1]

#### Justification

The ESP8266 is a low cost and low powered module that has built-in features that are suitable for this project like WiFi and GPIO pins. With this module being able to be programmed by Arduino IDE, it has community and documentation support resulting in a large compatibility with sensors and libraries. Therefore, this microcontroller can allow us to meet the objectives of the project.

### SEN0232

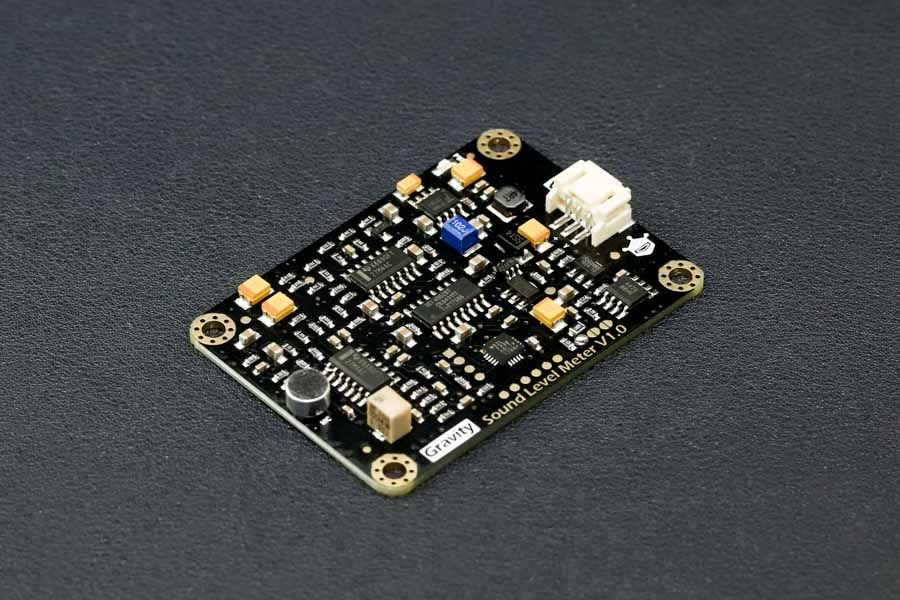


Figure 6 - Analog Sound Level Meter, SEN0232 [3]

#### Function

The SEN0232 is an Analog Sound Level Meter that is designed to accurately measure the sound level of the surrounding environment. This is achieved by using an instrument circuit and a low noise microphone. With a provided connector, this is a plug-and-play component which is user friendly and compatible with a lot of IoT devices due to its wide input voltage of 3.3-5V and outputs a maximum of 2.6V which linearly follows the decibel value of the surrounding environment. [3]

#### Specifications

|  |  |
| --- | --- |
| Input Voltage | 3.3V - 5V |
| Current consumption | 22mA @ 3.3V, 14mA @ 5V |
| Output Voltage | 0.6V - 2.6V |
| Measuring Range | 30dBA - 130dBA |
| Measurement Error | ±1.5dB |
| Module Size | 60mm \* 43mm |

Table 5 - Specification of SEN0232 [3]

#### Justification

Due to its ease of use, wide input voltage of 3.3V-5V and low input current of 22mA at 3.3V, the SEN0232 is an ideal Sound Level Meter to integrate with the ESP8266. It also has a wide measuring range of 30dBA ~ 130dBA with a measurement error of ±1.5dB. The module size is also easy to integrate with its size of 60mm\*40mm.

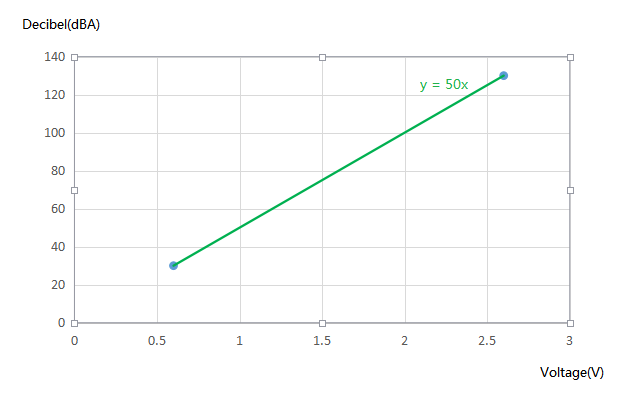


Figure 7 - Relation between Decibel Value and Voltage Output [3]

Additionally, from the figure above, the correlation between the decibel level and voltage output is linear, this simplifies the code without the need for complex algorithms. Suitable for new and experienced students taking on this project.

### GY-BME280



Figure 8 - GY-BME280 [4]

#### Function

This module is an environmental sensor that measures the surrounding pressure, humidity and temperature. Thanks to its low power consumption, it can be easily implemented in battery driven systems. This sensor uses an I2C communication protocol to exchange data with a microcontroller. [5]

#### Specification

|  |  |
| --- | --- |
| Input Voltage | 3.3V - 5V |
| Current consumption | 0.4mA |
| Temperature Range | -40°C to 85°C |
| Temperature Accuracy | ±1°C |
| Humidity Accuracy | ±3% |
| Pressure Range | 300 hPa – 1100 hPa |
| Pressure Accuracy | ±1.0 hPa |
| Module Size | 14mm\*11mm |
| Communication Protocol | I2C |

Table 6 - Specification of GY-BME280 [4]

#### Justification

This sensor can operate on a 3.3V source and communicate using I2C communication protocol that the ESP8266 is capable of. This would make the connection very simple. However, to receive readings, an Adafruit\_BME280 library needs to be installed and utilized.[5] This is an easy process to complete, and the library allows the code to be readable. The module size is also easy to integrate to any system with its size of 14mm\*11mm.

### PMS7003



Figure 9 - PMS7003 [6]

#### Function

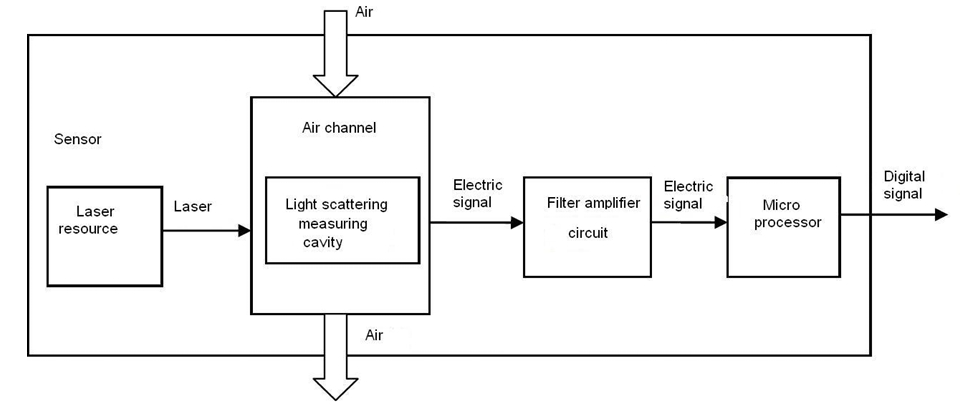


Figure 10 - Functional block diagram of sensor [7]

PMS7003 is a particle concentration sensor that obtains the number of suspended particles in the surrounding air and outputs the result digitally via I2C communication protocol. This is possible by the Laser scattering principle by using laser to radiate suspending participles which then produces scattering. With this, the microprocessor of the module can determine the particle diameter and number of particles based on the MIE theory calculation [7]

#### Specification

|  |  |
| --- | --- |
| Input Voltage | 4.5V - 5.5V |
| Current consumption (Active Mode) | ≤100mA |
| Range of measurement | 0.3μm - 1.0μm ；1.0μm - 2.5μm ；2.5μm - 10μm |
| Effective Range (PM2.5 standard) | 0μg/m³ - 500μg/m³ |
| Maximum Consistency Error (PM2.5 standard) | ±10%@100~500μ g/m³  ±10μ g/m³@0~100μ g/m³ |
| Module Size | 48mm\*37mm\*12mm |

Table 7 - Specification of PMS7003 [7]

#### Justification

This module is sensitive, accurate and reliable and has counting efficiency of 98% for particles larger than 0.5μm. Additional features of zero false alarms and high anti-interference capabilities allow this module to provide consistent data accuracy which makes this module suitable for varying environments. This module also has simple serial digital output that allows easy connection to the ESP8266. Lastly, its slim design allows easy integration into various devices.

### Li-Po Battery

hello


Figure 11 - Li-Po Battery 2500mAh [8]

#### Function

This battery is a compact and rechargeable source that is designed for portable applications that require a power source without the need for external power source. It provides an output voltage of 3.7V that makes it suitable for IoT applications. Its slim profile enables the battery to be lightweight and integrated into compact designs where space and weight are constrained. The battery includes a protective circuit to prevent overcharging which would allow the battery to have a longer life span.

#### Specifications

|  |  |
| --- | --- |
| Output Voltage | 3.7V |
| Working Current (Max) | 1500mAh |
| Battery Type | Lithium Polymer (Li-Po) |
| Connector | PH2.0 |
| Battery Size | 50mm\*35mm\*10mm |

Table 8 - Specification of Li-Po Battery [9]

#### Justification

To make this system portable, we needed an internal power source that is slim, lightweight and provides enough power to supply the ESP8266. With the ESP8266 and most of the modules used in this system only requiring 3.3V, this battery will be able to supply the required power. For the PMS7003 that requires 5V, a DC-DC step up module is used to output 5V to supply. Solar Panels will be integrated to the system to harvest power to charge the battery.

### DC-DC Step Up Booster Converter 2.5V - 5V (5W)

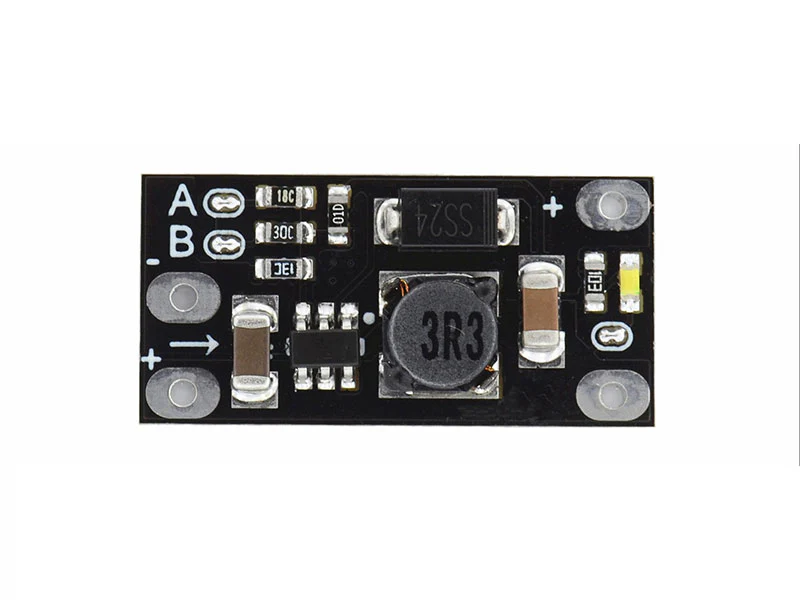


Figure 12 - DC-DC Step Up Booster Converter 2.5V - 5V (5W) [10]

#### Function

This is a module that ultilises a step-up topology to convert the low input voltage, starting from 2.5V, to a wide range of output voltage from 5V – 12V with a maximum power output of 5W. Selecting the output voltage can be done by cutting the A & B traces accordingly. Therefore, this module is useful for other modules that require an output voltage higher than the available source. [10]

#### Specification

|  |  |
| --- | --- |
| Input Voltage | 2.5V - Vout |
| Output Voltage | 5V / 8V / 9V / 12V |
| Output Current | 3.7V Input: 5V 1A / 8V 0.5A / 9V 0.45A / 12V 0.3A  5V Input: 8V 0.7A / 9V 0.7A / 12V 0.5A |
| Module Size | 22mm\*11mm\*3.6mm |

Table 9 - Specification of Step Up Converter [10]

#### Justification

The main purpose for choosing this module is to provide the correct input voltage to the PMS7003

sensor that requires 5V as the Li-Po battery used in the system is only able to output 3.7V. When this module outputs 5V, it can provide an output current of 1A which is sufficient for the PMS7003 module to operate.

### 2x Solar Panels – 6V, 1A

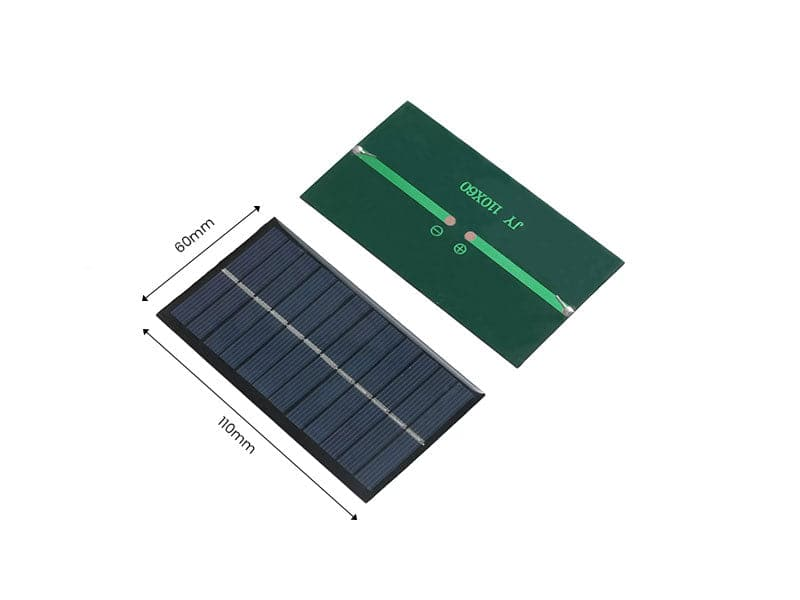


Figure 13 - Solar Panels 6V,1A [11]

#### Function

The solar panel is designed to be a renewable power source that converts sunlight to electrical energy. This solar panel is compact and efficient and can output a maximum power of 1W which makes it suitable for low-powered devices that would be exposed to the sun. When connected to a charge controller, it can be used to charge rechargeable batteries like Li-Po batteries.

#### Specification

|  |  |
| --- | --- |
| Output Voltage (Max) | 6V |
| Power | 1W |
| Module Size | 110m\*60mm\*2.5mm |

Table 10 - Specification of Solar Panel [11]

#### Justification

To make the system portable and operate in an outdoor environment, a solar panel is needed to charge the internal Li-Po battery. The solar panel is also lightweight and has a durable design which can be easily integrated into most systems. Two solar panels are used to make charging the battery faster which would make the system operate longer.

### Solar Charge Controller Module 6V 2A

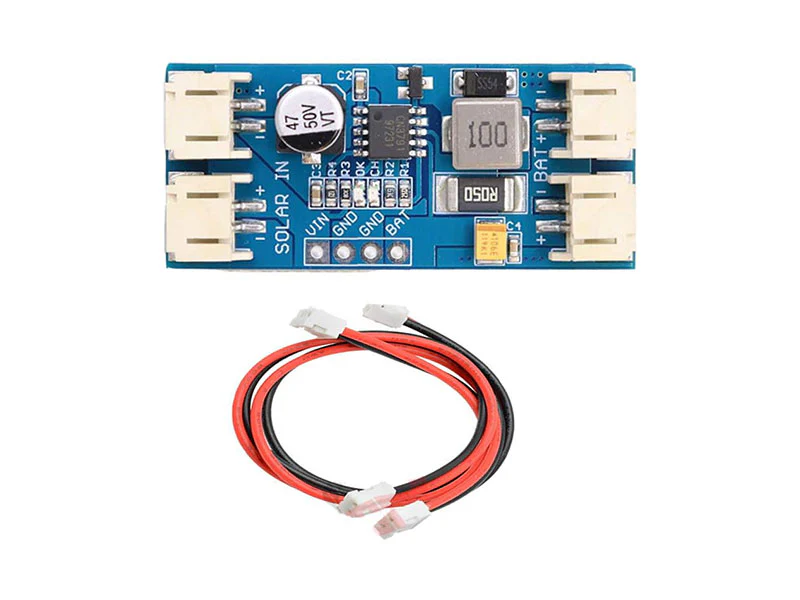


Figure 14 - Solar Charge Controller Module 6V 2A [12]

#### Function

The main function of this module is to regulate the voltage output of a 6V solar panel and manage the charging process of the Li-Po battery with Pulse Width Modulation (PWM). This would ensure that the battery receives a gradual and controlled charge which would result in a long-life span and better performance for the battery. [12]

#### Specification

|  |  |
| --- | --- |
| Input Voltage (Max) | 6V |
| Output Voltage | Nominal 3.7V, Full Charge 4.2V |
| Charging Current (Max) | 2A |
| PWM Frequency | 300kHz |
| Module Size | 45mm\*20mm\*9.5mm |

Table 11 - Specification of Solar Charge Controller [12]

#### Justification

To charge the Li-Po battery in the system, a controller module is needed. This module works nicely with the solar panel used as it can take up to 6V which is the maximum output voltage of the solar panel. Additionally, the small profile of the module makes it easy to integrate into the system.

### Custom PCB

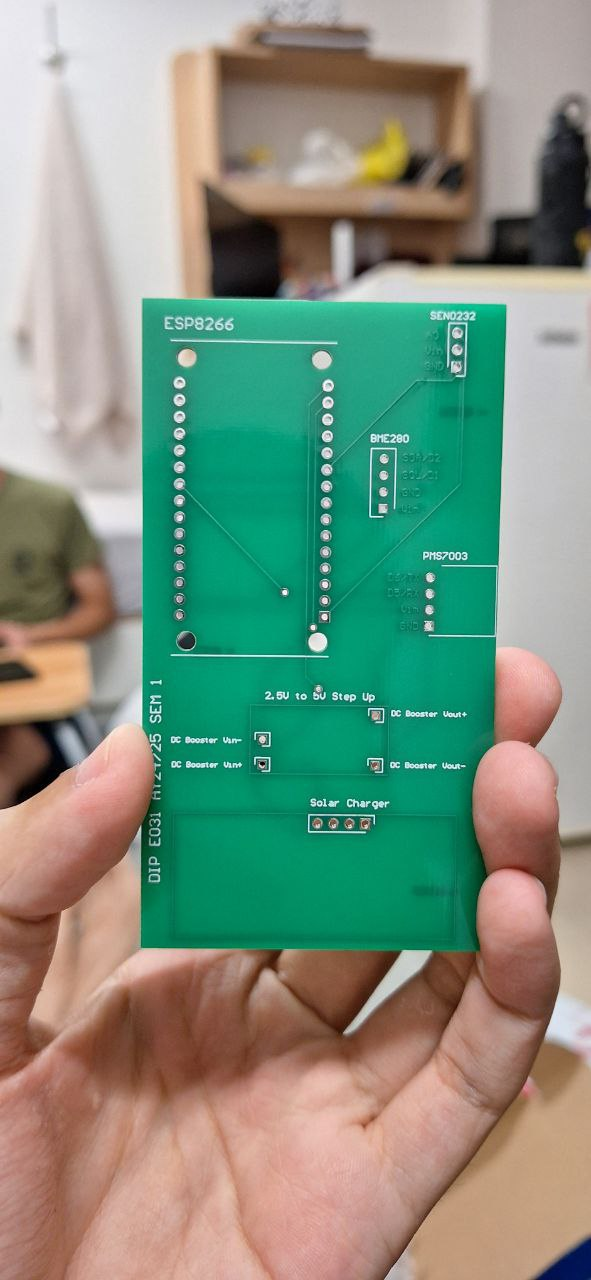


Figure 15 - Custom PCB

#### Function

The function of the custom PCB is to have a more compact and reliable circuit design for the system. By using PCB design software, human errors in the circuit design will be significantly reduced due to the error checking feature. Additionally, the traces in the PCB would be more reliable and are less prone to interference. Minimal soldering is required, and the general look of the circuit would be better.

#### PCB Design Software



Figure 16 - Altium Designer Logo [13]

Altium Design is a software tool used to create and manage PCB designs. It uses schematic design; PCB layout and 3D modeling to design and manage circuits. Features like auto routing, real-time error checks and extensive component libraries make this software one of the best in the industry. Allowing users to create their custom PCB faster while reducing mistakes done by users. Additionally, Altium Design supports team collaboration and version control which is a crucial feature for professionals working together on the same project.

#### Schematic Design & 3D Model

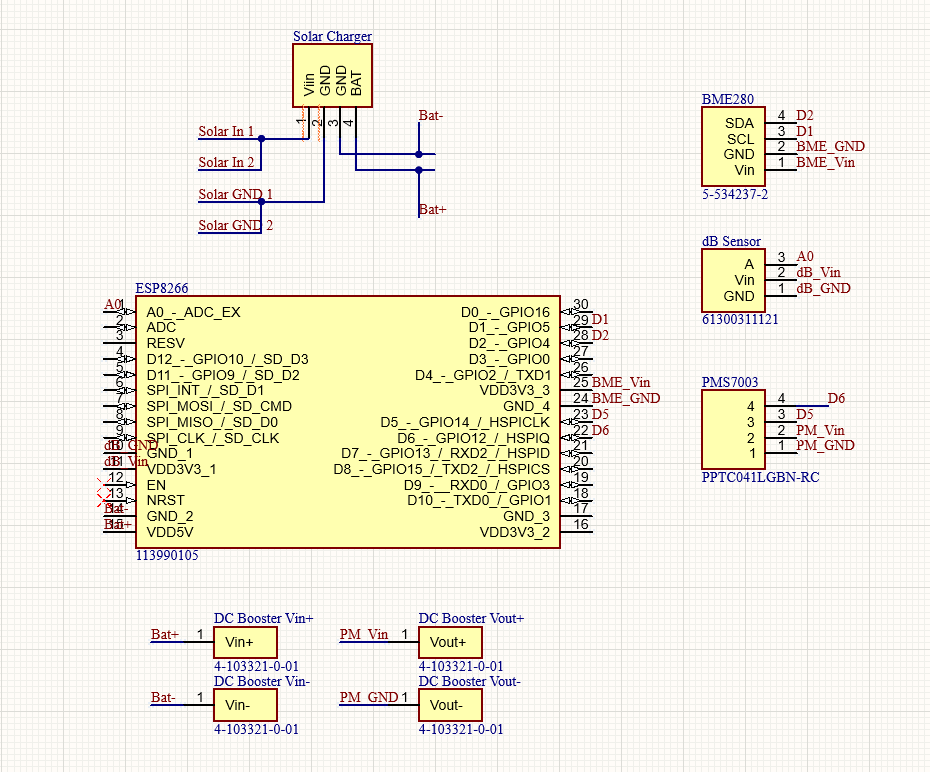


Figure 17 - Schematic Design of Custom PCB (Taken from Altium Designer)

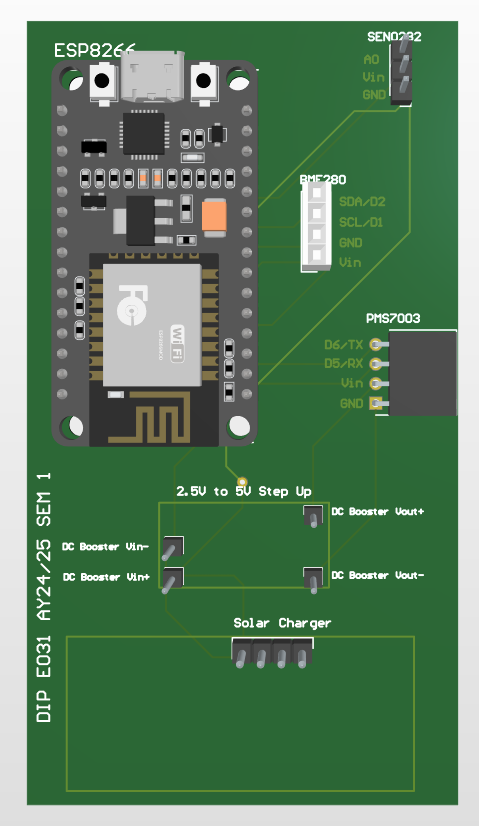


Figure 18 - 3D Model of Custom PCB (Taken from Altium Designer)

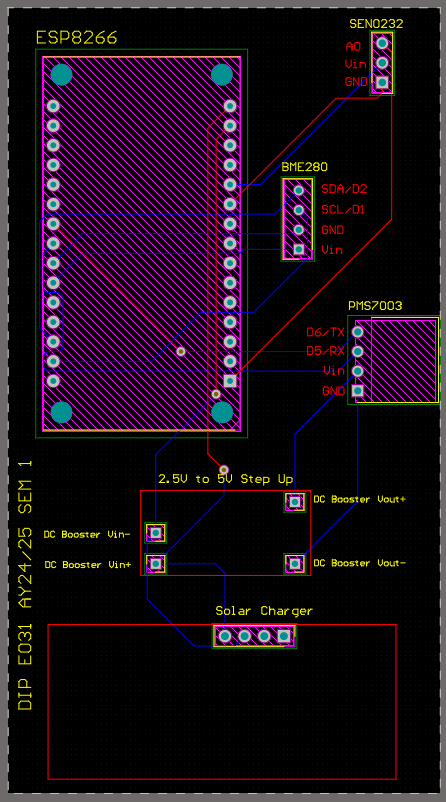


Figure 19 - Traces & Connections of Custom PCB (Taken from Altium Designer)

#### Justification

Custom PCB design is made to help make the system more compact and to increase reliability. This is done by soldering some sensors and modules to the PCB which improves durability. The components are also arranged specifically for optimal placement of each sensor which would make the overall system size to be smaller when compared to using a temporary breadboard for connections. Therefore, the overall system will look neater and easier for broader deployment if needed as PCBs are easy to duplicate.

## Software: ThingSpeak

A blue background with white text

Description automatically generated

Figure 20 - ThingSpeak Logo

We have utilised ThingSpeak as a platform for logging data and implementing a machine learning model using MATLAB for prediction analysis. ThingSpeak is an Internet of Things (IoT) platform which allows users to display, collect data and perform analysis through executing MATLAB code in it. In the context of our project, we use the cloud to store and display our readings of PM2.5, PM10, Humidity and Temperature readings collected from our sensors connected to ESP8266.

### Integrating and Sending Data to ThingSpeak

### Predicting Sound (dB) Levels

A screenshot of a computer program

Description automatically generated

Figure 21 - MATLAB code for Autogressive Model

A diagram of a graph

Description automatically generated with medium confidence

Figure 22 - Predicted Sound Data Trend

Using the MATLAB Analysis application in ThingSpeak, we have decided to use autoregressive modelling for predicting noise levels. Autoregression (AR) is a time series model where the current value is expressed as a linear combination of its past values. One of the key reasons as to why we used this model is that it is more effective for short term forecasting.

The AR model is ideal for real-time applications like ThingSpeak, where data is continuously being collected and immediate predictions are performed. Its fast computational output makes it a suitable fit for IoT-based environmental monitoring systems, where processing speed is crucial as opposed to more complex models such as neural networks. Here is a breakdown of the steps involved in the code:

The code begins by drawing noise level data from a specific ThingSpeak channel, which is connected to a sensor through an ESP8266. The channelID and fieldID parameters identify the data source, while numPoints specifies the amount of data to fetch. The function thingSpeakRead fetches the data in a structured table format for data analysis.

Since ThingSpeak timestamps are recorded in UTC, they need to be converted to the local time zone (GMT+8) to ensure the data is aligned correctly with the local time of the noise measurements. This conversion is done using the datetime function, which first interprets the data in UTC and then adjusts it to the desired time zone.

Before modelling, the data is cleaned to remove any invalid entries, such as missing NaN (Not-a-Number) or infinite values, which can affect data predictive analysis. The infinite function is used to filter out the valid data points, ensuring that only reliable data is used in the analysis.

To improve the performance of the AR model, the data is split into a train and test set: training (80%) and testing (20%). The training set is used to fit the AR model, while the test set is used for evaluating its performance on unseen data.

The AR model is built by regressing the current noise levels against their previous values. An AR(4) model is used, meaning the noise levels are predicted based on the previous four values. The AR model's order, determined by arOrder, specifies how many past data points are considered for making predictions. The coefficients of the model are then estimated through linear regression.

Once the model is fitted with the training data, it is used to predict the noise levels in the test set. The predicted values are compared to the actual noise levels to assess the model's accuracy. The performance of the model is evaluated using the Mean Absolute Percentage Error (MAPE). MAPE is a common metric for time series models, measuring the average percentage error between the predicted and actual values. A small epsilon value is added to avoid division by zero errors. The smaller the MAPE score, the higher the model's accuracy.

After testing, the model is used to predict future noise levels. The AR model makes these forecasts by iteratively using the most recent observed values to predict the next set of points. The number of future points to predict is specified by numFuturePoints.

The historical data and predicted future noise levels are plotted together. This visualisation gives a clear picture of how the model’s predictions align with the actual past values, as well as how the noise levels are expected to behave moving forward.

The code output consistently provides a range of MAPE score of 5 to 20%, suggesting that the predicted values are relatively close to values seen in the test set, which is a good sign.

### Predicting PM2.5 and PM10 Levels

A screenshot of a computer code

Description automatically generated

Figure 23 - MATLAB code for ARX Model

A graph showing the time and time

Description automatically generated with medium confidence

Figure 24 - Predicted PM2.5 Data Trend

A graph showing the time and time

Description automatically generated with medium confidence

Figure 25 - Predicted PM10 Data Trend

Similarly for the prediction of PM2.5 and PM10 levels, we have used Autoregressive-Exogenous (ARX) modelling as opposed to the AR model used to predict noise levels. The ARX model is a modified version of the autoregression model as it incorporates external inputs that influence the system in addition to the past values of the time series. As previously mentioned, we have concluded that humidity and temperature affects environmental dust levels in the air based on the study made. Therefore, this model is ideal as it can efficiently handle more than one input, making it more accurate in scenarios where external factors can affect the overall output.

The process begins by extracting PM2.5, PM10, temperature, and humidity data from ThingSpeak. The thingSpeakRead function retrieves the data from the designated channel and is fetched in a structured table format for data analysis. The humidity data, as retrieved from ThingSpeak, is stored as a string which is not suitable for data analysis. Hence, the string values need to be converted into a numerical format. This is done by using the cellfun function in combination with a regular expression (regexp) to extract the numeric part from the string and convert it to a number using str2double.

Similarly with the predictive analysis for environmental noise, the data is cleaned of any missing or invalid values to ensure only valid data points are used for reliable and accurate predictions. Additionally, the data is split into a 80% and 20% train test set so that the model is tested on data it has not seen before, allowing for a reliable and accurate predictive analysis.

The ARX model consists of an autoregressive component (na) and exogenous inputs (nb) for temperature and humidity. The model’s order is set by the parameters na, nb, and nk, where na defines the number of past PM2.5 values considered, and nb specifies the number of past values of temperature and humidity used in the model. The ARX function then fits the model to the training data using linear regression to estimate the model coefficients.

The trained ARX model is tested against the testing dataset to evaluate its accuracy. The predicted PM2.5 values are compared to the actual values in the test set using the Mean Absolute Percentage Error (MAPE) metric.

After evaluating the model on historical data, we forecast future PM2.5 levels by predicting 150 future data points based on trends in temperature and humidity. The future values for temperature and humidity are linearly increased to simulate changing environmental conditions, and the ARX model uses these inputs to predict PM2.5 levels.

Both the historical and predicted PM2.5 data are plotted together, where historical data shows the actual recorded PM2.5 levels, while predicted values provide forecasting into future PM2.5 trends.

The predictive analysis for PM10 (Appendix) is conducted in the same way as PM2.5 analysis, with both using the ARX model, which takes historical values of the target variable (PM10 or PM2.5) and external inputs (temperature and humidity) to predict future values.

The ARX models consistently achieve MAPE scores ranging from 5% to 20%, indicating a relatively accurate prediction of both PM2.5 and PM10 levels based on the test data.

### Average Predicted Values

A white background with colorful text

Description automatically generated

Figure 26 - MATLAB code for Mean & thingSpeakWrite Function

A screenshot of a computer screen

Description automatically generated

Figure 27 - TimeControl Settings

//Wait for their side

### Visualisation Gauges

A screenshot of a computer program

Description automatically generated

Figure 28 - Parameter Code for Current Gauges for Sound, PM2.5 & PM10

A close-up of a gauge

Description automatically generated

Figure 29 - Current Gauges for Sound, PM2.5 & PM10

A close-up of a gauge

Description automatically generated

Figure 30 - Average Predicted Gauges for Sound, PM2.5 & PM10

Visualisation gauges help provide a real-time display of current levels of noise (dB), PM2.5, and PM10. These gauges make it easy and provide an intuitive way for users to interpret and monitor the current as well as average predicted environmental conditions by using colour-coded segments that represent different threshold levels. The gauges are designed to visually represent the healthiness of the environment, from safe to hazardous levels. Here is the breakdown of the code:

Each gauge pulls its respective data from a ThingSpeak channel, such as sound, PM2.5 and PM10 respectively. The gauges are divided into different colour-coded segments, such as green for safe and healthy range, yellow for moderate or slightly elevated and red for hazardous, which represent various ranges of environmental health thresholds as mentioned above. The colour thresholds are defined within the code to highlight these ranges.

Each gauge uses a circular design which is drawn using parametric equations to represent the current and full range of noise, PM2.5, or PM10 values. A "needle" is plotted on the gauge to indicate the current value. The angle of the needle is calculated based on the current data, and it is dynamically updated as new data is received.

The gauges include tick marks that correspond to significant values (for example: 60 dB, 75 dB) to help the user easily understand the current value. This includes a label in large font, such as "dB" or "PM2.5", to show what is being measured.

The code arranges the gauges for dB, PM2.5, and PM10 in a clear and organised layout within the figure window. Each gauge is drawn with a different range, but they all share a similar design and structure, ensuring uniformity across the visualisations.

In addition to displaying current real-time data for noise (dB), PM2.5, and PM10 levels, we have also incorporated predictive modelling to forecast future values to provide insight into potential trends. (Appendix) These predicted values are then averaged and presented on dedicated gauges to allow users to view expected future values.

The prediction analysis for noise levels is conducted using the same autoregressive (AR) model as well as the same Autoregressive with Exogenous Inputs (ARX) model for PM2.5 and PM10, as used above in the MATLAB Analysis. After performing predictive modelling, the system calculates the average predicted values for each parameter-noise (dB), PM2.5, and PM10. This average is derived from a series of future predicted values generated by the AR or ARX models using the mean function in MATLAB.

## Software: Telegram Bot

# Schedule

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Planning/Week** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Research on Microcontroller & Sensors |  |  |  | ✔ |  |  |  |  |  |  |  |  |  |
| Project Charter |  |  |  | ✔ |  |  |  |  |  |  |  |  |  |
| Purchasing of Parts |  |  |  |  | ✔ |  |  |  |  |  |  |  |  |
| **Implementation/Development** |  | | | | | | | | | | | | |
| Assemble and integrate the ESP32 microcontroller with sensors |  |  |  |  |  |  | ✔ |  |  |  |  |  |  |
| Sensor and Firmware Calibration |  |  |  |  |  |  |  |  | ✔ |  |  |  |  |
| Circuit and Power Management |  |  |  |  |  |  |  |  | ✔ |  |  |  |  |
| Set up IoT platform (ThingSpeak) and enable Data Transmission |  |  |  |  |  |  |  |  | ✔ |  |  |  |  |
| Implement predictive analytics using MATLAB in ThingSpeak |  |  |  |  |  |  |  |  |  | ✔ |  |  |  |
| Create Visualisation Tool Gauges and Plotted Graphs |  |  |  |  |  |  |  |  |  |  | ✔ |  |  |
| Develop user interface (TeleBot) for data visualisation and alerting notifications |  |  |  |  |  |  |  |  |  |  |  | ✔ |  |
| Integration of all units and Housing Component |  |  |  |  |  |  |  |  |  |  |  | ✔ |  |
| **Testing** |  | | | | | | | | | | | | |
| Demonstration and Evaluation of Prototype |  |  |  |  |  |  |  |  |  |  |  | ✔ |  |

|  |  |
| --- | --- |
|  | Everyone |
|  | Hardware Team |
|  | Software Team (Data Analysis) |
|  | Software Team (TeleBot) |
| ✔ | Denotes Actual Completion |

Our team followed a planned schedule to meet various project milestones. Most of them were completed timely, however, during the implementation and development phases, we encountered several challenges, particularly in hardware setup and data analysis.

The initial battery setup for the microcontroller provided insufficient voltage for the sensors, delivering only around 3.7V instead of the required 5V. To address this, we installed a converter to step up the voltage and ensure stable sensor operation. Since our team was new to IoT platforms, we faced a learning curve while configuring ThingSpeak. We had to make several adjustments and troubleshooting to the settings, coding scripts as well as data upload intervals. Furthermore, we encountered limitations with Arduino's capabilities, particularly in its inability to take screenshots. This prevented us from directly capturing visualisations like gauges and plotted graphs created from MATLAB for display on Telebot. Instead, we worked around this issue and created public URLs for ThingSpeak visualisations and provided links to users on Telegram. The users could then view current and predicted values in gauges and graphs through the URLs.

In summary, while most project milestones were achieved within the scheduled time frame, we encountered delays in the development phase due to hardware issues, ThingSpeak integration challenges and limitations of the Arduino microcontroller. Nonetheless, we adapted to these obstacles with alternative solutions, such as using a voltage converter, adjusting data upload intervals, and providing ThingSpeak URLs for data visualisation. Despite the long delivery times for some components including the PCB, the team worked collaboratively to ensure the project was completed on time.

|  |  |  |
| --- | --- | --- |
| PHASE | Planned Milestone Date | Actual Milestone Date |
| Initiating Phase |  |  |
| Planning Phase |  |  |
| Execution Phase |  |  |
| Closing Phase |  |  |
| Project End Date |  |  |

<Explain differences in between the planned and actual schedules>

<Description of Benefits>

# Cost

|  |  |  |
| --- | --- | --- |
| PHASE | Planned Costs | Actual Costs |
| Planning Phase  (Week 1 – 4) | $0.00 | $0.00 |
| Execution Phase  (Week 5 – 11) | $200.00 | $165.00 |
| Closing Phase  (Week 12 – 13) | $0.00 | $0.00 |
| Project Total Costs | $200.00 | $165.00 |

Planned Costs

<Explain differences between the planned and actual costs>

# Project Outcomes

|  |  |
| --- | --- |
| Project Deliverables | Learning Outcomes |
| Assembly and integration of ESP8266 microcontroller with sensors | We were able to integrate the ESP8266 with the sensors used with the help of the data sheets provided by the manufacturer using a breadboard for testing. |
| Calibration of sensors and firmware development |  |
| Circuit and Power Management | For power management, we have used two  6V 2W solar panels with the goal to make charging the 3.7V Li-Po battery faster, prolonging the lifetime of the system in absence of an external power supply.We chose this battery for its compact size, making it ideal for powering the ESP8266. Additionally, due to our dust sensor (PMS7003) requiring 5V we had to use  a DC-DC step up module to power output of the battery. However, output power is more than the input power from the solar panels which will eventually cause the battery to deplete over time. The decrease in the voltage output of the battery will cause our sensors to malfunction over time. Therefore, an improvement made could be choosing a battery with a higher output voltage. |
| ESP8266 microcontroller link to ThingSpeak IoT platform |  |
| IoT data visualisation and analysis and implementation of predictive analysis using MATLAB on ThingSpeak | Data retrieved from the ESP8266 can be visualised effectively using the MATLAB Visualisations application on the ThingSpeak IoT platform. With options like charts, plots, and gauges,we can effectively monitor and verify that the microcontroller and sensors are consistently collecting data as well as observing trends at a glance. The MATLAB Analysis application, on the other hand, enables us to apply machine learning models for predictive analysis of our data as well as checking for the accuracy of the models used.  With its extensive library of built-in functions, we can accomplish this task efficiently. |
| Telebot for data visualisation and alerting notifications | The Telebot API provides our system the ability to send/receive messages on Telegram. Initially, it was planned for the Telebot to send images of the (current or predicted) visualisation gauges upon user prompts. However, due to the limitations of Arduino’s libraries, this was not feasible. Therefore, we programmed the Telebot to send users the same information in text format, with the image of the visualisation gauges embedded as a link in the text message. Additionally, the Telebot automatically alerts users via text message if and when noise or dust levels approach hazardous levels. This feature completes the overall system and enables us to achieve our goal of a smart monitoring system. |

# Project Management Review

This section should cover the following points

* Project initiation

Review if the initial objectives are specific, measurable, achievable, relevant, and time bound.

* Project planning

Is the project properly scheduled? Are the roles and responsibilities clearly defined?

* Project manager role

Review communication and motivation strategies and their impacts on project progress and outcome. Elaborate on monitoring and control activities necessary to ensure health progress, and how changes and challenges are managed if any.

* Cost management
* Risk management

# Reflection

This section should cover the following points

* Engineering knowledge
* Problem Analysis
* Design/development of Solutions
* Individual and Team Work
* Future Recommendations.

Within the 13 weeks, there was much engineering knowledge utilized to resolve and overcome any challenges faced. the engineering knowledge and techniques used was mainly from Circuit Analysis and Analog Electronics

# Acknowledging/Declaring the Use of GAI

Please refer to NTU's Current Policy & Guidelines on the Use of Generative AI available in NTUlearn home page and the link:

<https://entuedu.sharepoint.com/sites/Student/dept/ctlp/SitePages/Exploring-the-Impact-of-Generative-Artificial-Intelligence-(GAI)-Tools-on-Education.aspx>

* Complete the following declaration if applicable.
* Create a Paper Trail to document the input prompt, output obtained, and how you have used it

I \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (student name), \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_@e.ntu.edu.sg (NTU email) honestly and sincerely make the following declaration in relation to the following course submission:

1. Name of course:

2. Course Code:

3. Instructor:

4. Title of Assignment/Project Submission:

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

i. Used GAI as permitted to assist in generating key ideas only. ☐

ii. Used GAI as permitted to assist in generating a first text only. ☐

And/or

iii. Used GAI to refine syntax and grammar for correct language submission only. ☐

Or

iv. As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project. ☐

I also declare that I have :

a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that

b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,

c. Apart from the foregoing notices, the submission is wholly my own work.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Name & Signature Date

# References

[1] ‘NodeMCU ESP8266 Specifications, Overview and Setting Up’. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.make-it.ca/nodemcu-details-specifications/>

[2] I. Hübschmann, ‘Is the ESP8266 WiFi Module the Right Choice for You?’, Nabto. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.nabto.com/is-esp8266-wifi-module-right-choice-for-you/>

[3] ‘Gravity\_\_Analog\_Sound\_Level\_Meter\_SKU\_SEN0232-DFRobot’. Accessed: Nov. 11, 2024. [Online]. Available: <https://wiki.dfrobot.com/Gravity__Analog_Sound_Level_Meter_SKU_SEN0232>

[4] ‘Temperature Barometer Sensor BMP280 - Default Title’, Kuriosity. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/temperatureebarometer-sensor-bmp280?variant=447572991347777>

[5] ‘ESP8266 with BME280 using Arduino IDE (Pressure, Temperature, Humidity) | Random Nerd Tutorials’. Accessed: Nov. 11, 2024. [Online]. Available: <https://randomnerdtutorials.com/esp8266-bme280-arduino-ide/>

[6] ‘PM2.5 Dust Sensor PMS7003’, Kuriosity. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/pm2-5-dust-sensor-pms7003>

[7] P. Zhou, "PMS7003 Series Data Manual," Plantower, Ver. 2.5, Jun. 2016. [Online]. Available: <https://www.espruino.com/datasheets/PMS7003.pdf>

[8] ‘3.7V~High Temperature Series - Zhongshun Xinneng’, ZONCELL. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.zoncell.com/en/product/lp1s-t/>

[9] ‘3.7V Li-Po Lithium Polymer Battery 100mAh 600mAh 1200mAh 1600mAh 2000m’, Kuriosity. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/3-7v-li-po-lithium-polymer-battery-100mah-600mah-1200mah-1600mah-2000mah-5000mah>

[10] ‘DC-DC Step Up Boost Converter 2.5 - VOUT to 5 - 12V (1A) 5W’, Kuriosity. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/dc-dc-step-up-boost-converter-2-5-vout-to-5-12v-1a-5w>

[11] ‘Solar Panel 6V 1W’, Kuriosity. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/solar-panel-6v-1w>

[12] ‘Solar Charge Controller Module 6V 2A - Kuriosity’. Accessed: Nov. 11, 2024. [Online]. Available: <https://kuriosity.sg/products/solar-charge-controller-module-6v-2a?_pos=1&_psq=solar+contro&_ss=e&_v=1.0>

[13] ‘Altium Designer - PCB Design Software’. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.altium.com/altium-designer>

**Appendix A -** Project Members Information

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Project contributions | Report Contribution |
| 1 | Muhammad Azfar Nasri Bin Azman | * Group Leader * Hardware Team: Hardware Research * Hardware Team: Hardware Connection, Testing and Troubleshooting * Hardware Team: ESP8266 / Arduino Programming   + Reading of Hardware Sensors   + Sensor Library   + ThingSpeak API   + Telegram Bot API   + Read and Write to ThingSpeak Server * Hardware Team: PCB Design   + Schematic Design   + 3D model and Tracing * Software Team (Telebot): Telegram Bot Code   + Read user commands & execute accordingly     - Current Values from Sensors     - Predicted Values from ThingSpeak   + Custom Keyboard   + Auto send Hazard Alerts     - Implement 10mins delay in alert interval * Purchasing of Hardware Items | * Project Summary * Hardware |
| 2 | Danial Ong | * Treasurer |  |
| 3 | Darryl Tan Han Yu | * Software Team (Data Analysis): Visualisation Gauge Tools * Software Team (Telebot): Telegram Bot Code * Software Team: Debugging |  |
| 4 | Dylan Ser Zi Ler | * Software Team (Data Analysis): Predictive Analysis for PM2.5 & PM10 * Software Team (Data Analysis): Data Visualisation Plotted Data Trends * Software Team (Data Analysis): Average Predicted Values & ThingSpeak Dashboard * Software Team (Telebot): Telegram Bot Code * Software Team: Debugging |  |
| 5 | Ignatius Chin Zheng Hao | * Software Team (Data Analysis): Predictive Analysis for Sound (dB) * Software Team (Data Analysis): Data Visualisation Gauge Tools & Plotted Data Trends * Software Team (Telebot): Telegram Bot Code |  |
| 6 | Muhammad Saajud S/O  Jiaudden |  |  |
| 7 | Geng Zhiwei |  |  |
| 8 | Bu Yizhe |  |  |
|  |  |  |  |
|  |  | e.g. project video, final packaging, business proposal, group report editor, final project integration, final presentation presenter, secured industry sponsorship, |  |

Sub-Group Members List:

Hardware Team: Azfar, Danial

Software Team (Data Analysis): Darryl, Dylan, Ignatius

Software Team (Telebot): Azfar, Darryl, Dylan, Ignatius

Appendix B – (no page limit for appendices)

[1]

M. Mir, F. Nasirzadeh, H. Bereznicki, P. Enticott, S. Lee, and A. Mills, ‘Construction noise effects on human health: Evidence from physiological measures’, *Sustainable Cities and Society*, vol. 91, p. 104470, Apr. 2023, doi: [10.1016/j.scs.2023.104470](https://doi.org/10.1016/j.scs.2023.104470).