DEVELOPMENT OF IOT BASED MONITORING LAUNDRY SYSTEM USING NODEMCU ESP32 AND BLYNK APPLICATION

MUHAMMAD IRFAN BIN MOHD JASMI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF IOT BASED MONITORING LAUNDRY SYSTEM USING NODEMCU ESP32 AND BLYNK APPLICATION

MUHAMMAD IRFAN BIN MOHD JASMI

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

Faculty of Electronics and Computer Engineering and Technology

Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek	:	Development of IoT Based monitoring Laundry System Using NodeMCU ESP32 And Blynk Application
Sesi Pengajian	:	2024

Saya **MUHAMMAD IRFAN BIN MOHD JASMI** mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hak milik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓)

☐	SULIT*	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia sebagaimana yang termaktub dalam AKTA RAHSIA RASMI 1972)
☐	TERHAD*	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
✓	TIDAK TERHAD	

Disahkan oleh:

RAFIHAH BINTI MOHD ZAIN
Jurutera Pengajar
Jabatan Teknologi Kejuruteraan
Fakulti Teknologi dan Kejuruteraan
Elektronik dan komputer (FTKEK)
Universiti Teknikal Malaysia Melaka

....	....
(TANDATANGAN PENULIS)	(COP DAN TANDATANGAN PENYELIA)
Alamat Tetap:	
Tarikh: 12th January, 2025	Tarikh: 12th January, 2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this report entitled "Development of IoT Based monitoring Laundry System Using NodeMCU ESP32 And Blynk Application" is the result of my own research except for quotes as cited in the references.

Signature	:	....
Student Name	:	Muhammad Irfan Bin Mohd Jasmi
Date	:	12th January, 2025

APPROVAL

I declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours.

Signature	:	....
Supervisor Name	:	Raieha h B inti M o hdZ a in
Date	:	12th January, 2025

Signature	:	....
Co-Supervisor Name (if any)	:	N O T APPLIC AB LE
Date	:	....

DEDICATION

To my lovely parents,

Your unwavering love, support, and encouragement have been my guiding light throughout this project. Every success I achieve is a testament to the values you've instilled in me and the sacrifices you've made for my future. This project is dedicated to you, for being my inspiration and my greatest cheerleaders. Thank you for believing in me, even when I doubted myself. I love you more than words can express.

ABSTRACT

This project focuses on developing an IoT-based monitoring laundry system using NodeMCU ESP32 and the Blynk application. The system enables users to remotely monitor, enhancing convenience and efficiency in the laundry process. The NodeMCU ESP32 serves as the interface between the laundry machines and the internet, allowing users to access them via the Blynk application from anywhere. Key features include the ability to monitor machine usage, and receive alerts for maintenance or issues, saving time and energy and improving overall efficiency. This project showcases the potential of IoT technology to streamline household tasks and improve everyday life.

ABSTRAK

Projek ini bertujuan untuk membangunkan sistem dobi berasaskan IoT menggunakan NodeMCU ESP32 dan aplikasi Blynk. Sistem ini membolehkan pengguna untuk memantau, meningkatkan keselesaan dan kecekapan dalam proses cucian pakaian. NodeMCU ESP32 berperanan sebagai antara muka antara mesin dobi dan internet, membolehkan pengguna mengaksesnya melalui aplikasi Blynk dari mana-mana tempat. Ciri utama sistem ini termasuk keupayaan untuk menjadualkan kitaran cucian dari jauh, memantau penggunaan mesin, dan menerima notifikasi untuk penyelenggaraan atau masalah, yang dapat menjimatkan masa dan tenaga serta meningkatkan keseluruhan kecekapan. Projek ini menunjukkan potensi teknologi IoT untuk menyederhanakan tugas harian di rumah dan meningkatkan kualiti hidup secara keseluruhan.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to Madam Raiehah binti Mohd Zain, my project supervisor, for her invaluable guidance, unwavering support, and insightful feedback throughout the development of this project. Her expertise and dedication have been instrumental in shaping this work and enhancing its quality. I am truly grateful for her mentorship and for always pushing me to strive for excellence. Thank you, Madam Raiehah, for your patience, encouragement, and for believing in me for completion this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

C_{ox}	-	Gate-oxide capacitance
C_d	-	Depletion layer capacitance
SiON	-	Silicon oxynitride

LIST OF ABBREVIATIONS

ADC	-	Analogue-to-digital Converter.
ALU	-	Arithmetic Logic Unit.
IoT	-	Internet Of Things.
LMS	-	Laundry Monitoring Service.
SoC	-	System-On-Chip.
IMU	-	Inertial Measurement Unit.
QR	-	Quick Respond code.
SS	-	Soft Sensor.
WD	-	Washer-Dryer.
GPIO	-	General Purpose Input/Output.
IR	-	Infra Red sensor.
LCD	-	Liquid Crystal Display.
WiFi	-	wireless fidelity.
UART	-	universal asynchronous receiver / transmitter.
ADC	-	analogue-to-digital converter.
DAC	-	digital-to-analogue converter.
PWM	-	Pulse width modulation.
IDE	-	Integrated Drive Electronics.

CHAPTER 1

INTRODUCTION

The project describes the project background, problem statement, project objective and scope of project work for development of IoT based monitoring laundry system using NodeMCU ESP32 and Blynk Application.

1.1 Background

The implementation of an IoT-based monitoring laundry system has a significant impact on sustainable development by promoting environmental, economic, and social sustainability. Traditional laundry systems often result in resource wastage due to the lack of real-time monitoring and control. IoT technology optimizes machine usage, conserving energy and water. Economically, this system reduces operational costs and provides valuable data analytics. Socially, it improves user experience and safety through automated monitoring. In essence, IoT-based smart laundry systems drive sustainable development by making laundry management more efficient, convenient, and environmentally friendly while emphasizing the vital role of innovative engineering practices in fostering a more sustainable future.

1.2 Problem Statement

These systems rely heavily on manual oversight and coordination, leading to a range of issues. Firstly, the lack of real-time monitoring means that users must physically check for machine availability, leading to long wait times and inconvenience, especially during peak usage hours. This manual approach also results in suboptimal machine utilization, as machines may be left idle or overloaded, wasting resources such as water, electricity, and detergent.

Furthermore, the lack of feedback mechanisms means that users are often unaware of the status of their laundry, leading to uncertainty and potential conflicts over machine usage. The reliance on manual monitoring also incurs higher operational costs, as it requires dedicated personnel to manage and maintain the laundry facilities. Additionally, safety and security risks exist, as the absence of automated monitoring can lead to hazards from malfunctioning machines or unauthorized access to laundry facilities.

Addressing these challenges requires the development of an IoT-based laundry system that can automate machine monitoring, provide users with real-time status updates through a mobile application. Such a system would not only improve the efficiency and convenience of laundry management but also reduce operational costs and enhance safety and security. By developing a monitoring laundry system using NodeMCU ESP32 and the Blynk application, this project aims to address these challenges and revolutionize the way laundry facilities are managed and utilized.

1.3 Project Objectives

The main aim of this project is :

1. To develop a real-time monitoring system of laundry machines for self-service laundry shops, including sensors for detecting machine status.
2. To integrate the monitoring system with mobile application and provide users with real-time status updates and notifications.
3. To demonstrate the project efficient system for the users convenience and optimal machine utilization.

1.4 Scope of Project

The project scope of developing is as follows:

The project scope includes developing hardware interfaces using NodeMCU ESP32 and Blynk application to monitor laundry machines . A mobile application will be created using the Blynk platform to provide users with real-time updates . A backed system will give notifications to user . Testing will be conducted in a real-world environment, and documentation and support will be provided for deployment and maintenance. The project aims to improve resource efficiency, user experience, and operational costs in laundry management.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A literature review represent about past related project that are related or similarly with my project. At this stage of project planning process, the project's aim and objective are taken into account together with the raw resources required to finish the project.

2.2 Past Related Project

2.2.1 Design of IoT-Based Smart Laundry Applications Using Fuzzy Algorithms

This research aims to introduce innovative ideas in laundry services to support the development of smart cities [1]. It focuses on IoT-based Smart Laundry web applications that can simplify the use of laundry, making it easier and more practical for users. The implementation of the Fuzzy Algorithm, an Artificial Intelligence method, will support decision-making in the system. This will aid in sorting which laundry will be picked up and will also be used to determine the price of laundry to be paid by the user based on parameters such as Figure 2.1.

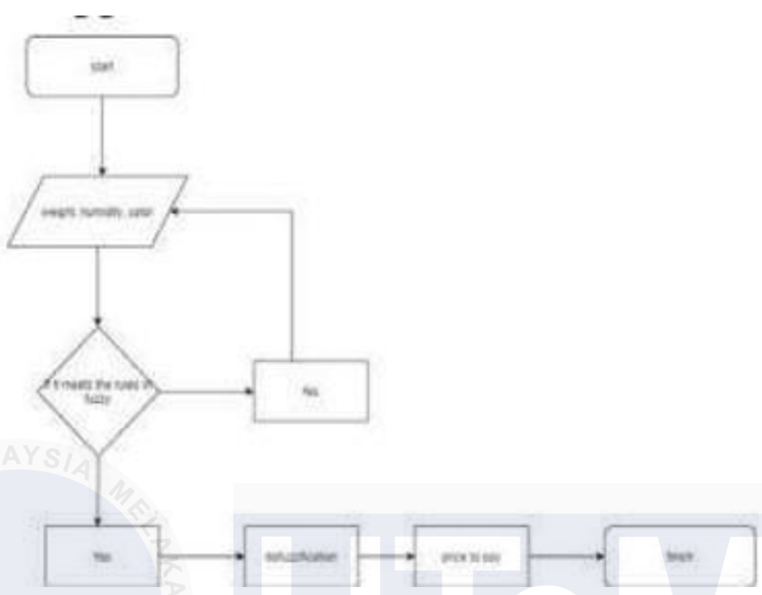


Figure 2.1: Flowchart for this Porject

2.2.2 Design of Laundry Box as Supporting Smart Laundry System based on Internet of Thinks

”In a recent study, Rizki *et al.* [2] designed the laundry box to help customers save time by allowing them to store their clothes without having to make trips to the laundromat. Figure 2.2 is equipped with several sensors, including a color sensor, a weight sensor, and a humidity sensor. The color sensor detects whether the clothes in the box are white (output 0) or not white (output 1). The humidity sensor monitors the moisture level of the clothes to prevent them from becoming too moldy. The weight sensor measures the weight of the clothes in grams. All the sensors send real-time data to Antares.”

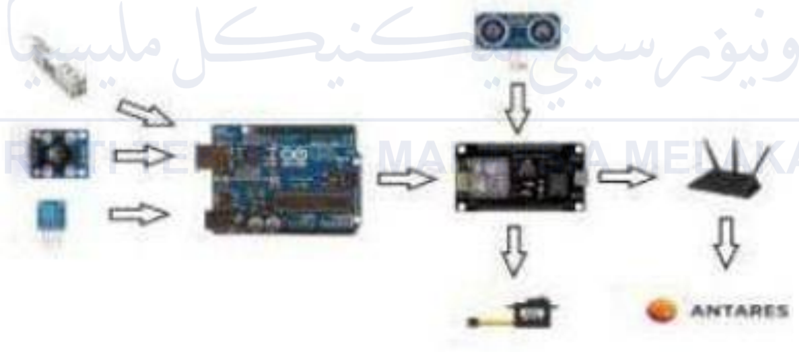


Figure 2.2: Block Diagram For Laundry Box

2.2.3 Web based application to search and manage Laundry shops

This study introduces web apps as a solution to the challenges faced by laundromats and their customers.According to [3], Customers who use laundry services will benefit from the proposed web app, as it will help them find nearby laundry services and easily leave reviews. This web application will also enable people to locate the best laundry shops in their area based on customer reviews and comments. Additionally, laundry shop owners can use the web app to advertise their services, discounts, and offers. The app can also help manage customers' laundry items, bills, and prevent mix-ups. Both customers and laundromat owners will need to sign up for the web app in order to use its features, such as searching for laundromats, adding items to a shopping basket, and providing feedback on their laundry experience.Figure 2.3 shows the flowchart about how the program work.

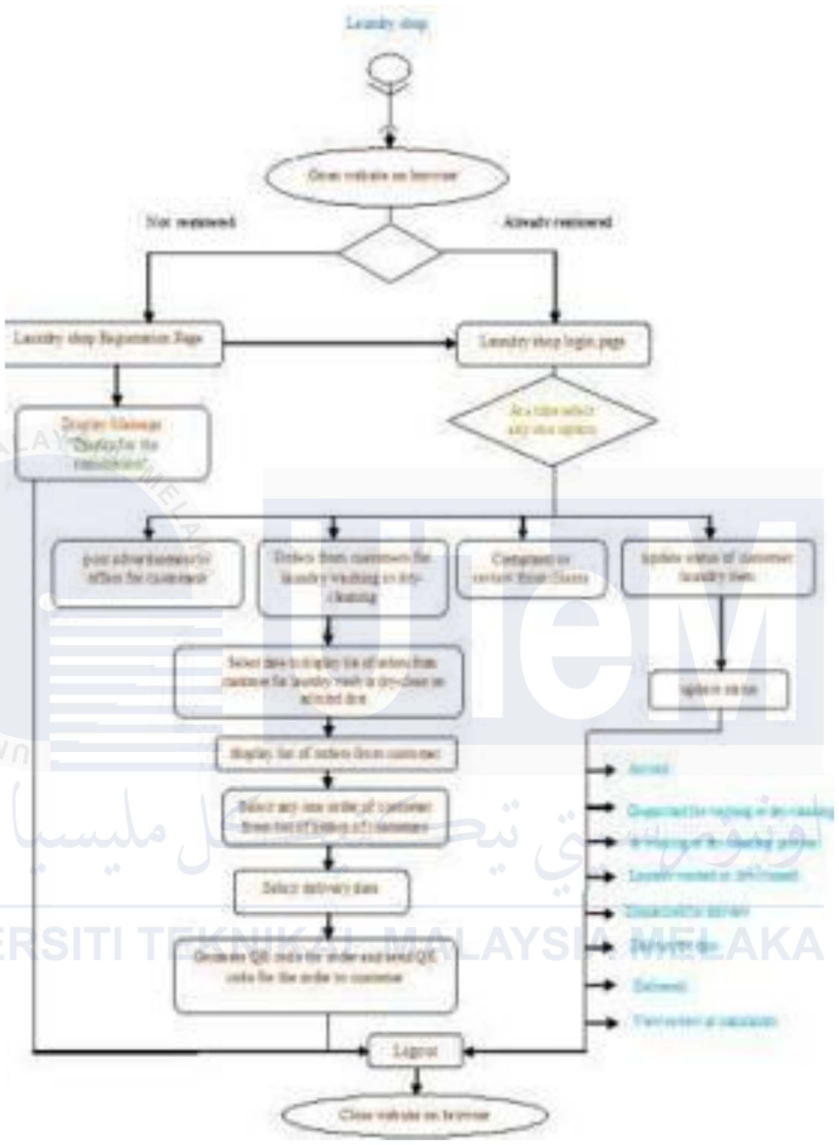

Laundry shop
Open website on browser
Not registered
Already registered
Laundry shop Registration Page
Laundry shop login page
Display Message
Orders from customers for laundry washing or dry-cleaning
Complaint or review from clients
Update status of customer laundry item
Select date to display list of orders from customer for laundry wash or dry-clean on selected date
Update status
Display list of orders from customer
Select any one order of customer from list of orders of customer
Select delivery date
Generate QR code for order and send QR code for the order to customer
Delivered
Logout
Close website on browser

Figure 2.3: Flow Diagram

2.2.4 Implementation of Naive Bayes Algorithm on IoT Based Smart Laundry Mobile Application System

The goal of this study [4] is to help customers with busy schedules, particularly in urban areas, who may not have time to visit the laundromat frequently. This Android application, based on IoT (Internet of Things), is designed to provide customers with automatic notifications, eliminating the need for manual processing. The laundry staff will automatically pick up the clothes by utilizing the Na¨ıve Bayes algorithm, which assesses the weight, distance, and moisture levels of the clothing to prioritize their processing order. Customers can track their clothes' progress and make payments directly through the smartphone. Moreover, in Figure 2.4, the laundry staff can easily identify and pick up customer clothing as there is a prepared list of customer names for processing.

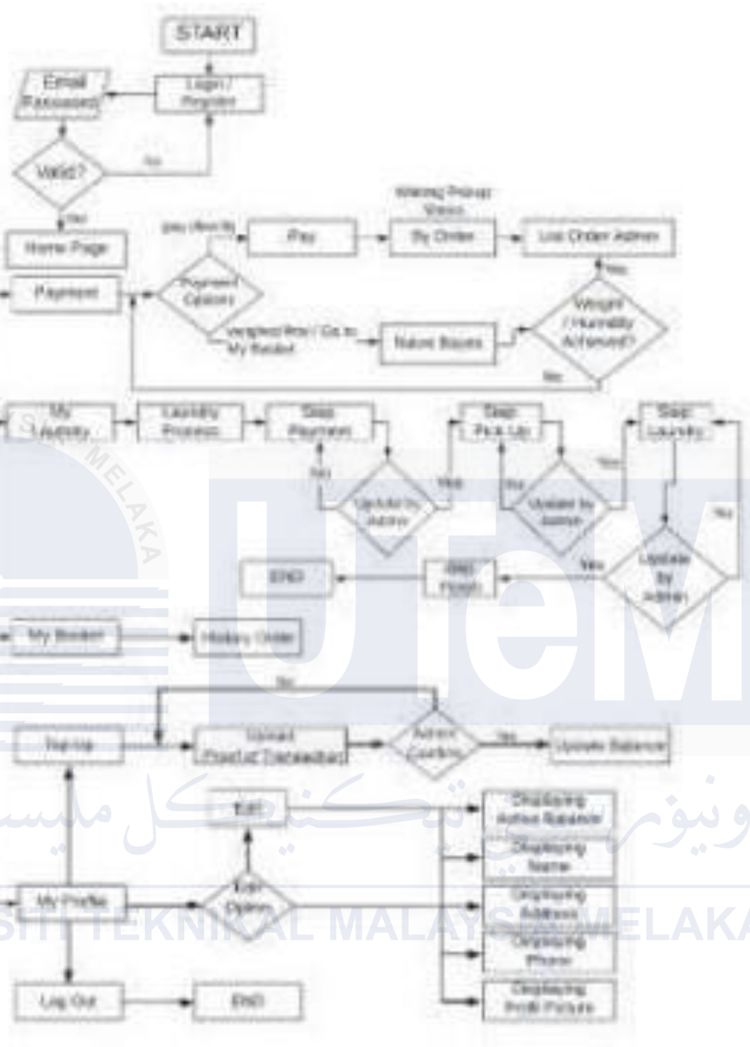


Figure 2.4: Application Flowchart

2.2.5 Monitoring the Status of Self-Operated Community Laundry Machines using IoT integration

This study outlines an Internet of Things automation system that uses cell phones to update consumers about the status of washing machines. A cell phone is attached to the rear panel of a washing machine or dryer to remotely detect changes in the magnetic field. The phone's built-in sensors provide data, which is collected and analyzed using computational techniques by a customized program [5]. This program can accurately determine whether the machine is on or off. Subsequently, the status of each laundry machine (refer to Figure 2.5) is sent to the cloud and made available to all users through mobile applications for iOS and Android. Testing has demonstrated that this system is reliable and suitable for practical use.

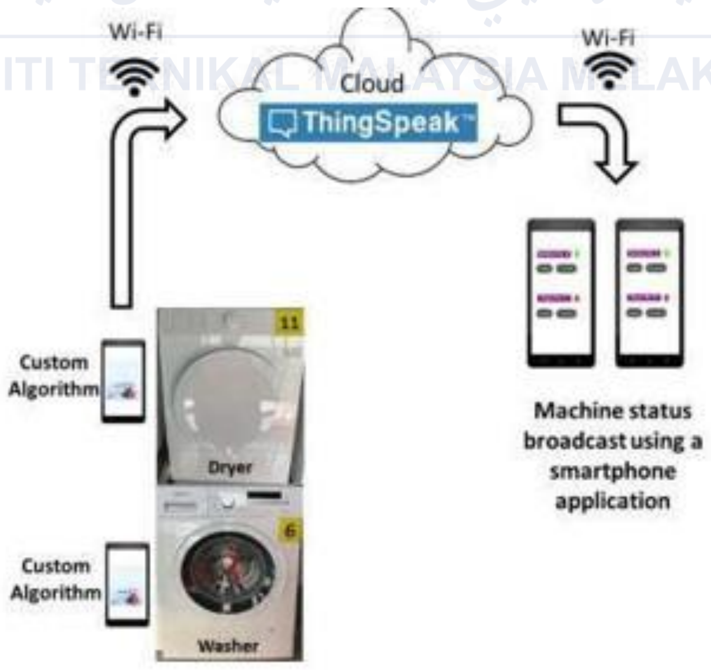


Figure 2.5: Top Level

2.2.6 An Internet of Things System for a Laundry Monitoring Service

Regarding [6], we provide a simple, economical, and novel method for classifying the Laundry Monitoring Service (LMS), which is a conventional washing machine's availability. This method includes listening for vibrations coming from the washing machine using a simple inertial measurement unit (IMU) sensor. Every LMS device has a system-on-chip (SoC) that supports WiFi that allows it to connect to the network. The mean acceleration amplitude along the three axes for a brief period of time, which is a summary statistic, is compared to a predetermined threshold for washing machine states. The suggested LMS system is built on open-source platforms with reasonably priced hardware components, scalable, easy to deploy, and secure for data. The LMS device's operation is verified by the creation and assessment of figures like Figure 2.6.

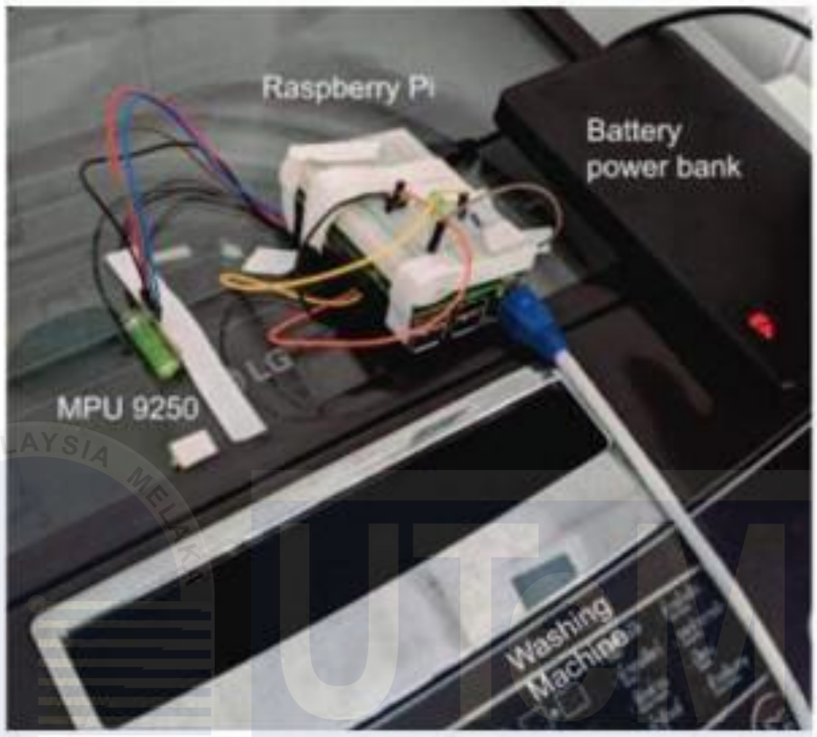


Figure 2.6: Prototype for the project

2.2.7 An IOT-Based Laundry System Application

This approach allows them to visit the store and retrieve their freshly laundered clothes from the service provider. Essentially, the model is an IoT-based Smart Laundry System Application that provides customers with the convenience of online payment and minimizes the need for human interaction.This project's main objective is to develop an Android mobile application for launderettes that will be connected to the Internet of Things (IoT) using ESP8266 and Arduino hardware. Customers will be able to pay for laundry services by scanning a QR code and utilising an e-wallet payment gateway thanks to this application, which is mentioned in [7]. When the money is received, it will link to an ESP8266 device. The mobile application will then direct users to a countdown timer website (see Figure 2.7), where they can track the status of their wash and get a notification when it's almost finished. They may visit the store and pick up their newly cleaned garments from the service provider thanks to this strategy.

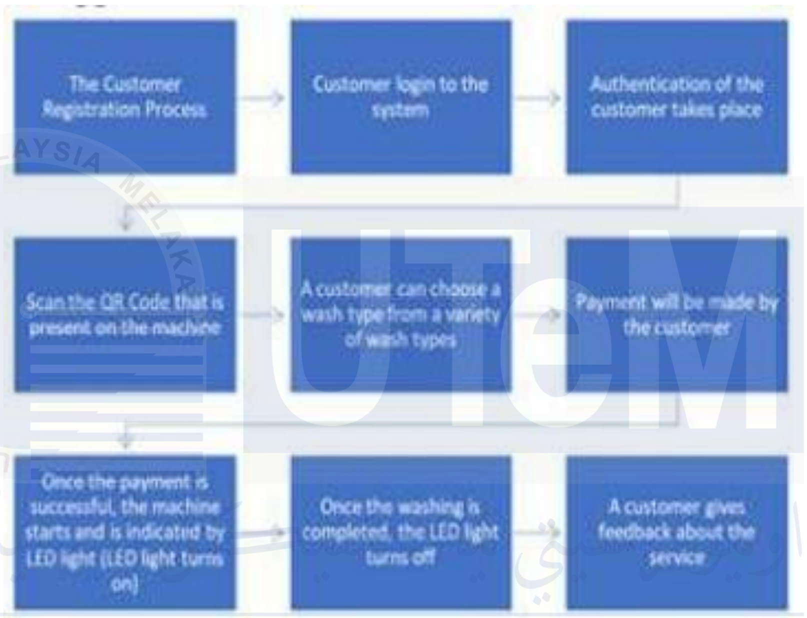

The Customer Registration Process
Customer login to the system
Authentication of the customer takes place
Scan the QR Code that is present on the machine
A customer can choose a wash type from a variety of wash types
Payment will be made by the customer
Once the payment is successful, the machine starts and is indicated by LED light (LED light turns on)
Once the washing is completed, the LED light turns off
A customer gives feedback about the service

Figure 2.7: Step For This Laundry Application

2.2.8 Machine Learning-based Soft Sensor for Laundry Load Fabric Typology Estimation in Household Washer-Dryers.

Fabric care manufacturers are working on creating more energy-efficient and user-friendly products. Based on this [8] The goal is to develop a Soft Sensor (SS) for a household Washer-Dryer (WD) that can differentiate between different fabrics loaded in the machine. Knowing the composition of the load could result in more precise drying, faster processing, and reduced energy consumption without increasing production costs. Additionally, automatically classifying fabric loads will enhance the user experience by requiring less input from users to achieve optimal drying processes. The SS developed in this work utilizes sensors already present in a commercial WD and employs regularization methods and Random Forests for classification from Figure 2.8 standpoint. The effectiveness of this approach has been tested on real data in various conditions.

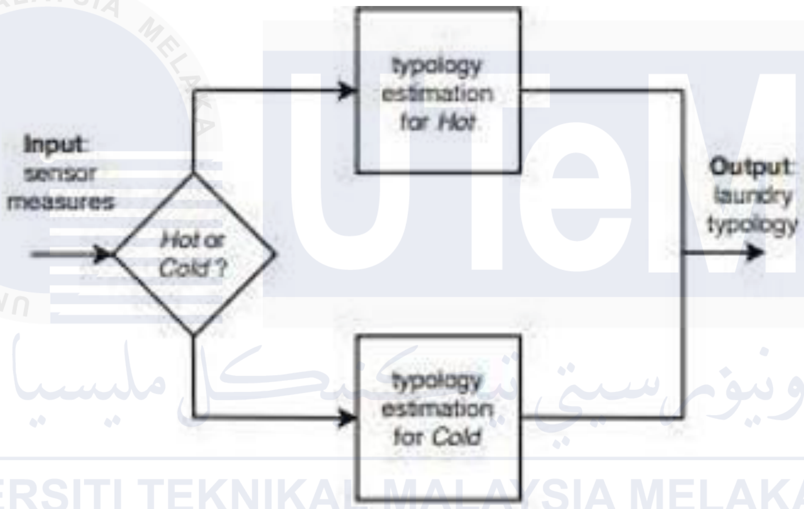


Figure 2.8: Step For This Laundry Application

2.2.9 A Wireless Intelligent Business Laundry Service System.

Network connectivity and wireless transmission are supported by the washing control centre. A research by [9] claims that in order to operate the washing machine, the system wirelessly communicates orders and payment information to the wireless terminal of the machine over the network after receiving them from the WeChat merchant platform.to control the machine's operations.

The MCU, wireless module, control module, power module, and touch screen make up the washing machine wireless terminal. When an order is formed, the wireless module in the laundry control centre sends the order information to the laundry wireless terminal, which relays it to the MCU. The order information includes WeChat Public Accounts, available washing machine numbers, the charge schedule, and the laundry service time. The MCU uses an internal technique to produce a 32-byte random integer, and it uses the MD5 code to construct a signature. It creates a QR code for payment by combining this data. Following this procedure, it uses CRC16 to encrypt the matching answer message, which it then transmits to the laundry control centre while it waits for payment confirmation.

Upon completion of the payment by the customer, the laundry wireless terminal receives the payment confirmation information from the server. The MCU analyzes the information and controls the washing machine through the control module, including the washing mode, time, etc. Simultaneously, the information is displayed to the customer through the touch screen, enabling autonomous laundry service (see Figure 2.9).

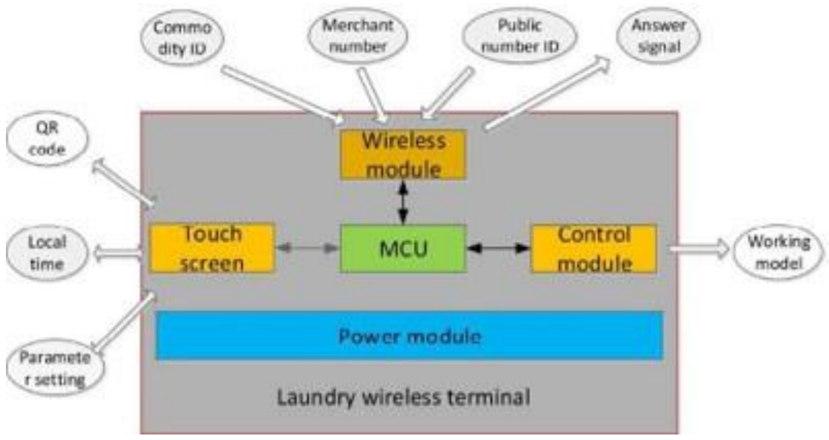


Figure 2.9: Estimation Algorithm Using Divided Approach

2.2.10 Intelligent Washing Machine Driven by Internet of Things.

According to Arivazhakan *et al.* [10], The primary aim of this research is to develop an intelligent washing machine that leverages IoT technologies. This innovative appliance aims to transform traditional washing machines by offering a wide range of advanced features and capabilities. Through the integration of IoT connectivity, sensors, and a master control chip, the intelligent washing machine becomes a highly adaptable device capable of delivering superior washing results . Figure 2.10 shows the core of the intelligent washing machine lies the machine body, housing essential components such as the washing drum, sensors, display screen, and function buttons. The incorporation of sensors within the water inlet and various sections of the washing drum enables precise monitoring of water levels, temperature, and other crucial parameters during the washing process. These sensors provide real-time feedback to the master control chip, ensuring precise and efficient control of the washing machine's operations, thus enhancing user interaction.

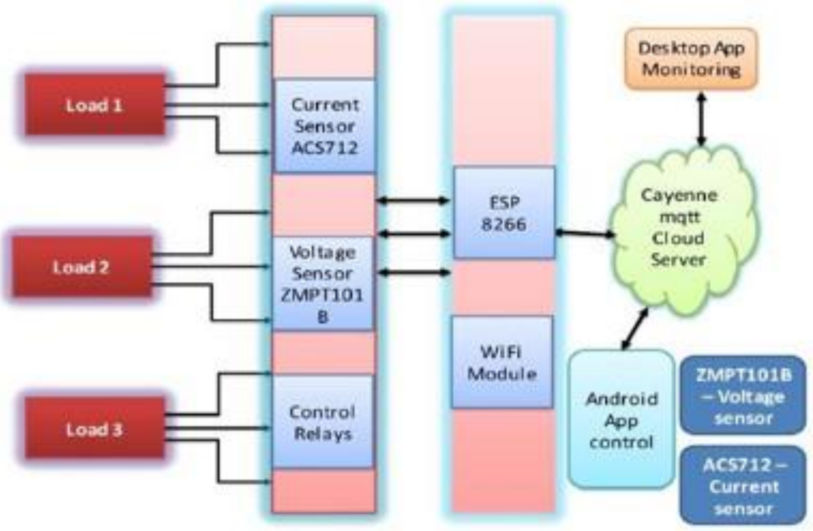


Figure 2.10: Estimation Algorithm Using Divided Approach

Table 2.1: Summary for past related project

Authors	Title	Advantage(s)	Disadvantage(s)
Saleha2020 *et al.*, [1]	Design of IOT-Based Smart Laundry Applications Using Fuzzy Algorithms.	• Time and Cost Efficiency . • Allowing them to easily determine the price to be paid based on the laundry parameters.	• Complex and require specialized knowledge. • Leading to potential errors in decision-making. • Require significant investment in terms of resources, infrastructure, and technology.
Rizki2020 *et al.*, [2]	Design of Laundry Box as Supporting Smart Laundry System based on Internet of Thinks.	• ensuring that the clothes are not moldy due to high humidity. • The Laundry Box provides a practical and efficient solution for customers with busy lives.	• Expensive product.
Singh2022 *et al.*, [3]	Web based application to search and manage Laundry shops.	• The web application is responsive and can be accessed from various devices, offering flexibility to users. • Easily locate nearby laundry shops and access information about their service.	• Concerns about the data security and privacy • The application may face technical glitches or downtime.
Akbar2020 *et al.*, [4]	Implementation of Naive Bayes Algorithm on IoT Based Smart Laundry Mobile Application System .	• Convenience for customers with high mobility . • Notifications for customers .	• Leading to potential issues if the system malfunctions .
Menachery2021 *et al.*, [5]	Monitoring the Status of Self-Operated Community Laundry Machines using IoT integration.	• Making it a cost-effective and widely available solution . • Centralized data acquisition and monitoring .	• The accuracy and reliability of the system are contingent on the quality. • Difficult for user to understand .

Continued from previous page . . .

Authors	Proposed Technique	Advantage(s)	Disadvantage(s)
Shakya2021 *et al.*, [6]	An Internet of Things System for a Laundry Monitoring Service.	• Cost-effective hardware components, making it an affordable solution for monitoring the status of traditional washing machines . • Allowing or easy expansion and integration with multiple washing machines.	• System could be complex .
Sai2022 *et al.*, [7]	An IOT-Based Laundry System Application.	• Enables customers to pay for laundry services through QR code scanning and e-Wallet. • Allowing them to pick up their clothes at the right time.	• The laundryman benefits more than the customers because the cost decision is in their hands. • There is an additional fee for delivery by a third party.
Susto2019 *et al.*, [8]	Machine Learning-based Soft Sensor for Laundry Load Fabric Typology Estimation in Household Washer-Dryers.	• lower energy consumption without increasing production costs. • Improve process handling or late to enhance end-cycle detection.	• Inaccurate information about the laundry load. • Requesting information from the user may be considered inconvenient and generate discomfort.
Qiao2019 *et al.*, [9]	A Wireless Interlligent Business Laundry Service System.	• low energy consumption • strong stability.	• Only user that have Wechat can make it thru online.
Arivazhakan2022 *et al.*, [10]	Intelligent Washing Machine Driven by Internet of Things.	• Users can customize washing modes. • providing users with an advanced and personalized washing experience.	• Complexity. • security concerns that users need to be aware of and address.

2.3 Summary

This literature review indicates that most projects have used the ESP8266 and Raspberry Pi for IoT-based laundry systems. In contrast, my project utilizes the ESP32 microcontroller due to its numerous advantages: it offers more GPIOs with multiple functions, faster Wi-Fi capabilities, and Bluetooth support. Although the ESP32 is considered more complex and challenging to handle compared to the ESP8266, its enhanced features provide significant benefits for the application. Moreover, while previous projects used humidity sensors, my project incorporates IR sensors and load sensors to better manage washing machines, providing a more comprehensive and efficient solution. for next chapter i will show about methodology that contain hardbare , software, block diagram and flowchart.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this section, we will discuss the major aspects of the project, as well as the methods and strategies that were utilised to ensure that the project requirement were met. This section provides a short explanation of the steps that need to be taken, in addition to an explanation of the primary components. This section is very important for determining whether the project step in this section is whether the system is works smoothly in accordance with what is anticipated.

3.2 Methodology

The main goal is to develop Development of IoT Based Monitoring Laundry System Using NodeMCU ESP32 And Blynk Application. This project's technique entails selecting a suitable title, carrying out research and reading journals, organising and developing the work, setting up hardware and software, putting the process into practice, testing it, and producing a report. To guarantee that this activity and process go well and that the project is completed according to schedule, all of these elements are required. The goal of the task inquiry is to familiarise oneself with the startup and acquire comprehension of the procedures. The next stage of the procedure is to identify the parts that will be used in the circuit's construction and programming, such as the hardware and programming.

Testing is the procedure that is used to guarantee that the circuit and the programmed are useful.

3.3 Block Diagram

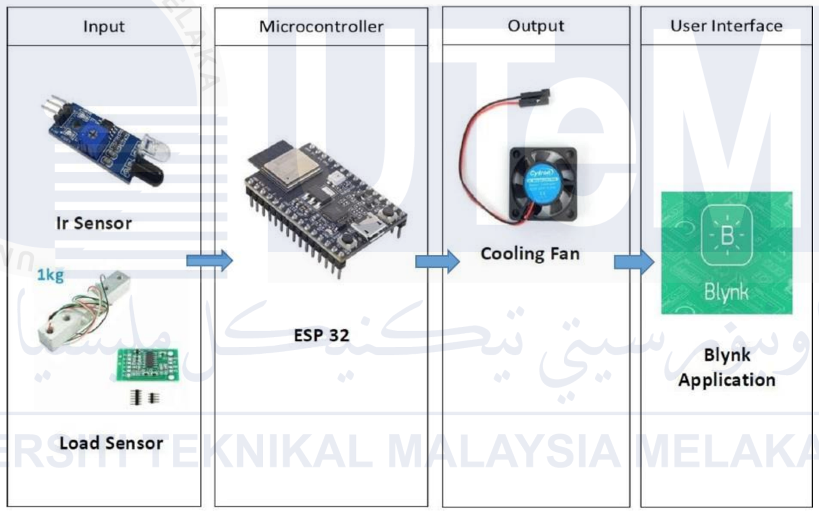


Figure 3.1: Block Diagram for Monitoring Laundry

Figure 3.1 shows the operation of an IoT-based monitoring laundry system using NodeMCU ESP32. The system comprises input devices including an IR sensor and a load sensor, which detect the status and weight of the laundry, respectively. These sensors transmit data to the NodeMCU ESP32 microcontroller, which processes the information. The microcontroller is linked to an LCD for real-time data display and controls a fan that mimics the washer machine blade. Additionally, the NodeMCU ESP32 interacts with the Blynk application, enabling remote monitoring and control via a mobile app. In summary, the system allows users to efficiently manage laundry operations using both local displays and remote interfaces.

3.4 Hardware

Table 3.1 shows the list for component that i used in my project. The table also share about price for each component and quantity component that i will need.

Table 3.1: List of materials that required in Developing the Prototype

No	Component	Price	Quantity	Total
1	NodeMCU ESP32	RM 36.98.00	1	RM36.98
2	Load Sensor	RM 15.00	3	RM 45.00
4	IR Sensor	RM 1.90	3	RM5.70
5	Power Supply Module	RM 7.90	1	RM7.90
6	Relay 5v	RM1.15	3	RM3.45
7	cooling fan 12v	RM 14.85	3	RM 48.74
	Total Spent			RM147.77

3.4.1 NodeMCU ESP32

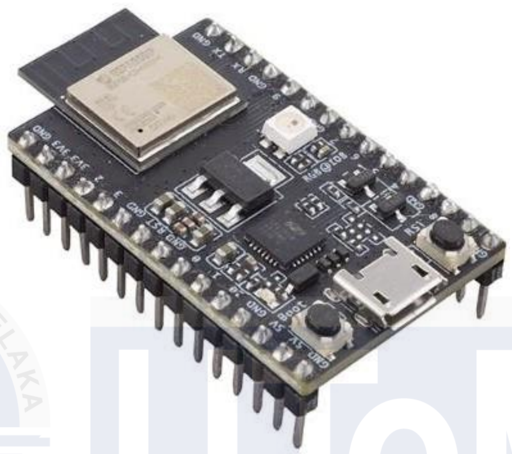


Figure 3.2: NodeMCU ESP32

The ESP32 is a dual-core SoC with two powerful Xtensa LX6 CPUs that can operate at up to 240 MHz, as shown in Figure 3.2. It is appropriate for a variety of Internet of Things applications that need wireless connectivity since it has integrated Wi-Fi and Bluetooth connectivity. The well-liked ESP8266 chip was replaced with the ESP32, which has many additional features and capabilities.

The ESP32 is also equipped with a number of peripherals, including SPI, I2C, UART, ADC, DAC, PWM, and GPIO. This versatility allows it to be used for a wide range of applications, from demanding jobs like MP3 decoding and speech encoding, to low-power sensor networks.

The low power consumption of the ESP32 is one of its biggest benefits. With its deep-sleep mode, which may lower power consumption to as little as 5 microamps, it is the perfect choice for battery-powered devices. The ESP32 is extremely simple to power and use because to its integrated voltage regulator, which runs on a broad variety of power supply, from 2.2V to 3.6V.

Table 3.2: Names of NodeMCU development boards

Series	Identifier	Processor speed (MHz)
ESP32	ESP32-S0WD	160
	ESP32-D0WD-V3	240
	ESP32-D0WDR2-V3	240
	ESP32-U4WDH	240

Table 3.3: Detail of NodeMCU ESP32

No	Specification	ESP32
1	Processor	Tensilica LX6 Dual-Core
2	Memory	4MB
3	Frequency	2.4GHz
4	Operating Voltage	3.3 V (operable via 5 V microUSB)
5	Operating Temperature	–40°C – 125°C
6	Dimensions	48 x 26 x 11.5 mm
7	Weight	10 g

3.4.2 Load Sensor

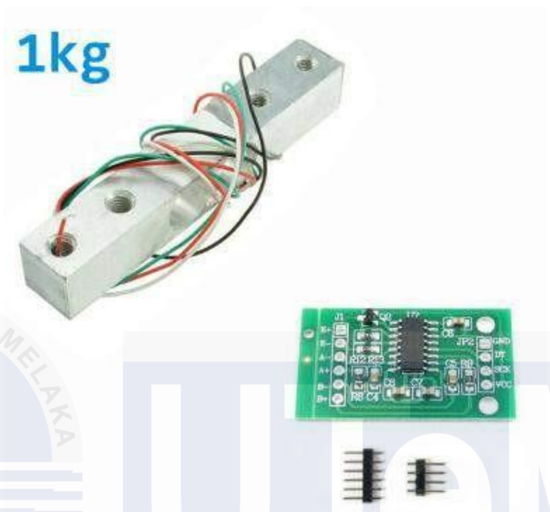


Figure 3.3: Load Sensor

A load sensor, commonly known as a ”load cell,” is an electronic device that converts tension and compression forces into an electrical signal. Load sensors are typically used to measure the weight of an object, such as in household or industrial scales, and to quantify tension in pulley cables and ropes. While designs and functions can vary among load sensors, they all measure resistance and/or deformation within the sensor to determine the magnitude of tension and compression forces. The manufacturing, medical, grocery, and automotive industries all benefit from load sensor technology.

In Figure 3.3, the load sensor is strategically placed to detect the weight of the laundry, sending this data to the NodeMCU. The ESP32 processes this information and communicates with the Blynk application, allowing users to monitor the load in real time. This functionality helps prevent overloading, ensure optimal washing conditions, and enhance overall efficiency. The integration of the load sensor with the NodeMCU and Blynk app provides a seamless, smart solution for managing laundry loads, making the system more user-friendly and effective.

3.4.3 Power Module Supply

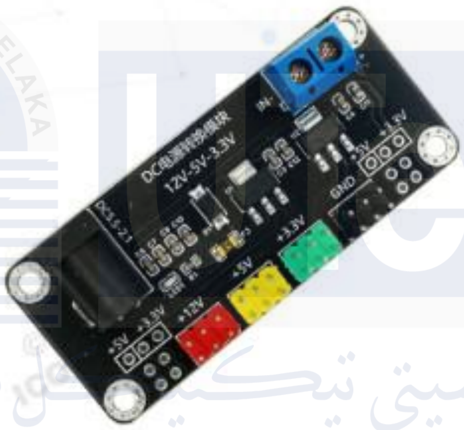


Figure 3.4: DC Power module

The DC-005 Power Supply Module plays a crucial role in ensuring a stable and reliable power supply for the ESP32 microcontroller and other connected components. This module allows you to conveniently connect an external DC power adapter, typically ranging from 6V to 12V, which can then be regulated and supplied to the ESP32. The DC-005 module's compact design makes it easy to integrate into system, whether working with a breadboard or a custom PCB.

Figure 3.4 shows the system gains flexibility in terms of power input, as it can seamlessly interface with commonly available DC adapters. This ensures consistent operation of ESP32, sensors, and actuators, while also minimizing the risks of power interruptions that could impact the functionality of the Blynk-based monitoring system. Additionally, it simplifies prototyping and deployment, as the VCC and GND terminals provide a straightforward way to distribute power across the system. The reliable power supply facilitated by the DC-005 module is essential for maintaining stable communication between the ESP32 and the Blynk application, ensuring smooth data monitoring and control.

3.4.4 IR sensor

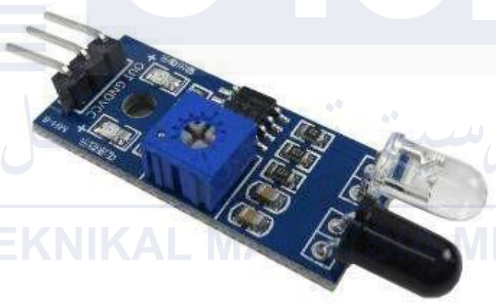


Figure 3.5: IR sensor Module

The IR sensor is an electronic device that emits light to detect objects in its surroundings. It can measure an object's heat and detect motion using Figure 3.5. In the infrared spectrum, all objects emit thermal radiation that is invisible to our eyes but detectable by infrared sensors.

The emitter is an IR LED (Light Emitting Diode), and the detector is an IR photodiode. The photodiode is sensitive to the same wavelength of IR light emitted by the IR LED.

When IR light falls on the photodiode, the resistance and output voltages change proportionally to the magnitude of the IR light received.

A typical infrared detection system consists of five basic elements: an infrared source, a transmission medium, an optical component, infrared detectors or receivers, and signal processing. Infrared lasers and infrared LEDs of specific wavelengths are used as infrared sources.

3.4.5 Relay 5V

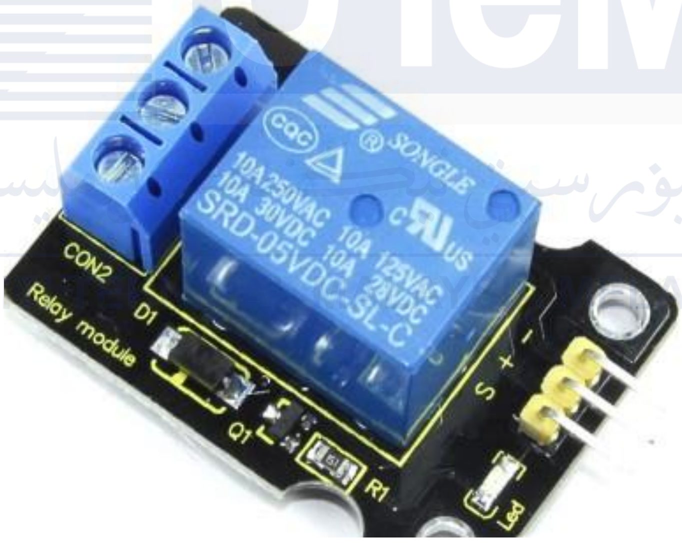


Figure 3.6: Relay 5V

A 5V relay module is used to control the operation of two washers and dryer in the IoT laundry system. Figure 00 shows a relay acts as an electrically operated switch that allows the ESP32 microcontroller to turn high-power devices, such as the washers, on and off using low-power control signals. Figure 3.6 are connected to GPIO pins 18 and 19 of the ESP32 and are controlled by sending a HIGH signal (5V) to activate the relay and turn the washer ON, or a LOW signal (0V) to deactivate the relay and turn the washer OFF. This control ensures the washers operate only when specific conditions are met, such as the detection of laundry weight and a closed door.

The relay module's electrical isolation, often provided by an optocoupler, safeguards the ESP32 from high voltages or current spikes in the washer's power circuit. During setup, the relay pins are configured as outputs, and their initial state is set to LOW to keep the washers OFF at startup. The relays are powered using a 5V supply, with their signal pins connected to the ESP32, ensuring compatibility with the microcontroller's logic levels. The washer circuit is connected to the relay's common (COM) and normally open (NO) terminals, so the circuit is completed and the washer is powered only when the relay is activated.

By using a 5V relay, the system can safely and effectively manage high-power devices, providing a reliable mechanism for automating washer control while maintaining the safety and integrity of the low-power ESP32 circuit.

3.4.6 Cooling fan 12V

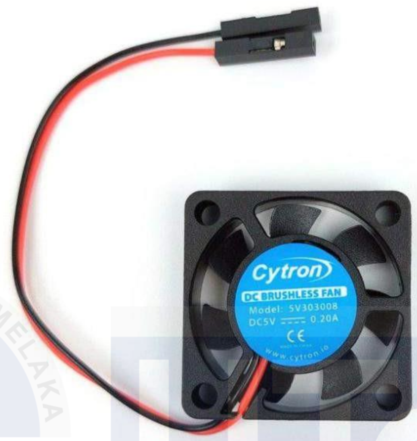


Figure 3.7: Cooling Fan 12V

A cooling fan is typically used in electronic devices to help dissipate heat, comprising a fan blade, a motor, and housing that directs airflow. These fans are commonly found in computers, power supplies, and other electronic equipment, where they play a crucial role in maintaining optimal operating temperatures and preventing overheating. Figure 3.7 , I repurposed a 12V cooling fan to act as the blade of a washing machine. This setup likely utilizes the fan's motor to drive the movement of the blades, enabling it to simulate the agitation or spinning action needed for washing clothes. By leveraging the cooling fan's compact size, ease of power supply, and consistent rotation speed, you've effectively created a functional component for your smart laundry system, demonstrating both resourcefulness and technical ingenuity.

3.5 Software

3.5.1 Blynk Application

Figure 3.8: Blynk

The Blynk application is a platform designed to control and monitor IoT devices via a smartphone. It works by allowing users to create custom interfaces using a variety of widgets like buttons, sliders, and graphs, which can then communicate with the hardware through the Blynk server. In my project, Blynk facilitates the interaction between the user and the smart laundry system. By using the Blynk app, you can monitor the status of the laundry process, and receive notifications about the laundry's progress. This integration not only enhances the user experience by providing remote access and control but also makes the system more convenient and efficient, aligning with the goals of modern IoT applications.

3.5.2 Arduino IDE Application

Figure 3.9: Arduino IDE

The Arduino Integrated Development Environment (IDE) is a versatile and user-friendly platform used for developing and programming microcontroller-based projects. It provides an intuitive interface where users can write, compile, and upload code to their microcontroller boards, such as the NodeMCU ESP32 used in my IoT-based smart laundry system project.3.9

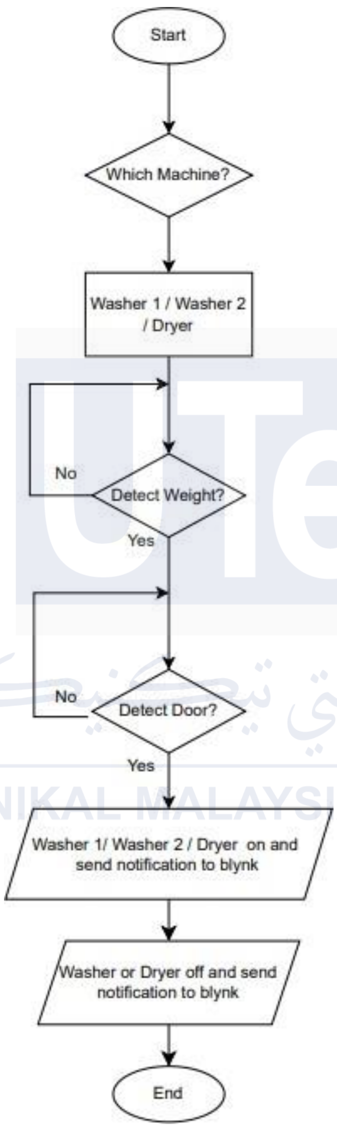
The Arduino IDE supports the C and C++ programming languages, offering a simplified syntax and a rich library ecosystem that makes it easier to interface with various sensors, actuators, and communication modules. The IDE includes a text editor for writing code, a message area for displaying errors and feedback, a text console, and a tool bar with buttons for common functions such as verifying and uploading code.

3.5.3 Fritzing

Figure 3.10: Fritzing Logo

3.10 is an open-source software application designed to help users create, document, and share electronic circuit designs. It provides an intuitive and visually appealing interface for designing circuits in three main views: Breadboard View, Schematic View, and PCB Layout View. The Breadboard View allows users to replicate the physical layout of a circuit on a virtual breadboard, showing how components like sensors, microcontrollers, and power modules are connected. The Schematic View generates a standard circuit diagram, which can be refined for documentation or analysis. The PCB Layout View enables users to design printed circuit boards (PCBs) for manufacturing.

3.6 Flowchart



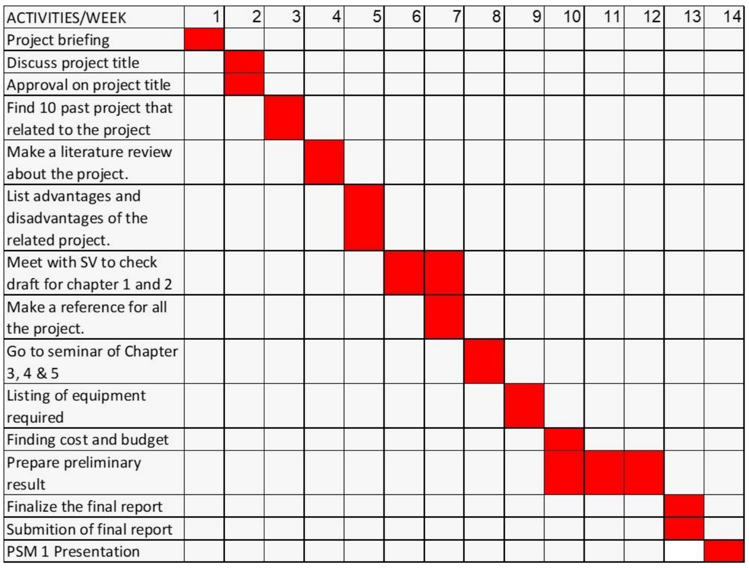
Start
Which Machine?
Washer 1 / Washer 2 / Dryer
No
Detect Weight?
Yes
No
Detect Door?
Yes
Washer 1/ Washer 2 / Dryer on and send notification to blynk
Washer or Dryer off and send notification to blynk
End

Figure 3.11: Flowchart for this project

As where, Figure 3.11 process of an IoT-based monitoring laundry system. It begins with the system being initiated and updating its status to the Blynk application, informing the user that it is ready. The system then checks if the door is closed which Ir sensor 1 will attach to the door. If the door is not closed, it waits until the door is properly closed before proceeding. Once the door is confirmed to be closed, the system checks if there is enough token, which likely represents a form of payment or authorization, to start the machine. If there is not enough token, the system will not proceed until the required token is present.

Next, the system verifies if the load is less than or equal to 1 kg. If the load exceeds 1 kg, the system will not start until the load is reduced to the acceptable limit fo safety machines. Once all conditions are met, the machine starts the washing or drying process. During the operation, the system continuously checks if the wash or dry cycle is finished. If the cycle is not yet complete, it keeps running. When the cycle is completed, the system sends a notification to the user via the Blynk application which states the ”laundry have finish” , and the process ends.

3.7 Gantt Chart



ACTIVITIES/WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project briefing	🟥													
Discuss project title		🟥												
Approval on project title		🟥												
Find 10 past project that related to the project			🟥											
Make a literature review about the project.				🟥										
List advantages and disadvantages of the related project.					🟥									
Meet with SV to check draft for chapter 1 and 2						🟥	🟥							
Make a reference for all the project.							🟥							
Go to seminar of Chapter 3, 4 & 5								🟥						
Listing of equipment required									🟥					
Finding cost and budget										🟥				
Prepare preliminary result										🟥	🟥	🟥		
Finalize the final report													🟥	
Submition of final report													🟥	
PSM 1 Presentation														🟥

Figure 3.12: Gantt Chart

3.8 Summary

The primary objective of this project is to develop an IoT-based Monitoring Laundry system using NodeMCU ESP32 and the Blynk application. The project involves several key steps, including research, hardware and software set-up, implementation, testing, and documentation. The goal is to create a functional system that enhances user convenience and efficiency in laundry management through monitoring provided by the Blynk application.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, which is result and discussion ,the analysis will be made according to data that have been collected and by observation of the development of IoT based Monitoring Laundry system using ESP32 and Blynk application. The hardware designd and software that have been implemented were explained. the function of this system will be tested using the prototype form and explained. several limitation of this project also were discuss in this chapter.

4.2 Schematic Diagram and Wiring Diagram

In the beginning, the project is construct using fritzing software to visualize how the connection between NodeMCU, IR sensor, Weight sensor, DC-module supply, relay 5v, and DC fan 12V. Figure 4.1 shows the connection between each hardware is connected using jumper wire .The schematic diagram represents an IoT-based monitoring system built around the ESP32 microcontroller, integrating several sensors and relay modules to automate and monitor a laundry system. At the heart of the system, the ESP32 serves as the central controller, handling inputs from sensors and managing outputs through relay modules. The system uses IR sensors specifically to detect whether the door of the washer or dryer is open or closed, ensuring safety by preventing the machine from running if the door is open. Additionally, weight sensors, implemented with load cells and HX711 amplifier modules, are used to detect the presence of objects in the washer or dryer. These sensors send signals to the ESP32 to determine if laundry is present, and the system notifies the user through the Blynk application when it detects laundry.

Relay modules are connected to the ESP32 to control external devices, such as motors or fans simulating washers and dryers. When the IR sensor detects that the door is closed and the weight sensor confirms the presence of laundry, the ESP32 activates the corresponding relay to operate the motor or fan. All components, including sensors and relays, share a common ground and are powered by the ESP32 or an external power source via a DC-005 module. Notifications and remote control are managed through the Blynk application, allowing users to monitor and control the system from their devices. This setup ensures efficient and automated operation while maintaining safety and providing user-friendly functionality.

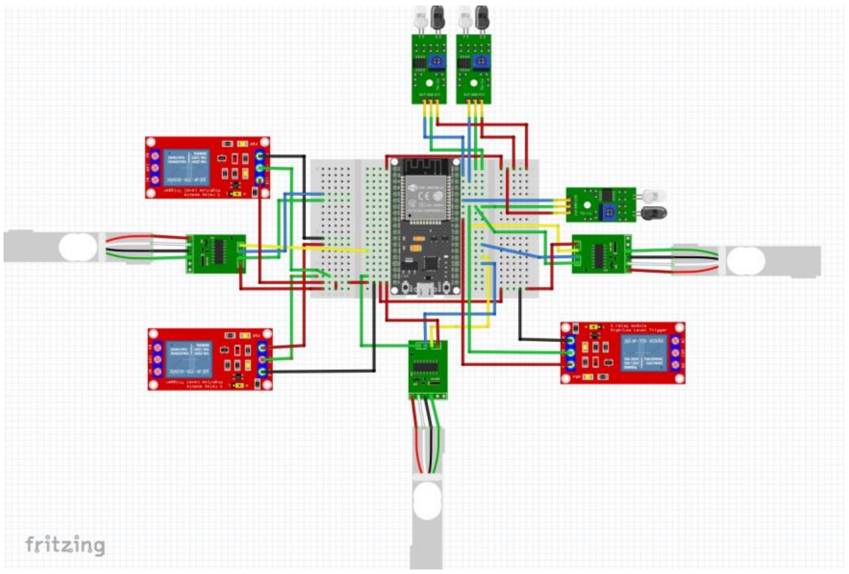


Figure 4.1: schematic diagram for this project

4.2.1 Hardware Implementation

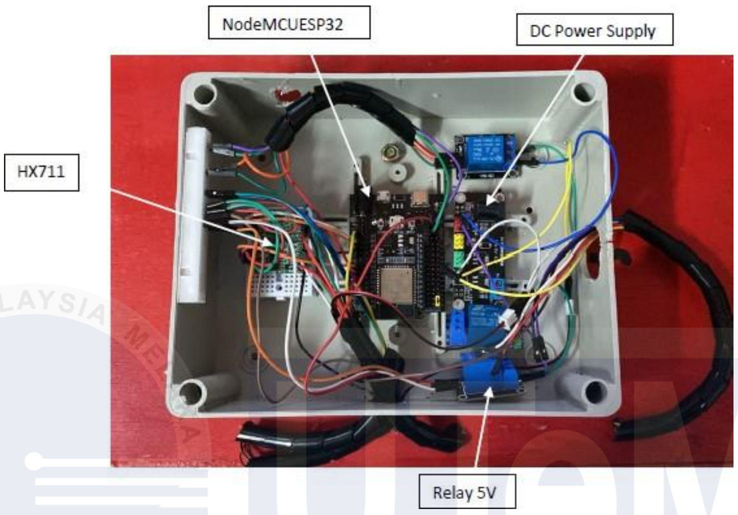


Figure 4.2: Hardware circuit

In this project, the NodeMCU, IR sensor, weight sensor, DC fan 12v and hx 711 will be used in order to assemble the prototypeas shown in Figure 4.2. All the component is play a very vital role to collecting and receiving information from the IoT platform which is Blynk that are transmit by ESP32 chip module inside the NodeMCU. All IR sensor will act as a sensor to detect when the door is open or closed which is IR1 will detect for washer 1 door and IR2 will detect for washer door 2 while IR3 will detect for dryer at door 3. Meanwhile, the NodeMCU will act as bridge to connect with the IoT platform.

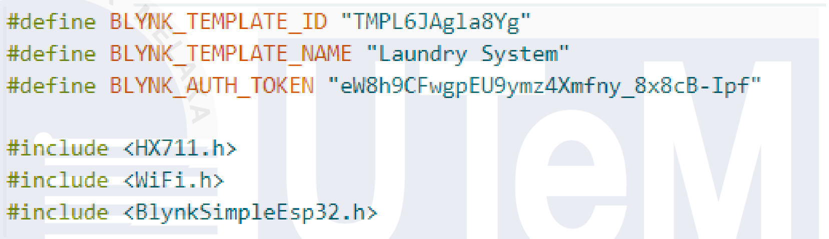
The Blynk application will receive data when the IR sensor detect the door either the door for each sensor closed or open. The code inside the NodeMCU will set the movement and weight. The Wi-Fi connection at the nodeMCU must be setup before connecting to the internet by setting the SSID and the password. other than that , this project using DC power supply selector for generate relay 12V that connect with dc fan 5V .DC fan responsible as washer or dryer fan.

4.2.2 Software Implementation

This section of the code is crucial for integrating the project with the Blynk application and ensuring proper functionality by including essential libraries. The first part of the code defines three parameters related to Blynk. The BLYNK-TEMPLATE-ID is a unique identifier that links the ESP32 microcontroller to the specific template in the Blynk cloud, which contains predefined widgets and data streams customized for this project. This ensures the hardware communicates with the correct project settings. The BLYNK-TEMPLATE-NAME, in this case, ”Laundry System,” serves as a label for the template, making it easier to identify when managing multiple projects. The BLYNK-AUTH-TOKEN acts as a secure key, uniquely linking the ESP32 to the user’s Blynk account. This token ensures only authorized devices can interact with the cloud, enabling secure communication for sending notifications or receiving commands.

The second part of the code includes three important libraries. The HX711.h library allows the ESP32 to interface with the HX711 module, which processes data from the weight sensors (load cells). This library is essential for interpreting raw sensor data into meaningful weight measurements, which play a critical role in the laundry system. The WiFi.h library is used to establish a Wi-Fi connection, enabling the ESP32 to connect to a wireless network. This connectivity is vital for real-time communication with the Blynk cloud. Lastly, the BlynkSimpleEsp32.h library serves as a bridge between the Blynk cloud and the ESP32, providing functions for initializing and maintaining a connection with Blynk and enabling data transfer between the microcontroller and the mobile application.

Overall, this section lays the foundation for the smart laundry system to communicate with the Blynk cloud and the user. It ensures that the system can operate in real-time, providing remote monitoring and control through the Blynk mobile application, as well as sending important notifications, such as when the laundry process is complete.(shows in Figure 4.3).



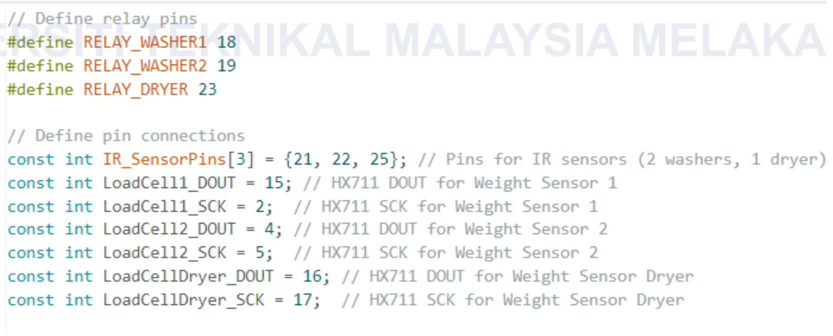
```
#define BLYNK_TEMPLATE_ID "TMPL6JAgla8Yg"
#define BLYNK_TEMPLATE_NAME "Laundry System"
#define BLYNK_AUTH_TOKEN "eW8h9CFwgpEU9ymz4Xmfny_8x8cB-Ipf"

#include <HX711.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
```

Figure 4.3: Header Definitions and Library that includes

Next, Figure 4.4 shows The initial part specifies the pins connected to the relays, which are responsible for controlling the operation of the washer and dryer motors. Two relays are defined here: RELAY-WASHER2, which is connected to pin 19 for the second washer, and RELAY-DRYER, connected to pin 23 for the dryer. These relays act as switches that allow low-power control signals from the ESP32 to operate high-power appliances, making them essential for safely managing the washers and dryer.

The code then defines the pins for the three IR sensors, stored in an array named IR-SensorPins. The sensors are connected to GPIO pins 21, 22, and 25, each corresponding to Washer 1, Washer 2, and the Dryer, respectively. IR sensors are used to detect whether the doors of the washer or dryer are open or closed. They typically work by emitting an infrared beam, which is interrupted when an object (such as the door) obstructs it. When the door is closed, the IR sensor sends a LOW signal to indicate the beam is uninterrupted. This functionality ensures the system does not operate when the doors are open, adding a layer of safety to the operation.

The code also includes pin configurations for three weight sensors, each connected through an HX711 module. Each weight sensor requires two pins: one for data output (DOUT) and another for the clock signal (SCK). The first washer's weight sensor is connected to pins 15 (DOUT) and 2 (SCK), the second washer's sensor to pins 4 (DOUT) and 5 (SCK), and the dryer's sensor to pins 16 (DOUT) and 17 (SCK). These sensors measure the weight of the laundry in each machine, with the HX711 module amplifying the signals from the load cells for precise weight measurements. By integrating these sensors, the system can detect whether a washer or dryer contains laundry before starting its operation, avoiding unnecessary energy usage and ensuring efficient performance.

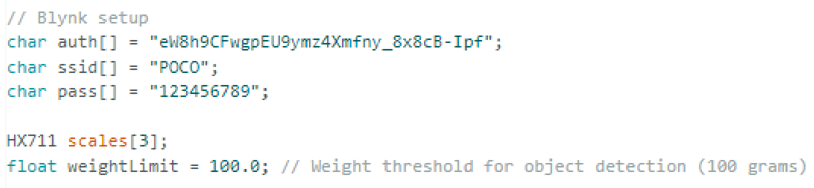


```
// Define relay pins
#define RELAY_WASHER1 18
#define RELAY_WASHER2 19
#define RELAY_DRYER 23

// Define pin connections
const int IR_SensorPins[3] = {21, 22, 25}; // Pins for IR sensors (2 washers, 1 dryer)
const int LoadCell1_DOUT = 15; // HX711 DOUT for Weight Sensor 1
const int LoadCell1_SCK = 2;  // HX711 SCK for Weight Sensor 1
const int LoadCell2_DOUT = 4; // HX711 DOUT for Weight Sensor 2
const int LoadCell2_SCK = 5;  // HX711 SCK for Weight Sensor 2
const int LoadCellDryer_DOUT = 16; // HX711 DOUT for Weight Sensor Dryer
const int LoadCellDryer_SCK = 17;  // HX711 SCK for Weight Sensor Dryer
```

Figure 4.4: Pin configuration and Component Definitions

In addition to hardware setup, this section of the code includes the configuration for the Blynk application. Figure 4.5 It begins with the declaration of a unique authentication token, stored in the variable auth. This token links the ESP32 microcontroller to the Blynk cloud, enabling seamless communication between the hardware and the user's mobile application. Alongside this, the code specifies the Wi-Fi credentials required for internet connectivity, using the variables ssid for the Wi-Fi network name and pass for its password. Connecting to Wi-Fi ensures that the system can send real-time notifications and updates to the Blynk app, keeping the user informed about the system's status.

The HX711 scales array is declared to manage the three weight sensors, allowing each washer and the dryer to have its own sensor for precise load monitoring. The scales are essential for interpreting the weight data from the HX711 modules, which is crucial for implementing the laundry detection functionality. Finally, the variable weightLimit is defined as 100.0 grams, setting the minimum threshold for detecting laundry. This threshold ensures that the system only activates when the weight of the laundry exceeds 100 grams, thereby preventing the washers or dryer from running when the drum is empty or the load is too light. By combining these features, this section of the code establishes the foundation for an efficient, safe, and user-friendly smart laundry system.



```
// Blynk setup
char auth[] = "eW8h9CFwgpEU9ymz4Xmfny_8x8cB-Ipf";
char ssid[] = "POCO";
char pass[] = "123456789";

HX711 scales[3];
float weightLimit = 100.0; // Weight threshold for object detection (100 grams)
```

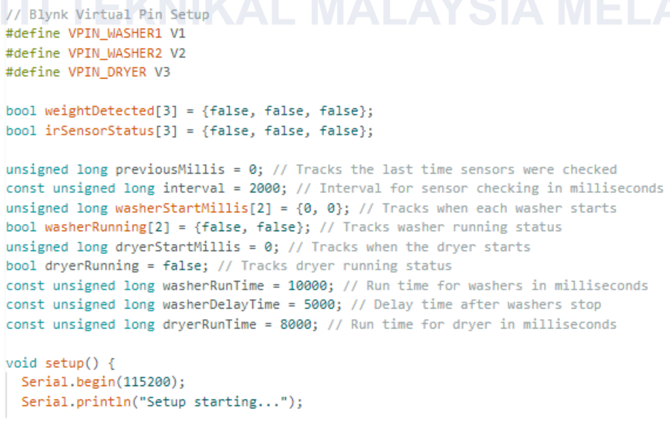
Figure 4.5: Blynk and Wi-Fi Configuration setup

Figure 4.6 shows the designed to manage and monitor the operation of washing machines and a dryer using the Blynk platform for IoT. It defines three virtual pins (VPIN-WASHER1, VPIN-WASHER2, and VPIN-DRYER) for communication. The arrays weight-Detected and irSensorStatus are initialized to store the states of sensors that monitor load and infrared status, respectively.

The code utilizes several variables and constants to track time and statuses. previousMillis is used to track the last sensor check in the main loop, while interval defines the interval for checking sensors (2 seconds). The washerStartMillis array tracks the start time for two washers, and washerRunning keeps their running status. Similarly, the dryer is monitored with dryerStartMillis and dryerRunning.

Constants define specific operational parameters: washerRunTime specifies how long a washer operates (10 seconds), washerDelayTime determines the delay after washers stop (5 seconds), and dryerRunTime sets the dryer runtime (8 seconds).

The setup() function initializes serial communication at a baud rate of 115200 and prints a message to indicate the start of the setup process. This forms the foundation for a system that can monitor and control laundry appliances in real-time, likely updating their status via Blynk.



```
// Blynk Virtual Pin Setup
#define VPIN_WASHER1 V1
#define VPIN_WASHER2 V2
#define VPIN_DRYER V3

bool weightDetected[3] = {false, false, false};
bool irSensorStatus[3] = {false, false, false};

unsigned long previousMillis = 0; // Tracks the last time sensors were checked
const unsigned long interval = 2000; // Interval for sensor checking in milliseconds
unsigned long washerStartMillis[2] = {0, 0}; // Tracks when each washer starts
bool washerRunning[2] = {false, false}; // Tracks washer running status
unsigned long dryerStartMillis = 0; // Tracks when the dryer starts
bool dryerRunning = false; // Tracks dryer running status
const unsigned long washerRunTime = 10000; // Run time for washers in milliseconds
const unsigned long washerDelayTime = 5000; // Delay time after washers stop
const unsigned long dryerRunTime = 8000; // Run time for dryer in milliseconds

void setup() {
  Serial.begin(115200);
  Serial.println("Setup starting...");
```

Figure 4.6: Monitoring Code.

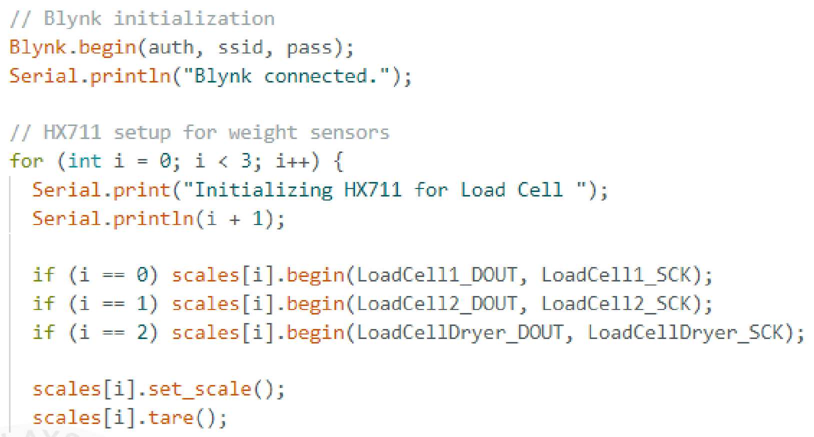
Figure 4.7 initializes the Blynk platform and configures three HX711 weight sensors used for load measurement. The first part initializes the Blynk IoT library with the credentials provided through auth, ssid, and pass, allowing the system to connect to a network and communicate with the Blynk server. Once the connection is established, it prints a confirmation message, ”Blynk connected,” to the Serial Monitor.

The second part of the code focuses on setting up the HX711 load cells, which are used to measure weight for applications such as monitoring laundry loads in washers and dryers. A loop iterates three times (indexed by i) to initialize each of the three load cells. For each iteration, the loop prints a message to the Serial Monitor indicating the initialization of a specific load cell, identified by its index.

Depending on the value of i, the corresponding load cell is initialized by calling the begin() function of the scales[i] object, which links the data (DOUT) and clock (SCK) pins for each load cell. For example:

When i = 0, the first load cell is initialized with LoadCell1-DOUT and LoadCell1-SCK. When i = 1, the second load cell is initialized with LoadCell2-DOUT and LoadCell2-SCK. When i = 2, the third load cell (for the dryer) is initialized with LoadCellDryer-DOUT and LoadCellDryer-SCK. After initializing each load cell, the set-scale() method is called to set the calibration factor for accurate weight measurement (though the specific factor isn’t defined in this snippet). Then, the tare() method is invoked to reset the scale, ensuring it starts at zero and eliminates any residual offset from the sensors.

This setup ensures the system accurately measures the weight of items in the washers and dryer, enabling real-time monitoring and decision-making based on load conditions. This is critical for automation and reporting tasks in smart laundry systems.



```
// Blynk initialization
Blynk.begin(auth, ssid, pass);
Serial.println("Blynk connected.");

// HX711 setup for weight sensors
for (int i = 0; i < 3; i++) {
  Serial.print("Initializing HX711 for Load Cell ");
  Serial.println(i + 1);

  if (i == 0) scales[i].begin(LoadCell1_DOUT, LoadCell1_SCK);
  if (i == 1) scales[i].begin(LoadCell2_DOUT, LoadCell2_SCK);
  if (i == 2) scales[i].begin(LoadCellDryer_DOUT, LoadCellDryer_SCK);

  scales[i].set_scale();
  scales[i].tare();
```

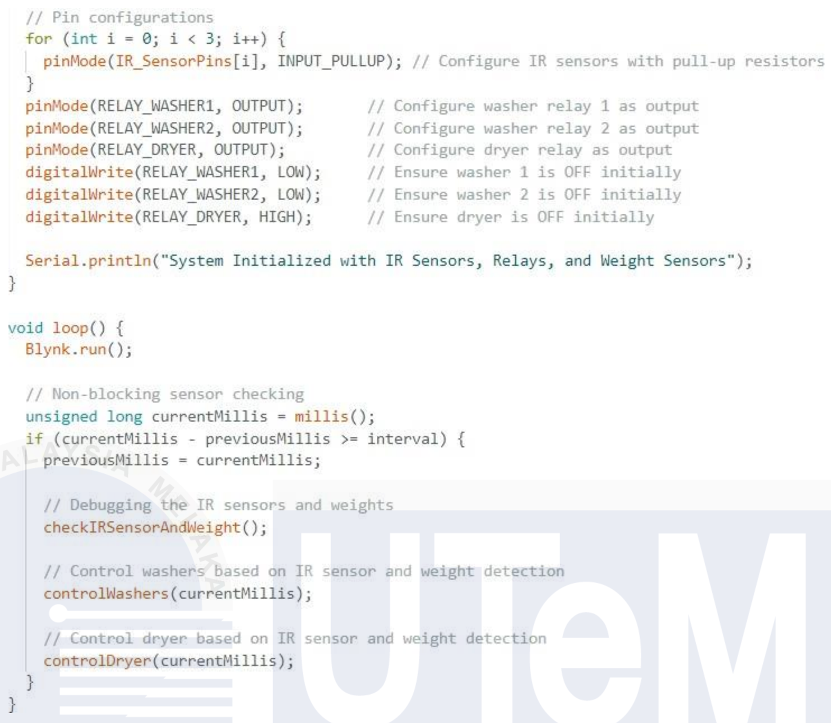
Figure 4.7: Setup Function

Figure 4.8 below shows first part of the code configures the necessary pins for the IR sensors and relays. A loop is used to set up three IR sensors as inputs with pull-up resistors, which helps stabilize the signal and ensures proper readings. Each sensor pin is defined in the array IR-SensorPins.

Relays for the washers and dryer are configured as outputs. The pinMode function sets the relay pins as output, enabling them to control the power flow to the machines. Initially, the washing machine relays (RELAY-WASHER1 and RELAY-WASHER2) are set to LOW, which turns them off. The dryer relay (RELAY-DRYER) is also initialized but set to HIGH to turn it off (assuming active-low relays are used). A message, ”System Initialized with IR Sensors, Relays, and Weight Sensors,” is printed to the Serial Monitor to indicate successful setup.

Next, The loop() function is the core of the program, continuously running during the operation of the microcontroller. It starts by calling Blynk.run(), which maintains the connection to the Blynk IoT platform, allowing real-time communication with the server.

Then, Non-Blocking Sensor Checking To ensure the program runs efficiently without delays, a non-blocking approach is implemented using millis(). The millis() function returns the number of milliseconds since the program started. A comparison is made between the current time (currentMillis) and the last recorded time (previousMillis). If the elapsed time exceeds a predefined interval (interval), the program updates previousMillis and performs sensor checks. Within this non-blocking interval, checkIRSensorAndWeight(): This function checks the status of IR sensors and weight sensors for debugging purposes. It likely ensures sensors are functioning correctly and provides feedback for troubleshooting. controlWashers(currentMillis): This function controls the washing machines based on the input from the IR sensors and weight detection. The logic likely determines when to activate or deactivate the washer relays depending on the detected load or operational status.controlDryer(currentMillis): This function operates similarly but is specific to the dryer. It ensures the dryer runs based on load detection or other predefined conditions. By combining real-time monitoring, IoT connectivity, and non-blocking execution, this system efficiently manages washing machines and a dryer. It ensures smooth operation by avoiding delays and maintaining precise control over appliances through relays and sensors.

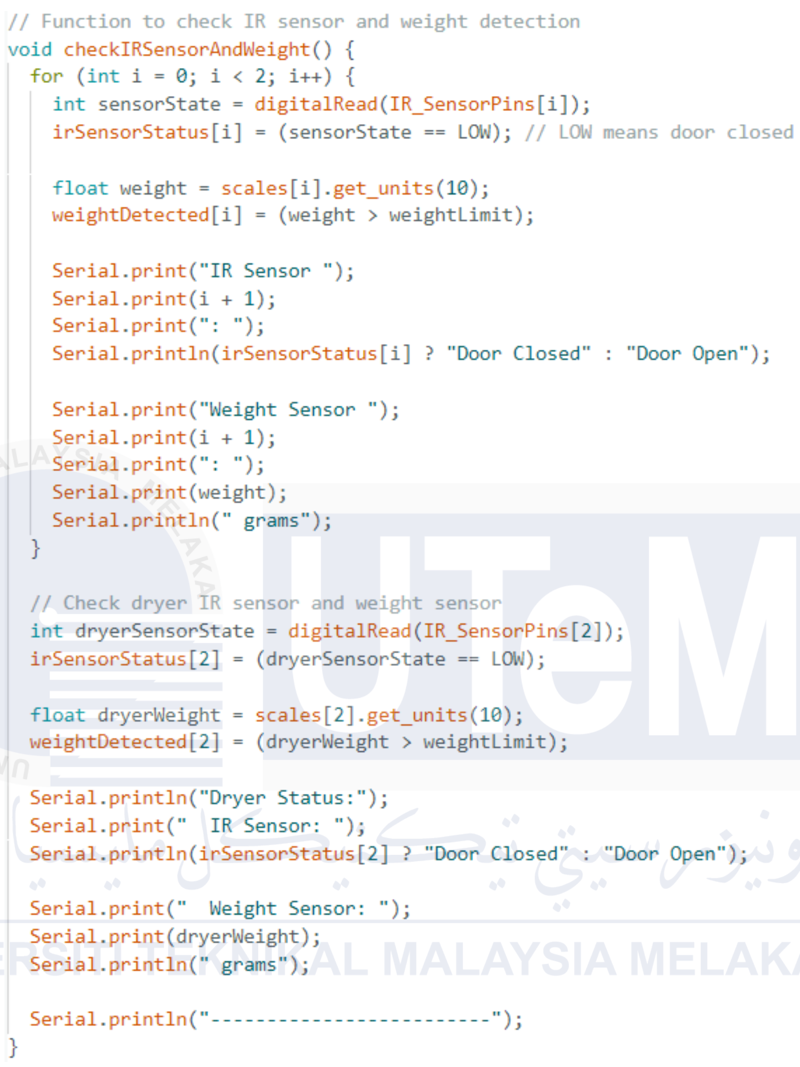


```
  // Pin configurations
  for (int i = 0; i < 3; i++) {
    pinMode(IR_SensorPins[i], INPUT_PULLUP); // Configure IR sensors with pull-up resistors
  }
  pinMode(RELAY_WASHER1, OUTPUT);       // Configure washer relay 1 as output
  pinMode(RELAY_WASHER2, OUTPUT);       // Configure washer relay 2 as output
  pinMode(RELAY_DRYER, OUTPUT);         // Configure dryer relay as output
  digitalWrite(RELAY_WASHER1, LOW);     // Ensure washer 1 is OFF initially
  digitalWrite(RELAY_WASHER2, LOW);     // Ensure washer 2 is OFF initially
  digitalWrite(RELAY_DRYER, HIGH);      // Ensure dryer is OFF initially

  Serial.println("System Initialized with IR Sensors, Relays, and Weight Sensors");
}

void loop() {
  Blynk.run();

  // Non-blocking sensor checking
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Debugging the IR sensors and weights
    checkIRSensorAndWeight();

    // Control washers based on IR sensor and weight detection
    controlWashers(currentMillis);

    // Control dryer based on IR sensor and weight detection
    controlDryer(currentMillis);
  }
}
```

Figure 4.8: main loop Function code

Figure 4.9 about function reads data from the IR sensors and weight sensors to determine the state of each appliance. The IR sensors detect whether the door of the washer or dryer is closed, with a LOW signal indicating a closed door. Meanwhile, the HX711 weight sensors measure the load in grams. If the load exceeds the defined weight threshold (weightLimit), the system interprets that laundry is present in the appliance.

For each washer and the dryer, the function prints the current state of the IR sensor (door open/closed) and the weight sensor (measured weight) to the Serial Monitor. This detailed debugging information helps developers monitor system performance and diagnose issues during testing. The function ensures that all sensors are checked within a single loop iteration, providing an efficient and unified way to update the system's state.



```
// Function to check IR sensor and weight detection
void checkIRSensorAndWeight() {
  for (int i = 0; i < 2; i++) {
    int sensorState = digitalRead(IR_SensorPins[i]);
    irSensorStatus[i] = (sensorState == LOW); // LOW means door closed

    float weight = scales[i].get_units(10);
    weightDetected[i] = (weight > weightLimit);

    Serial.print("IR Sensor ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.println(irSensorStatus[i] ? "Door Closed" : "Door Open");

    Serial.print("Weight Sensor ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(weight);
    Serial.println(" grams");
  }

  // Check dryer IR sensor and weight sensor
  int dryerSensorState = digitalRead(IR_SensorPins[2]);
  irSensorStatus[2] = (dryerSensorState == LOW);

  float dryerWeight = scales[2].get_units(10);
  weightDetected[2] = (dryerWeight > weightLimit);

  Serial.println("Dryer Status:");
  Serial.print("  IR Sensor: ");
  Serial.println(irSensorStatus[2] ? "Door Closed" : "Door Open");

  Serial.print("  Weight Sensor: ");
  Serial.print(dryerWeight);
  Serial.println(" grams");

  Serial.println("-------------------------");
}
```

Figure 4.9: sensor operation

Figure 4.10 shows about the function manages the operation of the two washers based on inputs from the IR and weight sensors. Each washer is controlled independently, allowing simultaneous operation if needed. The function uses the following logic:

Starting the Washer:

If the IR sensor detects a closed door and the weight sensor indicates a load, the washer starts. The relay controlling the washer is activated, and a notification is sent to the

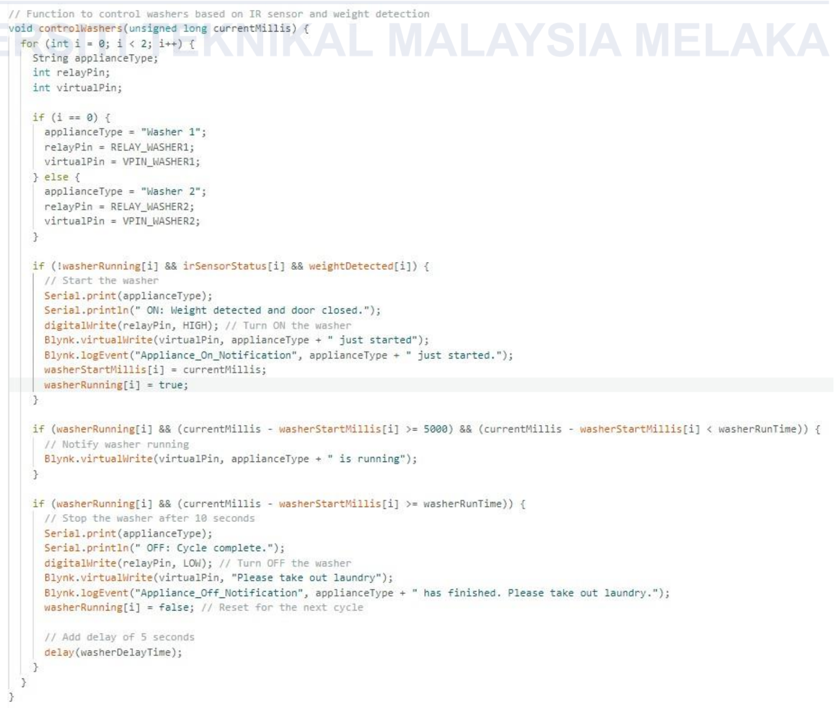
Blynk application stating that the washer has ”just started.” This notification ensures that users are immediately informed of the appliance’s status.

Running State:

While the washer is running, another notification (”is running”) is sent to the user. This message provides real-time updates during the appliance’s operation, enhancing the user’s awareness.

Stopping the Washer:

After completing its runtime (washerRunTime), the washer stops, and the relay is deactivated. A final notification, ”Please take out laundry,” prompts the user to retrieve their clothes. The function resets the washer’s state, allowing it to be used for the next cycle. A delay of 5 seconds (washerDelayTime) is added to ensure safe transitions between cycles.



```
// Function to control washers based on IR sensor and weight detection
void controlWashers(unsigned long currentMillis) {
  for (int i = 0; i < 2; i++) {
    String applianceType;
    int relayPin;
    int virtualPin;

    if (i == 0) {
      applianceType = "Washer 1";
      relayPin = RELAY_WASHER1;
      virtualPin = VPIN_WASHER1;
    } else {
      applianceType = "Washer 2";
      relayPin = RELAY_WASHER2;
      virtualPin = VPIN_WASHER2;
    }

    if (!washerRunning[i] && irSensorStatus[i] && weightDetected[i]) {
      // Start the washer
      Serial.print(applianceType);
      Serial.println(" ON: Weight detected and door closed.");
      digitalWrite(relayPin, HIGH); // Turn ON the washer
      Blynk.virtualWrite(virtualPin, applianceType + " just started");
      Blynk.logEvent("Appliance_On_Notification", applianceType + " just started.");
      washerStartMillis[i] = currentMillis;
      washerRunning[i] = true;
    }

    if (washerRunning[i] && (currentMillis - washerStartMillis[i] >= 5000) && (currentMillis - washerStartMillis[i] < washerRunTime)) {
      // Notify washer running
      Blynk.virtualWrite(virtualPin, applianceType + " is running");
    }

    if (washerRunning[i] && (currentMillis - washerStartMillis[i] >= washerRunTime)) {
      // Stop the washer after 10 seconds
      Serial.print(applianceType);
      Serial.println(" OFF: Cycle complete.");
      digitalWrite(relayPin, LOW); // Turn OFF the washer
      Blynk.virtualWrite(virtualPin, "Please take out laundry");
      Blynk.logEvent("Appliance_Off_Notification", applianceType + " has finished. Please take out laundry.");
      washerRunning[i] = false; // Reset for the next cycle

      // Add delay of 5 seconds
      delay(washerDelayTime);
    }
  }
}
```

Figure 4.10: Washer Control Logic Code

4.11below shows the code for dryer operation. The controlDryer() function operates similarly to the washer control function but is tailored for the dryer. It uses the same sensor inputs (IR and weight) to determine when to start and stop the appliance. The control logic includes:

Starting the Dryer:

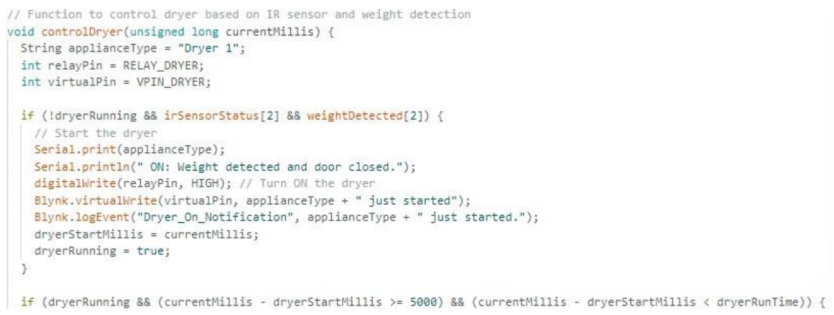
When the door is closed and a load is detected, the dryer starts. A notification is sent to inform the user that the dryer has ”just started,” ensuring they are aware of the process initiation.

Running State:

During the dryer’s runtime (dryerRunTime), the system sends a notification that the dryer ”is running.” This provides real-time updates to the user.

Stopping the Dryer:

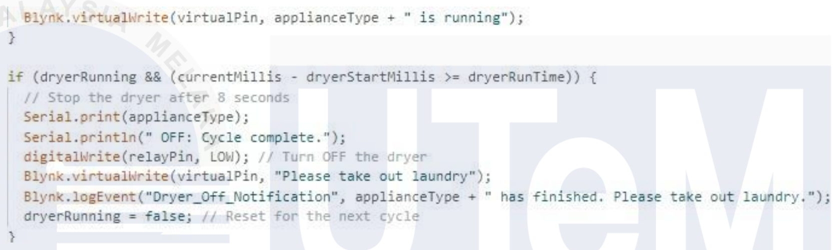
At the end of its cycle, the dryer turns off, and a notification prompts the user to ”Please take out laundry.” The function resets the dryer’s state, preparing it for the next use.



```
// Function to control dryer based on IR sensor and weight detection
void controlDryer(unsigned long currentMillis) {
  String applianceType = "Dryer 1";
  int relayPin = RELAY_DRYER;
  int virtualPin = VPIN_DRYER;

  if (!dryerRunning && irSensorStatus[2] && weightDetected[2]) {
    // Start the dryer
    Serial.print(applianceType);
    Serial.println(" ON: Weight detected and door closed.");
    digitalWrite(relayPin, HIGH); // Turn ON the dryer
    Blynk.virtualWrite(virtualPin, applianceType + " just started");
    Blynk.logEvent("Dryer_On_Notification", applianceType + " just started.");
    dryerStartMillis = currentMillis;
    dryerRunning = true;
  }

  if (dryerRunning && (currentMillis - dryerStartMillis >= 5000) && (currentMillis - dryerStartMillis < dryerRunTime)) {
```

Figure 4.11: Dryer Control Code

The integration with the Blynk platform enables seamless communication between the ESP32 and the user's smartphone or computer.4.12shows the code for Notifications that sent at key points in the appliance's operation, such as when the cycle starts, while it is running, and when it completes. These notifications provide a user-friendly experience, ensuring that users are always informed of the system's status. The Blynk.logEvent() function adds an additional layer of logging, which can be used for analytics or troubleshooting.

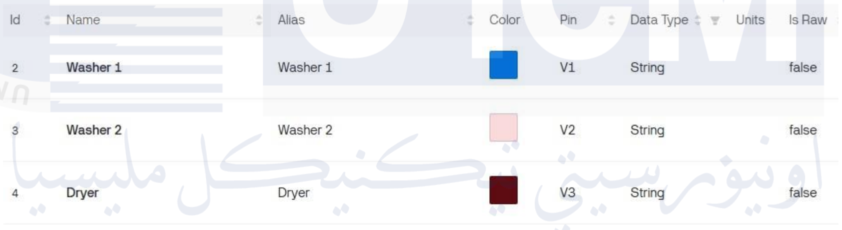


```
  Blynk.virtualWrite(virtualPin, applianceType + " is running");
}

if (dryerRunning && (currentMillis - dryerStartMillis >= dryerRunTime)) {
  // Stop the dryer after 8 seconds
  Serial.print(applianceType);
  Serial.println(" OFF: Cycle complete.");
  digitalWrite(relayPin, LOW); // Turn OFF the dryer
  Blynk.virtualWrite(virtualPin, "Please take out laundry");
  Blynk.logEvent("Dryer_Off_Notification", applianceType + " has finished. Please take out laundry.");
  dryerRunning = false; // Reset for the next cycle
}
```

Figure 4.12: Notifications and Cloud Integration

4.2.3 Application software

The software application that is implemented is known as Blynk and each software that includes Blynk has it own unique interface. In addition to this, the Blynk interface can be quickly developed because it already has awidget box, which can be supplied with any widget by dragging into the Blynk project. The configuration is very straight forward, and the example and explanation are given at each widget information tab inside the Blynk software. Based on the coding, each used function in Blynk will be send to the virtual pin such as washer1, washer2 and dryer status.



Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw
2	Washer 1	Washer 1	🟦	V1	String		false
3	Washer 2	Washer 2	🟥	V2	String		false
4	Dryer	Dryer	🟫	V3	String		false

Figure 4.13: Blynk Authorized Token

As can be seen in Figure 4.13, The Blynk pin arrangement has been build up. this is done to guarantee the connection between NodeMCU ESP32 and Blynk can function sucessfully and the data can be visualized in the correct manners. its application is widespread. In addition to that, it gives users the ability to save any value in the virtual pin.

4.2.4 Results

The result that has been obtained from the project prototype will be shown in this sub section. Project prototype has been develop to ensure that it can simulate how the project will function based on the objective. the prototype is build using plywood. Weight sensor based are custom using plywood and box while the door using a stick. 4.14 shows the front view of this project .Although the weight sensor are not so smooth but the function operation still can be shown and demonstrate.

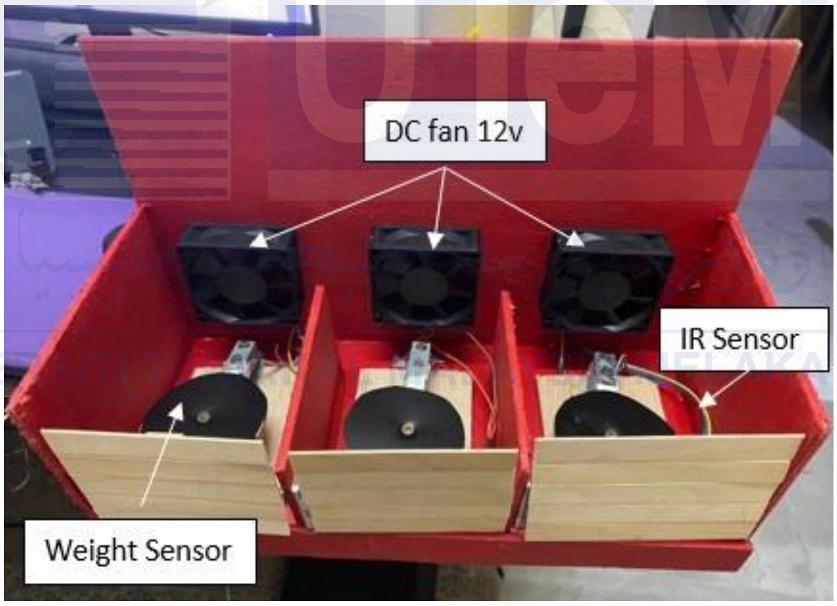
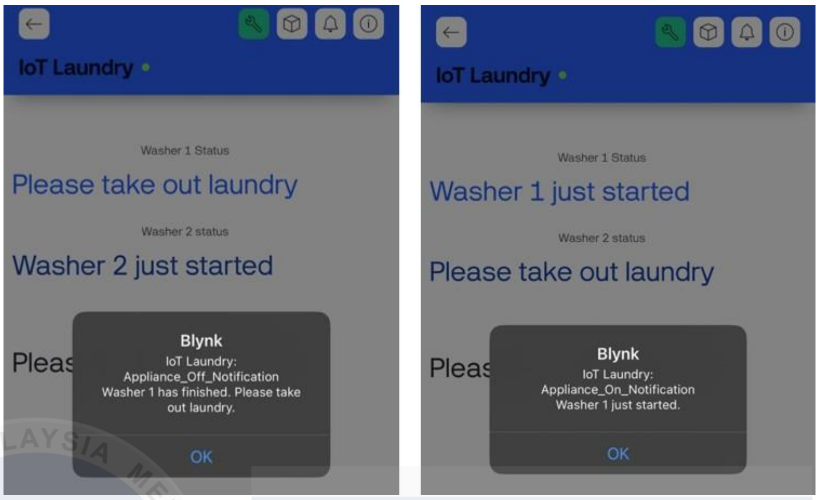


Figure 4.14: The Front View

Figure 4.15 below shows the result when users puts the laundry in the mechine and closed the door,Nodemcu willtrigger and send signal to blynk and wil gives notification for each machine. The internet connection for both must and it not compulsory for both device to connect to the same network.


IoT Laundry

Washer 1 Status

Please take out laundry

Washer 2 status

Washer 2 just started

Pleas

Blynk

IoT Laundry:

Appliance_Off_Notification

Washer 1 has finished. Please take out laundry.

OK

IoT Laundry

Washer 1 Status

Washer 1 just started

Washer 2 status

Please take out laundry

Pleas

Blynk

IoT Laundry:

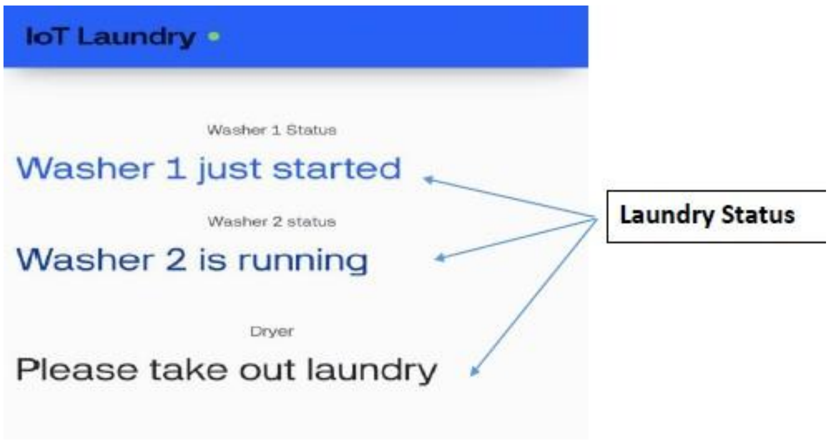
Appliance_On_Notification

Washer 1 just started.

OK

Figure 4.15: The Result for Push Notification in Blynk

Blynk also shows the current status at the STATUS display widget button. in the status widget, washer 1,washer 2 and Dryer will show the status ”Please take out your laundry”, ”washer is running”, washer just started” it will be shown according to the current condition like Figure 4.16 .


IoT Laundry

Washer 1 Status

Washer 1 just started

Washer 2 status

Washer 2 is running

Dryer

Please take out laundry

Figure 4.16: The Mechine status that been shown

4.2.5 Data Analysis

A survey has been conducted to gather several data that will be determined the importance of this project to consumers. Based on the question that has been given inside the survey, several key question have been selected with data. The data will be analysis in this chapter.

Firstly, The Figure 4.17 This question was designed to identify the age demographic of the participants to better understand their suitability as potential users of a smart laundry monitoring system. The survey results reveal that a majority of the respondents fall within the 21–25 age group, representing over (60%) of the total participants. This group likely consists of students, young professionals, or individuals who are more inclined to use technological solutions for everyday tasks. Additionally, a smaller number of respondents are in their late twenties, possibly early-career professionals or working adults, while one respondent is in their seventies, representing an outlier in this dataset. These results show that most participants belong to a demographic that is familiar with mobile applications and modern IoT-based systems, which suggests that the proposed laundry monitoring solution is well-targeted. The concentration of younger respondents also indicates that the product should cater to a tech-savvy audience who values convenience and efficiency in managing their chores.

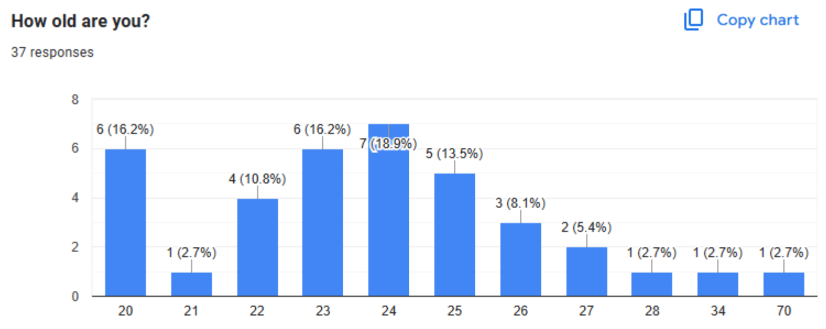


Figure 4.17: Result for Question 1

The second question explored laundry machine usage frequency, Figure 4.18 aimed to gauge the frequency of laundry machine usage among the participants, helping to assess the potential demand for a monitoring system. The responses show that (81.5%) of participants use a laundry machine weekly, indicating that laundry is a regular part of their routine. This high percentage suggests that the majority of users would benefit from a system that helps them better manage their laundry schedules. Meanwhile, (18.5%) of respondents reported using laundry machines daily, likely representing individuals with higher laundry needs, such as families or individuals with strict hygiene routines. None of the respondents selected ”rarely,” which further reinforces that laundry machine usage is a consistent activity for all participants. These findings highlight the relevance of introducing a smart solution that not only meets their current needs but also improves their overall experience by automating certain aspects of the process, such as notifications and monitoring.

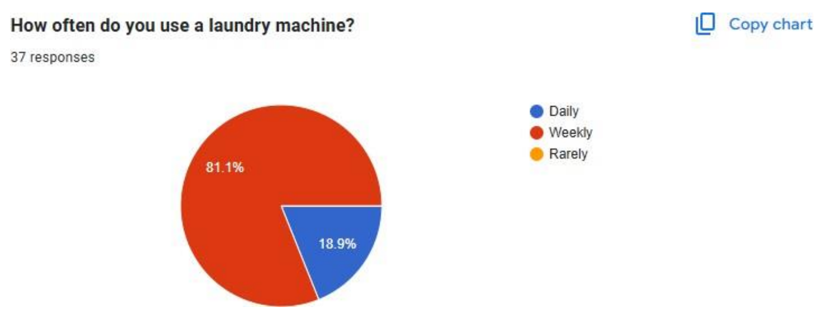


Figure 4.18: Result for Question 2

Next question Figure 4.19 to measure the level of inconvenience experienced by users when manually monitoring their laundry. A striking (88.9%) of respondents, or 24 out of 27 participants, stated that they find manual monitoring complicated. This highlights the frustration and inconvenience of traditional laundry management methods, such as waiting near the machine or repeatedly checking if the cycle is complete. Only (7.4%) of participants, or 2 respondents, reported that monitoring manually is easy, while (3.7%) were neutral. These results suggest that for the vast majority, manual monitoring presents a challenge that could be effectively addressed through automation. The data further emphasizes the need for a solution that alleviates this problem by providing real-time updates and notifications, allowing users to focus on other tasks. By reducing the need for constant physical monitoring, a smart laundry system could significantly enhance the user experience.

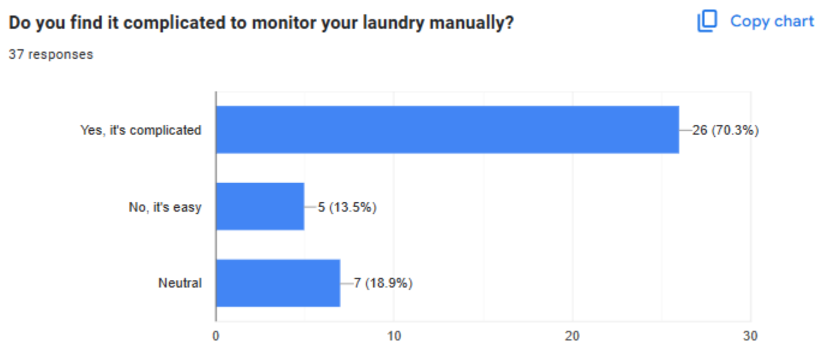


Figure 4.19: Result for Question 3

Figure 4.20 highlighted a common issue, This question explored participants' interest in receiving notifications about their laundry's status, a key feature of the proposed smart laundry system. The results were unanimous, with all 27 respondents (100%) expressing a preference for such a system. This overwhelming agreement underscores the high demand for time-saving and user-friendly solutions. Participants recognize the convenience of receiving notifications, which eliminates the need to wait around or repeatedly check on their laundry manually. This feature would allow users to manage their time more effectively, enabling them to engage in other activities without worrying about missing the end of a laundry cycle. The unanimous response not only validates the practicality of the proposed solution but also highlights the importance of incorporating notification features into laundry systems to meet user expectations and needs.

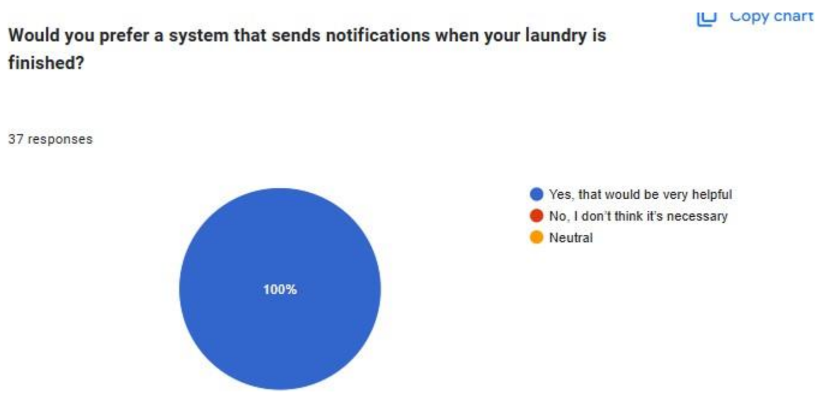


Figure 4.20: Result for Question 4

Finally, the fifth question to understand how much time participants typically spend waiting at the laundry shop, providing insights into the potential time savings offered by monitoring system. Figure 4.21 share he majority of respondents (60.7%) reported spending 30–45 minutes waiting, reflecting the typical duration of a laundry cycle. Another (13.3%) spend between 15–30 minutes, and an equal percentage spend 45 minutes to 1 hour. A smaller portion, (6.7%), reported spending less than 15 minutes, while another (6.7%) stated that they spend over an hour waiting at the laundry shop. These findings indicate that waiting times can vary widely depending on factors such as laundry load, machine efficiency, and user schedules. However, for the majority, waiting times are substantial, with many spending nearly an hour at the laundry shop. This reinforces the need for a system that allows users to leave the laundry shop and receive real-time notifications when their laundry is done. Such a system could save users significant time and provide greater flexibility in managing their schedules.

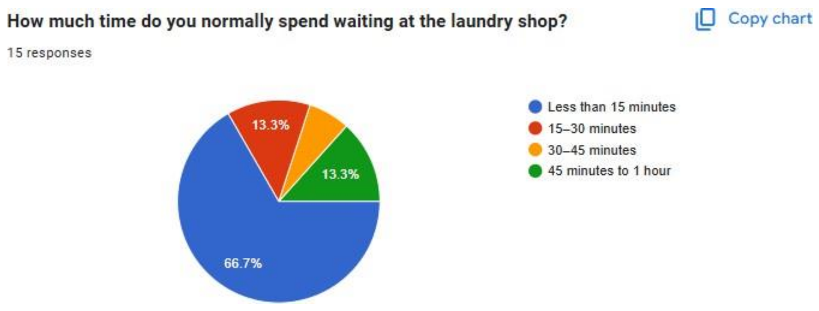


Figure 4.21: Result for Question 5

4.2.6 Discussion

The project, ”Development of IoT-Based Monitoring Laundry System Using NodeMCU ESP32 and Blynk Application,” successfully demonstrated how IoT technology can enhance laundry system functionality and convenience. Hardware components such as the NodeMCU ESP32, IR sensors, weight sensors, and relays were effectively integrated, ensuring reliable operation. IR sensors detected door statuses for safety, and weight sensors prevented overloading, while relays enabled precise machine control. Occasional calibration adjustments for the sensors highlighted sensitivity to environmental factors.

The software, implemented using Arduino IDE and the Blynk application, provided robust remote monitoring and real-time notifications like ”Laundry has done,” enhancing user convenience. Wi-Fi ensured seamless communication, though initial connection delays and minor notification timing discrepancies were noted but resolved. Data analysis confirmed the accuracy of sensor readings and validated the system’s efficiency.

Despite its success, limitations included reliance on stable Wi-Fi, limited scalability due to GPIO constraints, and potential sensor performance impacts from environmental factors. Compared to similar projects, this system stood out for its simplicity and cost-effectiveness, though adding features like predictive maintenance or SMS notifications could improve functionality. Future enhancements might also include energy monitoring and a more intuitive Blynk interface.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project successfully Developed an IoT-Based Laundry Monitoring System using NodeMCU ESP32 and the Blynk application. By integrating advanced IoT technologies, the system effectively addressed inefficiencies commonly found in traditional laundry setups, such as resource wastage, user inconvenience, and high operational costs. The project achieved several key milestones, including enabling real-time monitoring for users through the Blynk application, optimizing machine usage to reduce energy and water consumption, and providing timely notifications to enhance user experience. Additionally, the incorporation of safety features like IR sensors ensured that machines operated only under safe conditions, such as when doors were securely closed. The system's design also demonstrated scalability and adaptability, making it capable of supporting future enhancements to meet evolving user needs.

5.2 Future Works

For future development, several recommendations can further improve the system. Enhancing the user interface of the Blynk application could provide predictive maintenance alerts and detailed usage statistics for a better user experience. Incorporating energy-saving features, such as dynamic scheduling based on electricity tariffs, would improve resource efficiency. Machine learning algorithms could be utilized to predict laundry duration and detect anomalies in machine behavior, adding intelligence to the system. Additionally, the commercialization potential of the system could be explored by collaborating with laundromats to scale the solution for public use. Integration with other smart home devices, such as thermostats or virtual assistants, could also make the system more versatile.

This project lays a strong foundation for further innovations in IoT applications for household management, emphasizing convenience, efficiency, and sustainability. It demonstrates how IoT solutions can significantly improve everyday life by automating and optimizing routine tasks.

5.3 Project Commercialization

The commercialization potential of the IoT-based laundry monitoring system presents exciting opportunities for broader application. By collaborating with laundromats and other laundry service providers, the system could be scaled for public use, offering a more efficient and user-friendly experience for customers. Integrating features such as remote machine monitoring, automated notifications, and usage statistics would enhance the appeal of the system in a commercial setting. Additionally, implementing subscription-based services or pay-per-use models could generate steady revenue streams. Customization options, such as branding and tailored user interfaces, would allow businesses to adapt the system to their specific needs. Commercial deployment could also include partnerships with appliance manufacturers to embed the system into new laundry machines, creating a seamless, ready-to-use solution. With its focus on efficiency, user convenience, and IoT-driven automation, this system has the potential to transform traditional laundry services and establish a strong presence in the growing market for smart appliances and solutions.

REFERENCES

[1] B. Saleha, S. M. Nasution, and A. L. Prasasti, “Design of iot-based smart laundry applications using fuzzy algorithms,” in *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2020, pp. 393–397.

[2] E. M. Rizki, S. M. Nasution, and A. L. Prasasti, “Design of laundry box as supporting smart laundry system based on internet of things,” in *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2020, pp. 398–404.

[3] A. Singh and M. Dhaiya, “Web based application to search and manage Laundry shops,” in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, 2022, pp. 1150–1155.

[4] R. Akbar, S. M. Nasution, and A. L. Prasasti, “Implementation of naive bayes algorithm on iot-based smart laundry mobile application system,” in *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2020, pp. 8–13.

[5] A. Menachery and J. Christina, “Monitoring the status of self-operated community laundry machines using iot integration,” in *2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 2021, pp. 83–85.

[6] S. Shakya, O. Bajracharya, R. Padmanabhuni, S. D. A. P. Senadeera, A. Taparugssanagorn, and M. N. Dailey, “An internet of things system for a laundry monitoring service,” in *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, 2021, pp. 235–240.

[7] D. Sai, S. Reddy, S. Kulkarni, S. Pattankar, S. Chitragar, V. K. Annapurna, and U. Students, “An iot-based laundry system application,” vol. 9, pp. 475–479, 2022.

[8] G. A. Susto, L. Vettore, G. Zambonin, F. Altinier, D. Beninato, T. Girotto, M. Rampazzo, and A. Beghi, “A machine learning-based soft sensor for laundry load fabric typology estimation in household washer-dryers,” vol. 52, pp. 116–121, 2019.

[9] Y. Qiao, C. Zhang, R. Li, and Z. Liu, “A wireless intelligent business laundry service system,” *Journal of Computer and Communications*, vol. 07, pp. 105–113, 2019.

[10] A. D., M. M., N. M., and D. S.M., “Intelligent washing machine driven by internet of things,” *International Journal of Mechanical Engineering*, vol. 06, pp. 38–42, 9 2022.

APPENDICES

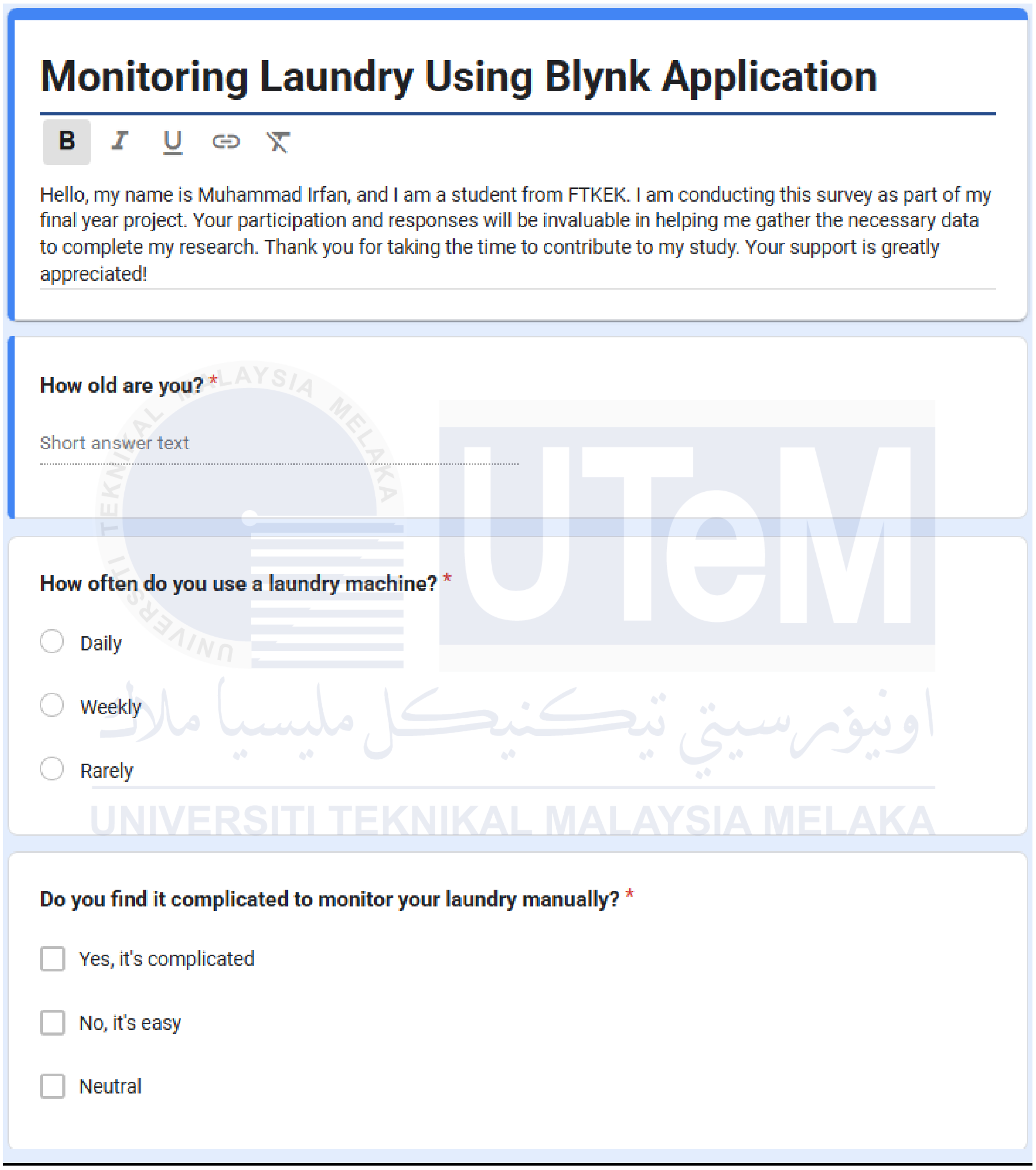
Appendix A: Full code for Monitoring Laundry System

```
#define BLYNK_TEMPLATE_ID "TMPL6JAgla8Yg"
#define BLYNK_TEMPLATE_NAME "Laundry System"
#define BLYNK_AUTH_TOKEN "eW8h9CFwgpEU9ymz4Xmfny_8x8cB-Ipf"

#include <HX711.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// Define relay pins
#define RELAY_WASHER1 18
#define RELAY_WASHER2 19
#define RELAY_DRYER 23

// Define pin connections
const int IR_SensorPins[3] = {21, 22, 25}; // Pins for IR sensors (2 washers, 1 dryer)
const int LoadCell1_DOUT = 15; // HX711 DOUT for Weight Sensor 1
const int LoadCell1_SCK = 2;   // HX711 SCK for Weight Sensor 1
const int LoadCell2_DOUT = 4; // HX711 DOUT for Weight Sensor 2
const int LoadCell2_SCK = 5;   // HX711 SCK for Weight Sensor 2
const int LoadCellDryer_DOUT = 16; // HX711 DOUT for Weight Sensor Dryer
const int LoadCellDryer_SCK = 17;   // HX711 SCK for Weight Sensor Dryer

// Blynk setup
char auth[] = "eW8h9CFwgpEU9ymz4Xmfny_8x8cB-Ipf";
char ssid[] = "POCO";
char pass[] = "123456789";

HX711 scales[3];
```

```
float weightLimit = 100.0; // Weight threshold for object detection (100 grams)

// Blynk Virtual Pin Setup
#define VPIN_WASHER 1 V1
#define VPIN_WASHER 2 V2
#define VPIN_DRYER V3

bool weightDetected[3] = {false, false, false};
bool irSensorStatus[3] = {false, false, false};

unsigned long previousMillis = 0; // Tracks the last time sensors were checked
const unsigned long interval = 2000; // Interval for sensor checking in milliseconds
unsigned long washerStartMillis[2] = {0, 0}; // Tracks when each washer starts
bool washerRunning[2] = {false, false}; // Tracks washer running status
unsigned long dryerStartMillis = 0; // Tracks when the dryer starts
bool dryerRunning = false; // Tracks dryer running status
const unsigned long washerRunTime = 10000; // Run time for washers in milliseconds
const unsigned long washerDelayTime = 5000; // Delay time after washers stop
const unsigned long dryerRunTime = 8000; // Run time for dryer in milliseconds

void setup() {
  Serial.begin(115200);
  Serial.println("Setup starting...");

  // Blynk initialization
  Blynk.begin(auth, ssid, pass);
  Serial.println("Blynk connected.");

  // HX711 setup for weight sensors
  for (int i = 0; i < 3; i++) {
    Serial.print("Initializing HX711 for Load Cell ");
    Serial.println(i + 1);

    if (i == 0) scales[i].begin(LoadCell1_DOUT, LoadCell1_SCK);
    if (i == 1) scales[i].begin(LoadCell2_DOUT, LoadCell2_SCK);
    if (i == 2) scales[i].begin(LoadCellDryer_DOUT, LoadCellDryer_SCK);

    scales[i].set_scale();
    scales[i].tare();
  }

  // Pin configurations
  for (int i = 0; i < 3; i++) {
    pinMode(IR_SensorPins[i], INPUT_PULLUP); // Configure IR sensors with pull-up resistors
  }
  pinMode(RELAY_WASHER1, OUTPUT);          // Configure washer relay 1 as output
  pinMode(RELAY_WASHER2, OUTPUT);          // Configure washer relay 2 as output
  pinMode(RELAY_DRYER, OUTPUT);            // Configure dryer relay as output
  digitalWrite(RELAY_WASHER1, LOW);        // Ensure washer 1 is OFF initially
  digitalWrite(RELAY_WASHER2, LOW);        // Ensure washer 2 is OFF initially
  digitalWrite(RELAY_DRYER, HIGH);         // Ensure dryer is OFF initially

  Serial.println("System Initialized with IR Sensors, Relays, and Weight Sensors");
}

void loop() {
  Blynk.run();

  // Non-blocking sensor checking
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Debugging the IR sensors and weights
    checkIRSensorAndWeight();

    // Control washers based on IR sensor and weight detection
    controlWashers(currentMillis);

    // Control dryer based on IR sensor and weight detection
    controlDryer(currentMillis);
  }
}

// Function to check IR sensor and weight detection
void checkIRSensorAndWeight() {
  for (int i = 0; i < 2; i++) {
    int sensorState = digitalRead(IR_SensorPins[i]);
    irSensorStatus[i] = (sensorState == LOW); // LOW means door closed

    float weight = scales[i].get_units(10);
    weightDetected[i] = (weight > weightLimit);

    Serial.print("IR Sensor ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.println(irSensorStatus[i] ? "Door Closed" : "Door Open");

    Serial.print("Weight Sensor ");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(weight);
    Serial.println(" grams");
  }

  // Check dryer IR sensor and weight sensor
  int dryerSensorState = digitalRead(IR_SensorPins[2]);
  irSensorStatus[2] = (dryerSensorState == LOW);

  float dryerWeight = scales[2].get_units(10);
  weightDetected[2] = (dryerWeight > weightLimit);

  Serial.println("Dryer Status:");
  Serial.print("  IR Sensor: ");
  Serial.println(irSensorStatus[2] ? "Door Closed" : "Door Open");

  Serial.print("  Weight Sensor: ");
  Serial.print(dryerWeight);
  Serial.println(" grams");

  Serial.println("--------------------------------------------------");
}

// Function to control washers based on IR sensor and weight detection
void controlWashers(unsigned long currentMillis) {
  for (int i = 0; i < 2; i++) {
    String applianceType;
    int relayPin;
    int virtualPin;

    if (i == 0) {
      applianceType = "Washer 1";
      relayPin = RELAY_WASHER1;
      virtualPin = VPIN_WASHER1;
    } else {
      applianceType = "Washer 2";
      relayPin = RELAY_WASHER2;
      virtualPin = VPIN_WASHER2;
    }

    if (!washerRunning[i] && irSensorStatus[i] && weightDetected[i]) {
      // Start the washer
```

```
        Serial.print(applianceType);
        Serial.println(" ON: Weight detected and door closed.");
        digitalWrite(relayPin, HIGH); // Turn ON the washer
        Blynk.virtualWrite(virtualPin, applianceType + " just started");
        Blynk.logEvent("Appliance_On_Notification", applianceType + " just started.");
        washerStartMillis[i] = currentMillis;
        washerRunning[i] = true;
      }

      if (washerRunning[i] && (currentMillis - washerStartMillis[i] >= 5000) && (currentMillis - washer
        // Notify washer running
        Blynk.virtualWrite(virtualPin, applianceType + " is running");
      }

      if (washerRunning[i] && (currentMillis - washerStartMillis[i] >= washerRunTime)) {
        // Stop the washer after 10 seconds
        Serial.print(applianceType);
        Serial.println(" OFF: Cycle complete.");
        digitalWrite(relayPin, LOW); // Turn OFF the washer
        Blynk.virtualWrite(virtualPin, "Please take out laundry");
        Blynk.logEvent("Appliance_Off_Notification", applianceType + " has finished. Please take out la
        washerRunning[i] = false; // Reset for the next cycle

        // Add delay of 5 seconds
        delay(washerDelayTime);
      }
    }
}

// Function to control dryer based on IR sensor and weight detection
void controlDryer(unsigned long currentMillis) {
  String applianceType = "Dryer 1";
  int relayPin = RELAY_DRYER;
    int virtualPin = VPIN_DRYER;

    if (!dryerRunning && irSensorStatus[2] && weightDetected[2]) {
      // Start the dryer
      Serial.print(applianceType);
      Serial.println(" ON: Weight detected and door closed.");
      digitalWrite(relayPin, HIGH); // Turn ON the dryer
      Blynk.virtualWrite(virtualPin, applianceType + " just started");
      Blynk.logEvent("Dryer_On_Notification", applianceType + " just started.");
      dryerStartMillis = currentMillis;
      dryerRunning = true;
    }

    if (dryerRunning && (currentMillis - dryerStartMillis >= 5000) && (currentMillis - dryerStartMillis
      // Notify dryer running
      Blynk.virtualWrite(virtualPin, applianceType + " is running");
    }

    if (dryerRunning && (currentMillis - dryerStartMillis >= dryerRunTime)) {
      // Stop the dryer after 8 seconds
      Serial.print(applianceType);
      Serial.println(" OFF: Cycle complete.");
      digitalWrite(relayPin, LOW); // Turn OFF the dryer
      Blynk.virtualWrite(virtualPin, "Please take out laundry");
      Blynk.logEvent("Dryer_Off_Notification", applianceType + " has finished. Please take out laundry.
      dryerRunning = false; // Reset for the next cycle
    }
}
```

Survey platform



Monitoring Laundry Using Blynk Application

B *I* U

Hello, my name is Muhammad Irfan, and I am a student from FTKEK. I am conducting this survey as part of my final year project. Your participation and responses will be invaluable in helping me gather the necessary data to complete my research. Thank you for taking the time to contribute to my study. Your support is greatly appreciated!

How old are you? *

Short answer text

How often do you use a laundry machine? *

○ Daily

○ Weekly

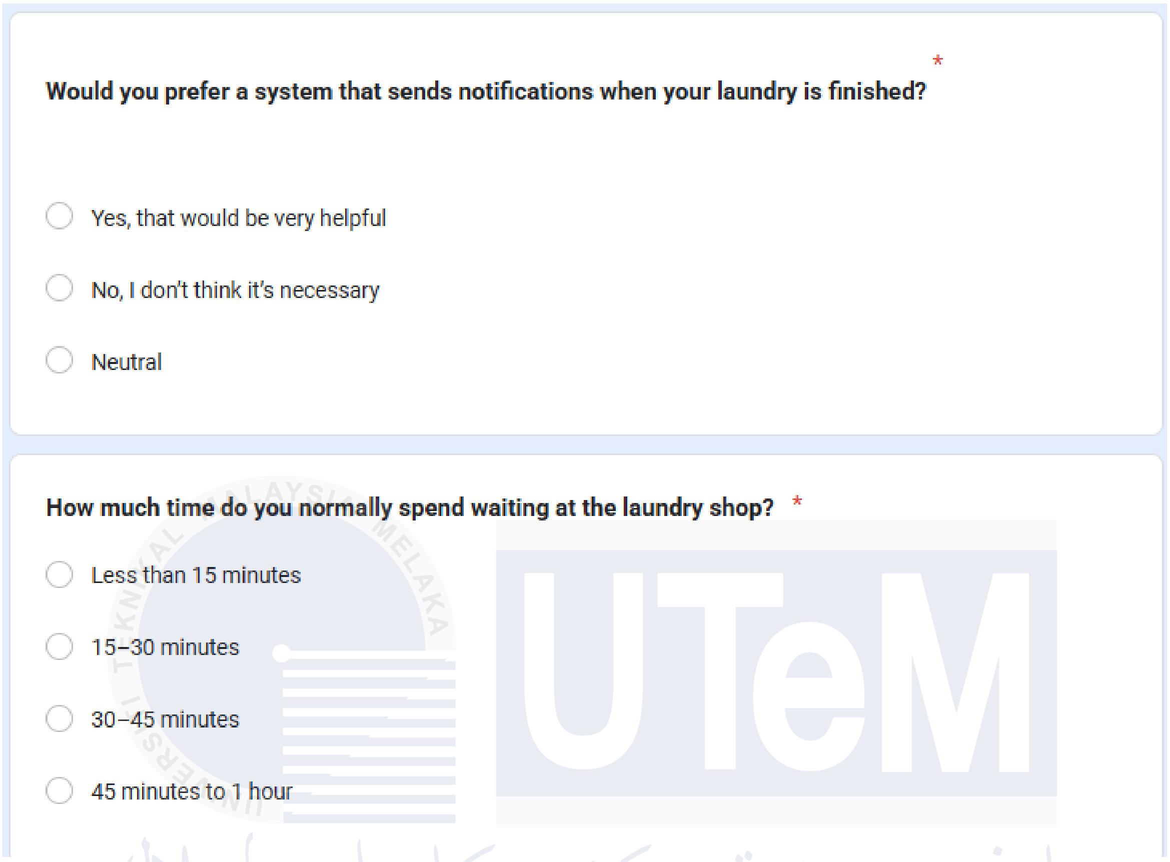
○ Rarely

Do you find it complicated to monitor your laundry manually? *

☐ Yes, it's complicated

☐ No, it's easy

☐ Neutral



Would you prefer a system that sends notifications when your laundry is finished? *

- ○ Yes, that would be very helpful
- ○ No, I don't think it's necessary
- ○ Neutral

How much time do you normally spend waiting at the laundry shop? *

- ○ Less than 15 minutes
- ○ 15–30 minutes
- ○ 30–45 minutes
- ○ 45 minutes to 1 hour