Project Num: A2037-231

Final Year Project (FYP)

Smart Noise Level Monitoring System with IoT Cloud

Student Name:	Jovester Koh Ming Jie
Matriculation Num:	U2120547A
Project Supervisor:	A/P Chan Pak Kwong

School of Electrical and Electronic Engineering

A final year project report presented to Nanyang Technological University
in partial fulfilment of the requirements of the degree of
Bachelor of Engineering

Academic Year 2023/24

Table of Contents

Abstract

This report presents the development of a "Smart Noise Monitoring System using IoT Cloud" specifically tailored for monitoring noise levels within school campuses. The system integrates ESP8266, Blynk dashboard, and SEN0232 sound sensor to provide real-time monitoring capabilities.

Motivated by the adverse effects of noise pollution on student and faculty well-being, the project aims to enhance awareness and enable informed noise management strategies. The ESP8266 microcontroller collects data from the SEN0232 sound sensor, which is then transmitted to the Blynk dashboard for visualization and analysis.

The report discusses the system's architecture, hardware setup, and software implementation. By leveraging Internet of Things technology and its potential, this project looks to foster a conducive learning and working environment by proactively addressing noise pollution concerns.

Acknowledgement

I would like to express gratitude to Associate Professor Chan Pak Kwong for his invaluable guidance, unwavering support, and mentorship throughout the course of this project. Associate Professor Chan's encouragement and constructive feedback have been instrumental in the development and execution of this Final Year Project.

I am grateful for Associate Professor Chan's generosity with his time, knowledge, and resources, which have assisted me in the process of this project. His advises have helped me think clearly with a constructive mind to focus on the fundamentals while building ground up.

I am honoured and grateful to complete the project and learn from Associate Professor Chan Pak Kwong.

List of Figures

List of Tables

1. Introduction

Being the densely populated country that Singapore is, it is not uncommon to run into large crowds or experience loud noises in our daily lives. Loud and irritable noises can generally be experienced even in one's residence from a nearby construction, heavy road traffic, low-flying aircraft, etc. In any busy city like Singapore, it is found that road traffic often amounts up to at least half of the noise pollution [1]. These exposures to noise pollution can exert a considerable impact on individuals' daily lives. Therefore, it is paramount to monitor and understand the level of noise pollution experienced in our daily lives. Studies have illustrated effects of noise exposure have led to deteriorated sleep quality, increased risk of cardiovascular disease, and hindered children's cognitive performance [2]. Result of the study showed an increase in the risk of cardiovascular disease at 7% to 17% per 10 decibels (dB) increment of transportation noise shown in Figure 1 [2].

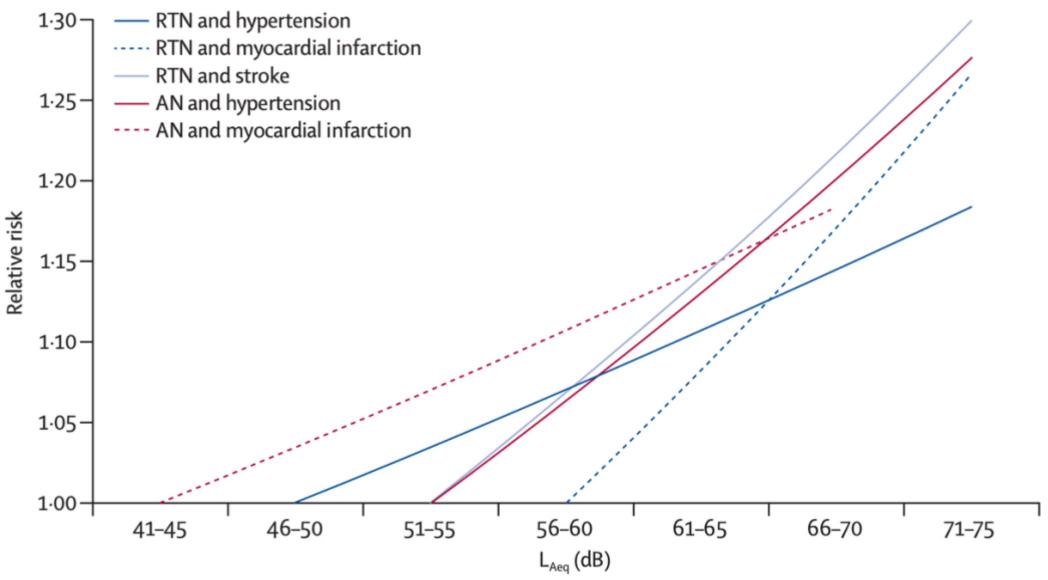


Figure 1: Correlation of transportation noise and cardiovascular disease risk

1.1 Project Objective

This project aims to enhance awareness and enable informed noise management strategies. The objective is to create a system that performs smart noise remote monitoring with an Internet of Things (IoT) cloud. The aim will be achieved by first measuring and analysing the noise level detected. Secondly, the system triggers an email alert to the administrator when the noise level reaches a certain threshold. These email alerts also help to act as a log for all of the triggered events and create a better understanding of the level of noise pollution experienced.

1.2 Project Scope

The usage of smart noise monitoring from this project will be crucial to one's understanding of where noise pollution can be expected and be minimised to lessen its impact. In this report, the loudness or the intensity of noise measured will be described in relative units of decibels (dB). The area of study chosen for this project will be done in the Nanyang Technological University (NTU) campus. The project will focus on the noise pollution present in a school affecting the students and faculty members. The project will set alert thresholds according to the recommendations from the United States' Centre for Disease Control and Prevention (CDC). Noise under 60dB is considered ideal for studying and exposure to noise above 70dB leads to hearing damage [3]. Hence, two thresholds will be set at 60dB and 70dB respectively for triggering the email alert.

1.3 Project Summary

The "Smart Noise Monitoring Level using IoT Cloud" project utilizes a microcontroller, IoT cloud dashboard, and a sound sensor to monitor the noise levels experienced in the NTU campus. Its aim is to increase awareness of noise pollution faced by students and faculty since noise has many negative impacts as discussed. With email alerts to administrators triggered from noise above 60dB and 70dB, the system helps the students and faculty to be more aware of the noise faced in the campus. Additionally, the IoT platform enables remote monitoring capabilities. This project addresses the importance of understanding and managing noise pollution for a healthier and more conducive learning environment.

2. Literature Review

In this report, the literature review on the components and software used for the developments of the project are presented. It includes the details of the components, diagrams, and the software used.

2.1 Hardware Components

ESP8266 NodeMCU

Figure 2: Illustration of an ESP8266 NodeMCU unit

In the development of a system with an IoT Cloud, it is necessary to leverage a board with an IoT platform and capabilities. Hence, the ESP8266 NodeMCU is selected to serve as the microcontroller (MCU), providing the computational power and connectivity capabilities essential for IoT applications.

The ESP8266 NodeMCU offers a cost-effective yet powerful microcontroller with built-in Wi-Fi connectivity [4]. Its compact size, ease of programming, and compatibility with the Arduino

Integrated Design Environment (IDE) make it an ideal choice for IoT prototyping and development.

The combination of hardware sensor components can be integrated with the ESP8266 NodeMCU. It forms a robust and versatile IoT platform capable of sensing, processing, and actuating in various applications, which will be instrumental to a successful implementation of an IoT cloud system. Specifications for ESP8266 NodeMCU is provided in Appendix A [4].

DFRobot Analog Sound Level Meter (SEN0232)

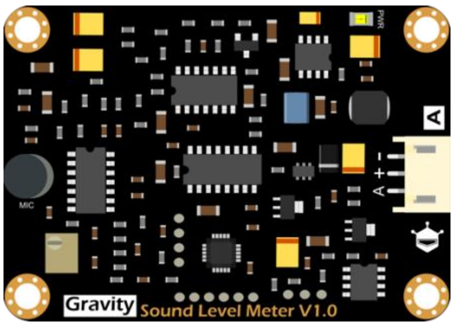


Figure 3: Illustration of a SEN0232 Analog Sound Level Meter

The DFRobot Analog Sound Level Meter, also referred to as a decibel meter or noise meter, serves as a fundamental tool for measuring environmental noise levels accurately. Utilizing an instrument circuit and a low-noise microphone, this device boasts high precision in its measurements. It operates within a wide input voltage range of 3.3-5.0V and provides a voltage output range of 0.6-2.6V [5]. The correlation between decibel value and output voltage is linear, simplifying the conversion process without the need for complex algorithms.

Sound level meters find extensive applications in detecting environmental noise, including monitoring highway noise and room noise levels. Therefore, this particular sound level meter proves suitable for integration into the project's scope. Specifications for the SEN0232 Analog Sound Level Meter is provided in Appendix B [5].

Solar Panel (6V 1.2W)

Figure 4: Illustration of the 115x90mm Solar Panel

This 6V 1.2W rated solar panel delivers a maximum of current output of 200 mA at 6V when exposed to ideal lighting conditions [6]. This solar panel measuring 115mm by 90 mm is made of polycrystalline material, which achieves a high level of solar energy conversion efficiency, ranging from 15% to 17% [6]. Furthermore, these polycrystalline solar cells are precision laser-cut to fit and encased within a durable resin surface, making it suitable in adverse weather conditions [6]. These polycrystalline solar cells offer significantly greater power output compared to amorphous thin-film solar panels, boasting two to three times the efficiency.

To integrate the solar panel into the system, the positive and negative terminals of the panel are soldered to the USB Voltage Regulator Module for a seamless connection.

DC-DC Voltage Regulator Module

Figure 5: Illustration of a DC-DC Voltage Regulator Module

The DC-DC Voltage Regulator Module operates by receiving an input of 6V-24V and stepping down to an output of 5V at the Universal Serial Bus (USB) [7]. Reverse input is prevented in this USB with a reverse polarity protection diode in the module [7]. This module serves the fundamental purpose of stabilizing and controlling the input voltage levels to the system, ensuring consistent power delivery to sensitive electronic components, thus safeguarding against voltage fluctuations and potential damage. The module is designed to operate with high efficiency of 97.5% and ensures a stable operation with overload protection. The specifications of the DC-DC Voltage Regulator Module is provided in Appendix C [7].

2.2 Compiler Software

Arduino Integrated Development Environment (IDE)

Figure 6: The open-source Arduino's Logo

Arduino is an open-source hardware and software provider. It offers a wide range of availability and resources and a comprehensive environment for developing, compiling, debugging, and uploading code to Arduino MCUs and other compatible boards [8].

Key Features:

1. Code Development: The Arduino IDE provides a powerful code editor based primarily on C and C++ language, or as Arduino refers as Arduino programming language [8]. The IDE is equipped with libraries and features, such as error checking and syntax highlighting, which facilitates the writing and debugging of code, streamlining the development process.
2. Board Management: The IDE offers seamless integration with a wide range of Arduino-compatible MCU boards, allowing the process of assigning the appropriate hardware

for application. This flexibility ensures compatibility and interoperability among the compiler, MCU, and other boards and sensor units.

3. Serial Monitor: The built-in serial monitor feature of the Arduino IDE facilitates real-time communication between the microcontroller and computer. It allows for debugging, accurate data visualization, and interaction with the running code, aiding in the testing and validation of our project.
4. Cross-Platform Support: The Arduino IDE is compatible with multiple operating systems (OS), including Windows, macOS, and Linux, ensuring accessibility and versatility across different development environments [8]. This will prove crucial as I am operating with macOS, which often comes with limitations on version availability.

The Arduino IDE serves as the primary software platform utilized in this project, to develop and upload firmware code to the ESP8266 NodeMCU, controlling various aspects of the system. From sensor data acquisition to communication with external devices and IoT cloud, the Arduino IDE provided the necessary tools and infrastructure to implement and test the functionality effectively. Its intuitive interface, extensive library support, and robust functionality have enabled efficient compiling of my program code has proven to be an invaluable asset in the development and implementation of this project.

2.3 Internet of Things (IoT)

The Internet of Things (IoT) has revolutionized the way that one interacts with everyday objects, enabling efficient and seamless connectivity and communication between devices over the internet [9]. In this project, I have leveraged IoT principles to create a smart and interconnected system, by choosing Blynk to serve as the chosen cloud platform. Blynk is a software firm offering cloud infrastructure and platforms tailored for IoT [10]. As Blynk connects a system to the cloud with IoT, it allows an user to generate a Graphic User Interface

(GUI) on their cloud platforms to approach the analog data collected from the sensors and devices. These analog data can be published as a datastream to the cloud for remote viewing on a dashboard.

Blynk dashboard is made available on a computer webpage and iOS and Android mobile applications. The live decibel data, coloured light indicators, and the graphic representation can be viewed remotely on the dashboard. In Figure 7, it shows the dashboard on Blynk's iOS application.

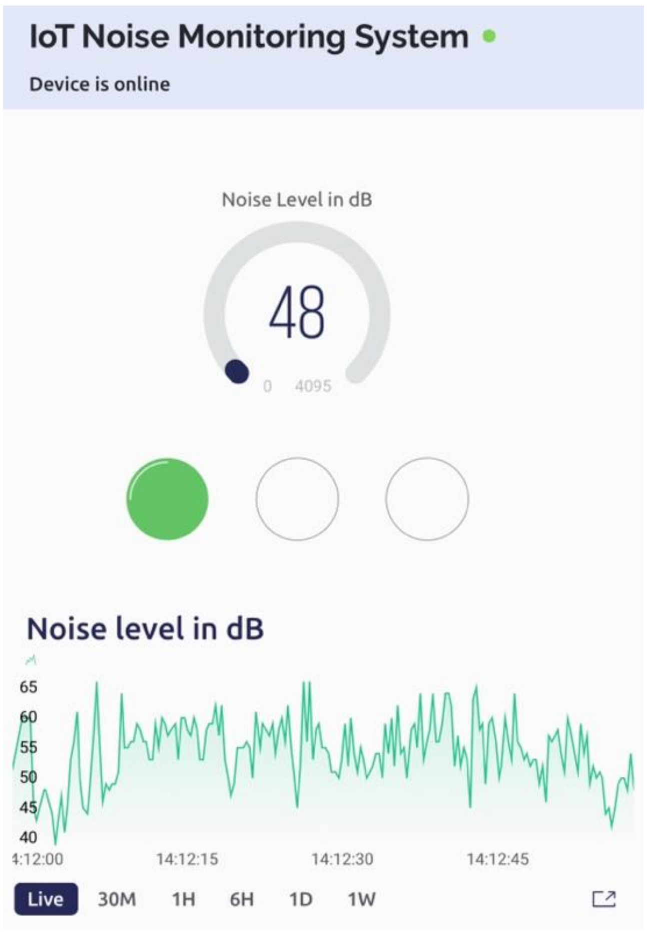


Figure 7: Cloud dashboard on Blynk iOS application

Similarly, the data can also be viewed on a web dashboard using a computer. The data view on both iOS and computer are the same, but only slightly represented differently as both use a different Application Programming Interface (API). In Figure 8, the web dashboard is shown with the live decibel data, coloured light indicators, and a graphic representation.

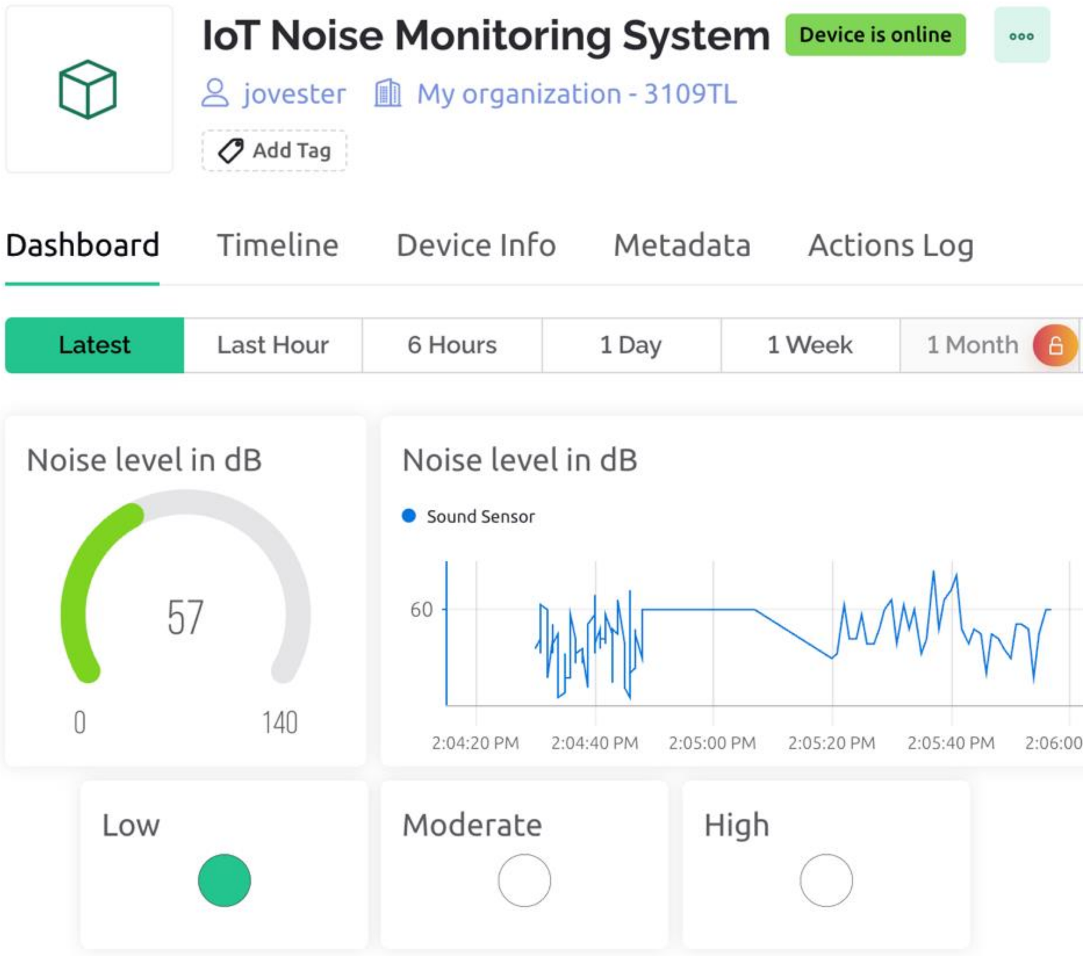


Figure 8: Blynk's web dashboard

Blynk's mobile iOS application is selected as the IoT cloud platform of choice for the deployment of this project. This is justified by its array of compelling features and advantages that align closely with the project's requirements and objectives. Utilising Blynk's cloud dashboard on the iOS application allows the deployment and usage of the Smart Noise Monitoring System with IoT Cloud to be more mobile and portable.

Besides, monitoring the live data can be done more conveniently and easily with a smart phone instead of hunkering down on a computer web dashboard. Hence, the comprehensive features of Blynk's dashboard on an iOS application offers the flexibility and scalability that will be crucial to the development of the project.

3. Self-reliant Power System

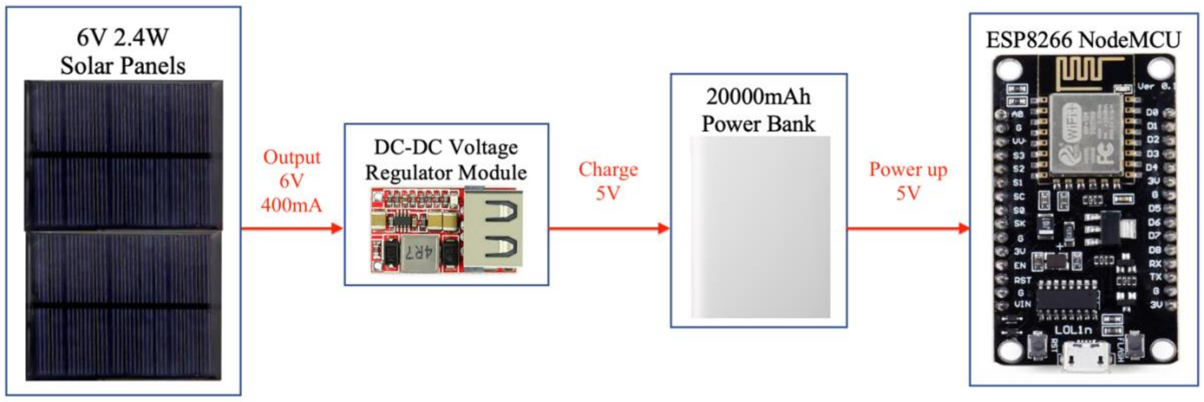


Figure 9: Flow diagram of the self-reliant power system

This project will operate independently on its own power source, eliminating the requirement for a connection to an AC power outlet. This feature renders the system portable and mobile, facilitating easy and rapid deployment in any location. The system relies on two units of solar panels, a DC-DC Voltage Regulator Module, and a 20000mAh power bank.

Firstly, the two units of the solar panel are connected in parallel to provide a maximum output of 6V and 400mA under ideal condition. The positive terminals of the two solar panels are soldered together and similarly the negative terminals of the two solar panels are also soldered together to provide the parallel connection. The solar panels were tested in the Project Laboratory and an output of 4V was measured with a multimeter, as shown in Figure 10. This can be expected as the laboratory is not under the direct sunlight and hence will not output anywhere near 6V.

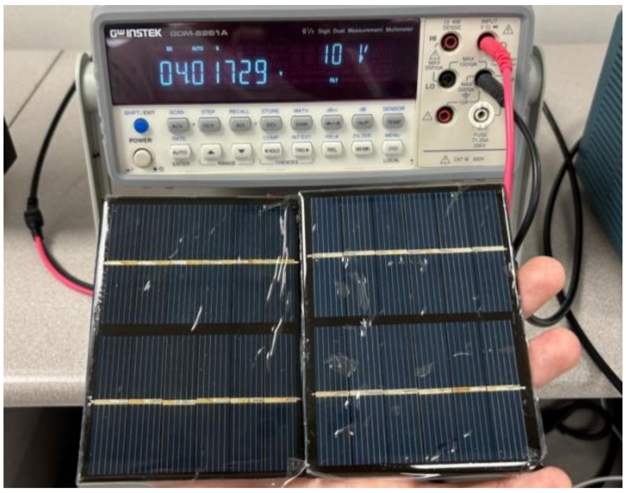


Figure 10: Voltage reading of two parallel solar panels

Secondly, the positive and negative terminals of the solar panel are soldered onto the DC-DC Voltage Regulator Module to step down the 6V input voltage from the solar panels to 5V at the output through the USB.

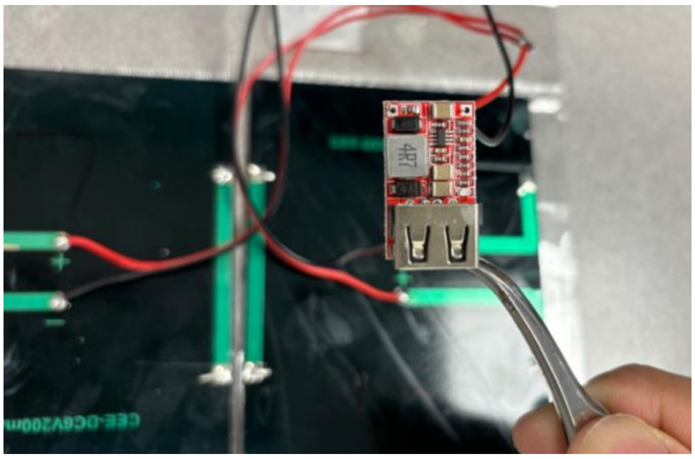


Figure 11: Soldering of the positive and negative terminals

Thirdly, the solar panels with the voltage regulator module were tested to measure the output by the solar panels with a digital multimeter. Under cloudy conditions, the solar panels were outputting currents of 196mA. In conditions when sunlight was much brighter, the solar panels were outputting currents of 321mA. These measurements are shown in the Figure 12.

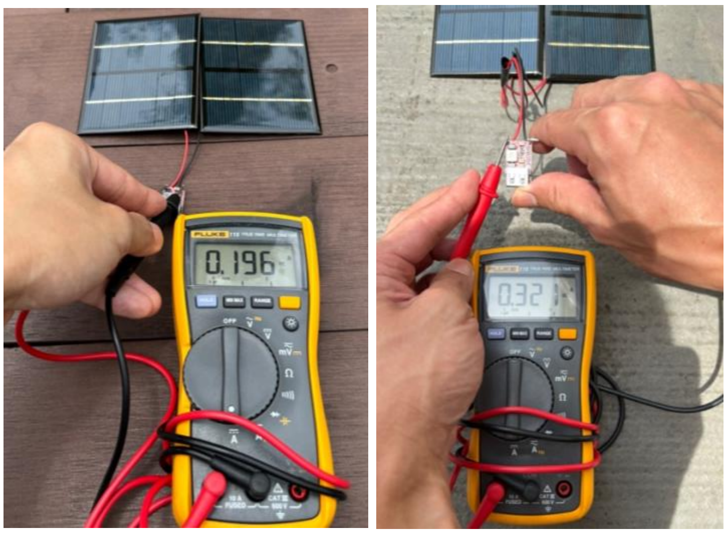


Figure 12: Current measurements of the solar panel in different conditions

The solar panel provides a maximum output of 400mA, under ideal condition, to charge the power bank. A 20000mAh power bank connects to the ESP8266 NodeMCU and powers the system with 5V. The supply voltage of 5V which is then regulated internally by the board to the required operating voltage of 3.3V. This internal voltage regulation ensures that the ESP8266 microcontroller and other components on the NodeMCU board receive a stable and appropriate voltage for their operation.

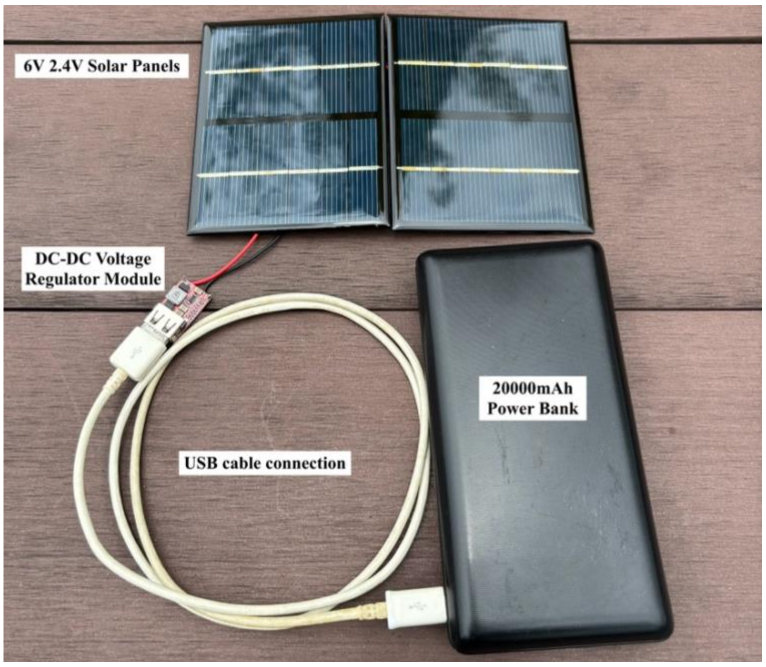


Figure 13: The connection of the self-reliant power system

To estimate the total current discharge for the system, assume the following:

- The system runs 12 hours a day
- There are 12 hours of sunlight a day
- The solar panels output a maximum of 400mA under ideal conditions
- The 20000mAh power bank is fully charged when used at the start of the day

The current consumption of each component is considered and then summed up.

1. ESP8266 active Wi-Fi transmission: 70mA [4]
2. SEN0232: 22mA [5]

To calculate net current consumption:

= ESP8266 active current + SEN0232 current − Charging current

= (70 mA + 22 mA) – 400 mA

= 704 mA

The effective current discharge from power bank for 12 hours a day:

= Net current × Sunlight hours

= 704 mA × 12 hours

= 8448 mAh

Number of days of power bank usage:

= Power bank capacity (mAh) / effective current discharge (mAh)

= 20000 mAh / 8448 mAh

≈ 2 Days 9 Hours

Therefore, the total estimated runtime for the system powered by the power bank and charged by the solar panels is approximately 2 days and 9 hours under ideal condition, before the power bank is depleted.

4. System Integration

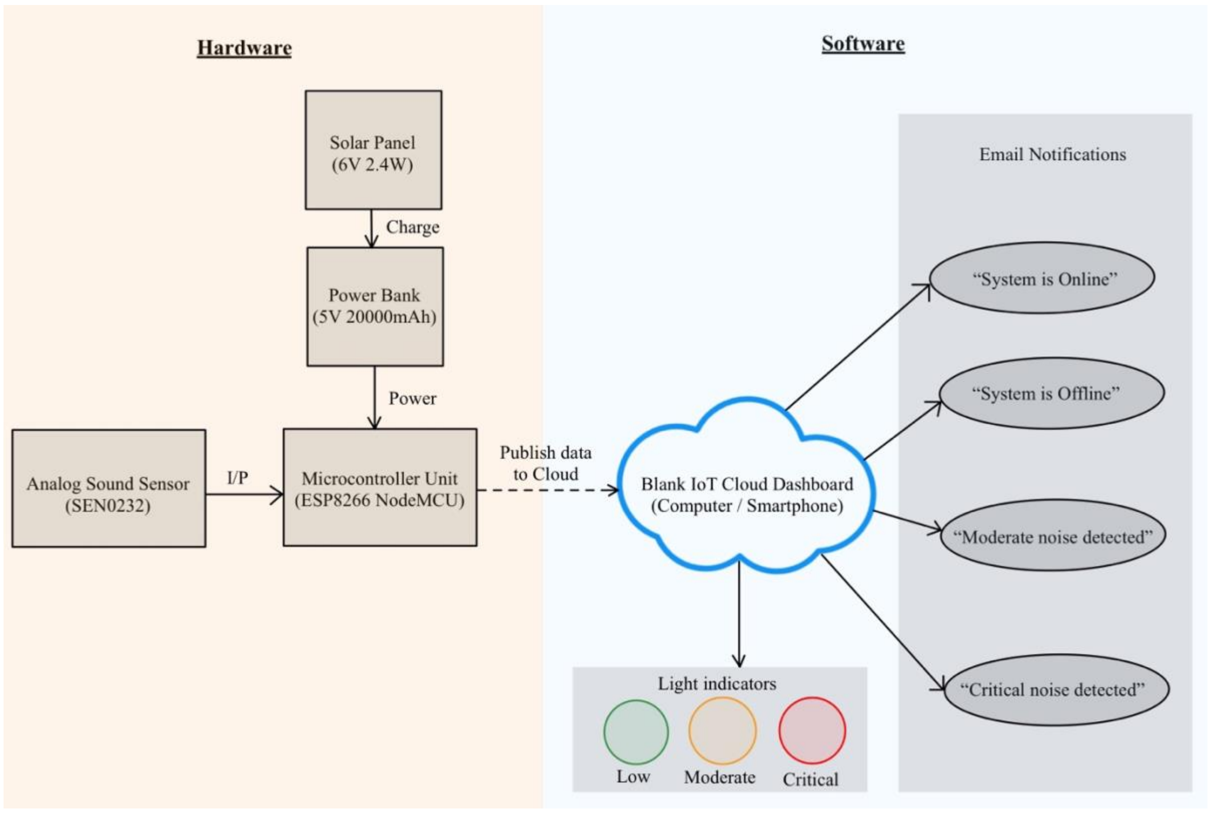


Figure 14: Block diagram of the general system

The block diagram for the Smart Noise Monitoring System with IoT cloud is presented above in Figure 14. The SEN0232 analog sound sensor detects surrounding noise accurately and this raw analog data is input to the ESP8266 NodeMCU. The MCU reads the analog raw data with the program and converts the analog data to display decibel units. The MCU is connected to Wi-Fi and publishes the decibel readings to Blynk's cloud platform, which can be monitored using a dashboard on Blynk. Blynk dashboard is able to monitor live data on the surrounding noise in decibels.

The system is programmed to publish alert notifications to the user when the system comes online or offline and when noise above a certain loudness is detected. Noise below 60dB is considered as normal, noise at 60dB and above will publish a "moderate noise" alert. Lastly, noise 70dB and above will publish a "critical noise" alert. The system also displays different light indicator colours according to the noise level. Green light is displayed when noise is below 60dB, orange light is indicated between 60dB-69dB, and red light is indicated 70dB and above. The connection between the components is shown in Figure 15.

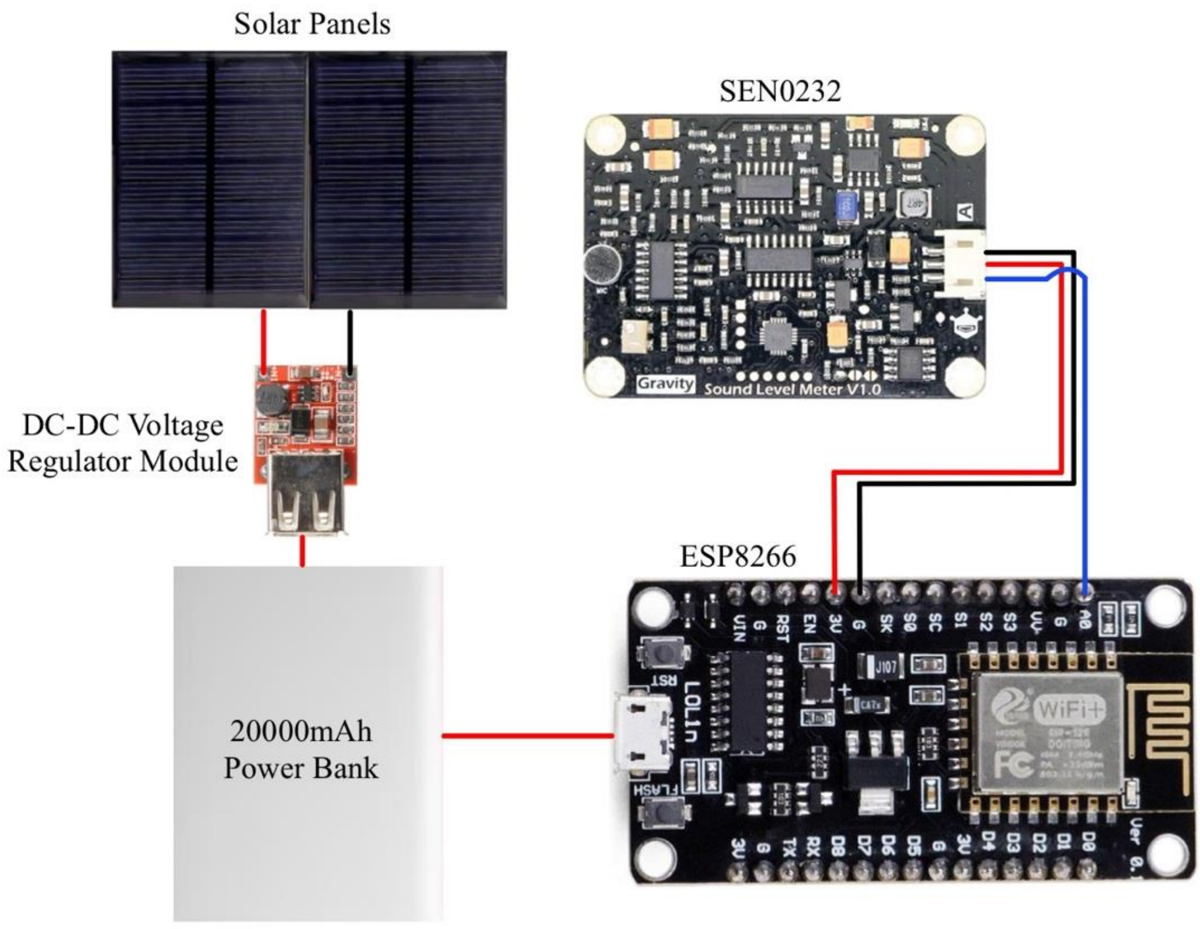


Figure 15: Components connection of the project

The sound sensor is first tested to detect voltage amplitudes from sound levels and display real-time data on a serial plotter using the Arduino IDE as shown in Figure 16. The amplitude mirrors the volume of the sound and shows a flat signal when no sound is present.

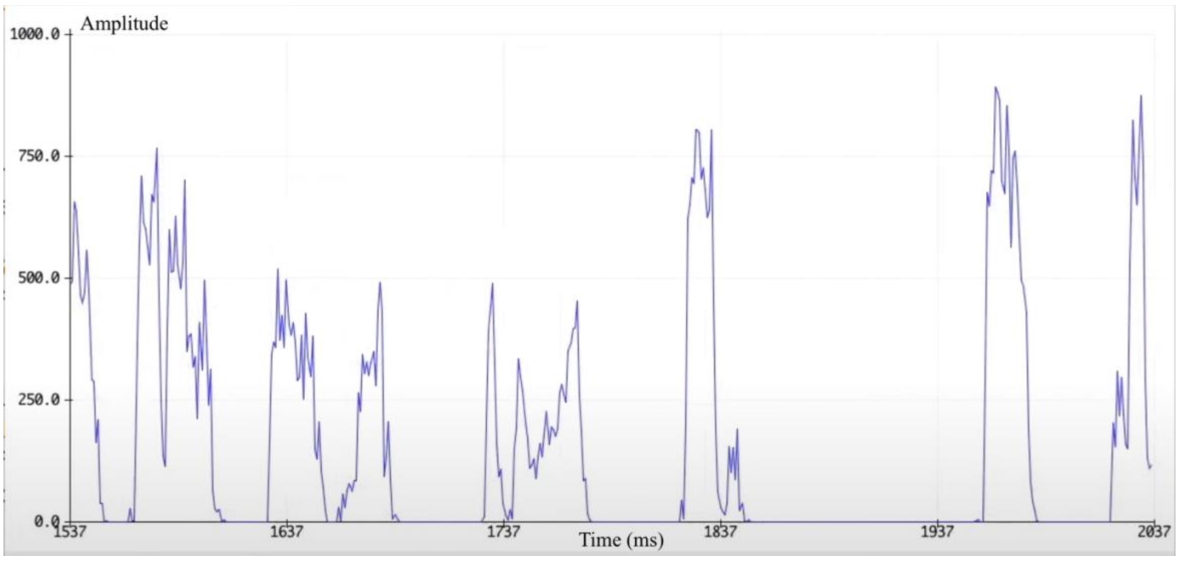
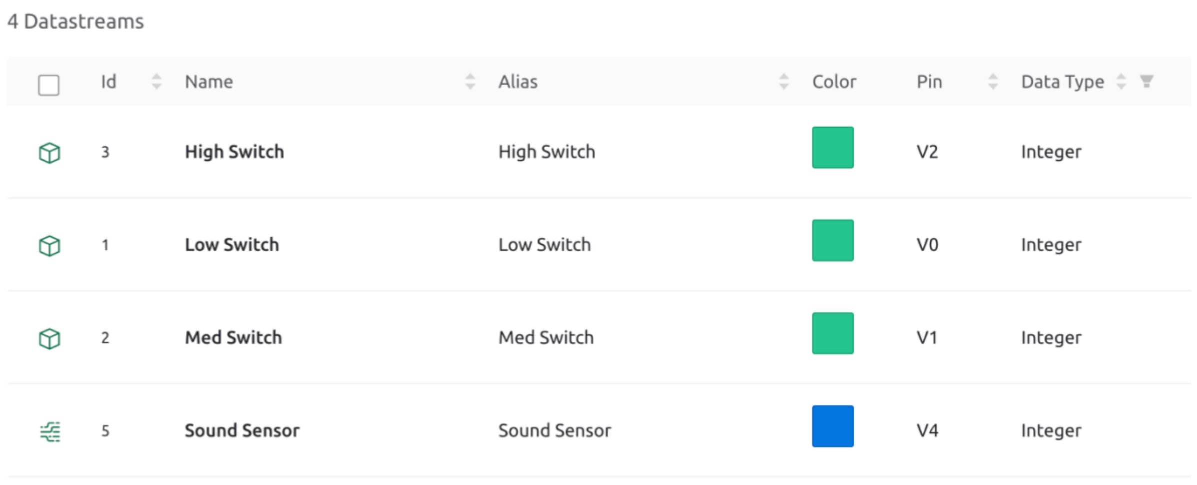


Figure 16: Real-time sound levels detected and displayed on a serial plotter

Once the sound sensor has been tested to be working, the raw analog data is then converted to decibel units in the program code and published to the cloud for monitoring. The raw data is programmed to be converted to decibel units, then published to the cloud datastream in Blynk.



4 Datastreams

	Id	Name	Alias	Color	Pin	Data Type
	3	**High Switch**	High Switch	🟩	V2	Integer
	1	**Low Switch**	Low Switch	🟩	V0	Integer
	2	**Med Switch**	Med Switch	🟩	V1	Integer
	5	**Sound Sensor**	Sound Sensor	🟦	V4	Integer

Figure 17: Blynk Datastream on the cloud

In Figure 18, the device's online and offline status is configured to publish an alert and email notification each time the device is online and goes offline.

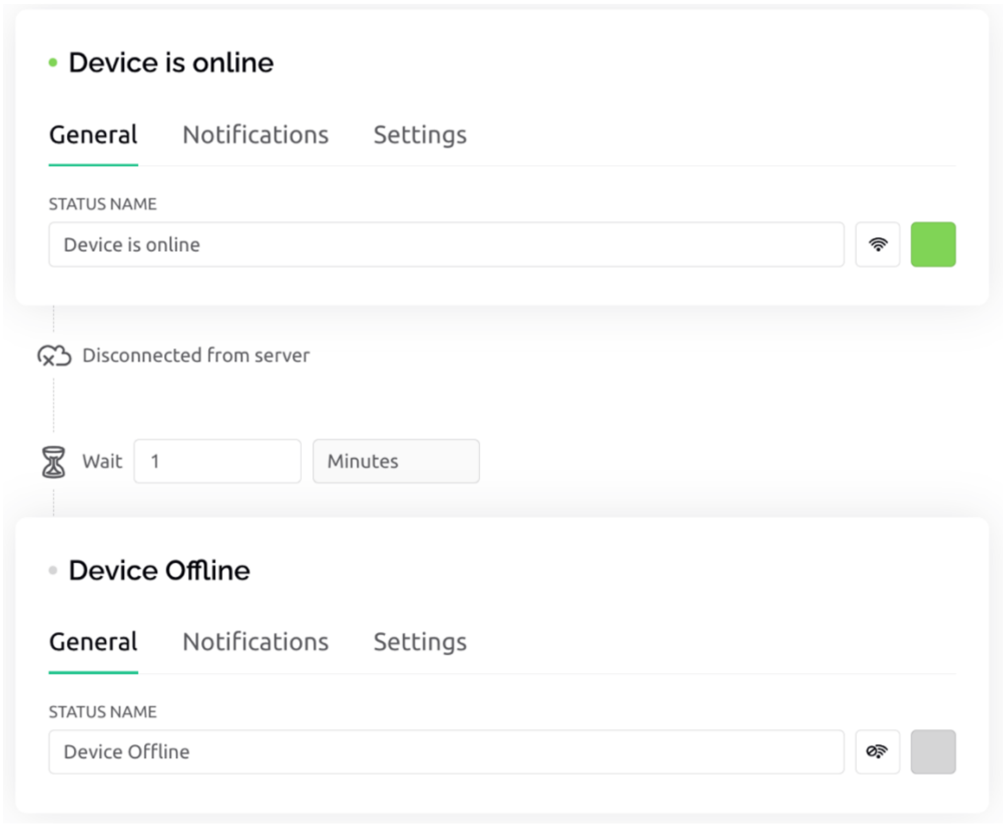
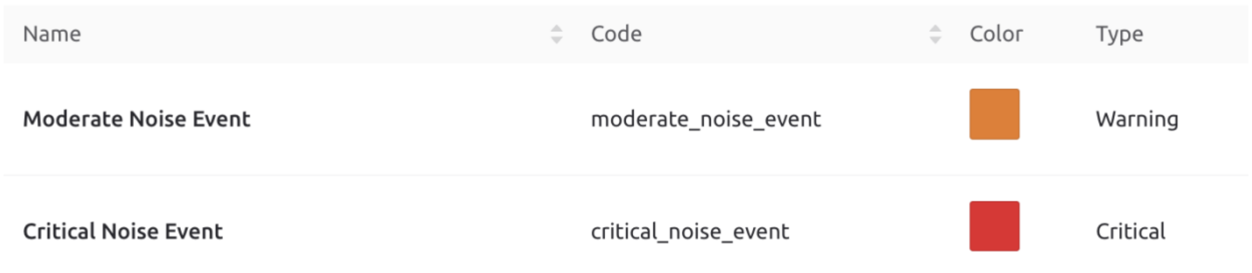


Figure 18: Online/Offline status notification of a device

In Figure 19, the events shown are configured to correspond to the notifications published from the MCU board via Wi-Fi transmission



Name	Code	Color	Type
Moderate Noise Event	moderate_noise_event	🟧	Warning
Critical Noise Event	critical_noise_event	🟥	Critical

Figure 19: Noise notification managed on the Blynk cloud platform

The system is tested in an environment with noise, such as drilling noises, and has shown high accuracy in noise detection and results publishing. The drilling noise is recorded at above 70dB and will light up the indicator to red and publish a "Critical Alert" notification to the user. In Figure 20, it shows the live decibel readings during the test on the Blynk iOS dashboard. The results are compared across an iOS decibel meter application to study the accuracy of the decibel reading published to the cloud. As shown in Figure 20, the decibel meter reading can be seen in red at the top of the screen. The drilling noise can be monitored live and seen across the chart with different timeline. The study has proved the results to be accurate by measuring the noise together with a decibel meter to prove its reliability and accuracy. The system automatically publishes alerts and email notifications to the user, as it is programmed according to requirements in the program code.

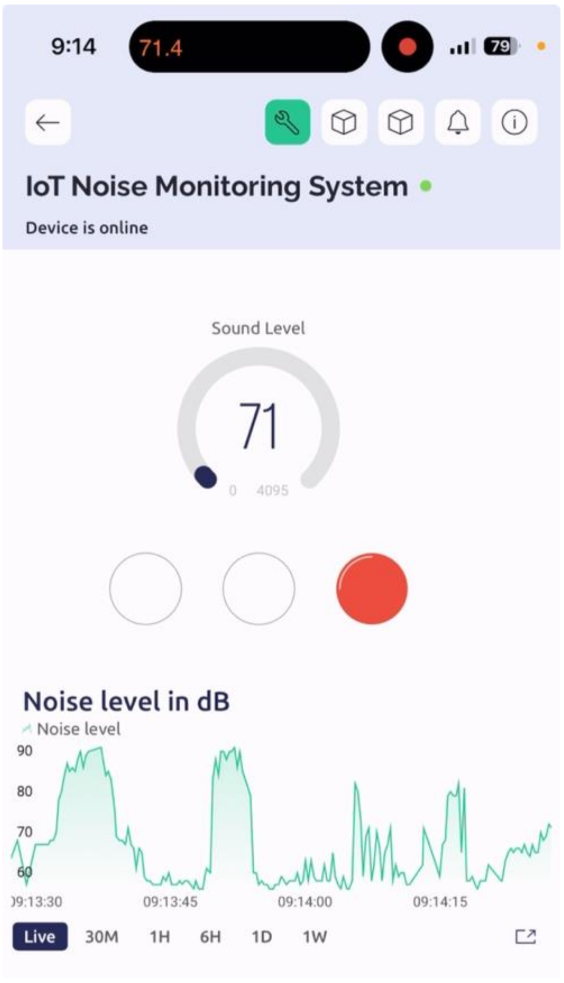
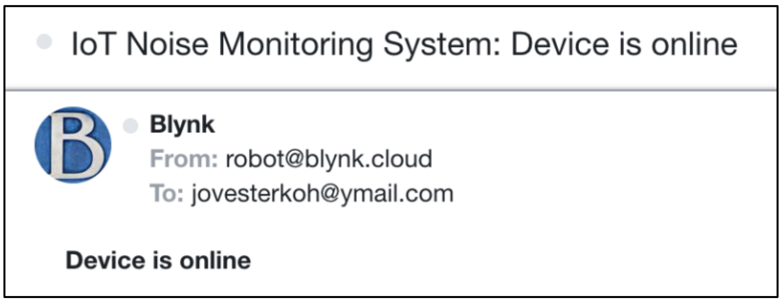


Figure 20: The Blynk iOS dashboard during a test with drilling noise

From Figure 21 to 24, different email notifications published by Blynk are displayed.

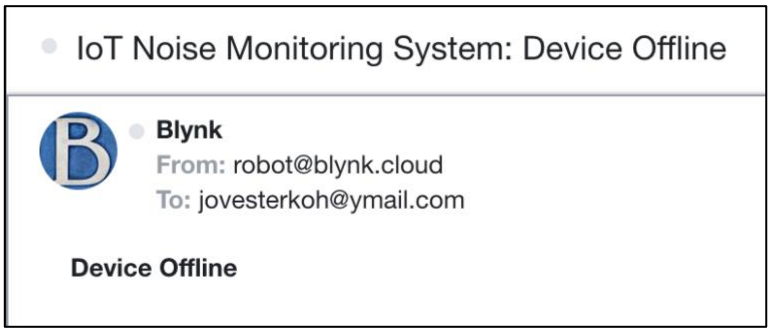


IoT Noise Monitoring System: Device is online

Blynk
From: robot@blynk.cloud
To: jovesterkoh@ymail.com

Device is online

Figure 21: Blynk email online notification

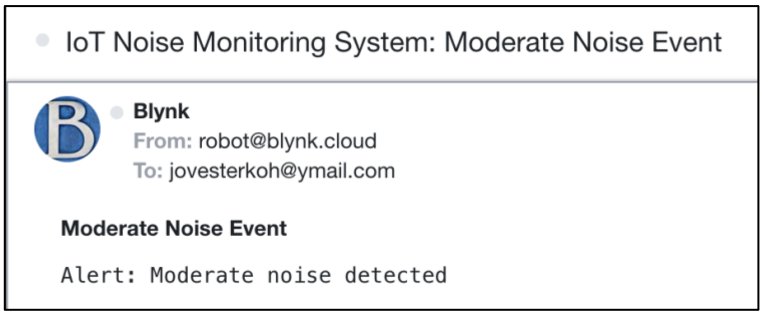


IoT Noise Monitoring System: Device Offline

Blynk
From: robot@blynk.cloud
To: jovesterkoh@ymail.com

Device Offline

Figure 22: Blynk email offline notification



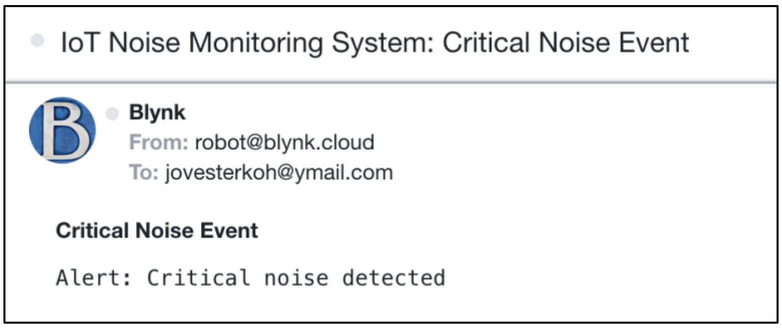
IoT Noise Monitoring System: Moderate Noise Event

Blynk
From: robot@blynk.cloud
To: jovesterkoh@ymail.com

Moderate Noise Event

```
Alert: Moderate noise detected
```

Figure 23: Blynk email moderate noise alert



IoT Noise Monitoring System: Critical Noise Event

Blynk
From: robot@blynk.cloud
To: jovesterkoh@ymail.com

Critical Noise Event

```
Alert: Critical noise detected
```

Figure 24: Blynk email critical noise alert

From Figure 25 to 28, it shows different notification alerts published by the Blynk cloud to the user through iOS application notifications.

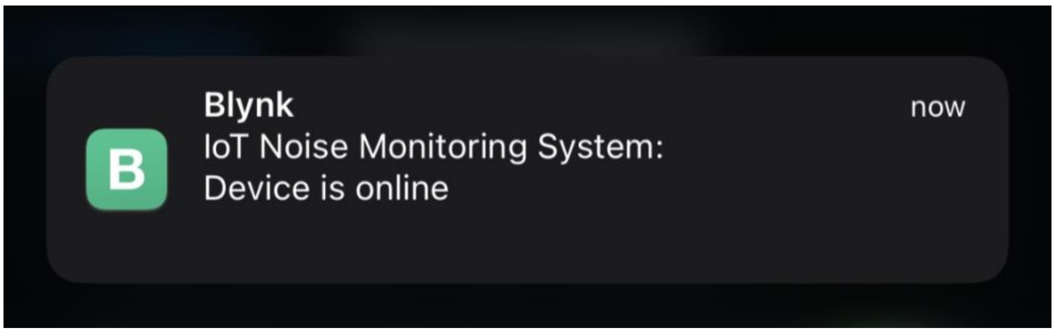

Blynk
now
IoT Noise Monitoring System:
Device is online

Figure 25: Blynk device online notification

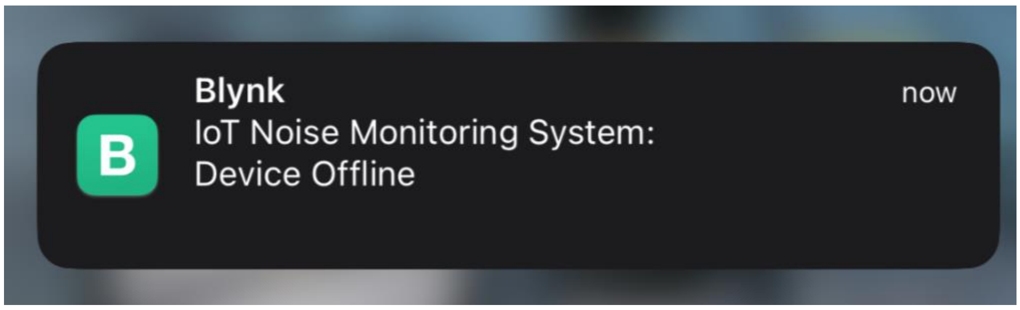

Blynk
now
IoT Noise Monitoring System:
Device Offline

Figure 26: Blynk device offline notification

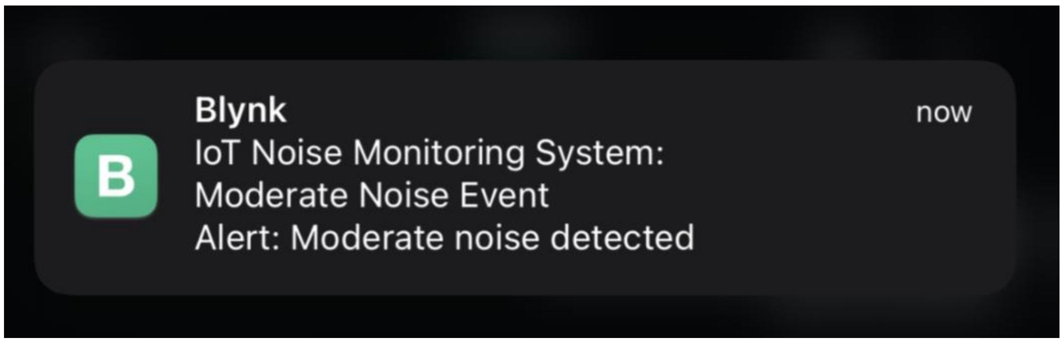

Blynk
now
IoT Noise Monitoring System:
Moderate Noise Event
Alert: Moderate noise detected

Figure 27: Blynk moderate noise alert

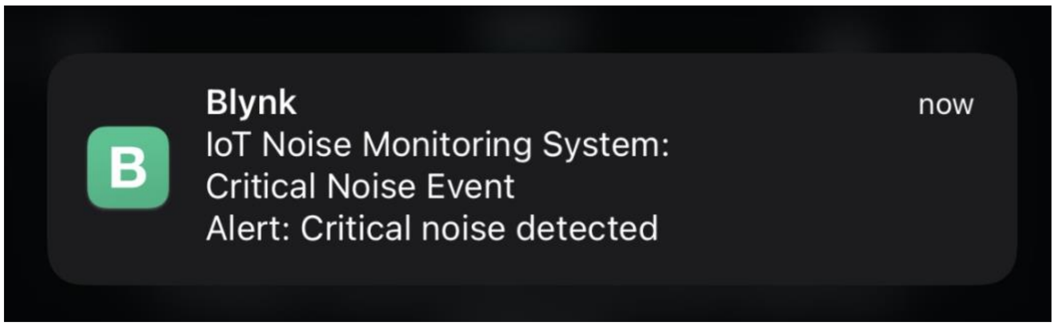

Blynk
now
IoT Noise Monitoring System:
Critical Noise Event
Alert: Critical noise detected

Figure 28: Blynk critical noise alert

5. Comparative Analysis

A comparative analysis is conducted to study the differences between the Smart Noise Monitoring System with IoT Cloud to other similar projects that encompass similar purposes and devices used. There are many projects that utilise a sound sensor to monitor noise, however the unique propositions offered by the Smart Noise Monitoring System with IoT Cloud are detailed as such.

Firstly, many projects detect the presence of noise in the environment by monitoring the change of output voltage value from the sensor. However, this change of output voltage value needs to be sufficiently large enough to be recognise by the sensor as noise. This means that these projects are not able to pick up noise that are softer and lower in volume. In contrast, noise detection with the Smart Noise Monitoring System with IoT Cloud has proven to be accurate and demonstrated a wide range of noise detection. The high accuracy of noise detection was proven in accordance with a decibel meter.

Secondly, since these projects perform noise detection by utilising the large differentiation of output voltage value, the purpose of these projects is commonly meant to detect a sudden change in noise. For example, these noise detected are loud clapping sound that can trigger a sudden change of voltage value. These projects monitor noise with a poor resolution and cannot detect small changes of volume. In comparison, the Smart Noise Monitoring System with IoT Cloud displays an excellent precision for the change of noise detected in decibel with a resolution of 0.01dB.

In addition, these projects that monitor noise are often demonstrated on a workstation when powered up with a wired connection to a computer. None of these projects has shown the ability and flexibility to deploy noise detection anywhere. Whereas, the Smart Noise Monitoring System with IoT Cloud offers the mobility of accurate noise detection with a self-reliant power system that enables the system to be deployed anywhere. Furthermore, the Smart Noise Monitoring System with IoT Cloud offers remote-monitoring of the real-time noise in decibel with a smartphone instead of a computer, further enhancing the mobility of this system.

These unique propositions offered by the Smart Noise Monitoring System with IoT Cloud is summarised in Table .

Table 1: Comparative analysis of similar noise-monitoring projects

	Similar projects	Smart Noise Monitoring System with IoT Cloud
Detection Resolution	Poor resolution	Precise resolution of 0.01 dB [5]
Noise detection accuracy	Poor accuracy of sound detection	Accurate wide range of detection from 30dB to 130dB [5]
Display of results	On a computer	Easy remote monitoring with a smartphone
Mobility	Only when powered by computer wired connection	Anywhere with the self-reliant power system

6. Schedule

Table 2: Project phase timeline

Project Phase	Planned Milestone Start	Actual Milestone Start
Literature Review	14 August 2023	14 August 2023
Conceptualization	22 September 2023	19 September 2023
Procurement	16 October 2023	23 October 2023
Development	06 November 2023	13 November 2023
Testing & Evaluation	11 December 2023	01 January 2024
Deployment	04 March 2024	01 March 2024
Completion	15 April 2024	08 April

Initial discussions and meetings with Associate Professor Chan was conducted in timely manner and started off in a good direction. However, the start of the development phase was slightly delayed as I was preparing for the final examination and was not able to provide much progress on the FYP. Additionally, the testing & evaluation phase was initially scheduled to start during the winter break but was pushed back by a month due to a personal health reasons. This delay caused a huge gap to between the planned and actual milestone and set me back as to where I should have been in the project. During the deployment phase, my sound sensor unfortunately became faulty and led to more time spent on troubleshooting. Without a spare sensor to swap out, this unseen circumstance caused a delay in the deployment. The project is aimed to be completed by 15 April 2024.

Table 3: Table of summary project milestones

Summary Project Milestones	Schedule Date
Project Plan/Strategy	18 September 2023
Interim Report & Video Presentation	14 November 2023
Draft Final Report	28 March 2024
Final Project Report	12 April 2024
Project Demonstration	15 – 19 April 2024
FYP Oral Presentation	14 May 2024
Final Report (revised)	22 May 2024

7. Cost

The budget allocated for a Final Year Project is S$400. The procurement of project parts was kept well within the allocated budget of S$207.57. The project's expenses were made solely for required project parts and spare redundancies. Any expense for cloud subscriptions were avoided in accordance to the faculty's claimable expenses. The details of the project's expenses are listed in Table 4.

Table 4: Expenses for the project

S/N	Item	Quantity	Spending
1	KY-038 Analog Sound Sensor Module	1	$ 7.58
2	ESP8266 NodeMCU	1	$ 16.74
3	USB DC-DC Stepdown Voltage Regulator Module	2	$ 13.08
4	5V 1.5W Solar Panel	1	$ 20.17
5	SEN0232 DFROBOT Analog Sound Level Meter	2	$ 120
6	6V 1.2W Solar Panel	2	$ 30
Total Spending:			$ 207.57

8. Reflection on Learning Outcome Attainment

Limitations & Challenges Faced

There were a few challenges faced in the course of this project. One of the challenge faced was finding a suitable cloud software that was suitable and compatible for use with the iOS and MacOS systems. As I am using MacOS and iOS system to couple with the couple for deploying the project, I was met with fewer options as there were lesser software that were made available to the Apple application store. In software that are compatible with a MacOS, it often comes with limited functionalities when compared to the Android version release. This has caused significant inconvenience as I had to work around the issue on version availability and limited functionalities. Due to this inconvenience, I am now more well-versed and knowledgeable in the functionalities and availability of cloud platform and software that are made available to MacOS and iOS users.

In addition, I encountered another challenge towards the end of my project. Due to the flimsy and fragile characteristics of the analog sound sensor, it became faulty and stopped outputting the way as it had been accurately displaying. After multiple attempts of troubleshooting, none of my efforts has shown to fix the issue of the unresponsive sound sensor and hence I determined it as faulty. Without any spare sensor in hand, I was led to additional lead time in the project due to the purchase of another new sensor component as an replacement. From this unexpected occasion, I have learnt the importance of preparing spare redundancy as an aspect of project management as electronic parts are subjected to failure anytime. I believe this valuable lesson will carry on to be applicable in all other projects.

Moreover, I met another challenge of incorporating smart features in the noise monitoring system which proved to be very challenging, since microcontrollers used in projects are generally a small board with limited memory and processing capabilities. In this course of the project, I tried to introduce a smart feature of predicting the type of noise by using Fast Fourier Transform (FFT) to analyse the frequency spectrum and predict types of noise using the certain spectrum generated by a type of noise. However, the results showed little to zero accuracy as MCUs do not have the appropriate amount of processing capability and memory needed for signal processing technique that require an intensive amount of memory. This feature may be realised with hardware components that are better equipped to handle signal processing techniques.

Possible Future Recommendations

Possible add-ons to the project can include using a better and more advanced MCU board, with more memory and processing capability, to handle signal processing techniques such as FFT.

If that is possible, FFT can be introduced to incorporate a smart noise detection to detect the type of noise picked up by the sensor.

Another recommendation can include using a higher-rated solar panel to increase the output by the panels to entirely support the current consumption required by the boards. This can eliminate the need for a power bank to power the operation.

9. Conclusion

The purpose of the project Smart Noise Monitoring System with IoT Cloud is to target the detection of the noise in our everyday lives. Noise has been proven to negatively impact health, with exposure to loud and prolonged noise, it has shown to lead to an increased in cardiovascular diseases and children's cognitive performances. Hence it is important to detect the noise and to be more aware of the noise we face every day in places of study and work.

The project targeted the issue of noise detection in the compounds of the NTU's campus and has proven to publish reliable results to Blynk's IoT cloud for monitoring. I have also successfully programmed the code to publish alerts and notifications to the user when the system becomes online or offline and when the noise level detected is above 60dB and 70dB respectively.

While the project has areas to improve on for future recommendations, such as using a more advanced MCU to introduce signal processing technique. This project is instrumental to noise detection with the usage of IoT cloud and has shown the accurate detection of noise and how clearly it can be heard and affects us.

References:

[1] Kundu Chowdhury, A. Debsarkar, and S. Chakrabarty, “Critical assessment of day time traffic noise level at curbside open-air microenvironment of Kolkata City, India,” *Journal of Environmental Health Science and Engineering*, vol. 13, no. 1, Sep. 2015. doi:10.1186/s40201-015-0219-6

[2] M. Basner *et al.*, “Auditory and non-auditory effects of noise on health,” *The Lancet*, vol. 383, no. 9925, pp. 1325–1332, 2014. doi:10.1016/s0140-6736(13)61613-x

[3] “Public Health and Scientific Information,” Centre for Disease Control and Prevention (CDC), www.cdc.gov (accessed Dec. 18, 2023).

[4] “NodeMCU ESP8266,” *Arduino Official Store* https://store.arduino.cc/products/nodemcu-esp8266 (accessed Feb. 19, 2024).

[5] “Gravity__Analog_Sound_Level_Meter_SKU_SEN0232-DFRobot,” *wiki.dfrobot.com*. https://wiki.dfrobot.com/Gravity__Analog_Sound_Level_Meter_SKU_SEN0232 (accessed Feb. 22, 2024).

[6] “Solar Panel 6V 200mA (115mmx90mm),” *Continental Electronics*, Mar. 11, 2024. https://continental.sg/product/solar-panel-6v-200ma-115mmx90mm/ (accessed Feb. 22, 2024).

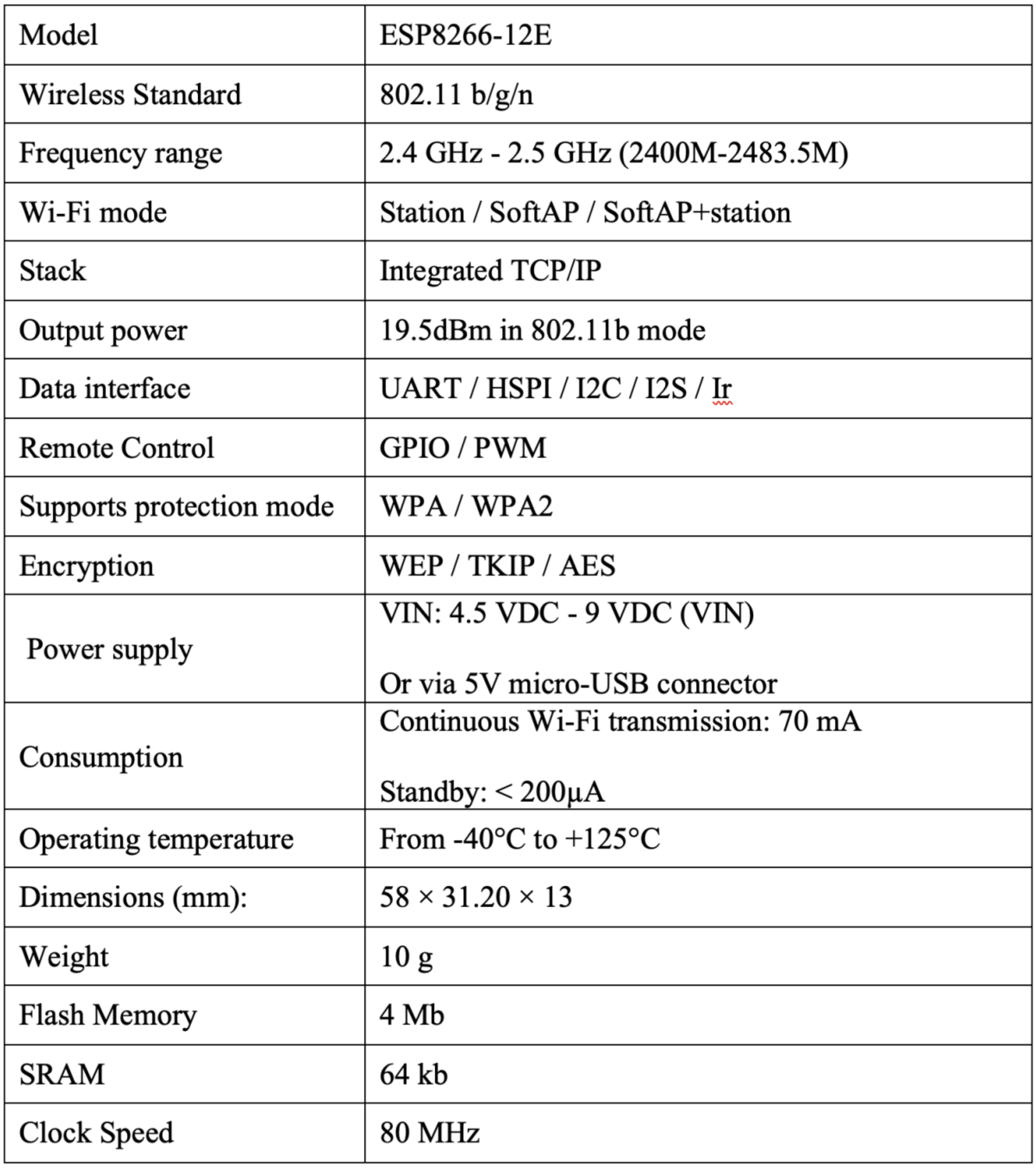
[7] “6-24V to 5V 3A USB DC-DC Buck Step-Down Converter,” *Makerlab Electronics*. https://www.makerlab-electronics.com/products/6-24v-to-5v-3a-usb-dc-dc-buck-step-down-converter (accessed Mar. 11, 2024).

[8] Ben, “What is an Arduino? - learn.sparkfun.com,” *Sparkfun.com*, 2019. https://learn.sparkfun.com/tutorials/what-is-an-arduino/all (accessed Feb. 02, 2024).

[9] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, S. Wook Baik, and M. Sajjad, “Enabling Automation and Edge Intelligence over Resource Constraint IoT Devices for Smart Home,” *Neurocomputing*, Nov. 2021, doi: https://doi.org/10.1016/j.neucom.2021.04.138.

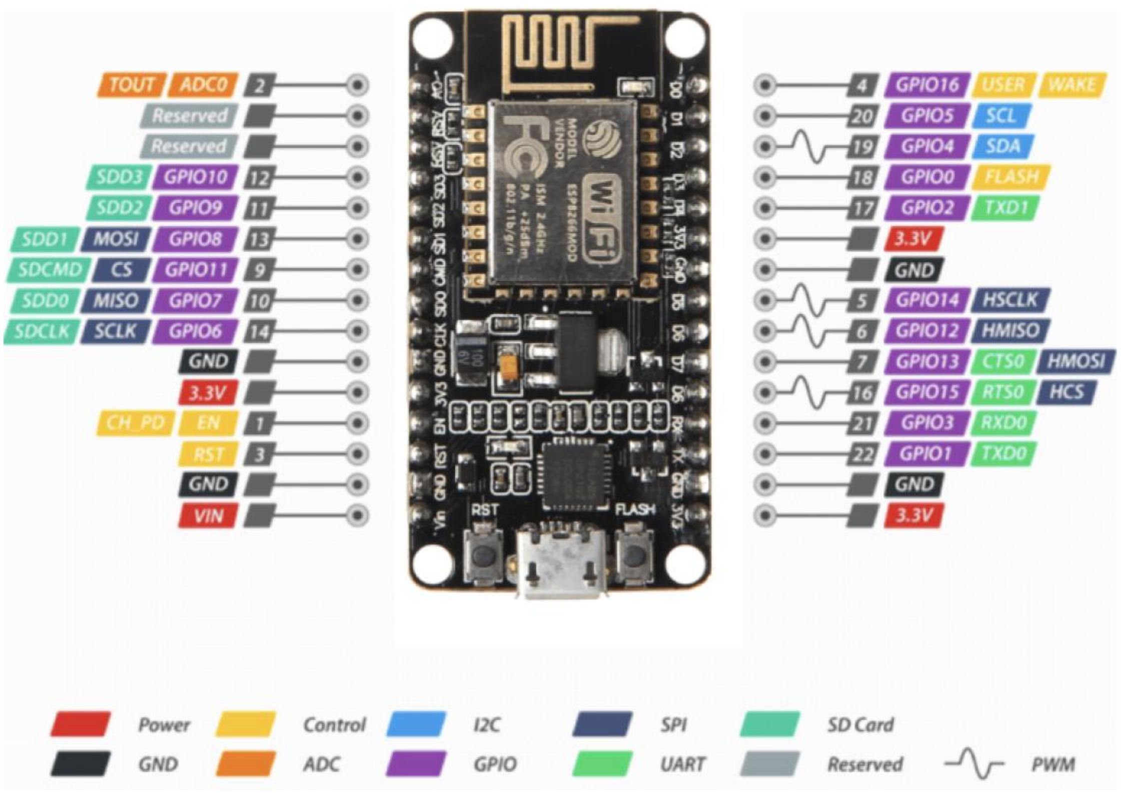
[10] “About us | Blynk,” *blynk.io*. https://blynk.io/about (accessed Feb. 29, 2024).

Appendix A: ESP8266 NodeMCU Technical Specifications



Model	ESP8266-12E
Wireless Standard	802.11 b/g/n
Frequency range	2.4 GHz - 2.5 GHz (2400M-2483.5M)
Wi-Fi mode	Station / SoftAP / SoftAP+station
Stack	Integrated TCP/IP
Output power	19.5dBm in 802.11b mode
Data interface	UART / HSPI / I2C / I2S / Ir
Remote Control	GPIO / PWM
Supports protection mode	WPA / WPA2
Encryption	WEP / TKIP / AES
Power supply	VIN: 4.5 VDC - 9 VDC (VIN) Or via 5V micro-USB connector
Consumption	Continuous Wi-Fi transmission: 70 mA Standby: < 200µA
Operating temperature	From -40°C to +125°C
Dimensions (mm):	58 × 31.20 × 13
Weight	10 g
Flash Memory	4 Mb
SRAM	64 kb
Clock Speed	80 MHz

Appendix B: ESP8266 NodeMCU Pinout Configuration

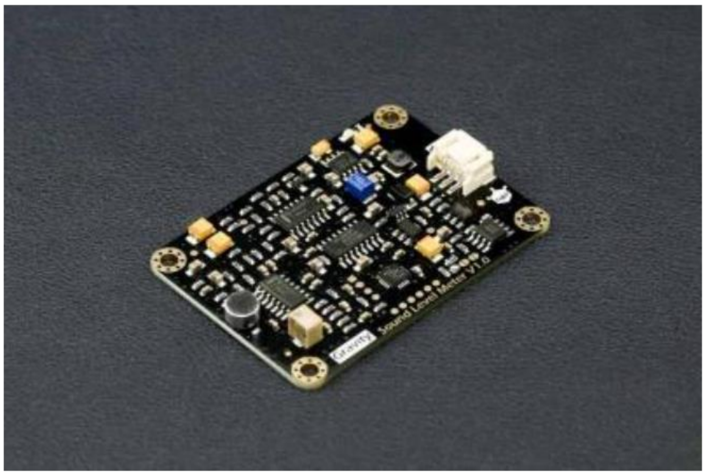


Appendix C: SEN0232 Analog Sound Level Meter Datasheet

DFROBOT
DRIVE THE FUTURE

Gravity: Analog Sound Level Meter SKU:SEN0232

Introduction



Analog Sound Level Meter SKU:SEN0232

In our environment, there are all kinds of sounds, some of which are noise. With the development of human civilization, the quiet environments are less and less, but more and more noisy environments instead. Staying in the noise for long time will have an impact on hearing, which is bad for health.

Sound level meter (also known as the decibel meter, noise meter) is a basic noise measurement instrument.We have launched a sound level meter, which is compatible with Arduino, plug-and-play . It can accurately measure the sound level of the surrounding environment. This product uses instrument circuit, low noise microphone, which makes it highly precious. It supports 3.3~5.0V wide input voltage, 0.6~2.6V voltage output. The decibel value is linear with the output voltage, which leads to a simple conversion, without complex algorithm.The connector is plug-and-play, without welding, so this product can be easily used in your application.

Sound level meter is widely used in environmental noise detection, such as highway noise monitoring station, room noise monitoring and so on. It's time for you to DIY a sound level detector to protect your hearing.

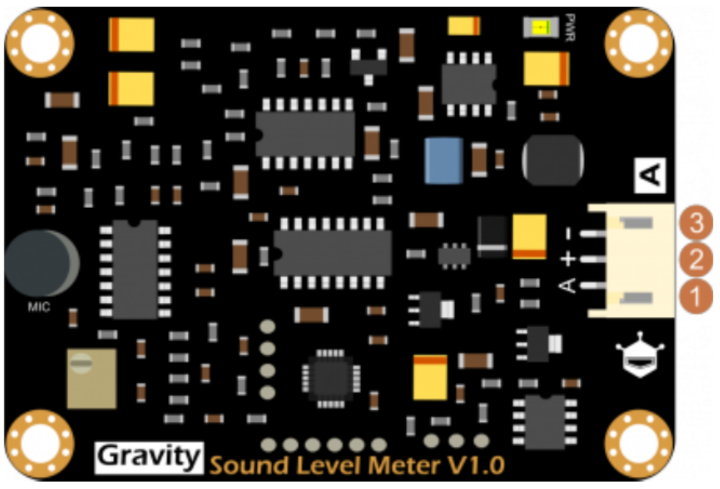
The black film on the microphone is a sensitive component, do not touch it with fingernails or other sharp objects.
Do not place this module on the surface of the conductor or the semiconductor, otherwise it will short the microphone's pins. It is recommended to place this module on a dry insulator's surface or secure with nylon columns to hang in the air.

Specification

- Measuring Range: 30dBA ~ 130dBA
- Measurement Error: ±1.5dB
- Frequency Weighted: A Weighted
- Frequency Response: 31.5Hz ~ 8.5KHz
- Time Characteristics: 125ms
- Input Voltage: 3.3 ~ 5.0V
- Input Current: 22mA@3.3V, 14mA@5.0V
- Output Voltage: 0.6 ~ 2.6V
- Module Size: 60mm * 43mm

Board Overview

	Num	Label	Description
Sound Level Meter	1	A	Analog Signal Output(0.6~2.6V)
	2	+	Power VCC(3.3~5.0V)
	3	-	Power GND(0V)

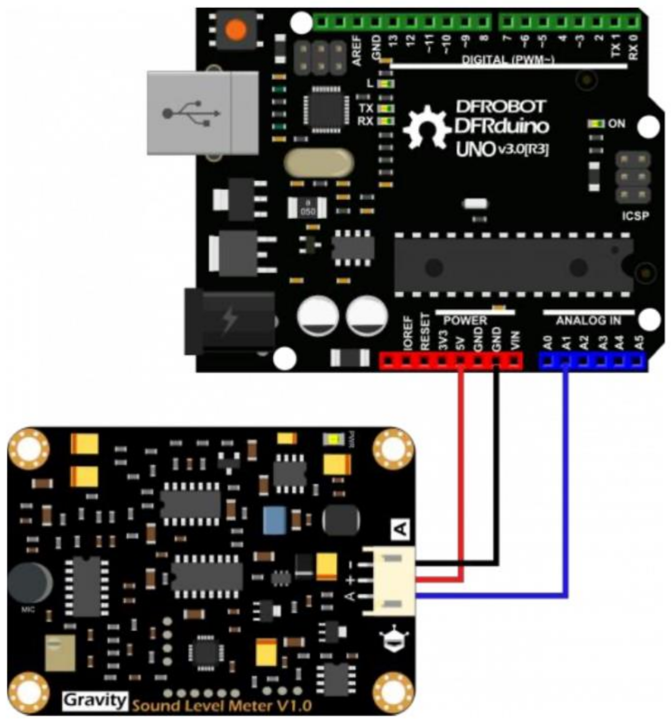


Requirements

- Hardware
  DFRduino UNO (or similar) x 1
  Sound Level Meter x1
  Gravity 3P Cable (or dupont Line) x 1

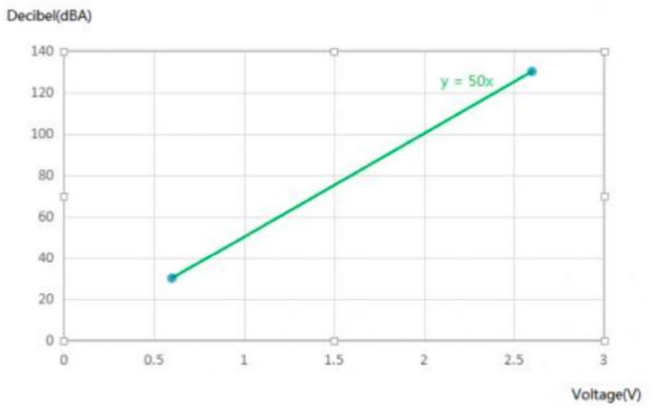
- Software
  Arduino IDE (Version requirements: V1.0.x or V1.8.x), **Click to Download Arduino IDE from Arduino®**

Connection Diagram

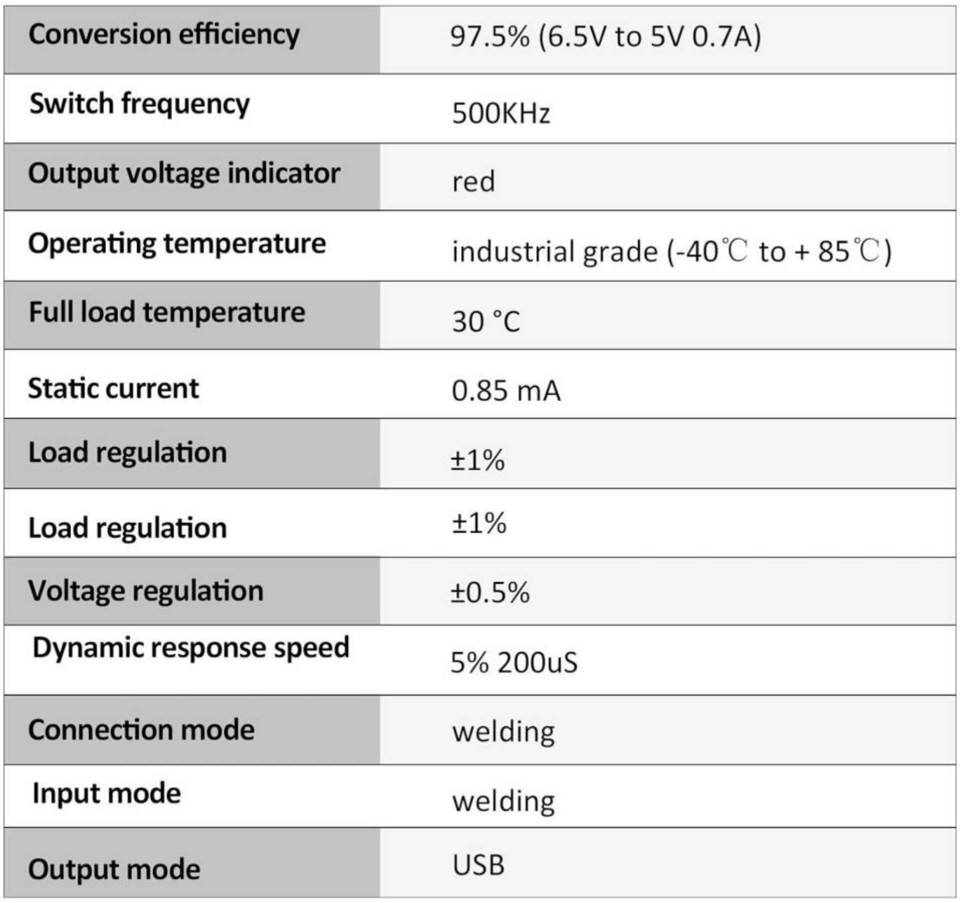


Relation between Decibel Value and Voltage Output

For this product,the decibel value is linear with the output voltage.When the output voltage is 0.6V, the decibel value should be 30dBA. When the output voltage is 2.6V, the decibel value should be 130dBA.
The calibration is done before leaving the factory, so you don't need to calibrate it.
So we can get this relation: Decibel Value(dBA) = Output Voltage(V) × 50, as shown below.

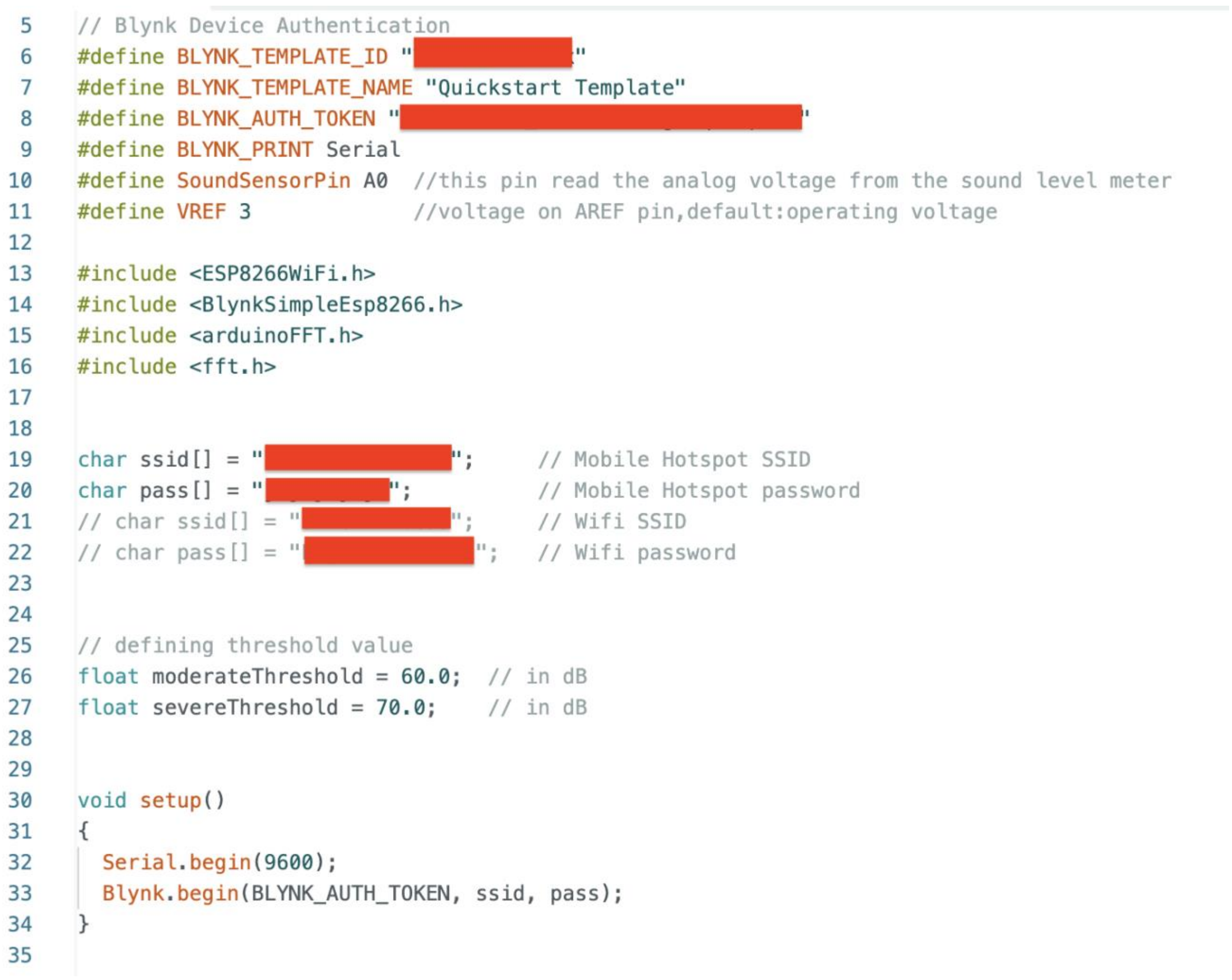


Appendix D: DC-DC Voltage Regulator Module Specifications

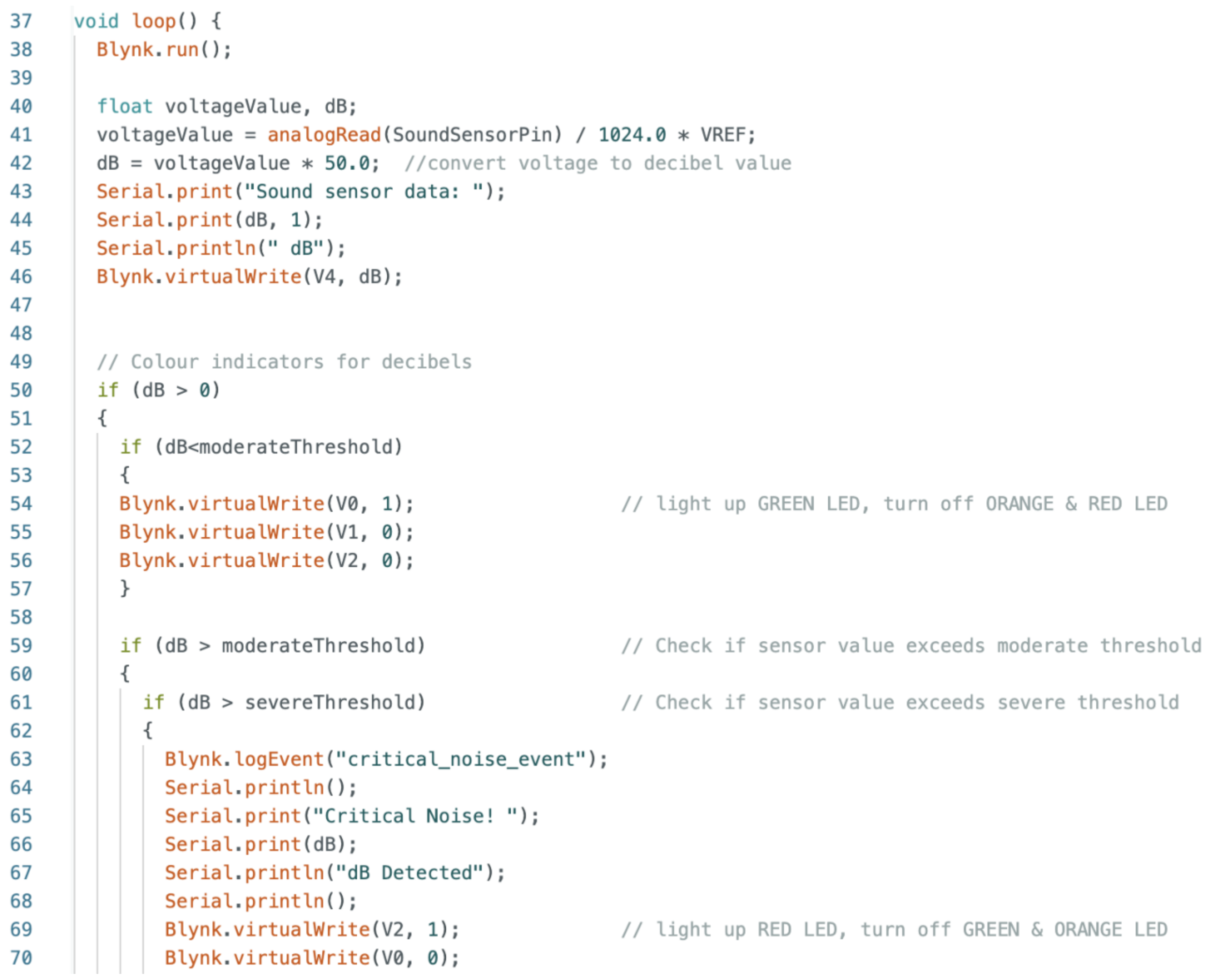


Conversion efficiency	97.5% (6.5V to 5V 0.7A)
Switch frequency	500KHz
Output voltage indicator	red
Operating temperature	industrial grade (-40℃ to + 85℃)
Full load temperature	30 °C
Static current	0.85 mA
Load regulation	±1%
Load regulation	±1%
Voltage regulation	±0.5%
Dynamic response speed	5% 200uS
Connection mode	welding
Input mode	welding
Output mode	USB

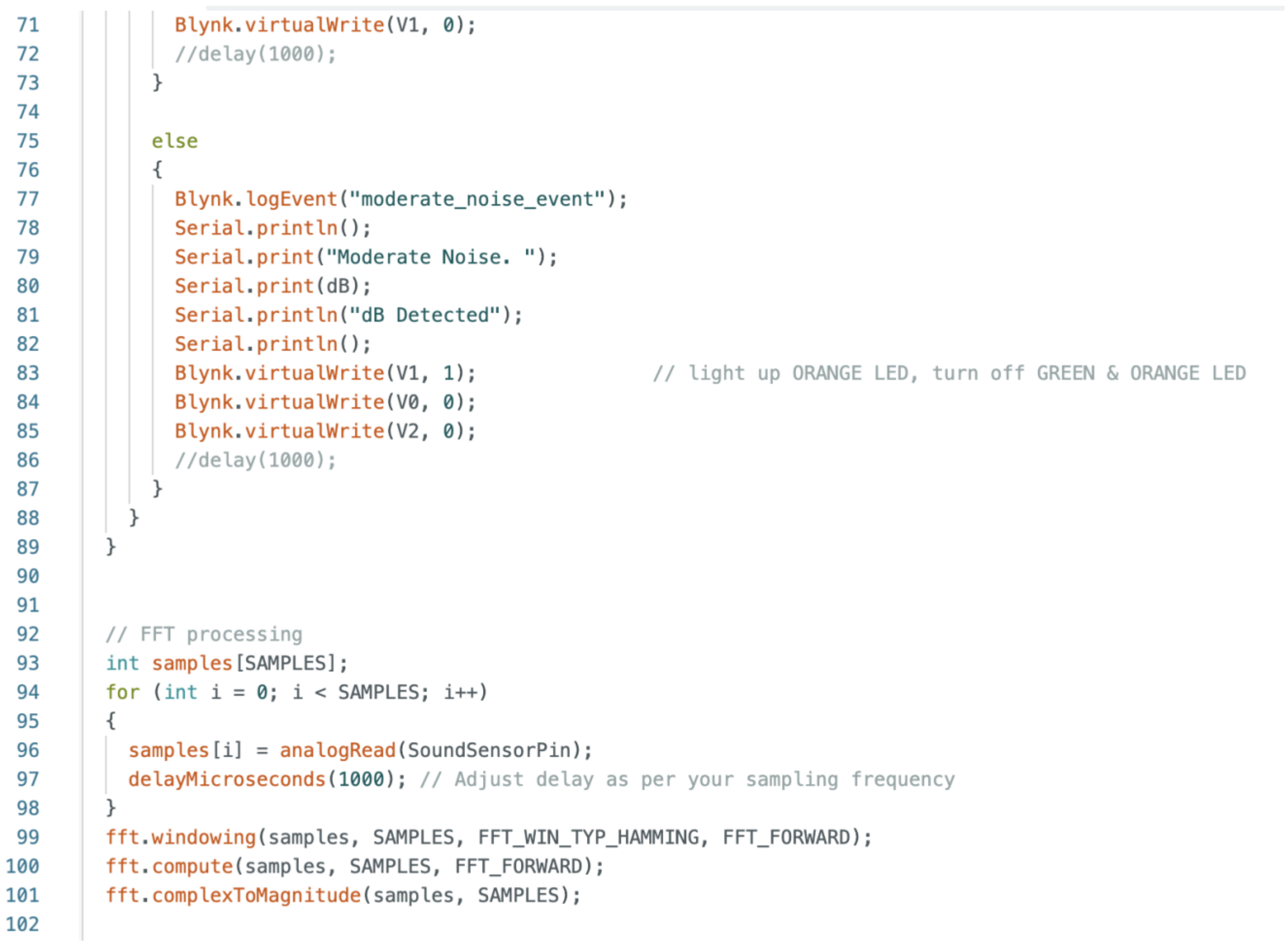
Appendix E: Program Code



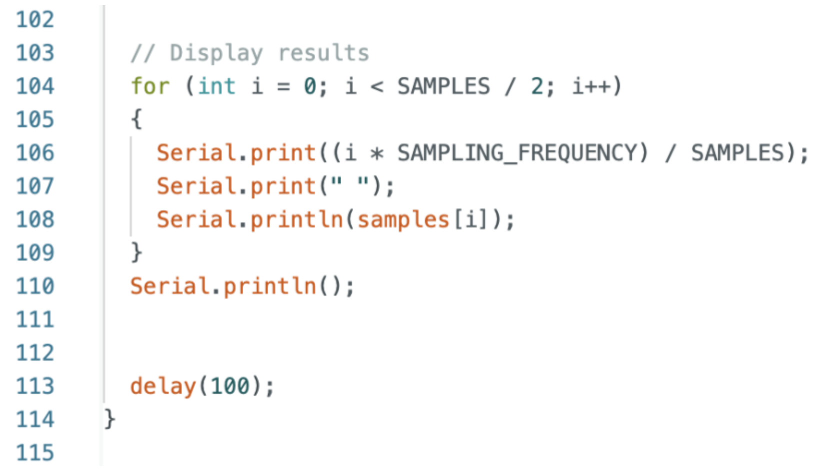
```
// Blynk Device Authentication
#define BLYNK_TEMPLATE_ID "[redacted]"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "[redacted]"
#define BLYNK_PRINT Serial
#define SoundSensorPin A0  //this pin read the analog voltage from the sound level meter
#define VREF 3             //voltage on AREF pin,default:operating voltage

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <arduinoFFT.h>
#include <fft.h>


char ssid[] = "[redacted]";      // Mobile Hotspot SSID
char pass[] = "[redacted]";          // Mobile Hotspot password
// char ssid[] = "[redacted]";     // Wifi SSID
// char pass[] = "[redacted]";     // Wifi password


// defining threshold value
float moderateThreshold = 60.0;  // in dB
float severeThreshold = 70.0;    // in dB


void setup()
{
  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}

```



```
void loop() {
  Blynk.run();

  float voltageValue, dB;
  voltageValue = analogRead(SoundSensorPin) / 1024.0 * VREF;
  dB = voltageValue * 50.0;  //convert voltage to decibel value
  Serial.print("Sound sensor data: ");
  Serial.print(dB, 1);
  Serial.println(" dB");
  Blynk.virtualWrite(V4, dB);


  // Colour indicators for decibels
  if (dB > 0)
  {
    if (dB<moderateThreshold)
    {
    Blynk.virtualWrite(V0, 1);                  // light up GREEN LED, turn off ORANGE & RED LED
    Blynk.virtualWrite(V1, 0);
    Blynk.virtualWrite(V2, 0);
    }

    if (dB > moderateThreshold)                 // Check if sensor value exceeds moderate threshold
    {
      if (dB > severeThreshold)                 // Check if sensor value exceeds severe threshold
      {
        Blynk.logEvent("critical_noise_event");
        Serial.println();
        Serial.print("Critical Noise! ");
        Serial.print(dB);
        Serial.println("dB Detected");
        Serial.println();
        Blynk.virtualWrite(V2, 1);              // light up RED LED, turn off GREEN & ORANGE LED
        Blynk.virtualWrite(V0, 0);
```



```
        Blynk.virtualWrite(V1, 0);
        //delay(1000);
      }

      else
      {
        Blynk.logEvent("moderate_noise_event");
        Serial.println();
        Serial.print("Moderate Noise. ");
        Serial.print(dB);
        Serial.println("dB Detected");
        Serial.println();
        Blynk.virtualWrite(V1, 1);              // light up ORANGE LED, turn off GREEN & ORANGE LED
        Blynk.virtualWrite(V0, 0);
        Blynk.virtualWrite(V2, 0);
        //delay(1000);
      }
    }
  }


  // FFT processing
  int samples[SAMPLES];
  for (int i = 0; i < SAMPLES; i++)
  {
    samples[i] = analogRead(SoundSensorPin);
    delayMicroseconds(1000); // Adjust delay as per your sampling frequency
  }
  fft.windowing(samples, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  fft.compute(samples, SAMPLES, FFT_FORWARD);
  fft.complexToMagnitude(samples, SAMPLES);

```



```

  // Display results
  for (int i = 0; i < SAMPLES / 2; i++)
  {
    Serial.print((i * SAMPLING_FREQUENCY) / SAMPLES);
    Serial.print(" ");
    Serial.println(samples[i]);
  }
  Serial.println();


  delay(100);
}

```