DEVELOPMENT OF PORTABLE VEHICLE NOISE PRESSURE LEVEL MONITORING SYSTEM

AHMAD NAZRIL HASAN BIN ZAINOL

BACHELOR OF MECHANICAL ENGINEERING TECHNOLOGY (BMMV) WITH HONOURS

2023

Faculty of Mechanical and Manufacturing Engineering Technology

DEVELOPMENT OF PORTABLE VEHICLE NOISE PRESSURE LEVEL MONITORING SYSTEM

Ahmad Nazril Hasan Bin Zainol

Bachelor of Mechanical Engineering Technology (BMMV) with Honours

2023

DEVELOPMENT OF PORTABLE VEHICLE NOISE PRESSURE LEVEL MONITORING SYSTEM

AHMAD NAZRIL HASAN BIN ZAINOL

A project submitted
in fulfilment of the requirements for the degree of
Bachelor of Mechanical Engineering Technology (BMMV) with Honours

Faculty of Mechanical and Manufacturing Engineering Technology

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I declare that this report entitled “DEVELOPMENT OF PORTABLE VEHICLE NOISE PRESSURE LEVEL MONITORING SYSTEM” is the result of my own research except as cited in the references. The Choose an item. has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	anas
Name	:	AHMAD NAZRIL HASAN BIN ZAINOL
Date	:	10/01/2023

APPROVAL

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the Bachelor of Mechanical Engineering Technology (BMMV) with Honours.

Signature	:	[illegible]
Supervisor Name	:	TS. DR. NOR AZAZI BIN NGATIMAN
Date	:	10/01/2023

DEDICATION

I dedicate my dissertation work with a unique feeling of gratitude to my family and my friends, especially my parents, Zainol bin Khalid and Rohana bt Yusof. Thank you for love and support. Also give me some space to complete this project. Special thanks to my supervisor, TS. DR. Nor Azazi Bin Ngatiman, with his tremendous support and patiently given me guidance and teachings. Their words encourage and push for more persistence. I also dedicate this dissertation to my many friends who supported me throughout the process. I will consistently acknowledge all they have done. I am very thankful for these opportunities.

ABSTRACT

Noise pollution can affect of loss hearing. Environment noise sum of all noises present like motor vehicle, airplanes, trains and industrial. This project explains a Portable Vehicle Noise Pressure Level Monitoring System that can be monitoring by wireless system and connect to Blynk application which can be downloaded from smartphone. Components can get from any store electronic for work tool and education purpose. Micro-controller with wireless system can easily do prototype by offers a simple one but powerful design be used everywhere and anytime. It would be convenient for beginners for develop for educational purposes since open source was provided. This project provides an overview of micro-controller type, noise monitoring sensors, wireless system implementation using WIFI, hardware used for the development of this project as well as hardware and software implementation. This project will generally explain the background of micro-controller and the wireless system and comparing this project to normal product without wireless system. With few components selected that suitable for students, students were able to learn and create this project, Microphone Sound Detector and Ultrasonic Sensor. These components were selected as the main components with micro-controller had bigger memory and lots of header compared with another micro-controller in the market. With bigger memory and many headers, we can simply use any pins number in the micro-controller according to the Microphone Sound Detector and Ultrasonic Sensor coding. This report also show the efficiency between this project value and sound level meter value. It can be acceptable cause the error percentage below than 10%. Not only focus to vehicle, it also can measure the surrounding environment noise use this project. User can alert about noise pollution by monitoring thru this project. For recommendation, student can also adding another sensor like air quality sensor for monitoring air surrounding to prevent air pollution.

ABSTRAK

Pencemaran bunyi boleh menjejaskan kehilangan pendengaran. Bunyi persekitaran jumlah semua bunyi yang hadir seperti kenderaan bermotor, kapal terbang, kereta api dan industri. Projek ini menerangkan Sistem Pemantauan Tahap Tekanan Bunyi Kenderaan Mudah Alih yang boleh dipantau melalui sistem wayarles dan menyambung ke aplikasi Blynk yang boleh dimuat turun dari telefon pintar. Komponen boleh didapati dari mana-mana kedai elektronik untuk alat kerja dan tujuan pendidikan. Pengawal mikro dengan sistem wayarles boleh membuat prototaip dengan mudah dengan menawarkan reka bentuk yang ringkas tetapi berkuasa untuk digunakan di mana-mana dan pada bila-bila masa. Ia adalah mudah untuk pemula untuk membangunkan untuk tujuan pendidikan kerana sumber terbuka disediakan. Projek ini menyediakan gambaran keseluruhan jenis pengawal mikro, penderia pemantauan hingar, pelaksanaan sistem wayarles menggunakan WIFI, perkakasan yang digunakan untuk pembangunan projek ini serta pelaksanaan perkakasan dan perisian. Projek ini secara amnya akan menerangkan latar belakang pengawal mikro dan sistem tanpa wayar dan membandingkan projek ini dengan produk biasa tanpa sistem wayarles. Dengan beberapa komponen terpilih yang sesuai untuk pelajar, pelajar dapat mempelajari dan mencipta projek ini, Pengesan Bunyi Mikrofon dan Penderia Ultrasonik. Komponen ini dipilih kerana komponen utama dengan pengawal mikro mempunyai memori yang lebih besar dan banyak pengepala berbanding dengan pengawal mikro lain di pasaran. Dengan memori yang lebih besar dan banyak pengepala, kita boleh menggunakan sebarang nombor pin dalam pengawal mikro mengikut pengekodan Pengesan Bunyi Mikrofon dan Pengekod Ultrasonik. Laporan ini juga menunjukkan kecekapan antara nilai projek ini dan nilai meter aras bunyi. Ia boleh diterima kerana peratusan ralat di bawah 10%. Bukan sahaja fokus kepada kenderaan, ia juga boleh mengukur bunyi persekitaran dengan menggunakan projek ini. Pengguna boleh memaklumkan tentang pencemaran bunyi dengan memantau melalui projek ini. Untuk cadangan, pelajar juga boleh menambah penderia lain seperti penderia kualiti udara untuk memantau udara sekeliling bagi mengelakkan pencemaran udara.

ACKNOWLEDGEMENTS

First praise is to Allah, the Almighty, on whom ultimately, we depend for sustenance and guidance. Secondly, my sincere appreciation goes to my supervisor TS. DR. Nor Azazi Bin Ngatiman, whose guidance, careful reading and constructive comments were valuable. His timely and efficient contribution helped me shape this into its final form and I express my sincerest appreciation for his assistance in any way that I may have asked. Without his guidance, this project may not be completed within the time given.

Next, I am also indebted to my colleagues. Without help from them, I may have not accomplished this project even from the start. Special thanks, tribute and appreciation to all those their names do not appear here who have contributed to the successful completion of this study. Finally, I'm forever indebted to my parents who, understand the importance of this work and even help to support the components that I needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

V	-	Voltage
SD	-	Secure Digital
USB	-	Universal Serial Bus
WiFi	-	Wireless Fidelity
IoT	-	Internet of Things
IDE	-	Integrated Development Environment
etc	-	et cetera
dB	-	decibel
RPM	-	Revolutions per minute
I/O	-	input/output
UARTs	-	Universal Asynchronous Reception and Transmission
MHz	-	megahertz
RST	-	Reset
AC	-	Alternating Current
DC	-	Direct Current
3D	-	three-dimensional
Vin	-	Voltage in
RAM	-	Random Access Memory
GPIO	-	General-Purpose Input/Output
PWM	-	Pulse Width Modulation
ICSP	-	In Circuit Serial Programming
VCC	-	Voltage Common Collector
TRIG	-	Trigonometry
USART	-	Universal Synchronous/Asynchronous Receiver/Transmitter
DIY	-	Do It Yourself
CPU	-	Central Processing Unit

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Commonly, noise is defined as unwanted sound, independent of loudness, that can produce a desired physiological or psychological effect in an individual and can interfere with the social goals of an individual or group. In industrial or densely populated areas, boundary and environmental noise will be a problem, and the sources of noise and their negative effects on our well-being can vary.

The Eurofins network of companies in Malaysia is able to perform noise measurements and monitor environmental noise over extended periods of time in order to measure and assess risk, as well as provide custom-tailored solutions to reduce exposure and comply with noise regulations. One of the most common types of noise monitoring, boundary noise monitoring is requested for regulatory, environmental, and measurement of noise exposure to neighboring properties.

The information gathered from noise level monitoring and testing enables us to recognize patterns and take steps to reduce noise pollution. Noise pollution is Low or High-frequency sound that is detrimental to human life. It can be caused by a variety of industrial machines, motor vehicles, and ships, etc. Noise Pollution Monitoring is a component of Environmental Monitoring & Testing, given that noise pollution has also increased exponentially in recent years.

1.2 Problem Statement

In this day and age of the Internet of Things, it is typical to be required to construct a project based on the loT. As a result, time was saved while still requiring them to complete a wireless system assignment. Because this project is based on a wireless system, students may monitor the sound pressure level using the Blynk program. As a result, they may repurpose this initiative for a future attempt.

Next, this project was made to monitor the presence of sound pressure level and analyzing the efficiency. With a cost-effective project, students can place this project placed in small or narrow place or confined space but still managed to take data with a suitable budget. Moreover, students managed to capture real time data and can be measured in long period of time as later can predict the future just by analyzing the data.(de Botton et al., 2000)

Meanwhile, because this project utilized an ESP32 as the input/output for the Microphone Sound Detector and Ultrasonic Sensor. ESP32 as the micro-controller, students can learn more about the loT and conduct additional research. Students can improve their abilities even further by understanding the programming code, the operation of sound pressure level monitoring, and how the Microphone Sound

Detector and Ultrasonic Sensor work. And finally, all these components can link to Blynk apps through the use of a wireless system.

1.3 Research Objective

An objective is defined as the purposes or target that will be obtained after finishing the project:

1) To create a sound module monitoring project using ESP32, Microphone Sound Detector and Ultrasonic Sensor.
2) To develop software using IDE and take the data from the Blynk application through smartphone.
3) To analyze the efficiency of project and monitoring the sounds pressure of exhaust motorcycle.

1.4 Scope of Research

The goal of this project is to create a sound pressure level monitoring system that makes use of ESP32 as the monitoring platform, a microphone sound detector and an ultrasonic sensor. The Arduino IDE is the software used in this project.

The workflow would be designing a microphone sound sensor to receive the input/output from the motorcycle's engine. Ultrasonic Sensor for determining object length. The inputs from microphone sound sensor and ultrasonic sensor send the data’s to Blynk apps by connect WiFi from ESP32. The ESP32 will be the monitoring platform as it receives both input/output from the microphone sound sensor. The data can be read from both Blynk apps and IDE serial monitor depends on circumstances. This project was separated into three phases which were software development, programming code and hardware fabrication system. Following the completion of these phases, the sound pressure level data from the exhaust motorcycles and the efficiency graph can be used to anticipate future development. Thus, this project can focus on sound pressure level analysis and the graph's efficiency such as the needs of a preventive noise pollution, and etc.

CHAPTER 2

LITERATURE REVIEW

2.1 Noise

2.1.1 Introduction

Noisy is any undesired sound *it* is disagreeable, loud, or disturbing to the ear. Since both noise and acceptable sound are vibrations across a medium such as air or water, there is no physics-based separation between the two. The distinction happens when the brain hears and processes a sound.

Acoustic noise is any intentional (music or speech) or accidental acoustic sound. Noise in electronics, on the other hand, may not be discernible with the human ear and may need the use of specialized equipment to identify. In engineering, noise may refer to an undesirable residual electronic noise signal that generates acoustic noise that is perceived as a hiss. Typically, signal noise is assessed by A-weighting. Noise is a term used in experimental sciences to describe any random fluctuations in data that interfere with signal comprehension.

2.1.2 Environmental Noise

Environment noise is the sum of all noises present in a given environment like surface of motor vehicles, aero planes, trains, and industrial sources are the primary sources of environmental noise.

This source of noise expose millions of people to noise pollution, which not only causes pain and it also has major health consequences, including an increased incidence of hearing loss, cardiovascular disease, and other illnesses. Although urban noise does not often result in hearing loss, it does interfere with sleep, talking, and other human activities.

Among the possible mitigation measures and controls for reducing sound levels include source intensity reduction, land-use planning strategies, noise barriers and sound baffles, time of day use regimens, vehicle operation controls, and building acoustics design elements.

2.1.3 Noise Pollution

Every day, millions of individuals are affected by noise pollution. Noise-induced hearing loss is the most prevalent health issue caused by noise (NIHL). In addition to hypertension, heart disease, sleep disturbances, and stress, loud noise may also cause hypertension and heart disease. These health issues may affect people of all ages, but children in particular. Numerous youngsters living near noisy airports or motorways have been discovered to suffer from stress and other difficulties, such as memory, concentration, and reading skill problems.

The health and well-being of animals is impacted by noise pollution. According to research, loud noises induce caterpillars' dorsal arteries (the equivalent of an insect's heart) to beat quicker and bluebirds to produce fewer young. Animals utilize sound for a number of purposes, including navigating, locating food, attracting mates, and avoiding predators. Noise pollution impedes their ability to do these activities, threatening their survival.

Not only does noise pollution damage terrestrial animals, but it is also becoming a major worry for aquatic species. Ships, oil drilling, sonar equipment, and seismic testing have made the previously tranquil sea environment noisy and chaotic. Whales and dolphins are especially susceptible to the effects of noise pollution. The ability of these aquatic animals to communicate, migrate, eat, and locate mates relies on echolocation, which is hindered by noise. (Singh, N., 2004).

2.1.4 Measurement

The loudness of sound is measured in decibels (dB). A whispering is roughly 30 dB, whereas a regular conversation is approximately 60 dB, and a motorcycle engine is roughly 95 dB. Long-term exposure in noise levels more than 70 decibels (dB) may result in hearing loss. Sounds over 120 decibels (dB) may cause immediate ear damage. The graph below demonstrates decibel levels and the potential impact of ordinary noise on hearing. (National Center For Environmental Health, 2019)

Table 2.1 dB Levels And Noise From Everyday Sources

Sounds and Noises	Average Sound Level (dB)	Response
The softest sound	0	These decibel levels normally do not cause hearing harm.
Ordinary breath	10	
Leaves rustling	20	
Whisper	30	
Quiet library	40	
Regular conversation	60	
Vacuums	70	You may notice the sounds annoying.
Traffic and alarm clock	80–85	You can feel really irritated.
Blenders and hairdryer	80-85	Possible hearing loss after two hours of exposure.
Motorcycle	95	Hearing impairment possible after around 50 minutes of exposure.
Concerts and car horns	100	Possible hearing loss after 15 minutes.
Mp3 player (full volume)	105–110	Possible hearing loss in less than 5 minutes.
Sports events	110	Possible hearing loss in less than 2 minutes.
Jet planes	120	Pain and ear damage.
Gun shots and fireworks	140–150	Pain and ear damage.

2.2 ESP32

The ESP32 System On a Chip (SoC) is a highly adaptable SoC that may be used as a general-purpose microcontroller with a huge assortment of peripherals, including WiFi and Bluetooth wireless capabilities. Espressif Systems located in Shanghai, makes it only costs around $5. Even though the ESP32 is a SoC, most users won't begin by utilising the device by itself. While employing the ESP32 SoC to develop a product is technically conceivable, it is not a typical strategy. Instead, the majority of ESP32-based designs employ pre-made modules that include a genuine ESP-32 SoC, external flash memory, a crystal, and a pre-tuned PCB antenna or an IPEX antenna connection. The entire structure is then set down beneath a protected container. The creator of this module is Espressif. The fact that Espressif has pre-loaded the low-level device drivers, the wireless protocol stacks for WiFi b, g, and n, Bluetooth and BLE, as well as FreeRTOS as the base OS, is a significant benefit of utilising this module rather than starting from scratch. To make downloading user programs relatively simple, a bootloader has also been installed.

2.3 ESP32 Types

For experimenters, there are a lot of ESP32 modules available. The experiments here can employ just about any of them. Programming will be made simpler by the inbuilt micro-USB port found on many of these boards. For programming, certain boards are missing this capability and need an additional FTDI adaptor. There are also ESP32 compatible boards that should really know about. After comparing and doing research this topic. When compared to other microcontrollers, the ESP32 was picked for its usefulness, specs, features, and applications. This section compared the ESP32 and ESP8266 to show why the ESP32 was chosen.

2.3.1 ESP32

Figure 2.1 ESP32

The bare ESP32 chip is referred to as ESP32. However, ESP32 development boards are also referred to as "ESP32" devices. It is difficult and impractical to use ESP32 bare chips, especially while learning, testing, and prototyping. Want to utilise an ESP32 development board the most of the time.

These development boards have all the circuitry required to programmer and power the chip, link it to your computer, connect peripherals, have built-in power and control LEDs, a wi-fi antenna, and other beneficial features. Others even include additional hardware, such as displays, specialized sensors or modules, or, in the case of the ESP32-CAM, a camera.

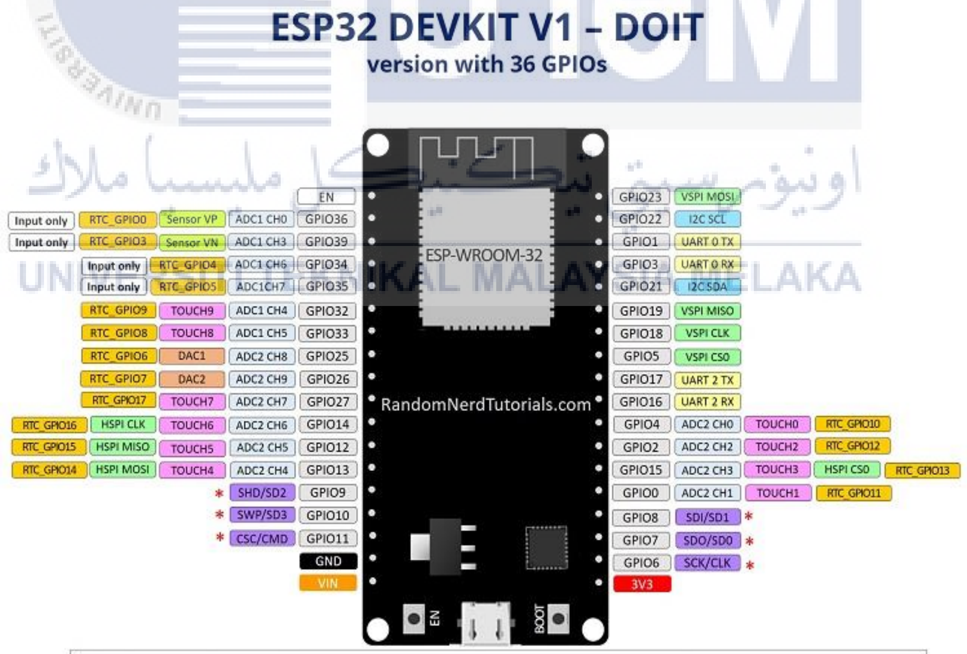
A new CPU core, faster Wi-Fi, additional GPIOs, compatibility for Bluetooth 4.2, and Bluetooth low energy are all added by the ESP32. The ESP32 also has a built-in hall effect sensor and touch-sensitive pins that may be used to bring it up from a deep slumber. The 48 pins on the ESP32 chip may be used for a variety of purposes. Not all ESP32 development boards expose every pin, and certain pins shouldn't be utilised. There are 36 accessible GPIOs on the ESP32 DEVKIT V1 DOIT board that may be used to connect peripherals.

Power Pins, all boards typically have three power pins: 3V3, GND, and VIN. If the USB port is not used to supply power, it can utilise these pins to power the board or other devices (if powering the board using the USB port).

Almost all General-Purpose Input Output Pins (GPIOs) have been given a number, therefore it is proper to refer to them by that number.

Which pins on the ESP32 are used for UART, I2C, or SPI may be chosen. Simply set it in the code. The ESP32 chip's multiplexing capability, which enables the assignment of many functions to a single pin, makes this feasible.

The pins will be set up by default as indicated in the image below if you wish to set them on the code (the pin location can change depending on the manufacturer). Additionally, certain pins have special characteristics that determine whether or not they are appropriate for a given project.


ESP32 DEVKIT V1 – DOIT
version with 36 GPIOs
ESP-WROOM-32
RandomNerdTutorials.com
EN
Input only RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
Input only RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
Input only RTC_GPIO4 ADC1 CH6 GPIO34
Input only RTC_GPIO5 ADC1 CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
* SHD/SD2 GPIO9
* SWP/SD3 GPIO10
* CSC/CMD GPIO11
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GPIO0 ADC2 CH1 TOUCH1 RTC_GPIO11
GPIO8 SDI/SD1 *
GPIO7 SDO/SD0 *
GPIO6 SCK/CLK *
3V3
EN
BOOT

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Figure 2.2 ESP32 Pin Configuration

Table 2.2 ESP32 Specifications

Number of cores	2 (dual core)
Wi-Fi	2.4 GHz up to 150 Mbits/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Clock frequency	Up to 240 MHz
RAM	512 KB
Pins	30, 36, or 38 (depending on the model)
Peripherals	Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.
Built-in buttons	RESET and BOOT buttons
Built-in LEDs	built-in blue LED connected to GPIO2; built-in red LED that shows the board is being powered
USB to UART bridge	CP2102

2.3.2 ESP8266

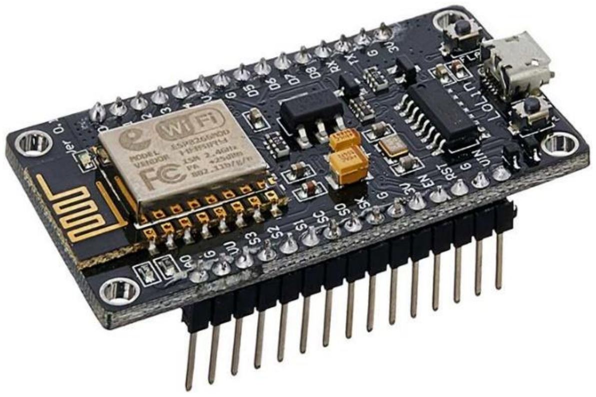


Figure 2.3 ESP8266

A 32-bit CPU with 16-bit instructions powers the ESP8266. Because of the Harvard design, instruction memory and data memory are entirely independent. The Read-Only Memory (ROM) of the ESP8266 contains a first stage boot loader and some library programmes. The remaining code has to be kept in an external Serial flash memory (provides only serial access to the data rather than addressing individual bytes, the user reads or writes large contiguous groups of bytes in the address space serially). The quantity of flash memory that is readily available might differ depending on your ESP8266.

The ESP8266 features a set of GPIO pins (General Purpose Input and Output pins), much like any other microcontroller, that may be used to "control" external sensors. Only 11 of the ESP8266's 17 GPIO pins may be utilised (among 17 pins, 6 are used for communication with the on-board flash memory chip). There is also an analogue input (to convert a voltage level into a digital value that can be stored and processed in the ESP8266). It also features WiFi connectivity, which may be used to link the ESP8266 to a WiFi network, access the internet, run a web server, enable smartphone connectivity, etc.

The ability to be programmed like any other microcontroller, notably any Arduino, is another benefit of an ESP8266.

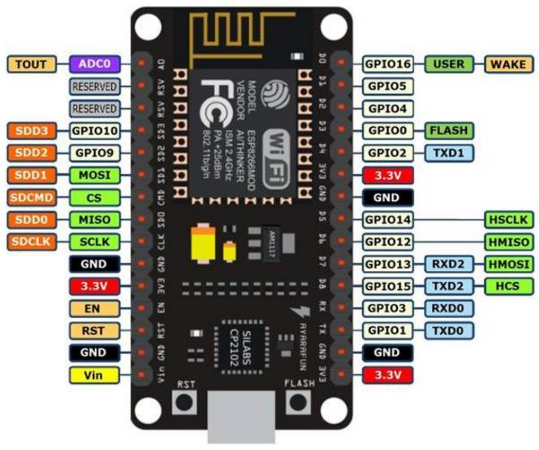


Figure 2.4 ESP8266 Pin Configuration

Table 2.3 ESP8266 Specifications

Microcontroller	ESP-8266 32-bit
NodeMCU Model	**Amica** (official)
Clock Speed	80-160 MHz
USB to Serial	**CP2102**
USB Connector	Micro USB
Operating Voltage	3.3V
Input Voltage	4.5V-10V
Flash Memory/SRAM	4 MB / 128 KB
GPIO Pins	17
Digital I/O Pins	11
Analog In Pins	1
PWM Pins	4
ADC Range	0-3.3V
UART/SPI/I2C	2 / 1 / 1
WiFi Built-In	802.11 b/g/n
Temperature Range	-40C – 125C

2.3.3 Comparison Specifications Table ESP32 And ESP8266

Table 2.4 Comparison Specifications Table ESP32 And ESP8266

ESP32	**SPECIFICATION**	**ESP8266**
Xtensa Dual-Core 32-bit LX6 with 600 DMIPS	**MCU**	Xtensa Single-core 32-bit L106
YES, HT40	**802.11 b/g/n Wi-Fi**	YES, HT20
12-bit	**ADC**	10-bit
1/16 channel	**Hardware/Software PWM**	None / 8 Channels
160MHz	**Typical Frequency**	80MHz
512kBytes	**SRAM**	160kBytes
36	**GPIO**	17
YES	**Touch Sensor**	NONE
Bluetooth 4.2	**Bluetooth**	none
4/2/2/3	**SPI/I2C/I2S/UART**	2/1/2/2
12-bit	**ADC**	10-bit
1	**CAN**	NONE
448kB of ROM for booting and core functions	**ROM**	No Programmable
-40°C to 125°C	**Working Temperature**	-40°C to 125°C

2.4 Hardware List and Functionality

1. ESP32 Dev Kit V1

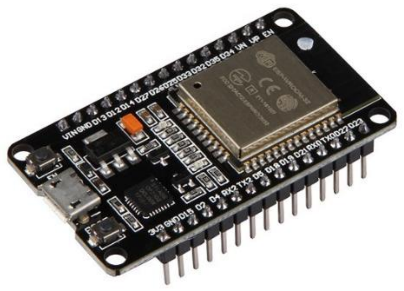


Figure 2.5 ESP32 Dev Kit V1

One of the development boards made to test the ESP-WROOM-32 module is the Esp32 Dev Kit v1. It is based on the ESP32 microprocessor, a single chip that supports WiFi, Bluetooth, Ethernet, and Low Power. Antenna and RF baluns, power amplifiers, low-noise amplifiers, filters, and a power management module are already included into the ESP32. The approach uses the least amount of space on the printed circuit board as a whole. This board uses TSMC 40nm low power 2.4 GHz dual-mode Wi-Fi and Bluetooth chips, which have the best power and RF attributes and are secure, dependable, and adaptable to a range of applications.

2. KY-037 Electret Microphone Audio Sensor Module

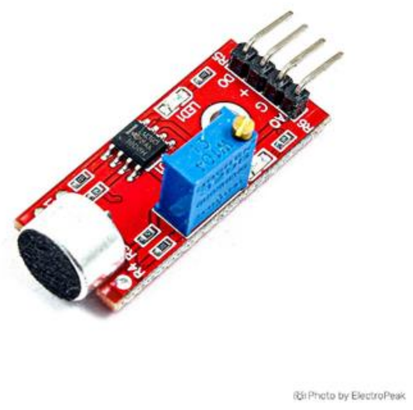


Figure 2.6 Microphone Audio Sensor Module

The audio detector included in the KY-037 Electret Microphone Audio Sensor Module package can detect analogue or digital sound with an adjustable audio level. This audio sensor board consists of three main sections: an electret condenser microphone (ECM) as an audio sensor to detect and measure the area's physical sounds and produce an analogue signal; an audio amplifier chip that receives the analogue signal from the ECM sensor and amplifies it; and a comparator that receives the amplified audio signal, compares it to a reference and adjusts the output level. The KY-037 is fitted with a multiturn potentiometer for adjusting the comparator's reference level, which allows the sensor sensitivity to be modified. When the ambient sound level exceeds a specified threshold, this sensor module's output level switches to High (reference level). This board is compatible with Arduino sensor boards and also compatible with ESP32 boards too.

3. **Ultrasonic Sensor HC-SR04**

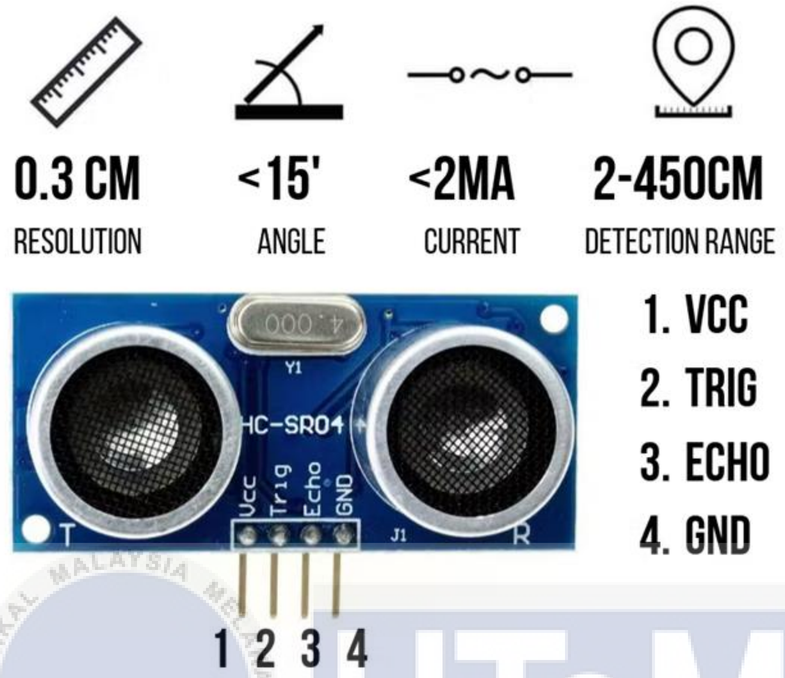


Figure 2.7 Ultrasonic Sensor HC-SR04

Ultrasonic Sensor HC-SR04 is a sensor for measuring distance. It emits a 40kHz ultrasonic signal that travels through the air and returns to the module if it finds an object or obstruction. You may calculate the distance by factoring in the travel time and the speed of sound.

The pins of the HC-setup SR04 are VCC (1), TRIG (2), ECHO (3), and GND (4). VCC's supply voltage is +5V, and the TRIG and ECHO pins may be connected to any Digital I/O on your Arduino board.

4. Power Supply (PowerBank)

Figure 2.8 Power Supply

The ESP32 may be fueled either by USB or an external power source. The power source is automatically chosen. A battery or an AC-to-DC converter (wall watt) may provide external (non-USB) power. Connecting the adapter requires inserting a micro-USB connector into the board's power port. The Gnd and Vin pin headers of the POWER connection are compatible with battery leads. The board can function with an external power source between 6 and 20 volts. However, if less than 7V is given, the 5V pin may provide less than five volts and the board may become unstable. If more than 12V is used, the voltage regulator may overheat and cause board damage. The suggested voltage range is 7 to 12 volts.

5. Bread Board

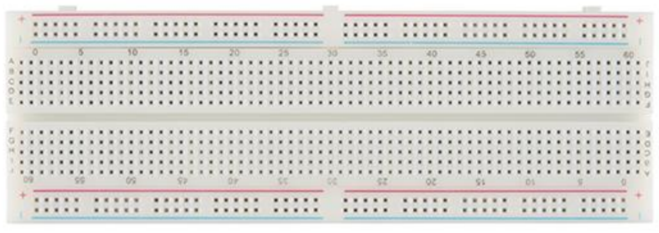


Figure 2.9 Bread Board

A breadboard or protoboard is a platform for prototyping electrical circuits. Originally, the term referred to a bread board, a piece of polished wood used for slicing bread. In the 1970s, the solderless breadboard (also known as a plugboard or terminal array board) was introduced, and the phrase "breadboard" is now often used to refer to them.

6. Jumper Wire

Figure 2.10 Jumper Wires

Wires are the connecting parts of a circuit. Jumper wires are small wire ducts that can be used to connect components to each other on breadboards or elsewhere. The male and female header of this wire provides easier connections without the need to solder.

7. Device (Smartphone or Tablet)

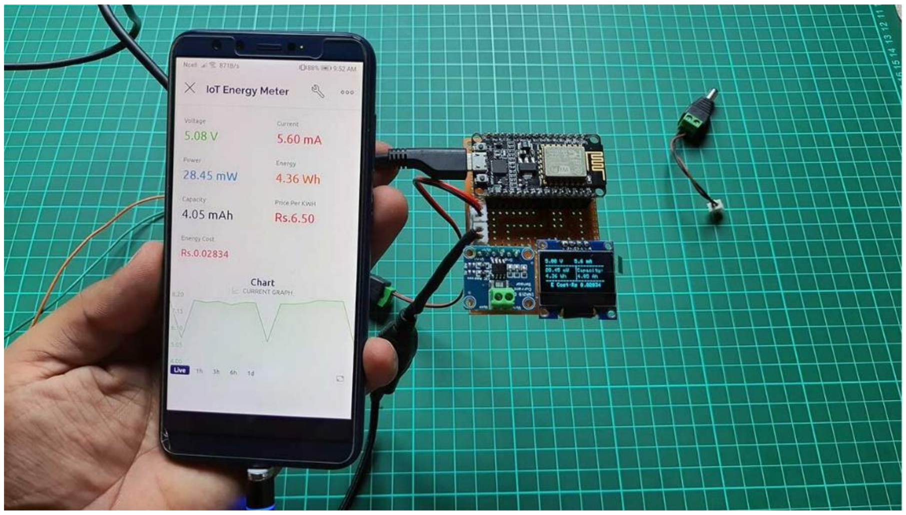


Figure 2.11 Device Connect To Micro-Controller

Besides a computer, smart IoT box gets along with smartphone. After the pairing, it sends notifications to it or displays graphs showing the data from micro-controller. It will connect them using the Blynk app.

CHAPTER 3

METHODOLOGY

3.1 Introduction

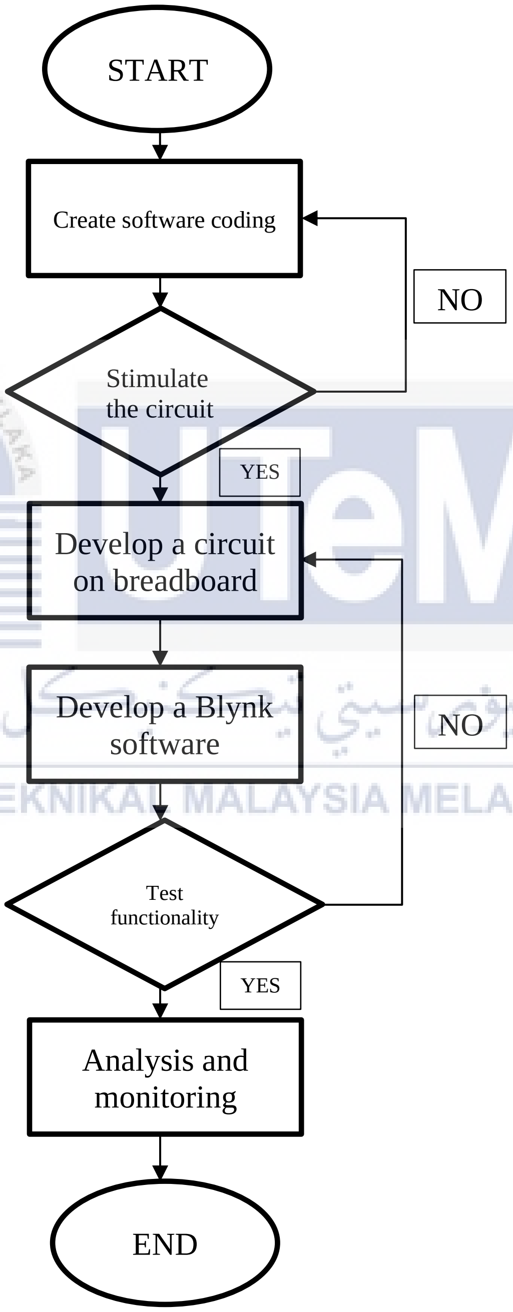


Figure 3.1 Project Planning

In this chapter, the design and methods can be explained to meet the objectives of this project which are to develop Blynk system to monitor the sound pressure level and of engine from few components by using the Arduino IDE programming, ESP32 board, KY-037 Electret Microphone Audio Sensor Module, and Ultrasonic Sensor HC-SR04.

This project was separated into three phases (Kondaveeti et al.. 2021): software development. programming code, and hardware fabrication system. The Arduino IDE software was utilized to develop the coding in the ESP32 for Phase 1. In order to build the software command, an assembly language from IDE was utilized to directly use it to specify instructions for the central processor unit (CPU).

In Phase II, the assembly language was downloaded to ESP32 utilizing the assembler for programming code. A software that translates assembly language into machine code transfers the essential commands and operations of the assembly code into a binary code which a particular processor can understand. In the second stage, it was then forwarded to PIC to interpret. After analysis is to identify and detect sound pressure level signal of component failures prior to loading to ESP32, input and output can be declared and launched in the Arduino IDE.

The hardware development is a production system in phase III. In addition, the current wireless project Arduino can be compared to the current model. The ESP32, breadboard, KY-037 Electret Microphone Audio Sensor Module, and Ultrasonic Sensor HC-SR04 were all components used for hardware building. This chapter discussed procedures from software development (assembly language) to hardware development, so this project may be electively completed. where solutions, ideas and designs can be defined.

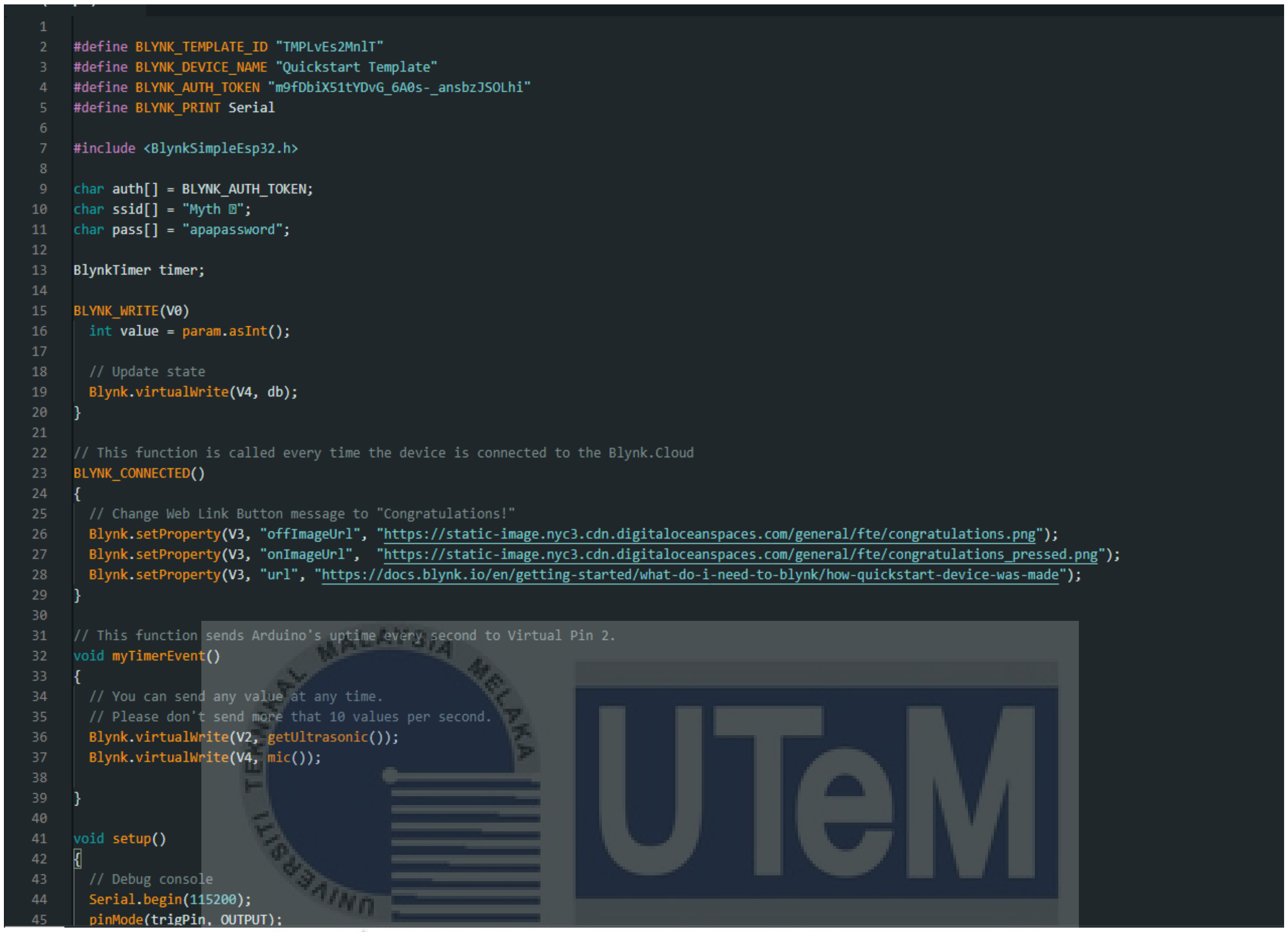
3.2 Arrangement of Circuit and Components

3.2.1 Programming Code

3.2.1.1 Setup Blynk

Blynk is an IOT platform useful for making projects connected to the Internet and accessible from a phone or mobile device. To monitor sound on smart phones, must create a project in the Blynk app and link it with our Arduino sketch. Blynk is a free to use app where the user can build software to control microcontrollers such as Raspberry Pi, Arduino, and ESP8266 NodeMCU, etc.

Firstly, create a new account on Blynk. Blynk Console is the easiest way to build a own IoT product with a mobile app that works with the hardware have been choice. After create a account, open Arduino IDE and install Blynk library. Blynk Library is an extension that runs on top of hardware application. It handles all the connection routines and data exchange between hardware, Blynk Cloud and app project. When all library's already install, connect the hardware to upload a sketch or coding. It for to get a hardware online and connect it to Blynk Cloud. In Blynk Console, Create a new project for build the hardware. After that, setup a hardware and customize it. For Blynk Auth Token can get from there at tab device info. Copy that token and put in IDE. Figure below show that code use for Blynk setup.

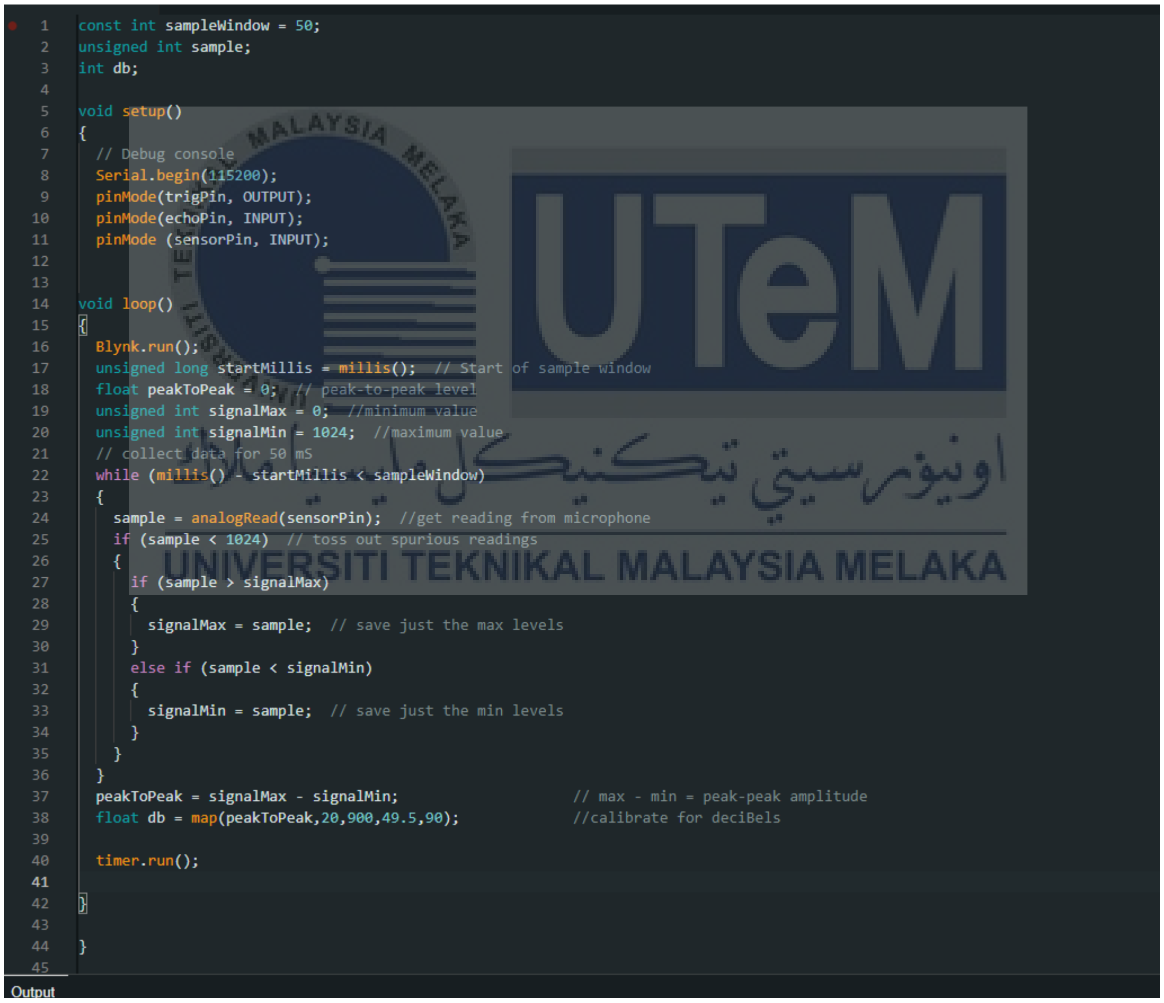


```
#define BLYNK_TEMPLATE_ID "TMPLvEs2MnlT"
#define BLYNK_DEVICE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "m9fDbiX51tYDvG_6A0s-_ansbzJSOLhi"
#define BLYNK_PRINT Serial

#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Myth ⍰";
char pass[] = "apapassword";

BlynkTimer timer;

BLYNK_WRITE(V0)
  int value = param.asInt();

  // Update state
  Blynk.virtualWrite(V4, db);
}

// This function is called every time the device is connected to the Blynk.Cloud
BLYNK_CONNECTED()
{
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, getUltrasonic());
  Blynk.virtualWrite(V4, mic());

}

void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
```

Figure 3.2 Code For Setup Blynk

3.2.1.2 KY-037 Electret Microphone Audio Sensor Module

The Sound Module was created for monitoring the sound level and intensity of any source. The sound level information is shown at Blynk app. Sound level is categorized based on graph and it show in numerical value. This sensor module for sound detection detects whether or not a threshold value has been exceeded. A microphone receives sound and inputs it to an LM393 operational amplifier. A built-in potentiometer is used to adjust the volume setting. For operation this sensor, they measured it by voltage to voltage and the show value in unit voltage.



```
const int sampleWindow = 50;
unsigned int sample;
int db;

void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode (sensorPin, INPUT);


void loop()
{
  Blynk.run();
  unsigned long startMillis = millis();  // Start of sample window
  float peakToPeak = 0;  // peak-to-peak level
  unsigned int signalMax = 0;  //minimum value
  unsigned int signalMin = 1024;  //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(sensorPin);  //get reading from microphone
    if (sample < 1024)  // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample;  // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample;  // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin;                    // max - min = peak-peak amplitude
  float db = map(peakToPeak,20,900,49.5,90);             //calibrate for deciBels

  timer.run();

}

}
```

Figure 3.3Code For Sound Module

3.2.1.3 Ultrasonic Sensor HC-SR04

Ultrasonic Sensor is a sensor that measures distance. It emits 40kHz ultrasonic waves that travel through the air and are reflected back to the module when they contact an object or barrier. Calculate the distance by multiplying the trip time by the speed of sound.

The pins of the HC-setup SR04 are VCC (1), TRIG (2), ECHO (3), and GND (4). The VCC supply voltage is +5V, and the TRIG and ECHO pins may be connected to any Digital I/O on the Arduino board.

Before beginning to write the programming code, the Trigger Pin and Echo Pin must be defined on the Arduino board. In this project, EchoPin is linked to D2, while TrigPin is connected to D3. Then, define distance (int) and duration (duration) variables (long).

In the loop, we must verify that the trigPin is clear by setting it to LOW for just two seconds. To create the ultrasonic wave, set the trigPin to HIGH for 10 seconds. Using the pulseIn() function, you must read the journey time and record it in the variable "duration."

This function receives two parameters: the name of the echo pin, and either HIGH or LOW for the second. HIGH indicates that the pulseIn() function will wait for the pin to become HIGH in response to the bouncing sound wave before beginning timing, and it will wait for the pin to become LOW after the sound wave finishes before ending timing. At its conclusion, the function will return the pulse length in microseconds. The distance will be calculated by multiplying the time by 0.034 and dividing by 2 as previously specified. Finally, the distance value will be reported on the Serial Monitor.

Table 3.1 Ultrasonic Sensor Specification

Operating Voltage (V)	DC 5V
Operating Current (mA)	15
Operating Frequency (KHz)	40
Max Range (m)	4
Min Range (cm)	2
Ranging Accuracy (mm)	3
Measuring Angle	15°
Trigger Input Signal	10µS TTL pulse
Dimension (mm)	45 x 20 x 15

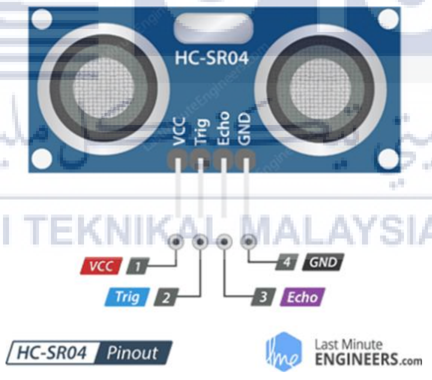
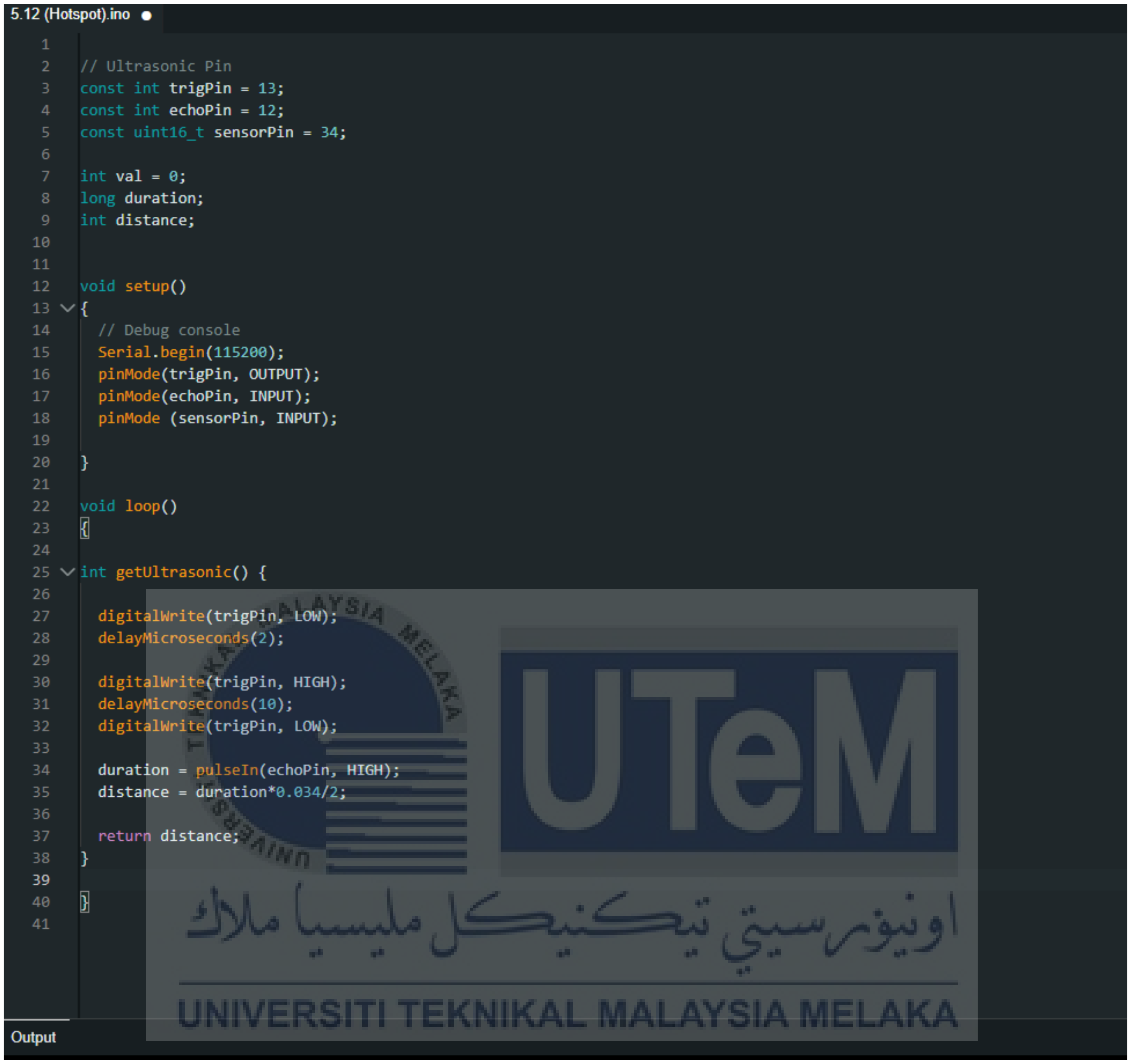


Figure 3.4 Ultrasonic Sensor HC-SR04 Configuration

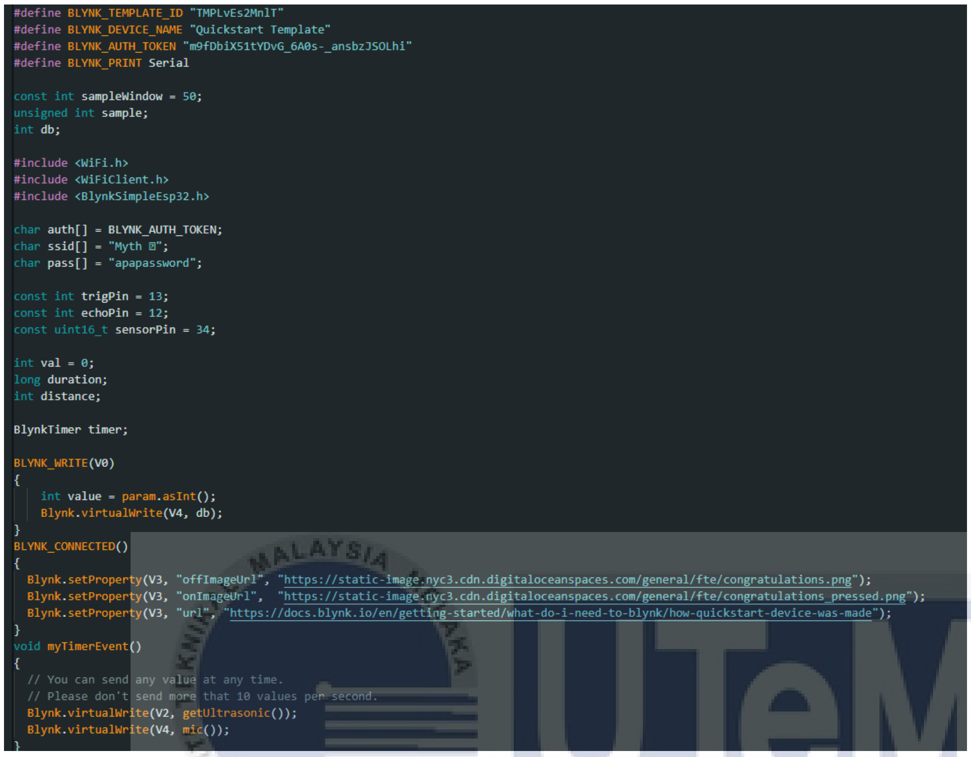

5.12 (Hotspot).ino

```

// Ultrasonic Pin
const int trigPin = 13;
const int echoPin = 12;
const uint16_t sensorPin = 34;

int val = 0;
long duration;
int distance;


void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode (sensorPin, INPUT);

}

void loop()
{

int getUltrasonic() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;

  return distance;
}

}

```

Output

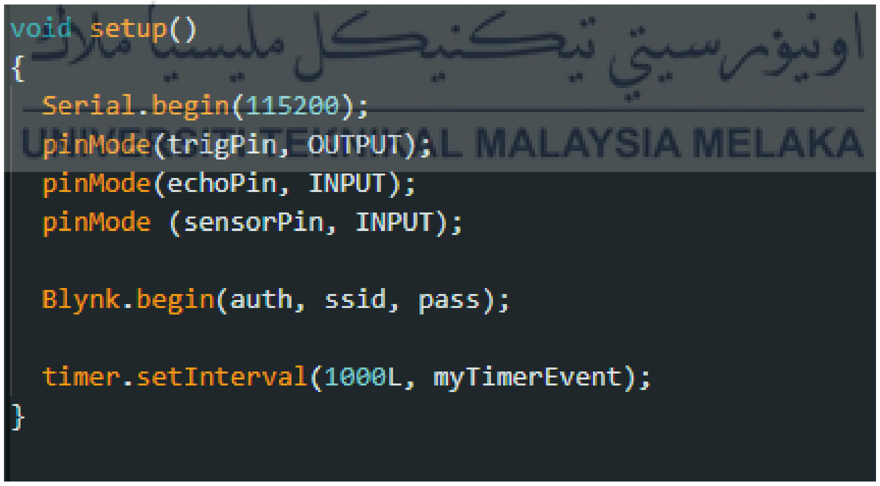
Figure 3.5 Code For Ultrasonic Sensor

3.2.2 Finalize Coding or sketch



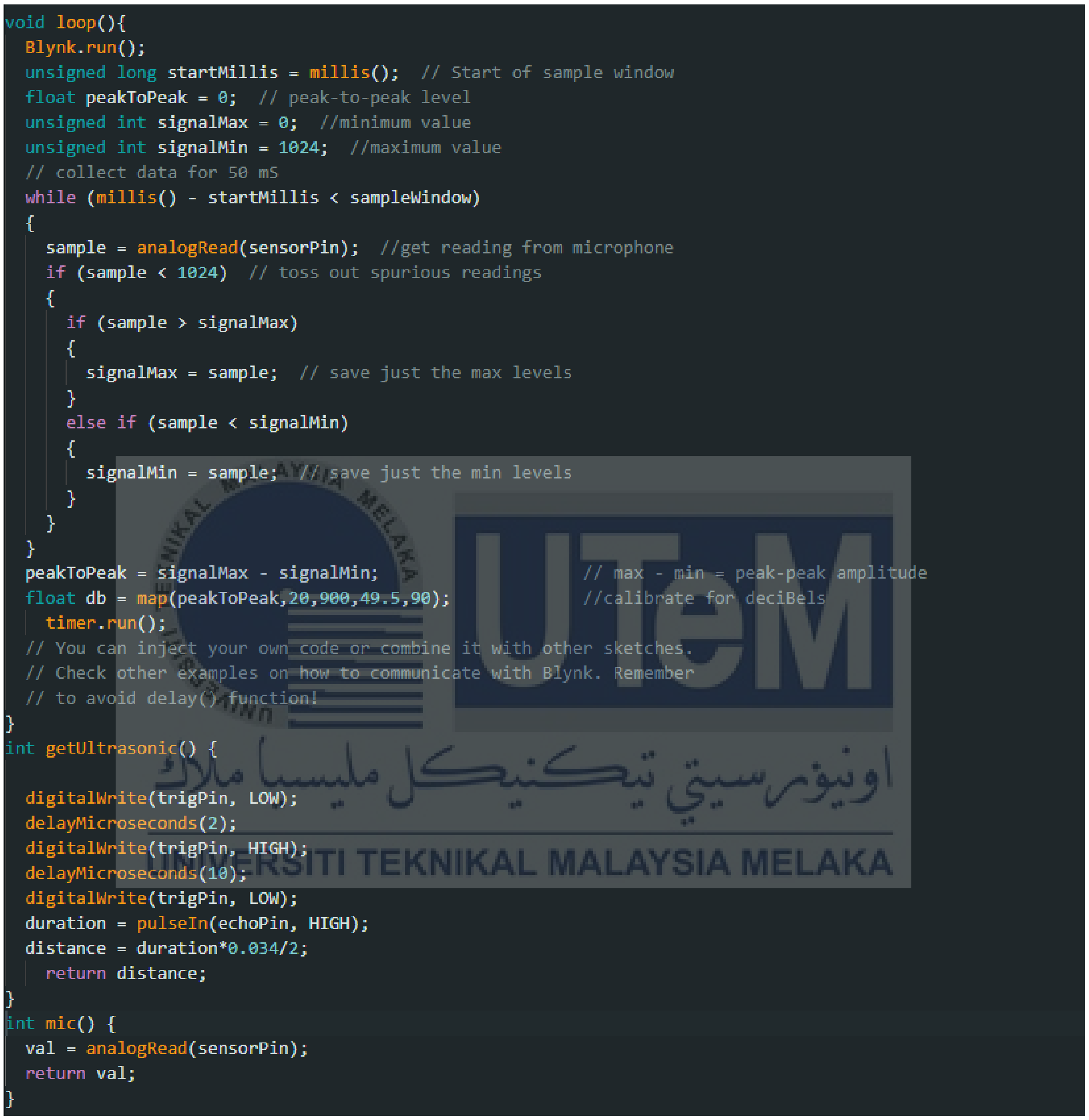
```
#define BLYNK_TEMPLATE_ID "TMPLvEs2MnlT"
#define BLYNK_DEVICE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "m9fDbiX51tYDvG_6A0s-_ansbzJSOLhi"
#define BLYNK_PRINT Serial

const int sampleWindow = 50;
unsigned int sample;
int db;

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Myth ⍰";
char pass[] = "apapassword";

const int trigPin = 13;
const int echoPin = 12;
const uint16_t sensorPin = 34;

int val = 0;
long duration;
int distance;

BlynkTimer timer;

BLYNK_WRITE(V0)
{
    int value = param.asInt();
    Blynk.virtualWrite(V4, db);
}
BLYNK_CONNECTED()
{
  Blynk.setProperty(V3, "offImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, getUltrasonic());
  Blynk.virtualWrite(V4, mic());
}
```

Figure 3.6 Describe Code



```
void setup()
{
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode (sensorPin, INPUT);

  Blynk.begin(auth, ssid, pass);

  timer.setInterval(1000L, myTimerEvent);
}
```

Figure 3.7 Code For Void Setup



```
void loop(){
  Blynk.run();
  unsigned long startMillis = millis();  // Start of sample window
  float peakToPeak = 0;  // peak-to-peak level
  unsigned int signalMax = 0;  //minimum value
  unsigned int signalMin = 1024;  //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(sensorPin);  //get reading from microphone
    if (sample < 1024)  // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample;  // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample;  // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin;                  // max - min = peak-peak amplitude
  float db = map(peakToPeak,20,900,49.5,90);           //calibrate for deciBels
    timer.run();
  // You can inject your own code or combine it with other sketches.
  // Check other examples on how to communicate with Blynk. Remember
  // to avoid delay() function!
}
int getUltrasonic() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;
    return distance;
}
int mic() {
  val = analogRead(sensorPin);
  return val;
}
```

Figure 3.8 Code For Void Loop

3.2.3 Blynk Apps

Blynk is a platform for iOS and Android that provides applications for controlling Arduino. Raspberry Pi and comparable gadgets are accessible through the Internet. It's a digital dashboard that lets you drag-and-drop widgets to create a graphical user interface for your project.

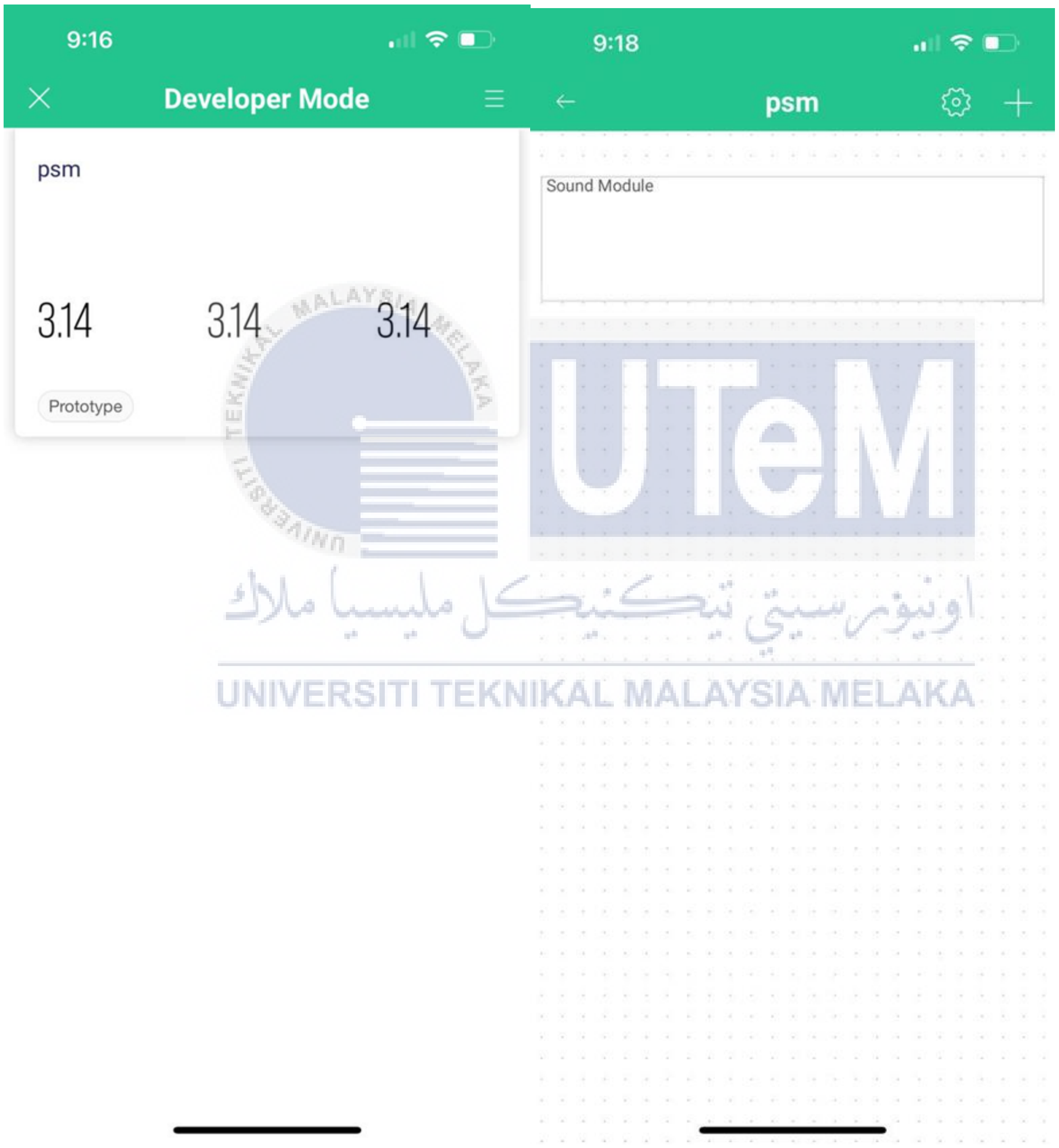
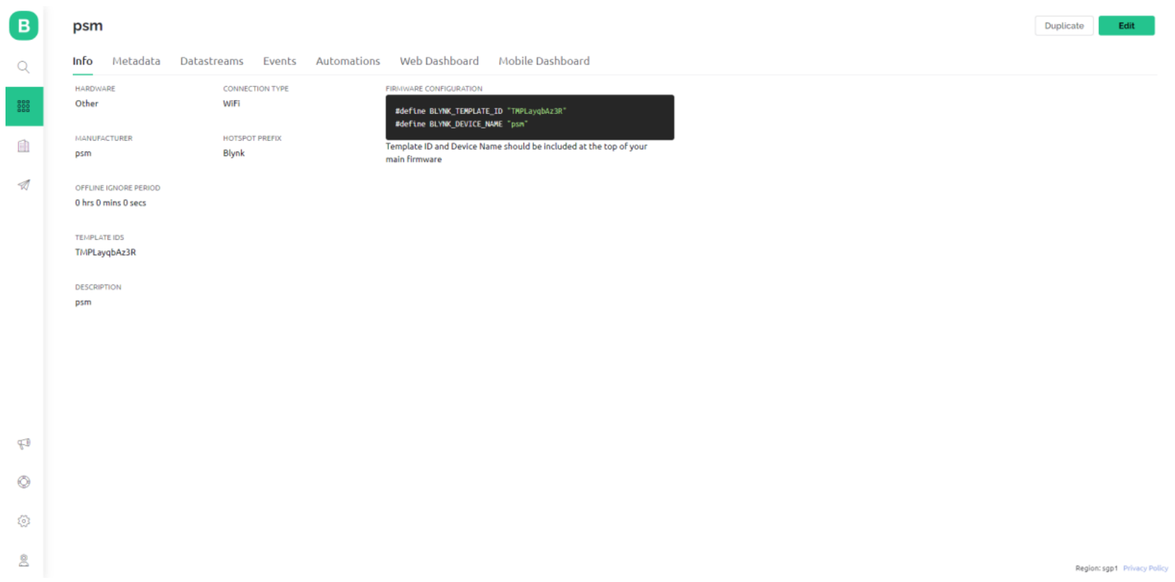


Figure 3.9 Blynk Mobile Dashboard



HARDWARE
Other

CONNECTION TYPE
WiFi

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLayqbAz3R"
#define BLYNK_DEVICE_NAME "psm"
```

Template ID and Device Name should be included at the top of your main firmware

MANUFACTURER
psm

HOTSPOT PREFIX
Blynk

OFFLINE IGNORE PERIOD
0 hrs 0 mins 0 secs

TEMPLATE IDS
TMPLayqbAz3R

DESCRIPTION
psm

Figure 3.10 Blynk Console Web Dashboard

3.2.3.1 Blynk App Function

- Blynk App - allows you to design aesthetically stunning interfaces for your projects by combining widgets from our collection.

- The Blynk Server oversees all smartphone-to-hardware connectivity. You have the choice of using our Blynk Cloud or establishing your own Blynk server. It is open source, able to manage thousands of devices, and even runs on a Raspberry Pi.

- Blynk Libraries allow server connectivity and the processing of all incoming and outgoing instructions on all popular hardware platforms.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Project Overview

The project is used to measure and monitor sound surrounding with length. It is developed to allow users to manage information efficiency of noise and preventive it. The main component for this project is:

1. ESP-32 Dev Kit
2. KY-037 Electret Microphone Audio Sensor Module
3. Ultrasonic Sensor HC-SR04
4. Power Supply (Power Bank)

The system will sense and measure sound through the cable and display the readings into Blynk display the reading from ESP32 as monitoring system. The user enables to monitor without any firewall.

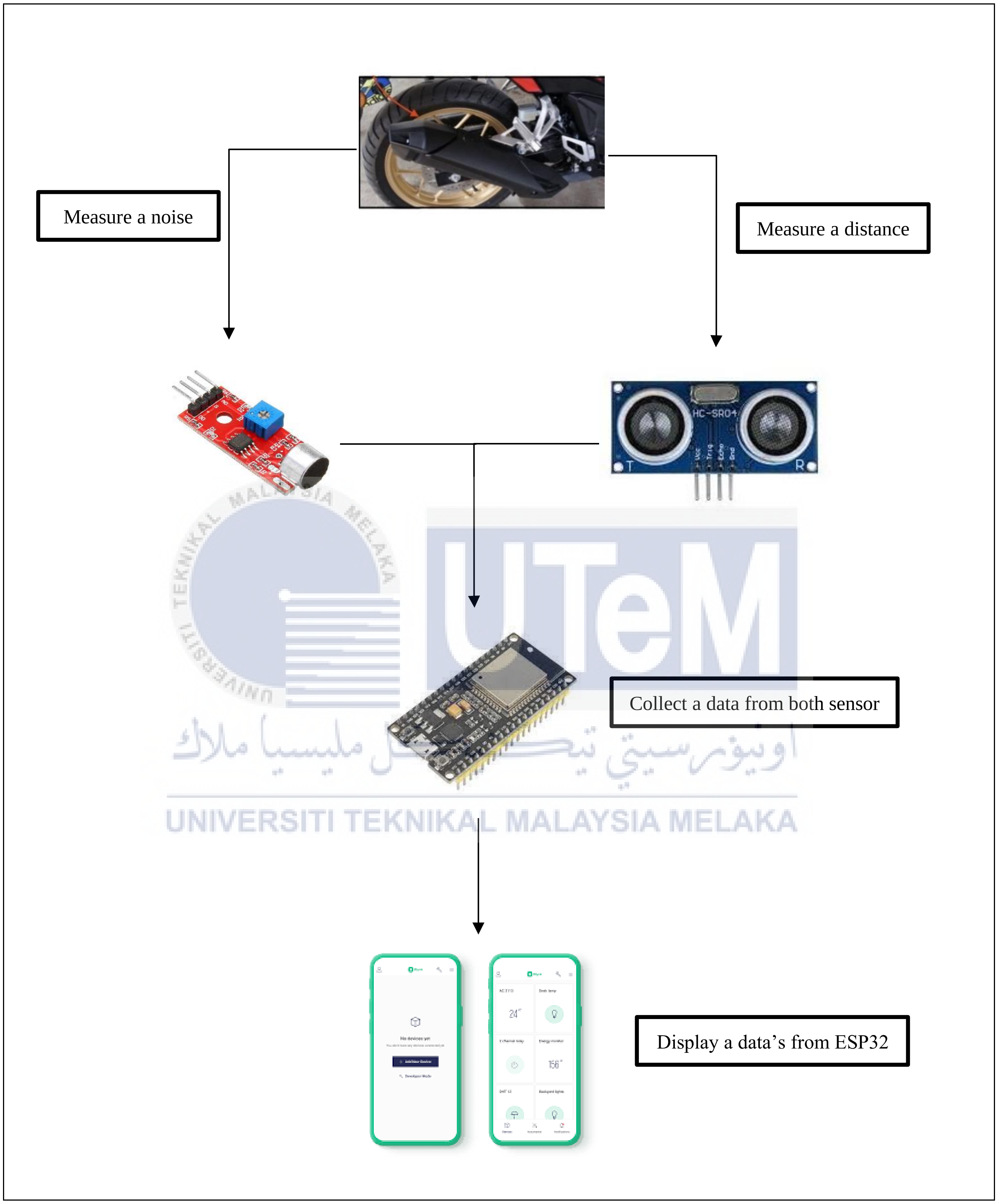


Figure 4.1 Flow Process Project

4.1.1 Result Of Completed Circuit

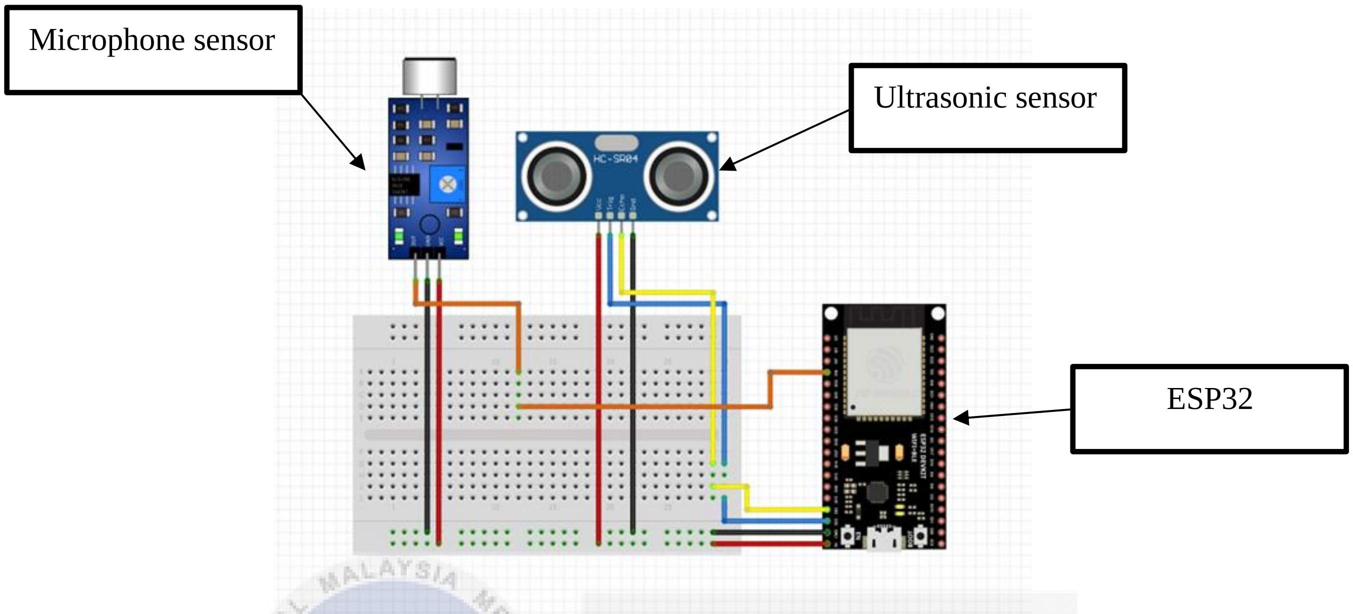


Figure 4.2 Completed Circuit In Fritzing

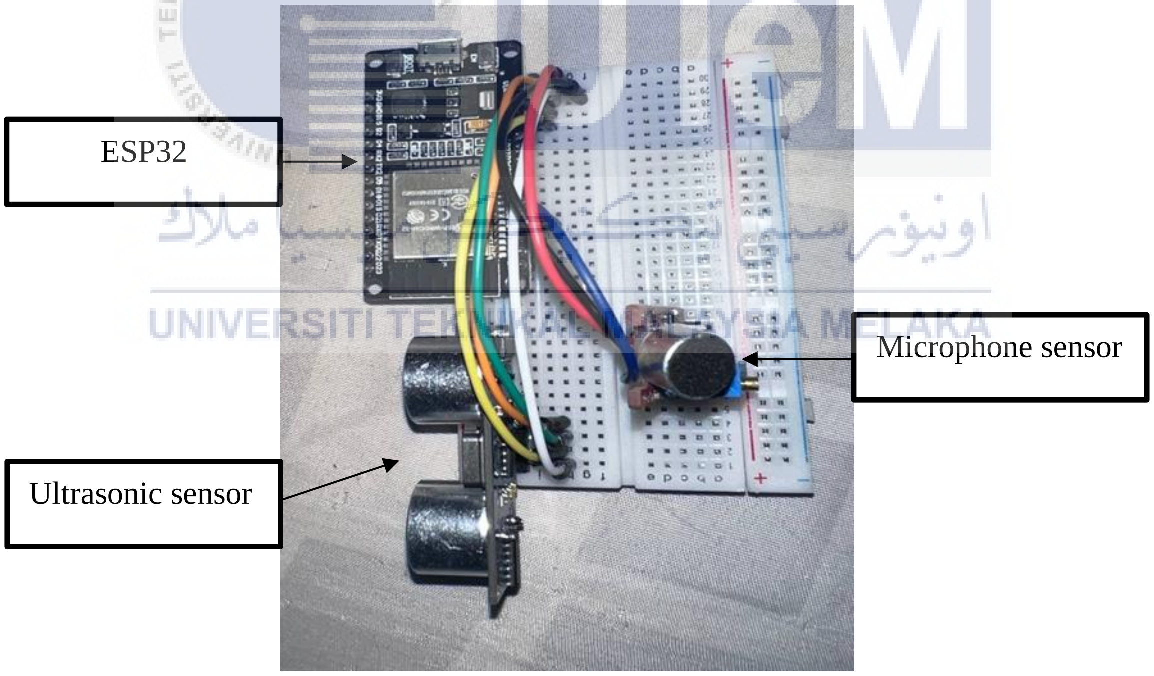


Figure 4.3 Completed Circuit

4.1.2 Schematic Circuit In Fritzing

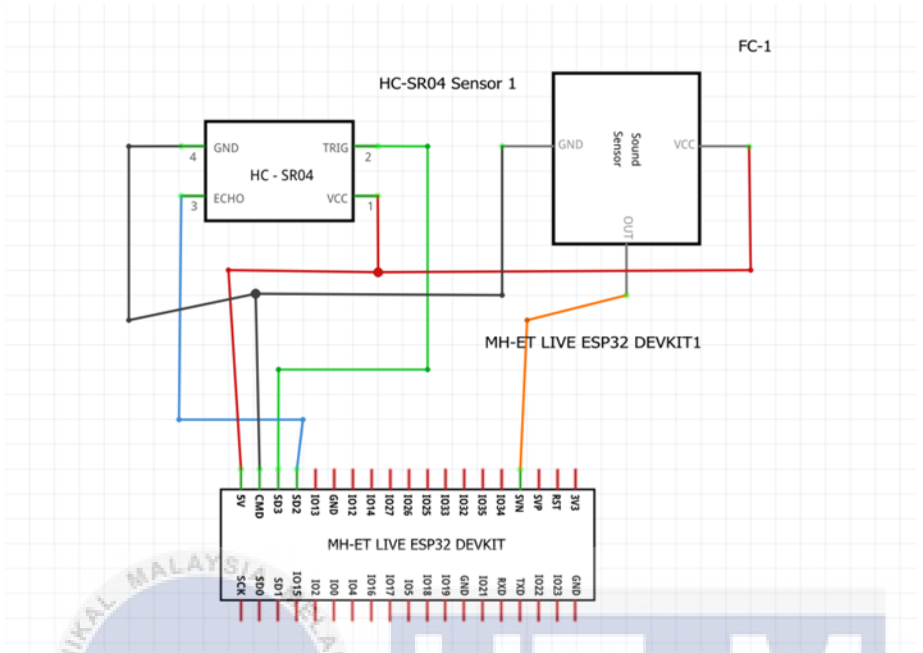

FC-1
HC-SR04 Sensor 1
HC - SR04
GND
TRIG
ECHO
VCC
Sound Sensor
GND
VCC
OUT
MH-ET LIVE ESP32 DEVKIT1
MH-ET LIVE ESP32 DEVKIT
5V CMD SD3 SD2 IO13 GND IO12 IO14 IO27 IO26 IO25 IO33 IO32 IO35 IO34 SVN SVP RST 3V3
SCK SD0 SD1 IO15 IO2 IO0 IO4 IO16 IO17 IO5 IO18 IO19 GND IO21 RXD TXD IO22 IO23 GND

Figure 4.4 Schematic Completed Circuit

The schematic circuit in the figure above is schematic for sound module monitoring where the ultrasonic and sound module were connected to ESP32. The sound sensor will receive input from test sound engine and ultrasonic will receive input of length then transmit the data from both input and ESP32 using Bluetooth, thus the output will appear in Blynk. The sound sensor requires 4V, ultrasonic sensor requires DC 5V. The power A source from ESP32 was connected directly to laptop using USB because the IDE needs to run simultaneously when sound module monitoring. For this project, simulation will not require as it has been tested directly on the hardware.

The ESP32 will connect each input/output of ultrasonic sensor connect to ground, Trig to D13. Echo to D12 of ESP32 and VCC of 3.3V. For sound sensor, the ESP32 will connect each input/output of sound sensor to grounded, digital output to pin D34 (A0) of ESP32 and VCC of 5V.

4.2 Final Prototype

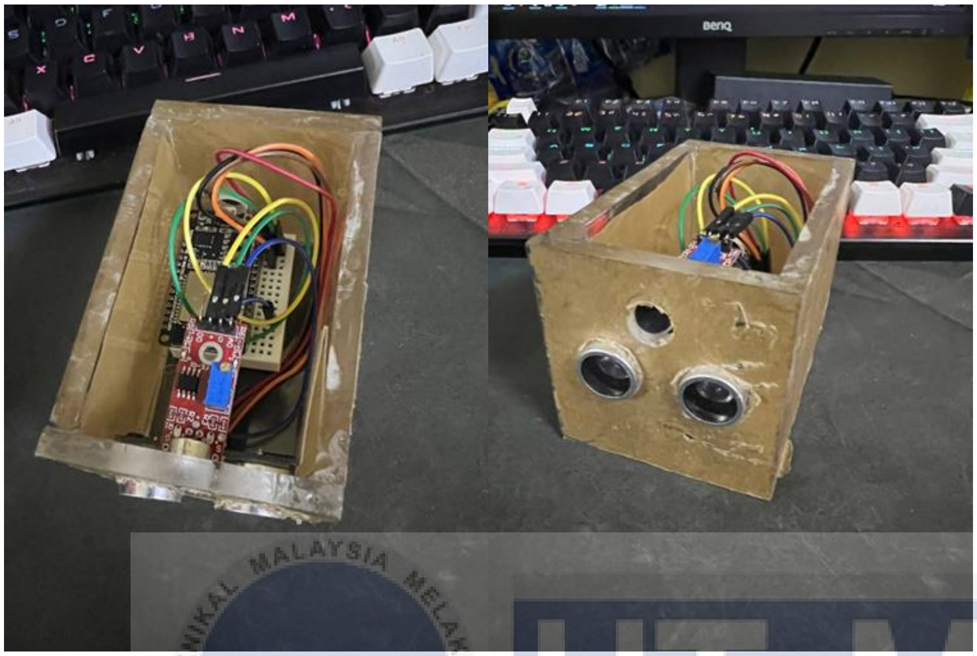


Figure 4.5 Final Prototype Project

For the prototype, this project was considered successful as it fulfills project objectives which it to monitor the noise and analyze the efficiency of this project with sound level meter. Even with few components, it can still manage to read the output of the noise by using ESP32, microphone sensor and ultrasonic sensor. Casing made from Acrylic Perspex.

4.3 Field Testing

4.3.1 Analysis in Lab Environment

For analysis noise in lab environment, Selective Laser Sintering (SLS) lab it quite noisy lab. In SLS lab, the activity inside lab mostly heavy process like welding, grinding, drilling, cutting and fabrication. Not include machines running inside this lab like compressor, sand blasting and SLS 3D printing.

Based on **figure 4.5** is a point of field testing for noise environment inside lab. This point is a middle of the lab. For analysis, it takes time around 30 minutes to measure noise of environment this lab. Measurement in decibel (dB) and every 2-minute measurement will take as a result.

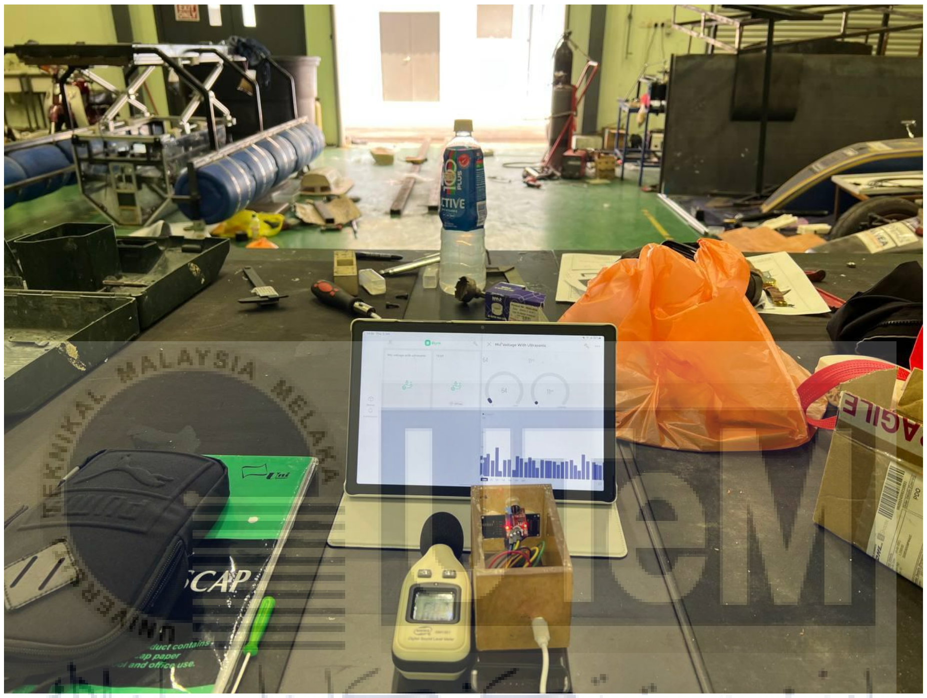
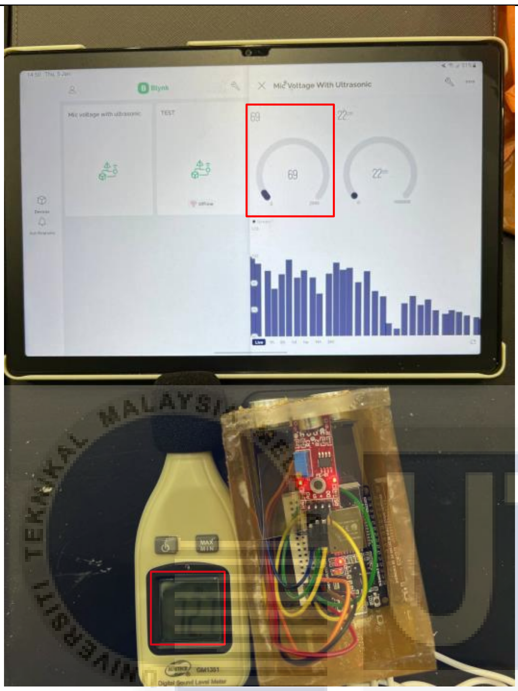
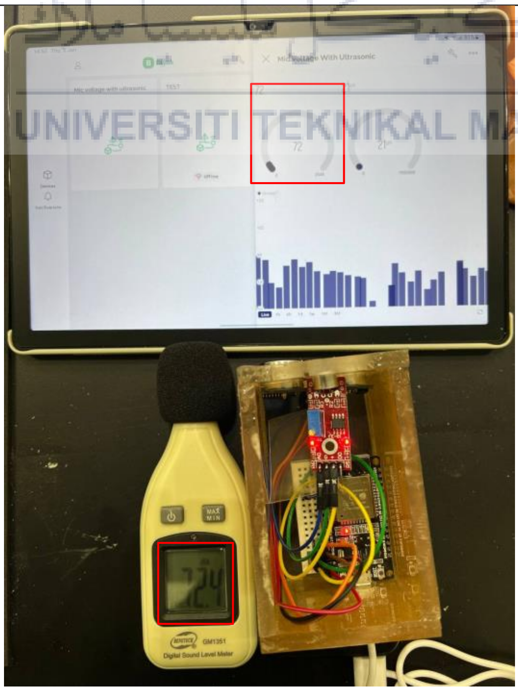
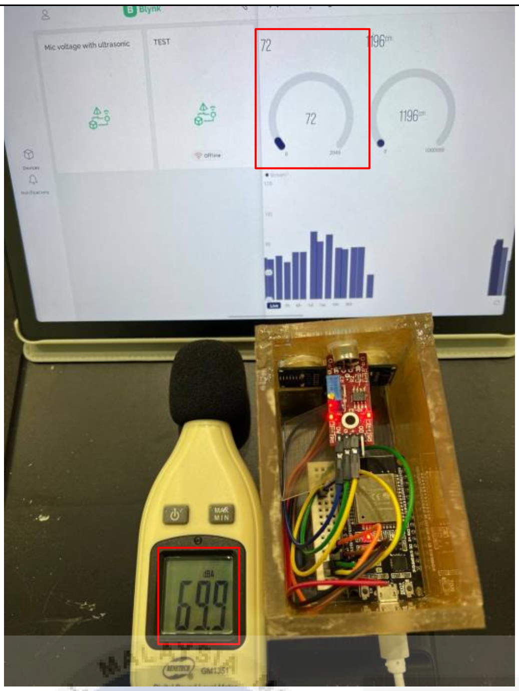
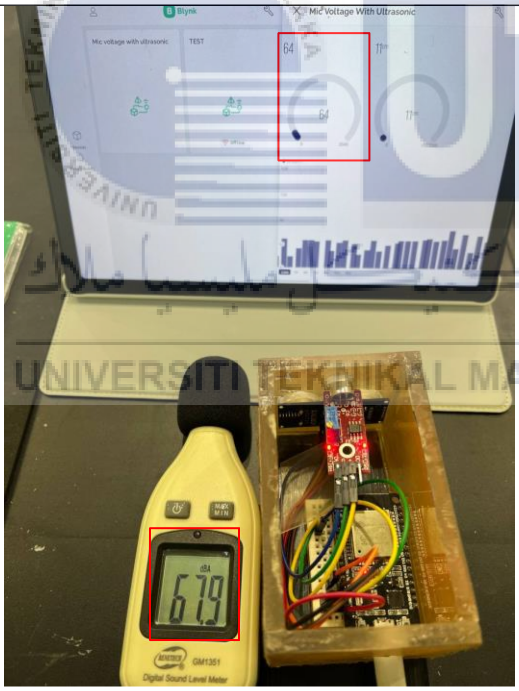


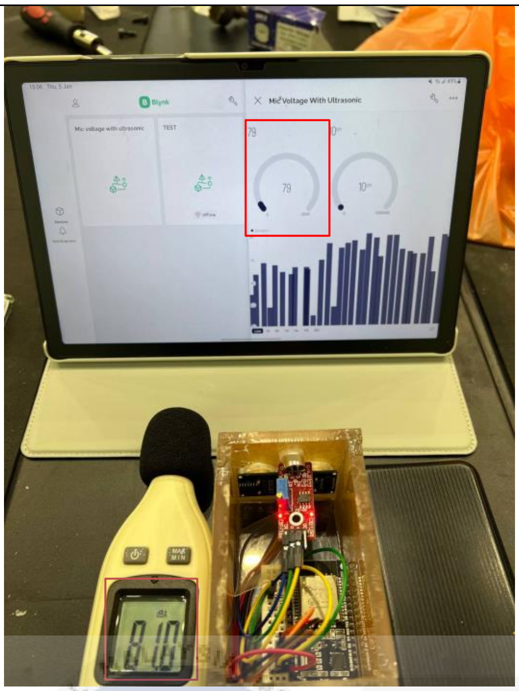
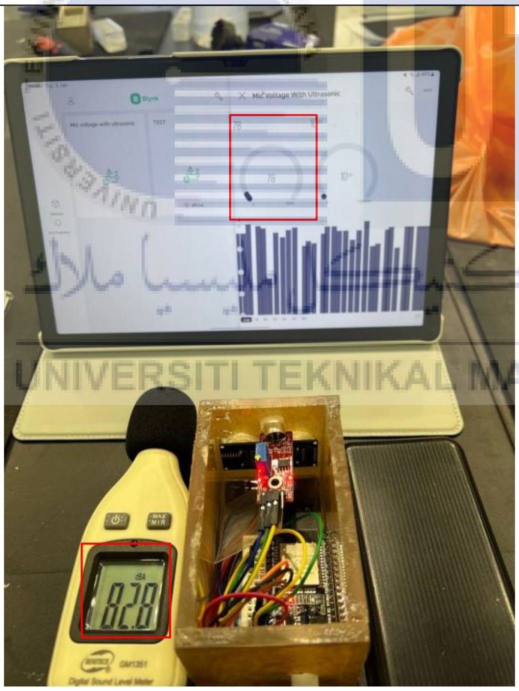
Figure 4.6 Point of Field Testing

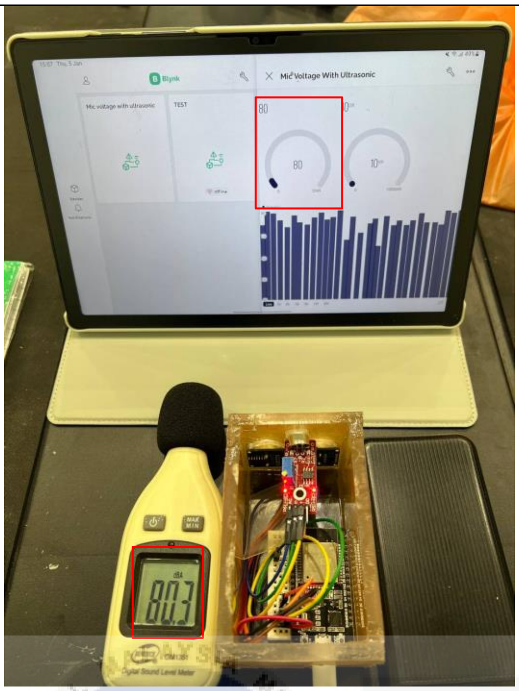
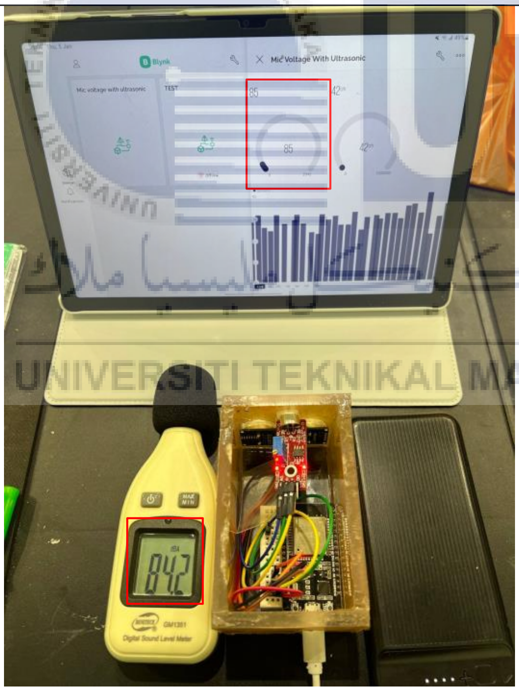
a) Result Of Measurements in SLS Lab

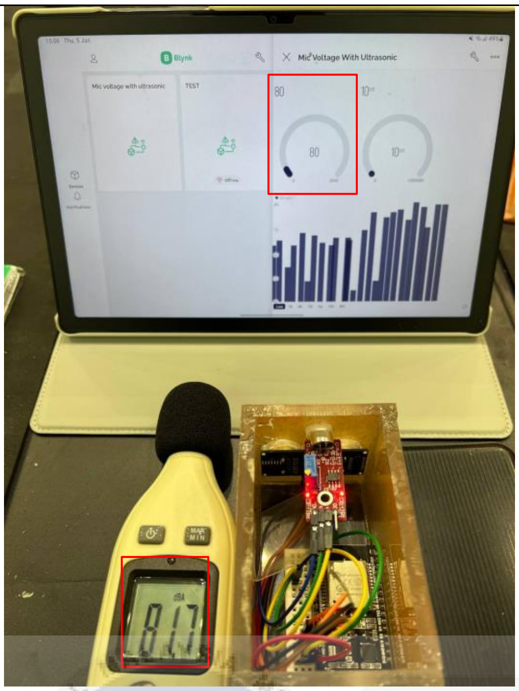
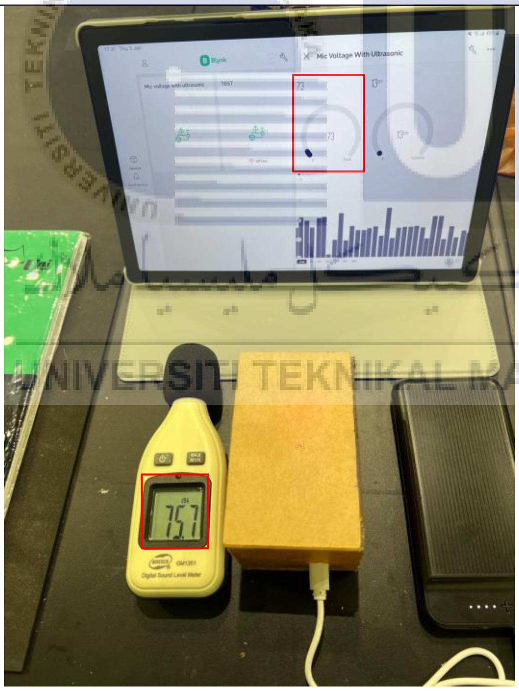
Table 4.1 Result of Measurement in 30 minutes

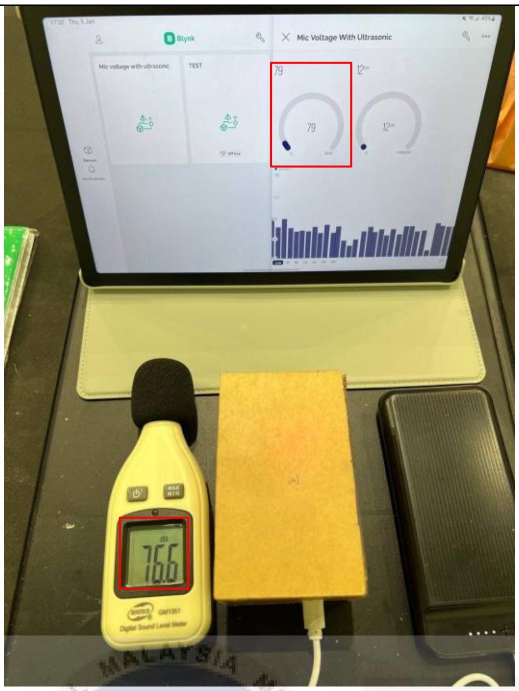
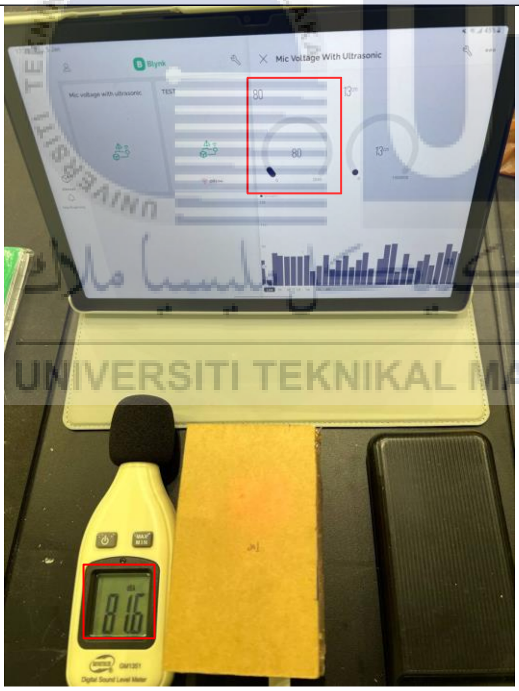
Duration	**Result**	**Reference Value (dB)**	**Experimental Value (dB)**
2 Minute		72.7	69
4 Minute		72.4	72

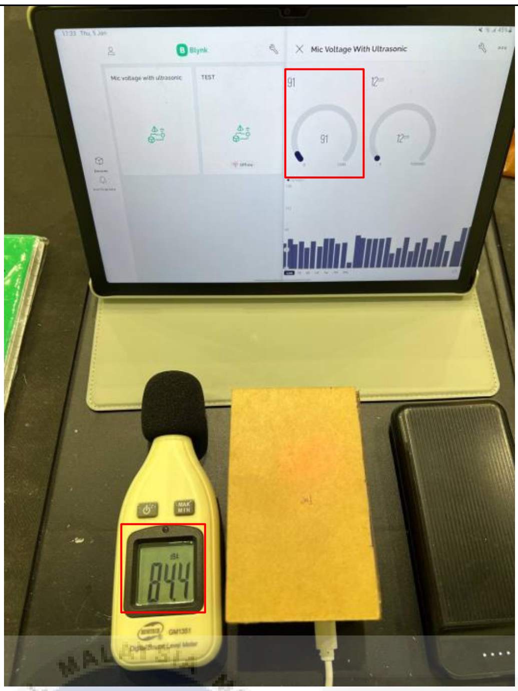
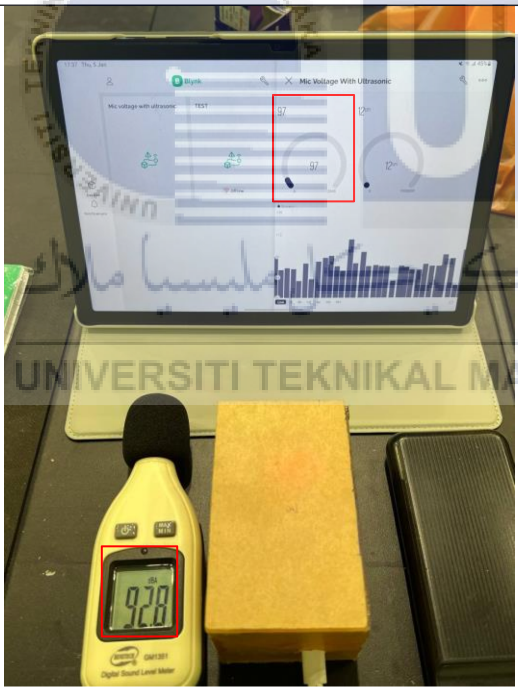
6 Minute		69.9	72
8 Minute		67.9	64

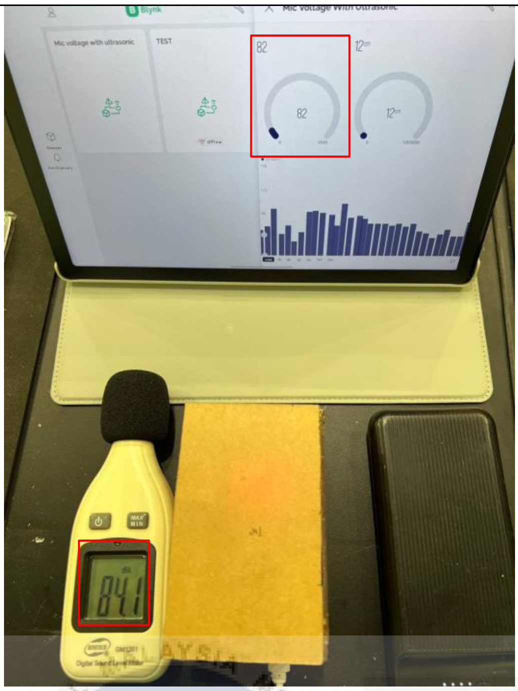
10 Minute		81	79
12 Minute		82.8	78

14 Minute		80.3	80
16 Minute		84.2	85

18 Minute		81.7	80
20 Minute		75.7	73

22 Minute		76.6	79
24 Minute		81.6	80

26 Minute	 Blynk Mic Voltage With Ultrasonic 91 12 84.4 GM1351 Digital Sound Level Meter	84.4	91
28 Minute	 Blynk Mic Voltage With Ultrasonic 97 12 92.8 GM1351 Digital Sound Level Meter	92.8	97

30 Minute		84.1	82

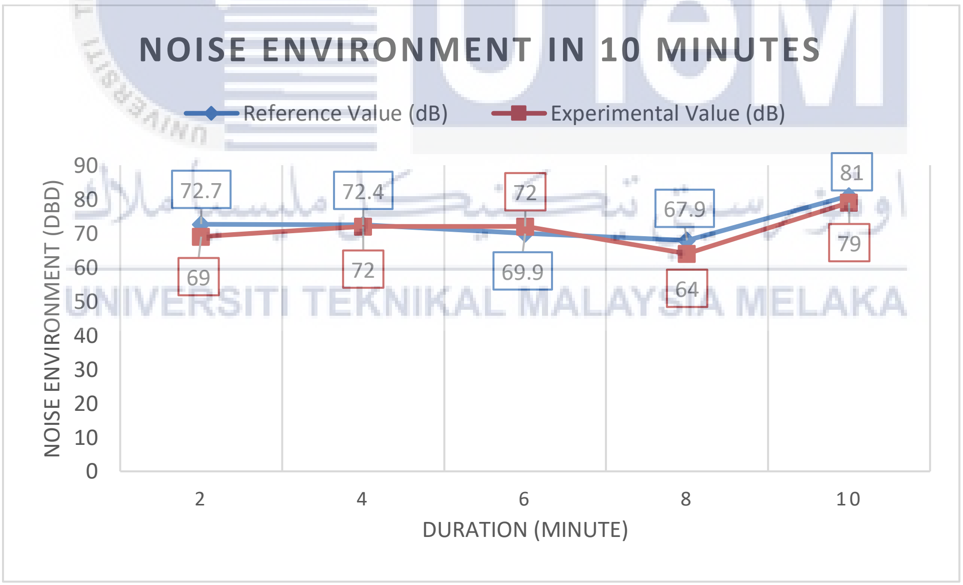


Figure 4.7 Noise Measurement in 10 Minutes

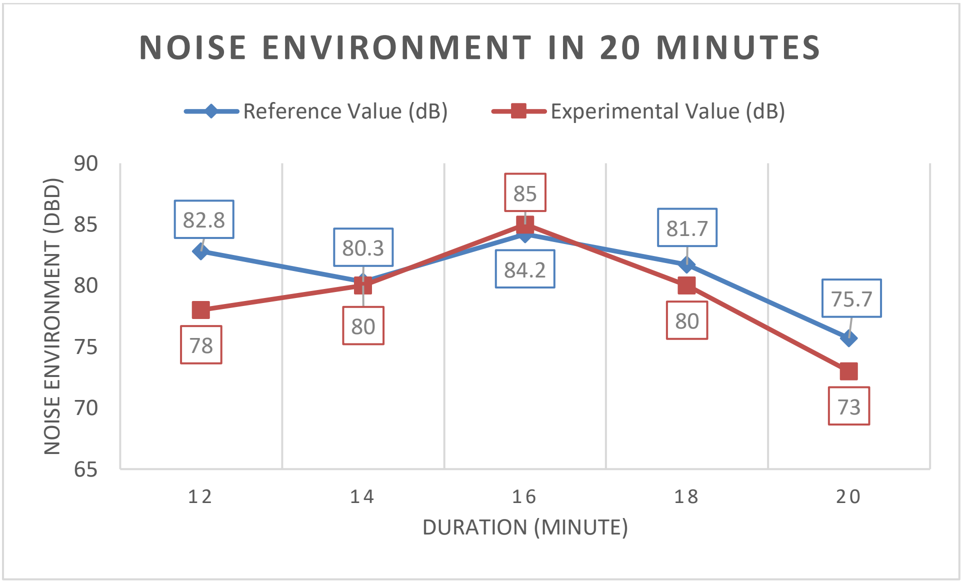


Figure 4.8 Noise Measurement in 20 Minutes

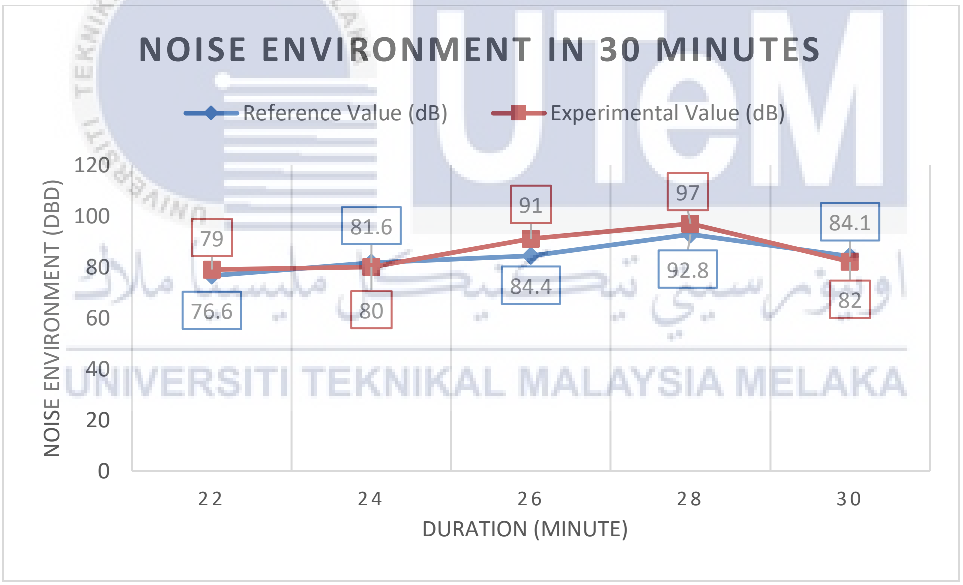


Figure 4.9 Noise Measurement in 30 Minutes

As a figure show above, the measurement environment in 30 minutes. The efficiency between this project and sound level meter almost same value. The far different value gap between this product at 26 minutes with 7.82% error percentage. It can be acceptable cause error of percentage still under 10%.

b) Percentage Of Error

Percent of error is the difference between estimated reference value and experimental value in comparison to the reference value and is expressed as a percentage. In other words, the percentage of error is the relative error multiplied by 100. The percentage of error must not higher than 10% to make sure the experimental value close to the reference value.

$$\% \text{ Error} = \frac{Reference\ Value - Experimental\ Value}{Reference\ Value} \text{ X } 100\%$$

Example of calculation:

$$\% \text{ Error}_{\ 2\ minute} = \frac{72.7-69}{72.7} \text{ X } 100\% = 5.09\%$$

Table 4.2 Result Percentage of Error Lab Environment

Duration (Minute)	Reference Value (dB)	Experimental Value (dB)	Percentage of Error (%)
2	72.7	69	5.09
4	72.4	72	0.55
6	69.9	72	3
8	67.9	64	5.74
10	81	79	2.47
12	82.8	78	5.8
14	80.3	80	0.37
16	84.2	85	0.95
18	81.7	80	2.08
20	75.7	73	3.57
22	76.6	79	3.13
24	81.6	80	1.96
26	84.4	91	7.82
28	92.8	97	4.53
30	84.1	82	2.5

4.3.2 Analysis on Motorcycle

For analysis on motorcycle, Yamaha Lagenda Fi 115 with open exhaust has been tested. As figure 4.9 below show the distance between project and sound level meter to exhaust motorcycle. The measurement takes in afternoon and cannot do more higher than 4000 rpm of engine motorcycle. It will make a lot noisy sound for neighbors. The measurement will take from environment, idle engine and 1000-4000 rpm.

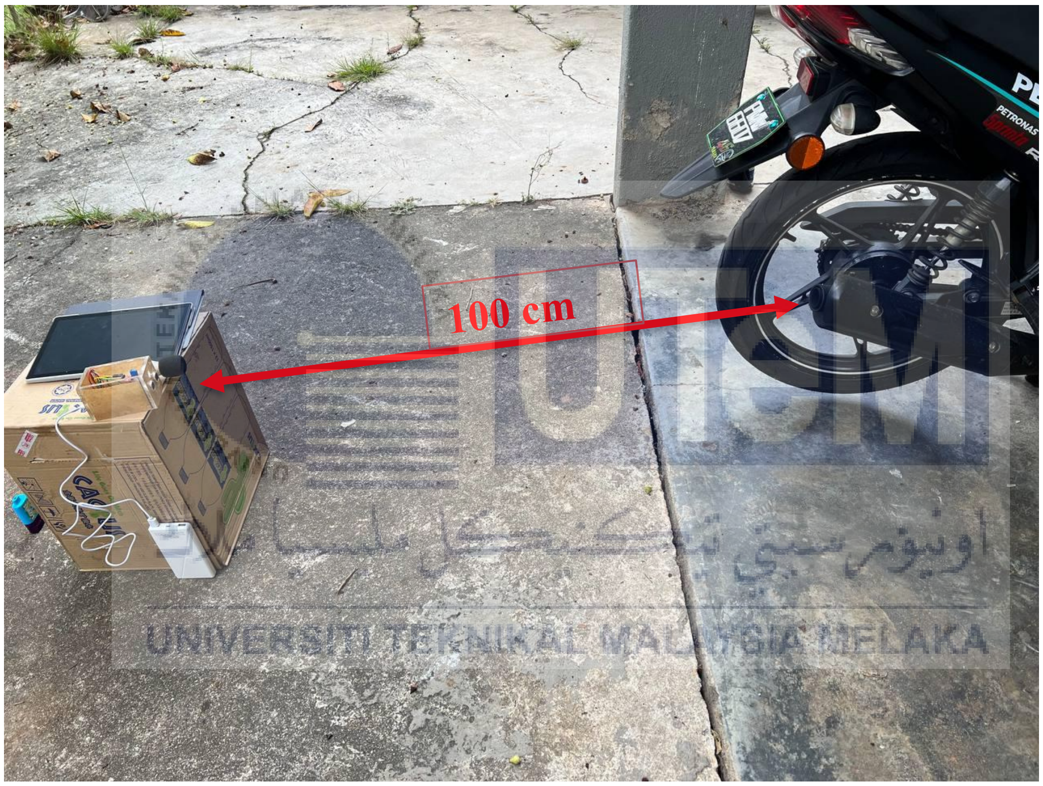
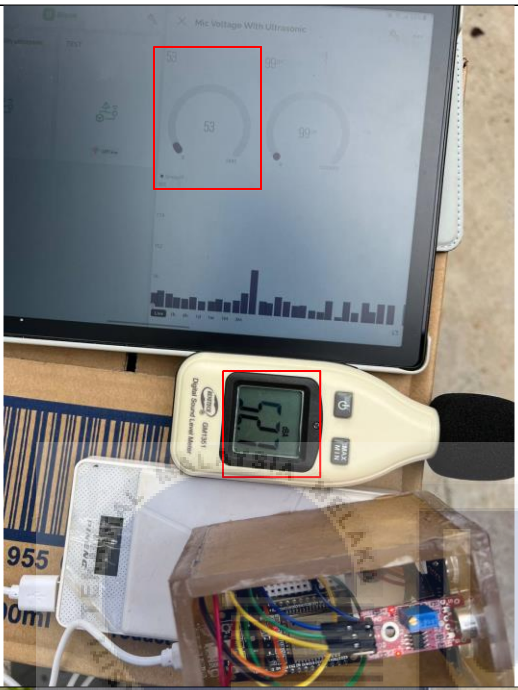
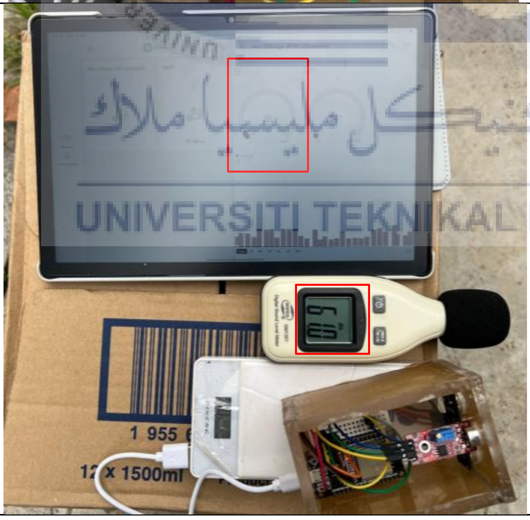
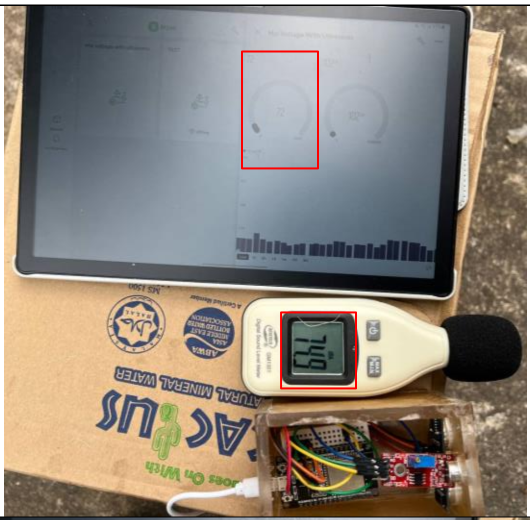
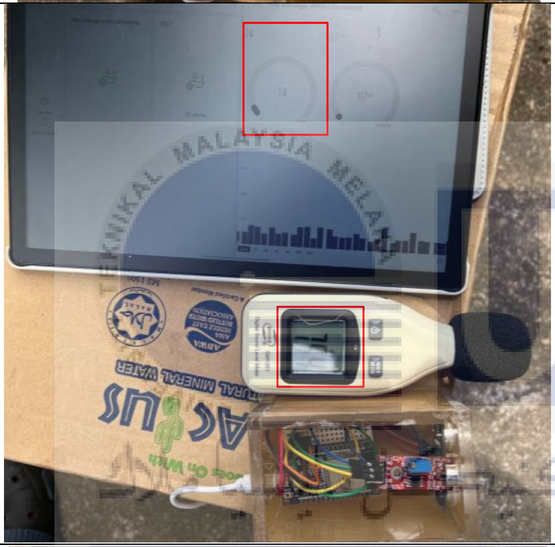
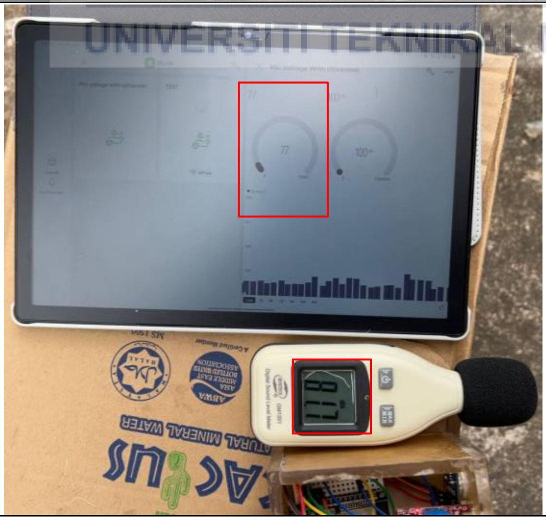


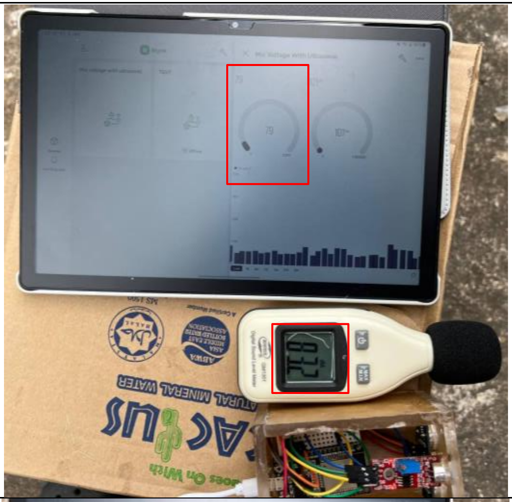
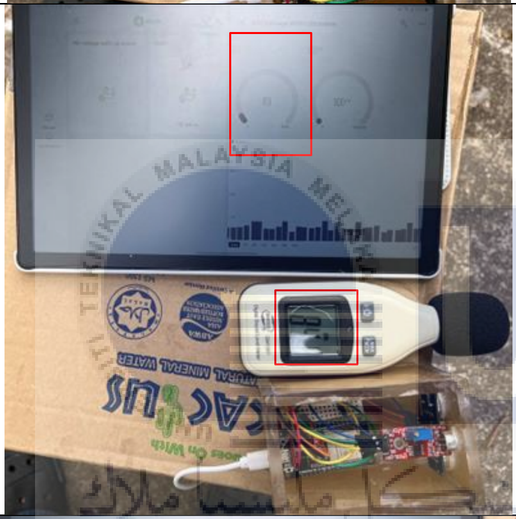
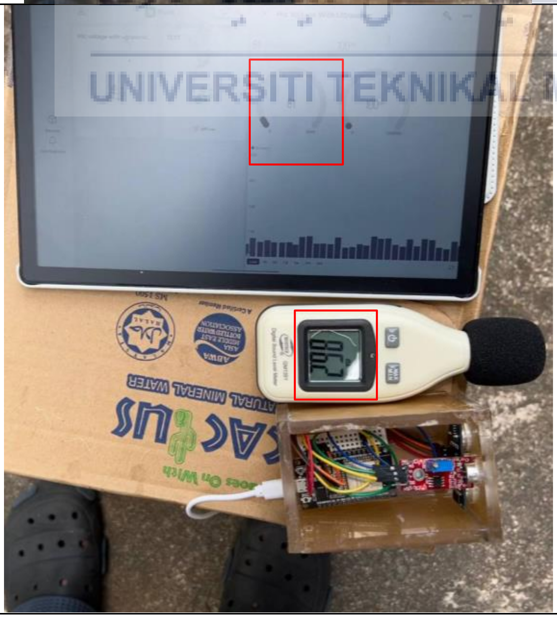
Figure 4.10 Distance between tester and exhaust motorcycle

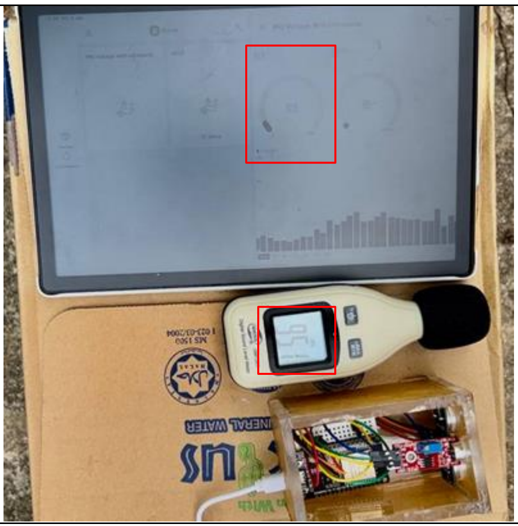
a) Result Of Measurements on exhaust motorcycle

Table 4.3 Result measurement on exhaust motorcycle

Description	Result	Reference Value (dB)	Experimental Value (dB)
Environment		52.7	53
800 rpm (idle)		61	59

1200 rpm		74.9	72
1500 rpm		75.4	74
2000 rpm		81.7	77

2300 rpm		83.2	79
2500 rpm		84.1	83
3000 rpm		86.2	81

4000 rpm		95.1	103

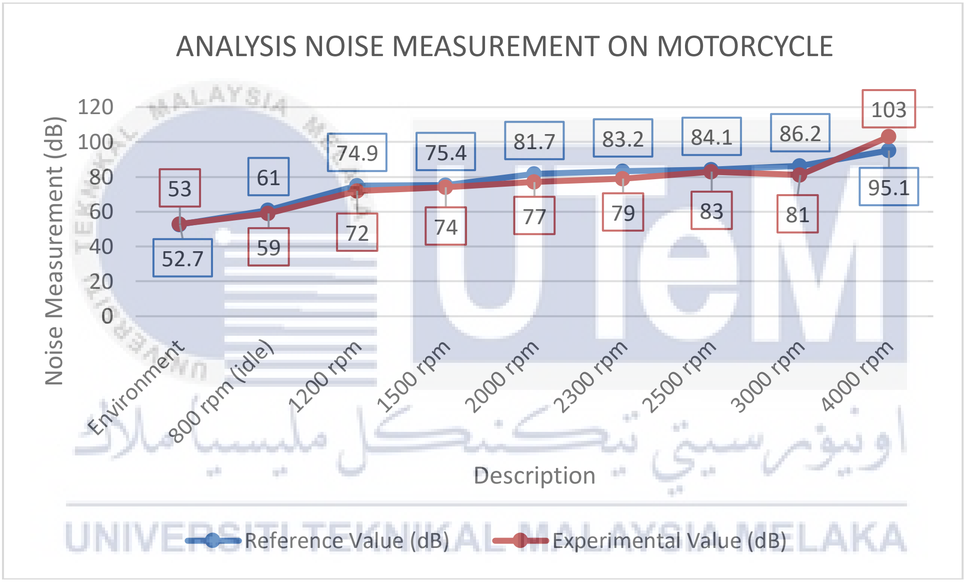


Figure 4.11 Graph Analysis On Motorcycle

At figure 4.11 show the measurement of motorcycle exhaust. It quite difficult to measure it cause noise environment can interrupt too. The higher of error percentage at 4000 rpm with 8.31%. the pattern of these 2 lines almost same. This project value can be acceptable too because it under 10% of error percentage.

b) Percentage Of Error

Percent of error is the difference between estimated reference value and experimental value in comparison to the reference value and is expressed as a percentage. In other words, the percentage of error is the relative error multiplied by 100. The percentage of error must not higher than 10% to make sure the experimental value close to the reference value.

$$\% \text{ Error} = \frac{Reference\,Value - Experimental\,Value}{Reference\,Value} \text{ X } 100\%$$

Example of calculation:

$$\% \text{ Error}_{Enviroment} = \frac{52.7 - 53}{52.7} \text{ X } 100\% = 0.6\%$$

Table 4.4 Result Percentage of Error Motorcycle

Description	Reference Value (dB)	Experimental Value (dB)	Percentage of Error (%)
Environment	52.7	53	0.6
800 rpm (idle)	61	59	3.28
1200 rpm	74.9	72	3.87
1500 rpm	75.4	74	1.86
2000 rpm	81.7	77	5.75
2300 rpm	83.2	79	5.05
2500 rpm	84.1	83	1.31
3000 rpm	86.2	81	6.42
4000 rpm	95.1	103	8.31

4.4 Discussion

There were many problems encounter when programming the ESP32, wiring from the breadboard, sensitiveness of the sensor or component itself, few trials and error for coding and sensitiveness of microphone sensor. All of these problems were finally overcome after doing more research and asking for helped. For wiring, it is suspected that the breadboard itself has loose connection. For confirmation, connecting the jumper wire at other connecting port in the breadboard or replace with another breadboard for trial and error.

Next, for a small component as microphone sensor, it is one of the factors that should be handle carefully because of sensitiveness sensor itself. For this one, bought another microphone sensor for backup of plan if first microphone problem to continue this project. When conducting this project, there were many factors effecting the results. Environment noise also make a result not be to accurate focusing on motorcycle cause another vehicle moving surrounding at field testing place.

From observation show on table 4.1 and 4.2, there not much different between this project and sound level meter. The value from both almost same. Percentage of error to make sure the value from both not higher than 10%. It causes factor of the manufacturing and sensitiveness the sensor. At table 4.1, noise environment drop because activity inside lab stop example at 20 minute. In table 4.4, show noise measurements increase because when rpm increase same with noise will increase too.

After considering the factor from the environmental, human errors and manufacturing of sensor, this project managed to analyze the efficiency of the project.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, all requirements for this project must be met in this chapter. The entire sequence shown in the flowchart which this project mainly divided into three phase, software development, programming code and hardware fabrication system. This project was inspired from existing systems which are used in measured noise surrounding like sound level meter. For this project upgrade it to wireless system. Student also able to confront hardware subjects related to sensors and programming. Besides, the software development using IDE and Blynk it required the most attention. Indirectly, students are able to learn and had the opportunity from the self-learning programming with dedicated Arduino functions. This project can help student to develop their own DIY monitoring Arduino project and understanding more also go deeper about IoT. Besides, this project is appropriate to practice because it can be monitoring far away by using the Blynk app. Moreover, this project also can achieve IoT development. This monitoring noise project really helped user to help preventive a noise pollution. Even though this project were not able to compare the efficiency between the industrial specification tools.

5.2 Recommendations

For future improvement, it is recommended to use the higher sensitivity of microphone sensor could help improve the result. If the sensor is already had high sensitivity, it can be adjusted to set the minimum range in the operating system. It also can add another sensor like air quality sensor for measure air surrounding to preventive from air pollution. Besides that, try to comparison ESP32 and ESP8266. ESP8266 more accuracy because ESP8266 have Analog Pin for input of microphone sensor. For this recommendation, prefer use ESP8266 than ESP32 because ESP32 only have digital input. Microphone sensor not suitable by using digital output.

5.3 Project Potential

This project basically from system sound level meter. With this project, add some new feature and sensor. It can be monitoring online and can measure a distance of project. This project also can be uses in industry for measuring noise environmental. This project right now in prototype, it can be upgrade for more purposed like adding new sensor for measuring air surrounding. Sound level meter in market right now do not have online monitoring, this project can comparable with tool in market right now. Quality checker can use this project also for more easily to handle the situation like from 2 people and can be handle by 1 person only. Example situation like Jabatan Pengangkutan Jalanraya (JPJ) inspection of motorcycle exhaust.

REFERENCES

Stansfeld, S. A., & Matheson, M. P. (2003). Noise pollution: non-auditory effects on health. *British medical bulletin*, 68(1), 243-257.
https://academic.oup.com/bmb/article-abstract/68/1/243/421340

Morillas, J. M. B., Gozalo, G. R., González, D. M., Moraga, P. A., & Vílchez-Gómez, R. (2018). Noise pollution and urban planning. Current Pollution Reports, 4(3), 208-219.
https://link.springer.com/article/10.1007/s40726-018-0095-7

Slabbekoorn, H. (2019). Noise pollution. Current Biology, 29(19), R957-R960.
https://www.cell.com/current-biology/pdf/S0960-9822(19)30863-2.pdf

Babiuch, M., Foltýnek, P., & Smutný, P. (2019, May). Using the ESP32 microcontroller for data processing. In 2019 20th International Carpathian Control Conference (ICCC) (pp. 1-6). IEEE.
https://ieeexplore.ieee.org/abstract/document/8765944/

Muliadi, M., Imran, A., & Rasul, M. (2020). Pengembangan tempat sampah pintar menggunakan esp32. *Jurnal Media Elektrik*, *17*(2), 73-79.
http://download.garuda.kemdikbud.go.id/article.php?article=1747009&val=4329&title=PENGEMBANGAN%20TEMPAT%20SAMPAH%20PINTAR%20MENGGUNAKAN%20ESP32

Kurniawan, A. (2019). *Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32*. Packt Publishing Ltd.
https://books.google.com/books?hl=id&lr=&id=v86PDwAAQBAJ&oi=fnd&pg=PP1&dq=esp32+sound+sensor&ots=RzTUhcU2_8&sig=B9j3lc1h6AFmMc6o77l6GbFQKuk

Kanakaraja, P., Sundar, P. S., Vaishnavi, N., Reddy, S. G. K., & Manikanta, G. S. (2021). IoT enabled advanced forest fire detecting and monitoring on Ubidots platform. *Materials Today: Proceedings*, *46*, 3907-3914.
https://www.sciencedirect.com/science/article/pii/S2214785321014449

Dhiya, D. A. A., & Taufik, I. (2022). The Design of a Noise Detection Automatisation Tool in Library–Based on Internet of Things. *Inspiration: Jurnal Teknologi Informasi dan Komunikasi*, *12*(2), 17-31.
https://ojs.unitama.ac.id/index.php/inspiration/article/view/12

Seneviratne, P. (2018). *Hands-On Internet of Things with Blynk: Build on the power of Blynk to configure smart devices and build exciting IOT projects*. Packt Publishing Ltd.
https://books.google.com/books?hl=id&lr=&id=ZHteDwAAQBAJ&oi=fnd&pg=PP1&dq=blynk+application+setup&ots=K008d6QnbN&sig=P-85z6NTH4xZWO7Lnu9tcJ2fZe8

Durani, H., Sheth, M., Vaghasia, M., & Kotech, S. (2018, April). Smart automated home application using IoT with Blynk app. In *2018 Second international conference on inventive communication and computational technologies (ICICCT)* (pp. 393-397). IEEE.
https://ieeexplore.ieee.org/abstract/document/8473224/

Celestina, M., Hrovat, J., & Kardous, C. A. (2018). Smartphone-based sound level measurement apps: Evaluation of compliance with international sound level meter standards. *Applied Acoustics*, *139*, 119-128.
https://www.sciencedirect.com/science/article/pii/S0003682X17309945

APPENDICES

APPENDIX A Table 1 Gantt Chart PSM 1

TIME (WEEK) / ACTIVITY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Attend Projek Sarjana Muda (PSM) briefing	🟧	🟧	🟧												
Identify title for PSM	🟧	🟧													
Choosing supervisor	🟧	🟧													
Brainstorming		🟧													
Supervisor meeting and report discussion		🟧		🟧		🟧	🟧			🟧			🟧		
Hardware & Software research/preparation			🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧					
Draft report & slides submission										🟧		🟧			
Final eLogbook, report and slide submission													🟧	🟧	
Presentation, Q&A with a panel															🟧

APPENDIX B Table 2 Gantt Chart PSM 2

TIME (WEEK) / ACTIVITY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PROJECT BRIEFING	COMPLETED								COMPLETED						
EXPERIMENTAL SETUP		COMPLETED													
DATA COLLECTION				COMPLETED											
RESULT ANALYSIS						COMPLETED									
RESULT VERIFICATION										COMPLETED					
CONCLUSION AND RECOMMENDATION													COMPLETED 80%		
WEEKLY REPORTING (LOGBOOK)	COMPLETED 80%														
PROJECT REPORTING					COMPLETED 80%										
4 PAGES SUMMARY														COMPLETED	
PRESENTATION AND SLIDES														COMPLETED	