



Expertise
and insight
for the future

Danh Nguyen

Introduction to Industry 4.0 and Data Visualization in Embedded IoT System

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Bachelor's Thesis

Author	Danh Nguyen
Title	Introduction to Industry 4.0 and Data Visualization in Embedded IoT system
Number of Pages	48 pages + 2 appendices
Date	4 October 2020
Degree	Bachelor of Engineering
Degree Programme	Electronics
Professional Major	
Instructors	Janne Mäntykoski, Senior Lecturer
<p>The project goal concerns designing an IoT system for Industrial Management and Smart Home Application, which acquires various sensors' data, such as temperature, humidity, and pressure, visualizes them on the line chart, stores them in MySQL database of the web server domain, and sends email alert based on the control limit.</p> <p>At first, a local web server was hosted by the ESP32 which displayed the current temperature readings from a DS18B20 sensor, allowed the users to input the upper, lower temperature threshold, and the particular person's email which is responsible to ensure that the temperature is in the normal range. If the current temperature went to the dangerous field, an email would be sent. Moreover, the user can turn off the email function by checking the radio button on the web server.</p> <p>Secondly, the above design is enhanced to run on the user's specific web server domain provided by a free web hosting service to have some more useful and efficient functions. First of all, one extra sensor module was added to this project - the BME280 sensor kit which gains the humidity, temperature, and pressure data. Then, this system will send an email whenever the temperature, humidity, and pressure exceeds the control limit. Subsequently, the data from the sensors will be stored to the SQL database of the server and visualized in line chart and table forms in the web browsers which can be seen on all devices connected to the Internet from anywhere in the world. These two websites for this design are attached to the appendix 2 of this report.</p> <p>Based on the theoretical knowledge of the IoT Industry requirement, sensor module, this web server project is developed for a better understanding of the Industry 4.0 and IoT data visualization as a result of an embedded IoT system operating on a web server domain.</p>	
Keywords	Industry 4.0, ESP32, Internet of Things, MySQL, PHP

Contents

List of Abbreviations

1	Introduction	1
2	Theoretical Study	2
2.1	Industry 4.0 and the Development of IoT.	2
2.1.1	The First Industrial Revolution and the Second Industrial Revolution	2
2.1.2	The Third Industrial Revolution	3
2.1.3	Industry 4.0	3
2.2	IoT and Industrial IoT	5
2.2.1	IoT Development in the Industry	5
2.2.2	IoT Architecture	6
2.2.3	IoT Data Visualization	9
2.3	The Requirement for IoT Sensor	13
2.3.1	Durability	13
2.3.2	Accuracy	13
2.3.3	Versatility	14
2.3.4	Power Consumption	14
2.3.5	Cost	14
3	Introduction to the Hardware	14
3.1	ESP32 and ESP32 Development Board	14
3.1.1	Ultra-Low-Power Solution	15
3.1.2	Integration Method	15
3.2	Specification of ESP32 board	15
3.2.1	Wifi Connectivity	15
3.2.2	CPU and Memory	15
3.2.3	Peripherals, and Pinout Guide	17
3.3	Sensor Units	18
3.3.1	BME 280	18
3.3.2	DS18B20	22
4	Introduction to Software	23
4.1	Web Host Service	23

4.2	PHPMYAdmin	24
4.3	MySQL	24
4.4	Arduino Software	25
5	Introduction to the Programming Part	26
5.1	Local Web Server	26
5.2	Web Server with a Free Domain.	31
5.2.1	Programmed in the Arduino Software	33
5.2.2	Programmed by PHP Scripts.	34
6	Results	39
6.1	Local Web Server Control for the First Project.	39
6.1.1	Description	39
6.1.2	Flow Specification	40
6.2	The Final Web Server with Data Visualization	41
6.2.1	Description	41
6.2.2	Flow Specification	45
7	Conclusion	47
	References	49

Appendices

Appendix 1. Local web server prototype for the first project whose name is "first_project.ino".

Appendix 2. Second project prototypes for the web's server domain.

List of Abbreviations

API	Application Programming Interface
BI	Business Intelligence
°C	Degree Celsius
DBMS	Database management system.
°F	Degree Fahrenheit
GPIO	General Purpose Input and Output
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
IT	Information Technology
I2C	Inter-Integrated Circuit
ms	millisecond
MOSI	Master Output/Slave Input
MISO	Master Input/Slave Output
SCLK	Clock
SCL	Serial Clock
SDA	Serial Data
SS/CS	Slave Select/Chip Select

SoftAP	Software Enabled Access Point
SPI	Serial Peripheral Interface
SQL	Structured Query Language

1 Introduction

The first industrial revolution occurred when people started using water and steam power to operate machines in production, and then the use of electricity in wide manufacturing triggered the second industrial transformation. After that, the third industrial revolution occurred when automatic operation appeared with the help of modern computer and electrical automation software. The most recent transformation is Industry 4.0 which is the combination of all technologies including Science, Technology, Physics, and Biology.

The development of Industry 4.0 focuses mainly on four main areas which are Information Technology, Engineering, Biotechnology, and Physics. In the field of Engineering, artificial technology, big data, and IoTs are mostly targeted by industrial companies. The theoretical background of this project is based on the development of Industry 4.0, the architecture, and the requirements when designing an Industrial IoT system.

There are two practical parts in this project. The first one is to establish a local web server that includes the ESP32 connect to the DS18B20 sensor to display the current temperature data and allows the user to input two temperature thresholds and the Gmail's account. In addition, these input fields are required to send an alert email when the current temperature is too low or too high. The second part is to generate an IoT system with the writer's web domain administered by a free web host service. The web server will display several sensors data from BME280 and DS18B20 which are connected to the ESP32 and visualize them on the control line chart and in table form. When the humidity, temperature, and pressure data exceeds the normal range, the web service will send numerous emails to specific people until the sensor's data become normal. Owing to that system, users can manage information on websites with numerous mobile devices connected to the internet from all over the world.

Overall, through this project, various sensor data such as temperature, humidity, and pressure can be got via only one small IoT platform. It helps save a lot of space in some factories, and be more convenient for the employees, especially in manufacturing engineering to check the condition of the industrial environment more effectively. The sensor

data which is collected from the electronic device is not only in the number format, but also in the line chart form in order to improve in data visualization and analysis.

2 Theoretical Study

2.1 Industry 4.0 and the Development of IoT.

Figure one provides a brief introduction of four distinct industrial revolutions that the world has experienced until today.

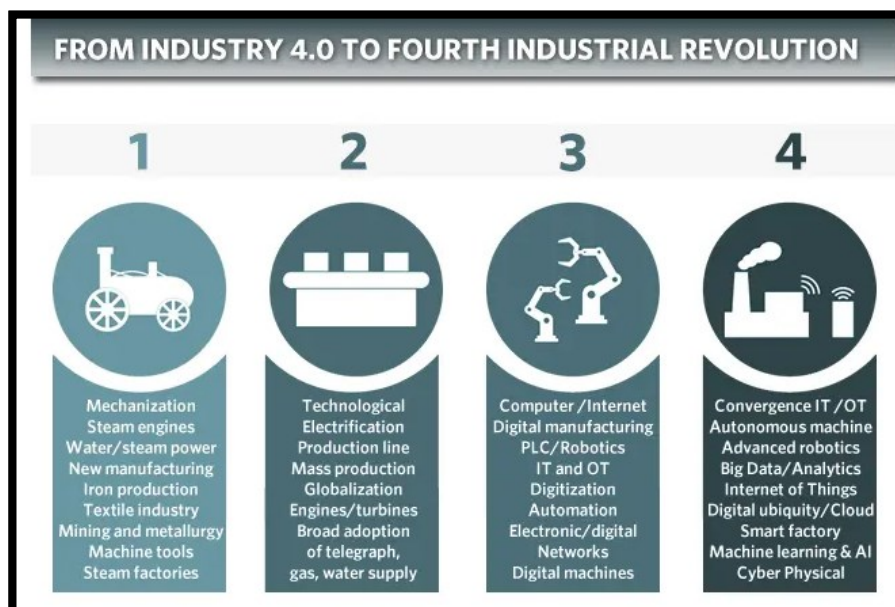


Figure 1. The Development of the Industry. [1]

2.1.1 The First Industrial Revolution and the Second Industrial Revolution

From the 1700s to the early 1800s [1], production transformed from concentrating on manual labor operated by individuals to the application of steam-powered engines, and other machine tools.

Consequently, between the late 19th and the early 20th century, the introduction of steel, electricity, gas, and oil in the factories enabled the abilities to construct the electrical assembly line for mass production in manufacturing to boost efficiency and increase productivity.

2.1.2 The Third Industrial Revolution

From the second half of the 20th century, as a result of the development of electronics, computers, and telecommunications, the industry had changed from putting less emphasis on mechanical technology to automating in production. In this transformation, there were two essential inventions which are Programmable Logic Controllers (PLCs) and high-level electrical automation Robots. [2].

2.1.3 Industry 4.0

The term Industry 4.0 was first created in Hannover Industrial Workshop in Germany in 2011 and mainly concentrates on digital technology with the help of the Internet of Things and cyber-physical systems for real-time data. In this way, products, machines, and processes can 'communicate' with manufacturers and engineer through intelligent networking. Figure 2 defines all the current trends in manufacturing technologies of the Industry 4.0.

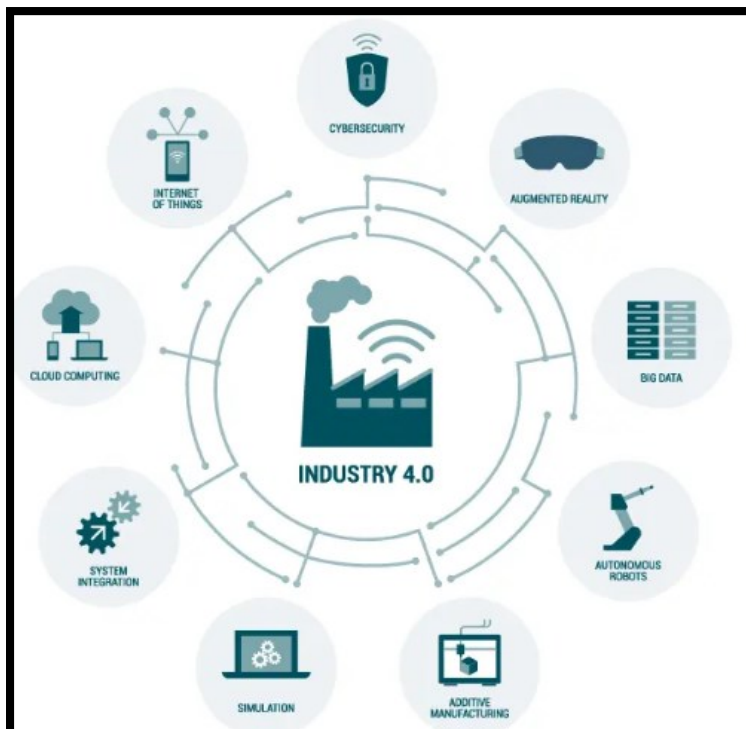


Figure 2. Main trends of the Industry 4.0 [1]

Industry 4.0 combines the latest scenarios of production and business processes from product life cycle to sales, quality, maintenance, and customer services. In particular, first of all, it improves the productivity and efficiency in manufacturing by decreasing the downtime and automated or semi-automated decision making with the assembly lines. Secondly, it increases the collaboration between your team members, departments, operators, and managers to fully control real-time data and ideas to acquire the most appropriate decisions. Thirdly, one of the most important advantages of industry 4.0 is the ability of the company to handle hidden problems which can become worse in the future. With the help of data predictive analytics and internet-connected machinery, manufacturers can predict the specific issues happening in the production or supply chain process. Finally, customers also gain a huge benefit from these “smart” factories as less machine and production line downtime comes with less overall operating costs. Furthermore, by combining technology and business, customers will have more high-quality and innovative products.

2.2 IoT and Industrial IoT

IoT is the use of the Internet for network-connected devices that are embedded in the physical environment. Besides, Industrial IoT is the application of IoT technology in manufacturing by controlling sensors and devices through cloud technology, or wireless automation [3]. These IoT applications provide some best features in the industry such as improving the speed of production through the high-efficient machine, enabling companies to take into consideration the health and safety of their employees, especially with those who work in a hazardous environment.

2.2.1 IoT Development in the Industry

Figure 3 demonstrates the global market spending on the Industrial IoT system for manufacturing. The trend of this manufacturing is supposed to rise dramatically from \$1.670B in 2018 to approximately \$12.44B in 2024. In a nutshell, in seven years, the average of the compound annual growth rate is roughly 40%.

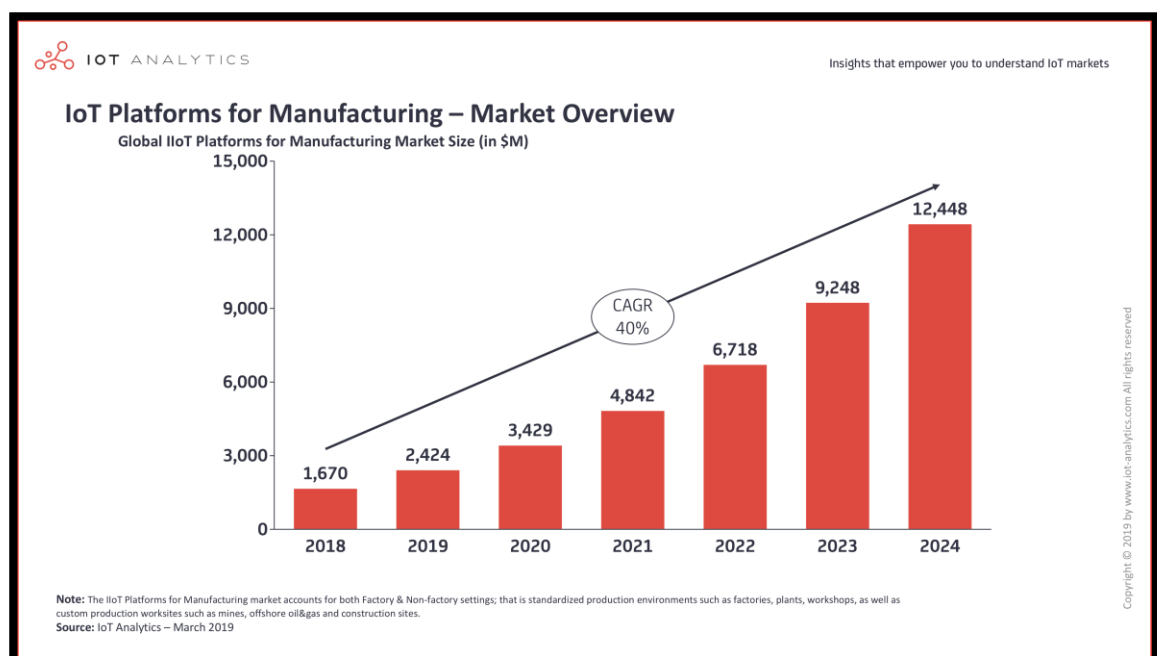


Figure 3. IoT Platforms for Manufacturing – Market Overview [4]

2.2.2 IoT Architecture

In general, to execute or design an efficient IoT system design, the architecture of IoT is required to be considered primarily. Figure 4 is the pictorial representation of the four main stages in IoT architecture

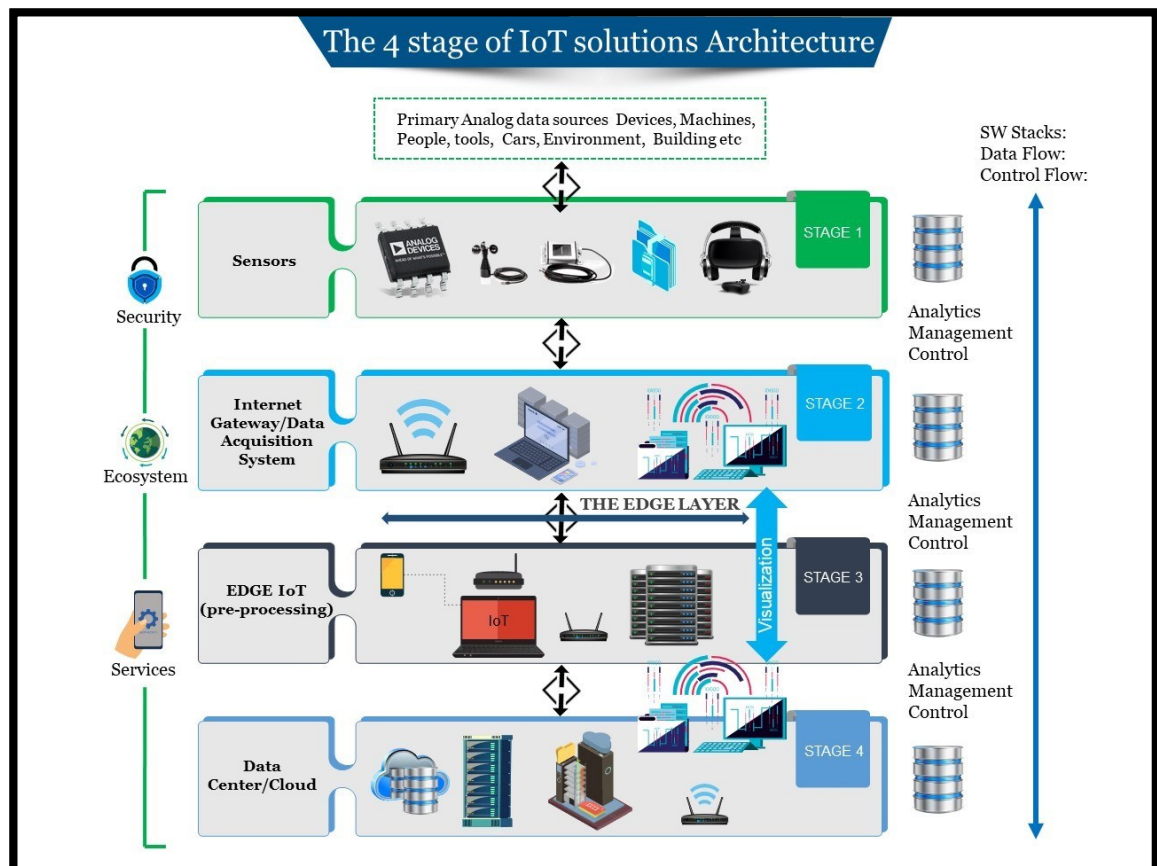


Figure 4. Four stages of IoT solutions Architecture [5]

2.2.2.1 Connected Devices

Sensors can convert the electrical signals into information or data that can be analyzed, while actuators can operate automatically based on the data from them. These devices are in the first stage of the IoT architecture to continuously collect data from numerous regions and then send them to the following layer. In particular, some popular sensors in the IoT system now are temperature, pressure, humidity, and RFID tags sensors. They

sent data to the server through some low power wireless networks which are Wi-Fi, ZigBee, Bluetooth, etc. whose logos are indicated in figure 5 below.



Figure 5. Low Power Wireless Network [6]

2.2.2.2 Internet Gateway/Data Acquisition System

The IoT gateway has two main functions. The first one is to control the bidirectional data between network and protocol, and the second is to translate these protocols to ensure the interoperability of the multiple sensors and electrical equipment. Furthermore, a certain degree of security is provided by the gateway to transmit data from one system to others with high encryption techniques [6]. In other words, it is a middle protection layer that prevents data sent to the cloud from detrimental and illegal access.

This acquisition system is mandatory to collect, filter, and transfer data to the next layer which is edge and cloud-based infrastructure. In particular, there are various IoT devices in this system which connected to the sensors to transform the electrical signal to the data signal before transferring to the next step, such as Microchip AVR-IoT WG development board, the low-cost board TI CC3220, and Arm-based IoT Kit for Cloud IoT Core-Raspberry BI.

2.2.2.3 Edge IT Data Processing

This stage is considered as a significant part of large-scale IoT projects while a huge amount of data is transferred from the sensors to the IoT Cloud [7]. In particular, the edge system includes the service, hardware, and software to analyze and pre-process the

information before sending it to the cloud [8]. Besides, edge computing prevents transferring raw data to the cloud by carrying out data cleaning, aggregation, and analysis on the device itself [9], and this will result in reducing bandwidth costs, traffic delays, and quicker response times in the system

When determining the appropriate IoT edge system, three main criteria are based on the ability to connect to modern and new cybersecurity technology, adapt to numerous legacy assets, and work on a variety of edge-cloud hybrid environments [10].

2.2.2.4 Analysis, Storing and Visualizing Data

One of the essential advantages of a modern IoT system is real-time data analysis which is extremely useful for system management and maintenance. In particular, the organizations can predict the tendencies of their production to invent new, innovative, and successful implementation.

To execute data analysis, the first required process is storing the data. Because of the requirement to accumulate and manage massive data efficiently in real-time, the IoT cloud has been used as a high-performance network to store data from different servers. However, one challenge in storing data on the cloud is security issues. Figure 6 below is one of the most comprehensive samples of IoT architecture for data protection.

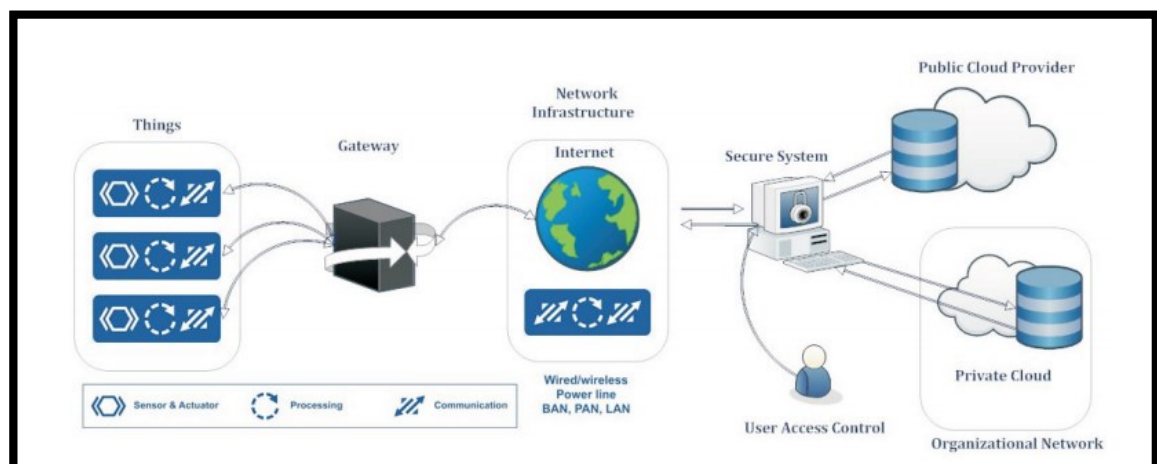


Figure 6. Common IoT Architecture for data protection [11]

In the above-proposed diagram, various data and information can be acquired by the IoT devices and if there is no internet connection, an intermediate gateway is used to provide

the important connection between the device and cloud. For security reasons, the administrator of the system will define appropriate roles for individuals in the organization according to their job functionalities to access data on the cloud. Besides, information from IoT equipment is in the encrypted format which allows only specific persons to access [11]. Two types of cryptography algorithms that can be used in the IoT system are symmetric and asymmetric algorithms. The first one encrypts and decrypts the data with the same secret key, while the second allows two keys for encryption and decryption separately. One of the most essential parts of the IoT cloud is the database management system. For example, SQL Server Management is widely used in some businesses nowadays to collect, analyze, and administer data.

The second stage to prepare for data analysis is data visualization. All the data generated from different IoT devices are quite complicated and impractical to analyze and control. However, for this reason, data in the IoT system is necessary to present in visual language. Data visualization helps businesses to realize hidden patterns and discover customers' current trends which change dramatically nowadays, based on real-time data analysis. Therefore, companies can get the appropriate messages from the customers conveniently and quickly to determine their adaptive methods and plans for revenue growth [12]. Some common data visualization tools in IoT systems nowadays are Power BI for real-time data visualization, Grafana for metrics visualization, and Kibana for Logs Visualization.

2.2.3 IoT Data Visualization

In the era of the Internet of Things, data visualization is one of the most significant challenges and barriers for numerous industrial companies to transform their ways of manufacturing to adapt to Industry 4.0. In particular, when a huge amount of raw data from several electronic devices needed to be visualized, not only does the company require plentiful competences from their employees, but also considers the benefits of this transformation, especially in the overall costs. In a small company, since a small amount of data and few customers' requirements are needed, data visualization can be executed in some simple tools such as Microsoft Excel or used some sample programming librar-

ies such as the Plotly, Matplotlib Python library. However, in a medium-size, some powerful data visualization tools such as Tableau and Power BI are required to use because of two essential reasons.

Firstly, these tools provide different functional protocols for the company to adapt to the customer requirements which change hastily nowadays. For example, there is a variety of functions for the users to calculate the total benefits of each month, year, and week, and the protocol to predict the device's condition based on the sensors' statistics. Secondly, to utilize all the advantages of data visualization, large companies require to hire numerous skillful specialists with a large amount of money since these competencies are still not quite prominent at present. In addition, when using programming languages in data visualization, it will be extremely difficult for the companies to insert new functions, or maintain if these languages are not compatible with the requirements of higher versions of the wireless protocols, or new and modern electronic devices. Therefore, these visualization tools are widely used by mid-sized companies for IoT designs because they always develop to adapt to customer's requirements, and they are also cost-effective for a large IoT system.

This report will summarize the use of Power BI as an example of the use of powerful software in IoT data visualization. The practical part of this thesis also uses Google Chart Libraries to visualize the data from the ESP32.

2.2.3.1 Power BI

It is a popular Business Intelligence Tool created by Microsoft to provide detailed analysis reports for large Enterprises. There are three products included in a Power BI suite which are Power BI Desktop, Power BI Services to design and publish specific reports into the Web Service, and an app on the mobile devices to manage the reports and dashboards. [12]

Power BI first gets the data from various sources such as Facebook, Google Analytics, Azure Cloud, SQL Server, Excel, etc. and then the raw data is transformed into power query data to use for visualizing and analyzing. Once the report is developed, it can be published to an internal web portal for security reasons. It means that only certain groups

of people with the web link can access to read, modify, or export the report in any preferred format such as pdf, excel, tables, etc. These groups can also create some different features for Power BI reports, such as refreshing updated data based on minutes, hours, days, months, etc. for real-time analysis and calculating some basic features of the business such as total sales, average salary per employee, and so on. Figure 7 describes the general workflow of Power BI.

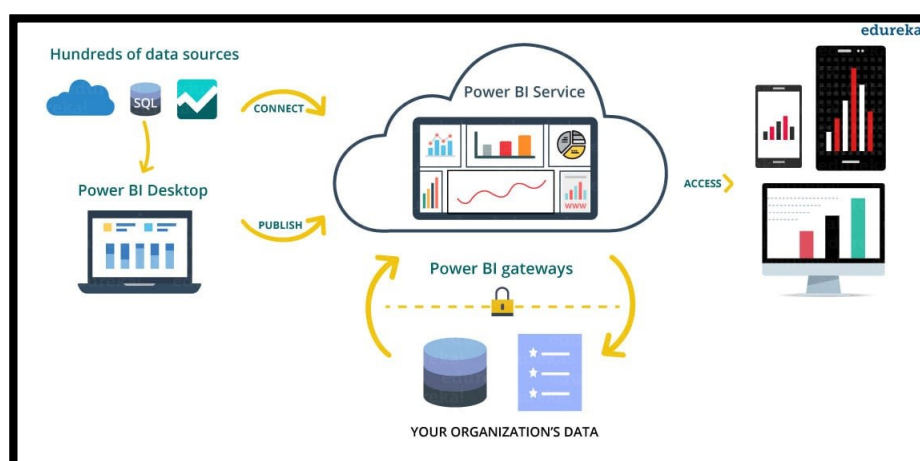


Figure 7. General workflow of Power BI service [13]

Although Power BI Suite is a paid service, it has a lot of highlights compared to other BI tools. Firstly, it is comparatively cheaper than other data visualization tools and it also offers various free services up to 1GB storage. Figure 8 below demonstrates all the free services of Power BI. Secondly, it can analyze big data in both streaming and static way, and thirdly, it has rich data graphs templates. For example, the library includes the bar, line, donut, column, map (online and filled map) charts, and developers may write Python code or use the R scripting language to design the chart much easier, especially when uploading into the internal web server. Finally, Power BI has a short learning curve with the use of power query natural language. Everyone who has used Microsoft Office before can learn Power BI quickly.

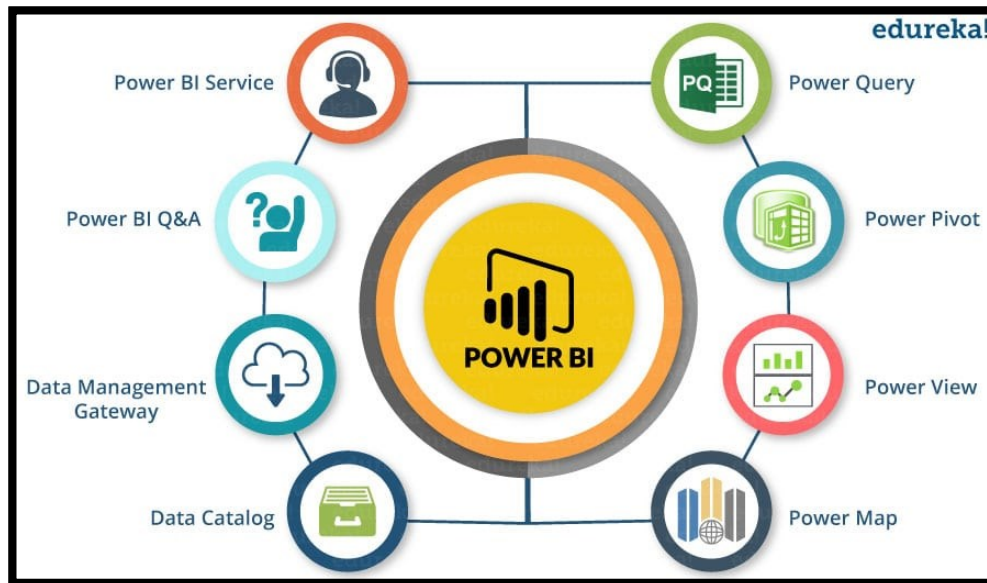


Figure 8. Main components of Power BI [13]

2.2.3.2 Google Visualization

In this thesis project, Google chart tools are used to visualize the various sensors' data from ESP32 because of numerous reasons. First of all, although it is a free software compared to Power BI, it provides numerous chart galleries from the simple pie chart, line chart, and bar chart to complex hierarchical treemaps. Figure 9 below demonstrates the sample dashboard of Google Chart.

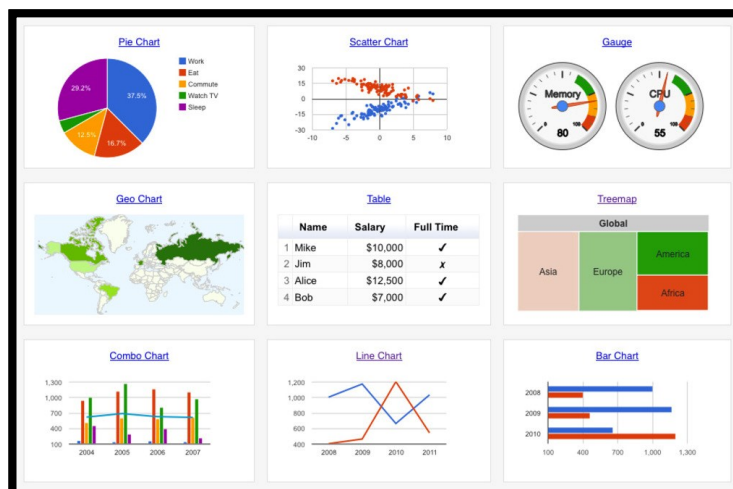


Figure 9. Example of Google Charts' Dashboard [14]

Secondly, the Google Charts can be embedded in the writer's web page with the help of JavaScript which is one of the fastest-growing and most common web programming languages for a junior developer to learn. Thirdly, thanks to the high interaction, these charts can create events that enable the programmer to create complex dashboards and other experiences such as sending an email, sorting, modifying, and filtering data integrated with the webpages [15]. At last, by applying the HTML5 technology which provides cross-browser compatibility to iPhones, Ipads, and Android, these charts are more convenient for the users to observe and control the data from their devices from all over the world.

2.3 The Requirement for IoT Sensor

The location of the IoT sensor and device depends mostly on human interaction. They can be placed in a remote location or embedded in a system, where people cannot access it. In general, the IoT sensors' selecting process is depending on five factors which are durability, accuracy, versatility, power consumption, and cost.

2.3.1 Durability

It is essential to ensure that the IoT device can operate for a certain period without causing unnecessary costs. For example, a water-resistant temperature sensor can be used for a remote weather station, but it is unable to use to control water temperature in a pool because it is not waterproof.

2.3.2 Accuracy

Accuracy is depending on mostly the environment in which the IoT device is used. For example, when designing a remote medical device system, the medical temperature sensor requires to have accuracy from approximately ± 0.2 degrees while for smart home applications, the sensor can be accurate with a ± 1 degree error [16].

2.3.3 Versatility

In the IoT network, it is essential to have sensors that can operate in different variations of the environment. For example, in remote weather stations for a country which have four seasons, it is significant to find some sensors which be capable of operating in extremes temperatures in summer and winter.

2.3.4 Power Consumption

For an efficient-power IoT system, a low-power, high-power, or even very low-power devices must be selected based on different conditions. For example, a sensor or device powered by solar-charged batteries may need to spend a great portion of its life in sleep mode to prolong battery life during low-light times. It may also require fast wake-up times to capture data appropriately even when small solar energy is provided.

2.3.5 Cost

Cost is considered one of the most important impacts when designing an IoT system because, in this network, there are hundreds or even thousands of sensors and devices. These costs involve more than just the price of the sensor because some considerations must be given to the cost of placement, maintenance, reliability, etc.

3 Introduction to the Hardware

3.1 ESP32 and ESP32 Development Board

In this thesis, the ESP32 DEVKIT DOIT board was mainly used which demonstrates numerous features of one IoT kit for junior developers. In particular, two main features of this board are the ultra-low-power and complete integration solution.

3.1.1 Ultra-Low-Power Solution

Thanks to the characteristics of low-power chips, the ESP32 has been used widely in the IoT system and some other wearable electronic mobile applications. To reduce the amount of energy used by the chip to a minimum, it is mandatory to modify the duty cycle to a low value [17]. In addition, the adjustability characteristic of the power amplifier's output can create an efficient optimal trade-off among the communication range, data rate, and power consumption. From the ESP32 datasheet [18], the minimum power supply voltage for the ESP32 is 3.0V.

3.1.2 Integration Method

ESP32 has approximately twenty external components, some of them are antenna switch, filters, power amplifier, and the power management modules. However, these parts can be handled in a small Printed Circuit Board, and those also make the ESP32 module has a high-integrated solution, especially in Wifi, and Bluetooth.

3.2 Specification of ESP32 board

3.2.1 Wifi Connectivity

ESP32 implements Wifi at a 150.0 Mbps data rate and supports software that enables a computer to become a router or a wireless access point and some other basic service set under numerous distributed control functions [19].

3.2.2 CPU and Memory

ESP32's processor integrates a Tensilica Xtensa Dual-Core 32-bit LX6 microprocessor, running at 240 MHz (in the DOIT development board). In addition, it has 448KB Rom, 520kB SRAM, and 4MB of the flash memory to execute a program.

Table 1 and Table 2 below demonstrate the similarities and differences in using ROM, Flash, and RAM which includes two types-DRAM and SRAM memory.

Table 1. Difference between ROM and Flash Memory [20]

Difference between ROM and Flash Memory	
ROM	FLASH
Data stored in are nonvolatile which will not be removed when there is no power supply.	
- It is read-only memory, so data cannot be written or modified.	- The data can be read and write.
- The data cannot be removed by any electronic devices.	- The data can be erased or written by electronic devices or by applying the electrical field to the chip.
- Higher read speed.	- Slower read speeds compared to ROM/RAM.

Table 2. Difference between SRAM and DRAM Memory [21]

Difference between SRAM and DRAM Memory	
SRAM	DRAM
Data stored in are volatile which means that it is lost when power is removed	
- SRAM has better performance in terms of speed of operation because of the lower access time.	- DRAM has lower performance because of the higher access time.
- Transistors and latches are mainly used in SRAM.	- Capacitors are mainly used in DRAM, with few or no transistors.
- SRAM needs a constant power supply, so it consumes more power.	- DRAM can store the information in the capacitor, so it reduces power consumption.
- SRAM's form is on-chip memory.	- DRAM's form is off-chip memory.

3.2.3 Peripherals, and Pinout Guide

Two essential components of the microcontroller are the processor and peripherals. The core of the integrated circuit in the microcontroller is called the processor which performs various functions such as controlling and communicating with other electronic devices. Nevertheless, this core requires some other parts to finish executing specific tasks. Therefore, some other external or internal peripherals have been soldered in the breadboard to help the microcontroller interface directly or indirectly with other systems such as relays, motor controller, keypad, seven-segment display, and electronic and electrical sensors.

ESP32 was used in this project which has numerous peripherals indicated in table 3 below.

Table 3. ESP32 In General [22]

ESP32	
Digital Input/Output Pins	25
Analog Input Pins	6
Analog Outputs Pins	2
UARTs	3
SPIs	2

In order to select specific pins for different functions of the ESP32, figure 10 below is one of the most essential parts for the developers to consider carefully when designing an appropriate program. In particular, to use the I2C communication of the ESP32, two pins are needed which are GPIO22 and GPIO21.

humidity, and pressure sensor and a successor to sensors like BMP180, BMP085, or BMP183.

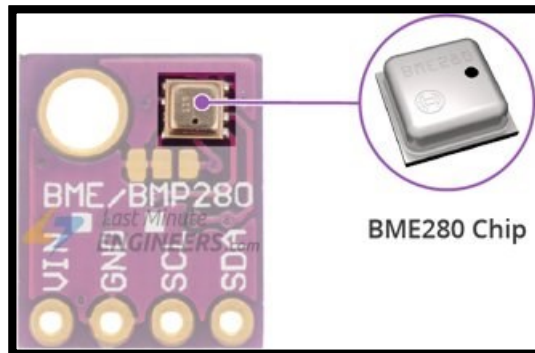


Figure 12. BME280 Bosch Chip [24]

This module can measure the relative humidity, pressure, altitude, and temperature with the precision indicated in figure 13 below.





	Temperature: -40°C to 85°C ($\pm 1.0^\circ\text{C}$ accuracy)
	Humidity: 0 to 100% RH ($\pm 3\%$ accuracy)
	Pressure: 300hPa to 1100hPa ($\pm 1\text{hPa}$ accuracy)
	Altitude: 0 to 30,000ft (± 1 meter accuracy)

Figure 13. BME280's precision [24]

Regarding the power requirement, this module can be used without problems with the 3.3V or 5V logic microcontroller such as the Arduino board, and it consumes less than 1 mA during continuous measurements.

3.3.1.1 BME280 Sensor Pinout

Figure 14 demonstrates the pinout guide of BME280.

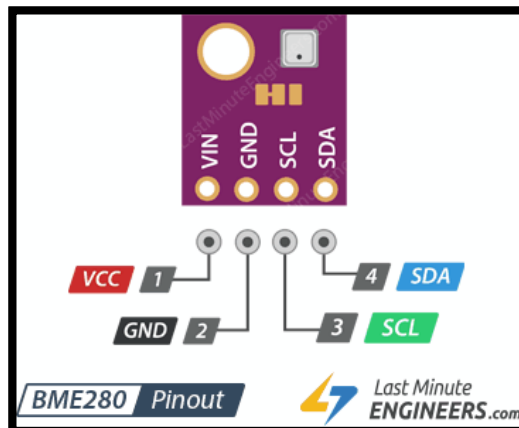


Figure 14. BME280 Pinout Guide [24]

Furthermore, the BME280 module features a simple two-wire I2C or SPI interface to communicate with the microcontroller. However, only the I2C communication protocol will be used in this project.

3.3.1.2 BME280's communication protocol

BME280 provides two efficient communication protocols which are the SPI and I2C. This part will cover some basic introductions of these protocols to explain why the I2C was used in this project for the ESP32 to receive and transmit data from the BME280 sensor module.

In electronic communication, there are two types of communication which are serial and parallel interfaces. While the serial communication requires that one bit is sent continuously via a single wire, the parallel protocol allows that all data bits are transmitted concurrently through different wires. Figure 15 below demonstrates the differences between the serial and parallel communication protocols.

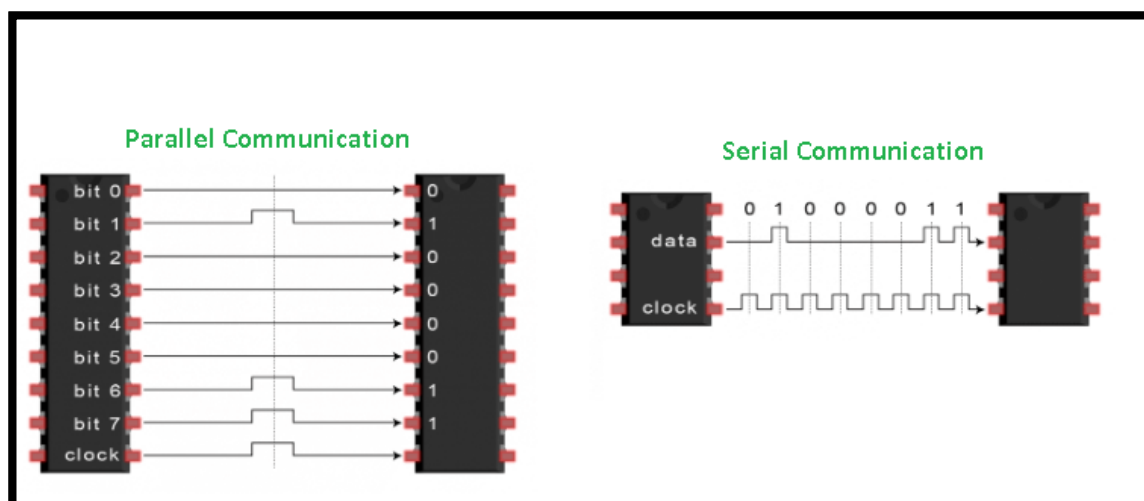


Figure 15. Difference between serial and parallel communication protocols [25]

SPI is the serial master-slave communication interface that requires three or four wires to operate efficiently for different purposes. In SPI, a master can control and connect with one or multiple slaves, but only one master is allowed to use. Furthermore, compared with I2C, the data in SPI is transmitted at a higher rate with the maximum data rate up to 10 Mbps [25]. However, it requires four lines which are MOSI, MISO, SCLK, SS. To be more specific, MOSI and MISO are the signals from the master to slave and reverse respectively. The data is sent based on the SCLK clock signal created by the master. If multiple slaves are used, the CS and SS wire will be used for the Master to check for the appropriate slave.

I2C is also a serial interface which requires two wire for the clock and data signal. These two wires' symbols are SDA and SCL. Compared to the SPI, I2C can transfer and receive data between multiple masters and slaves [26]. Particularly, in I2C, based on the start condition, a message was sent to the address defined in the address frame. After that, if the message was sent successfully, the acknowledge bit would be returned. At the final stage, If the stop condition was sent from the master, the data transmission would be stopped. The disadvantage of I2C is that it has a slow data rate whose maximum full speed is 400 kbit/s. However, it is widely used since it has only two wires and its simple setup in the programming parts.

3.3.2 DS18B20

The DS18B20 is a one-wire digital temperature sensor that needs only one line to send or receive data with another micro-controller [27]. Figure 16 below describes the pinout of this sensor, the left sensor in the picture is the waterproof version of DS18B20.

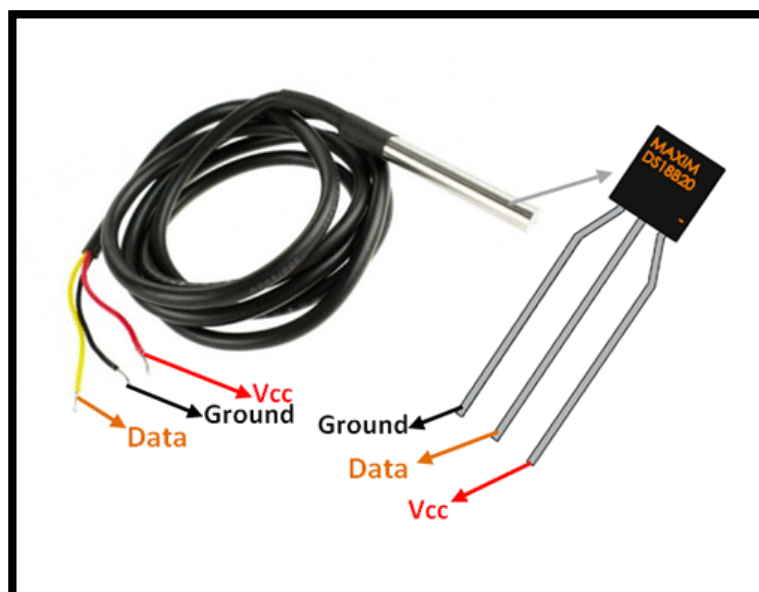


Figure 16. Two types of DS18B20 and pinout guide [28]

In this project, the DS18B20 Digital temperature Sensor (TO-92) from Dallas Semiconductor will be applied, which includes in figure 17. The power supply of this sensor ranges from 3.0V to 5.5V and temperatures can be calibrated between -55°C and $+125^{\circ}\text{C}$. Compared to other temperature sensors such as BME280, the accuracy of this sensor is higher which is $\pm 0.5^{\circ}\text{C}$ accuracy from -10°C to $+85^{\circ}\text{C}$. In addition, it reads thermometer data from 9 to 12-bit in a maximum of 750 ms to convert to a digital word [29]. To be more specific, figure 17 below demonstrates the pin description of the DS18B20 TO-92 package.

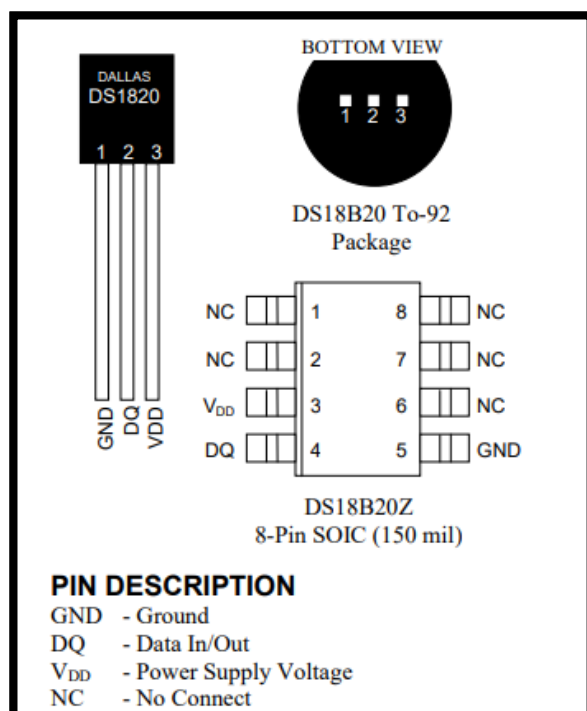


Figure 17. Dallas DS18B20's pin description.

4 Introduction to Software

4.1 Web Host Service

The goal of this thesis was to design the writer's web domain name which indicates in the appendices and a global hosting account that enables users to save sensors' information from the ESP32 and visualize them on the web browser. However, building its web server domain requires various things such as paying a huge amount of money every month, registering the website based on each country's rule, encrypting the website with a high-security password in order not to be hacked or stolen data by numerous anonymous organizations. Therefore, in this project, a hosting service has been used. Compared to other services, such as Bluehost, Digital Ocean, the 000webhost has some specific features which are providing the user to have one free website with 300 MB disk space, and create one database on MySQL using the PHPMyAdmin software. Figure 18 below is the logo icon of the 000webhost service.



Figure 18. 000webhost logo. [30]

4.2 PHPMyAdmin

PhpMyAdmin, which is also called hypertext preprocessor my admin, is a free software tool developed in PHP programming language to design the web interface and operate on the MySQL database. In particular, it supports a huge range of operations on MySQL such as browsing, creating, updating, and dropping databases, tables, and managing all users' accesses and stored procedures. In addition, PHPMyAdmin on the web server can create some functions to send emails, and connecting to the data of the IoT module. Figure 19 below is the logo of the phpMyAdmin tool.



Figure 19. PHPMyAdmin software. [31]

4.3 MySQL

SQL stands for Structured Query Language is a database accessing and manipulating language. In general, each database requires one or more distinct APIs for creating, searching, managing, and replacing the data it holds. In order to execute SQL, and use the database's API, PHPMyAdmin is a fast, open-source license, and easy-to-use relation database management system software. Figure 20 below demonstrates the use of

SQL query on the PHPMyAdmin platform to create “ThesisSensorData” table which includes two columns of character types which are “sensor” and “temperaturedata”.

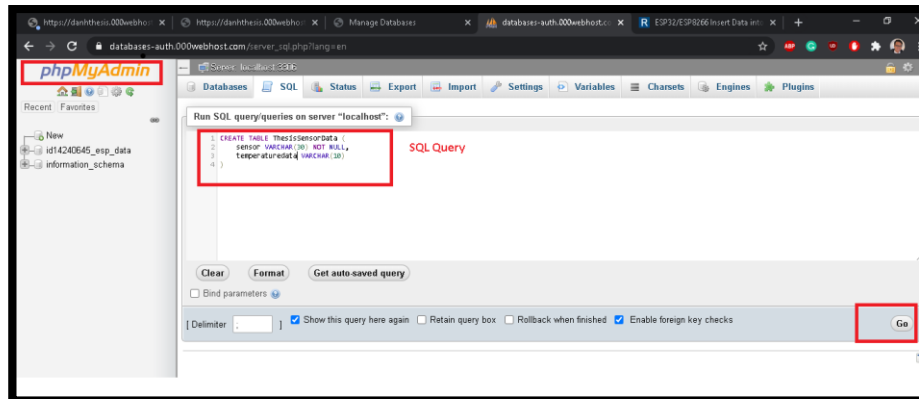


Figure 20. SQL Query on PHPMyAdmin.

4.4 Arduino Software

This project uses Arduino 1.8.13 software which is the latest version to operate with the ESP32 and connect to the web's database. However, to execute these thesis projects, it is essential to install numerous libraries into this platform. Figure 21 demonstrates the use of the Arduino version 1.8.13 to program the practical parts of this thesis.

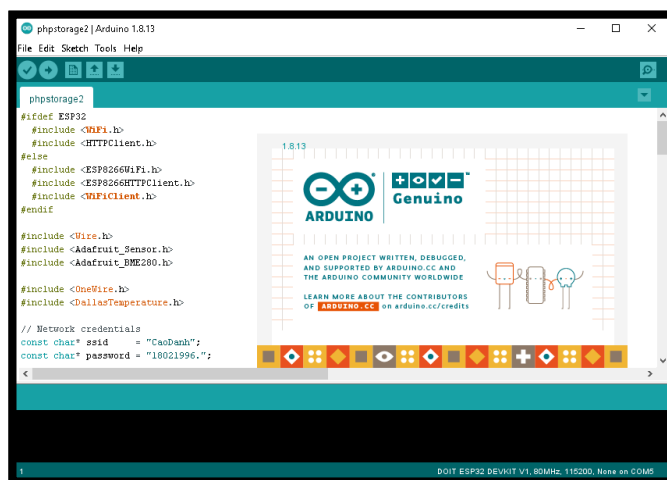


Figure 21. The Arduino Platform.

5 Introduction to the Programming Part

5.1 Local Web Server

In this part, only one file executed in Arduino CC software has been used which is named `first_project.ino` and attached in appendix 1 of this report. However, before starting writing code into the IDE, there are numerous steps needed to be executed. These tasks are summarized in figure 22 below.

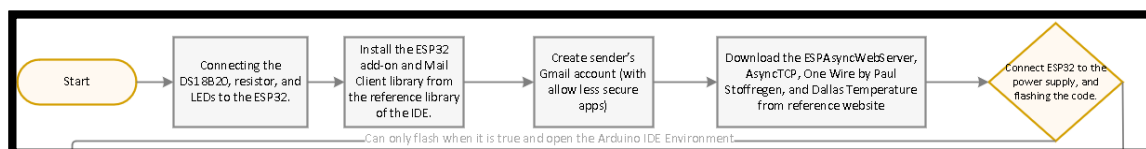


Figure 22. First project's preparing steps.

In the preparation process, a Gmail account should be created to enable the option “Allow less secure app” for the ESP32 to login and send the emails. This stage is indicated in figure 23 below.

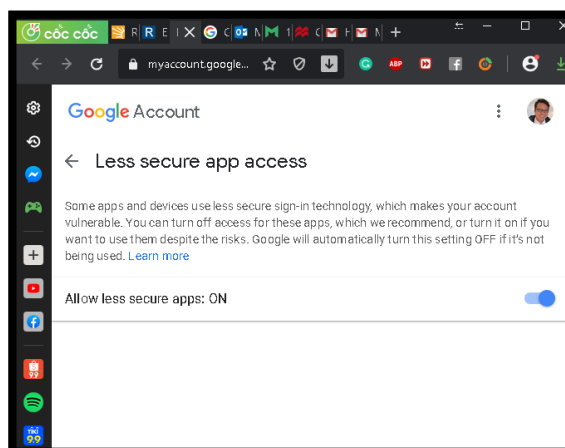


Figure 23. Less secure app access in Gmail.

In addition, to communicate with the DS18B20 temperature sensor, One Wire by Paul Stoffregen and Dallas Temperature libraries have to be installed. To create an asynchronous web server and send email alerts, the ESPAsyncWeb server, AsyncTCP, and the Mail Client libraries are also instituted. These libraries are indicated in figure 24 below.

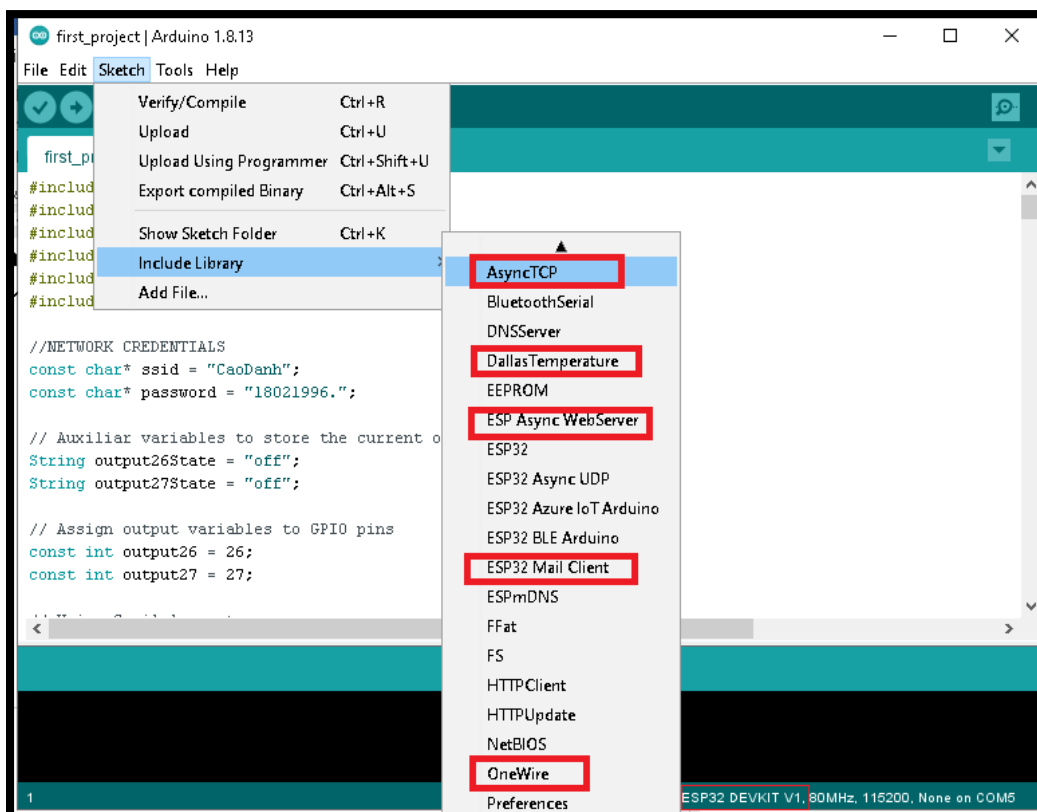


Figure 24. First project's necessary libraries.

Regarding the programming part, the first step is to write the code for the setting of the simple mail transfer protocol which comprises the email account's name, password, email subject, and Gmail's SMTP port. Then it is essential to define some variables to store the default receiver's email address, the minimum, and maximum temperature thresholds. Figure 25 describes the code for all these settings.

The screenshot shows the Arduino IDE interface with the following code and annotations:

```

// Using Gmail Account
#define emailSenderAccount "danwin1802@gmail.com"
#define emailSenderPassword "danh18021991."
#define smtpServer "smtp.gmail.com"
#define smtpServerPort 465
#define emailSubject "[ALERT] ESP32 Temperature"

// Default Recipient Email Address
String inputMessage = "caodanh1802@gmail.com";
String enableEmailChecked = "checked";
String inputMessage2 = "true";

// Default Threshold Temperature Value
String inputMessage3 = "21.0"; //low threshold
String inputMessage4 = "27.0"; //max threshold
String lastTemperature;

```

Annotations on the right side of the code:

- SMTP's setup protocol**: Points to the first block of code (SMTP server and account details).
- Define the receiver email address**: Points to the second block of code (recipient email and related variables).
- Define default threshold and variable to store the last temperature of the DS18B20**: Points to the third block of code (temperature thresholds and last temperature variable).

Figure 25. SMTP protocol, default receiver, and threshold variables' declaration.

In the second stage, the AsyncWeb server protocol has been used in this project to build the front end characteristic of the webpage and assign the data input from the webserver to all the appropriate variables in the code. Particularly, the layout of the webpage is programmed in HTML and CSS code, stored in an array, and sent by the HTTP request function to the local web server. These codes are also constituted the preprocessor variables which are the users' inputs from the website. The coding for this part is demonstrated in figure 26.

```

first_project | Arduino 1.8.13
File Edit Sketch Tools Help

first_project

// HTML web page
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html><head>
<title>Email Notification with Temperature</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}
form {
  display: inline-block;
  width: fit-content;
  height: fit-content;
  background-color: rgb(218, 165, 32, 0.5);
}
</style>
<body>
<h2>DS18B20 Temperature</h2>
<h3>TEMPERATURE: <deg>C</h3>
<h2>ESP Email Notification</h2>
<form action="/get">
  Email Address <input type="email" name="email_input" value="%EMAIL_INPUT%" required<br>
  Enable Email Notification <input type="checkbox" name="enable_email_input" value="true" %ENABLE_EMAIL%><br>
  Low Temperature Threshold <input type="number" step="0.1" name="low_threshold_input" value="%THRESHOLD%" required<br>
  Max Temperature Threshold <input type="number" step="0.1" name="max_threshold_input" value="%THRESHOLD%" required<br>
  Turn off minimum threshold device <input type="checkbox" name="low_threshold_device" value="true" %TURN_OFF_LOWTHRESHOLD%><br>
  Turn off maximum threshold device <input type="checkbox" name="max_threshold_device" value="true" %TURN_OFF_MAXTHRESHOLD%><br>
  <input type="submit" value="Submit">
</form>
</body></html>rawliteral";

```

Figure 26. Front-end characteristic of the first project.

After that, the processor function replaces all the input values from the users with the software variables, such as lastTemperature, inputMessage, enableEmailChecked, inputMessage3 (for the low threshold), inputMessage4 (for the high threshold), lowthresholddevice, and maxthresholddevice variables. The preprocessor function is indicated in figure 27.

```

first_project | Arduino 1.8.13
File Edit Sketch Tools Help

first_project
// program the processor variable of the web server with the data from the device
String processor(const String var) {
  //Serial.println(var);
  if (var == "TEMPERATURE") {
    return lastTemperature;
  }
  else if (var == "EMAIL_INPUT") {
    return inputMessage;
  }
  else if (var == "ENABLE_EMAIL") {
    return enableEmailChecked;
  }
  else if (var == "THRESHOLD") {
    return inputMessage3;
  }
  else if (var == "THRESHOLD1") {
    return inputMessage4;
  }
  else if (var == "TURN_OFF_LOWTHRESHOLD") {
    return lowthresholddevice;
  }
  else if (var == "TURN_OFF_MAXTHRESHOLD") {
    return maxthresholddevice;
  }
  return String();
}

// Using flag variable to keep track if email notification was sent or not, and set the output state
bool emailSent = false;

```

Figure 27. Processor function of the first project.

Then some pointer variables are used to check whether HTTP GET request was sent, for example, `if(request->hasParam(PARAM_INPUT_3))` condition to verify if the user inputted the value low threshold temperature. When the request contains input parameters, they will be assigned to the variables, and those which have no request will remain their previous values. In the `loop()` function, the timer of the ESP32 is used to acquire new temperature readings every five seconds, and another protocol is to set up the conditions to send the email alerts.

In this project, the email alert will be sent only in these three conditions. First of all, the current temperature is above, or below the threshold. Secondly, the checkbox of enabling the email notifications is ticked on the web page, and at last, the email has not been sent yet based on the boolean variable `emailSent`.

5.2 Web Server with a Free Domain.

Compared to the first project, this design introduces numerous good features. First of all, the users can get more data such as temperature, humidity, pressure from the BME280 sensor module, and the DateTime value when updating the number to the table "SensorData1" of the database. Secondly, this server uses the HTTP POST request to publish all the sensor data from the ESP32 to the web domain, and these statistics will be stored in the SQL database for further analysis, such as for predictive maintenance and quality management. Thirdly, one issue is that this project requires a long time to execute all the above functions, so the time interval between its post is thirty seconds while the response period in the first project is just only five seconds. Lastly, this project will comprise two websites for visualizing the data in the table format and line chart form, and they can be seen by using all devices connected to the Internet globally through the web browser.

Before starting to program this second design, there are numerous essential preparation processes. Firstly, equivalent to the first project, the Arduino platform required to install all the important libraries which are One Wire by Paul Stoffregen, Dallas Temperature for the DS18B20, the Wifi, HTTP clients to send and receive requests to the web server, and the Adafruit_Sensor for the BME280 sensor. In addition, a Gmail account which allows the less secure app access should be created. After that, the breadboard should be connected based on the circuit and schematic diagram in figure 28 and 29 below.

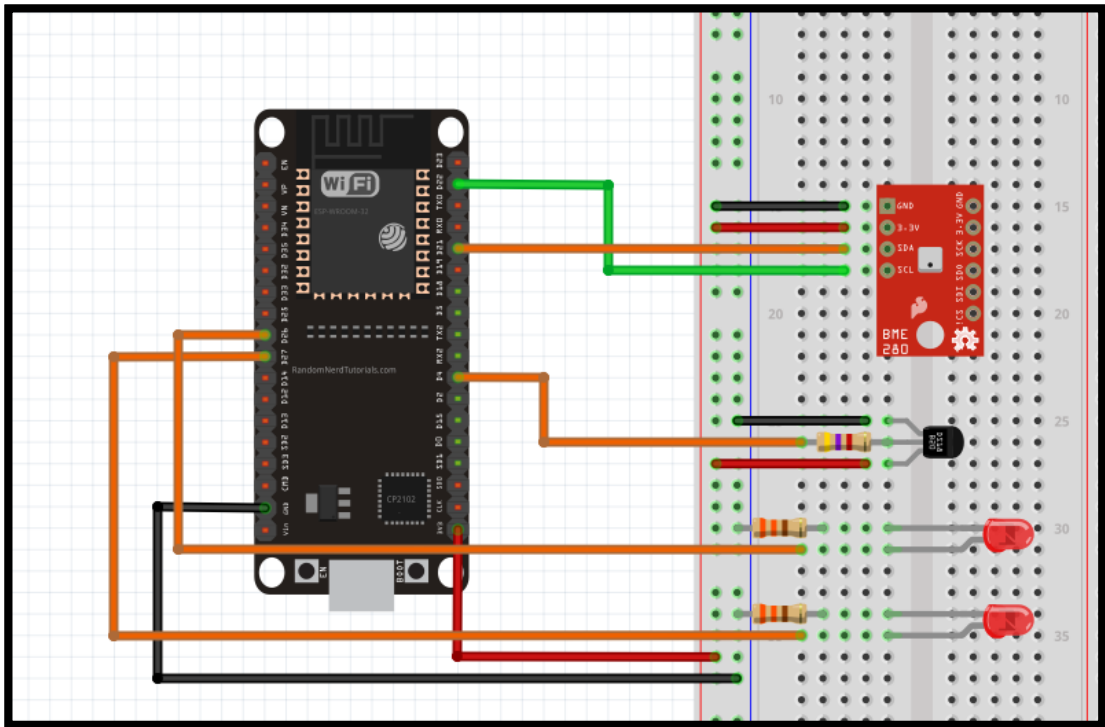


Figure 28. Circuit Diagram

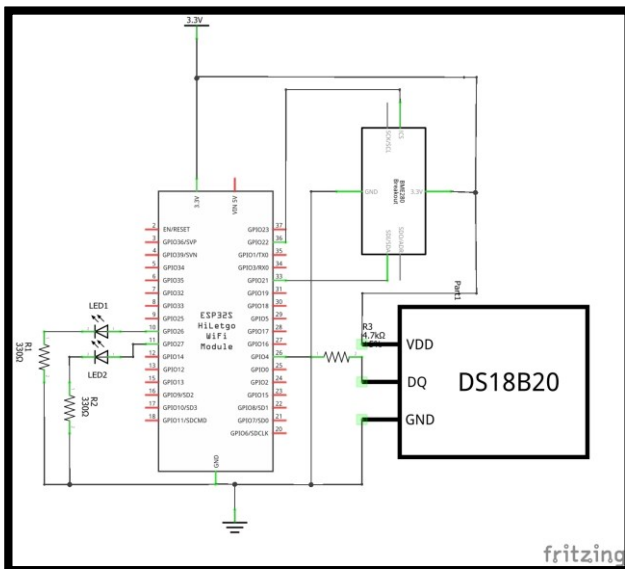


Figure 29. Circuit Schematic

In a real experiment, the breadboard with all essential connections should present equivalent to figure 30.

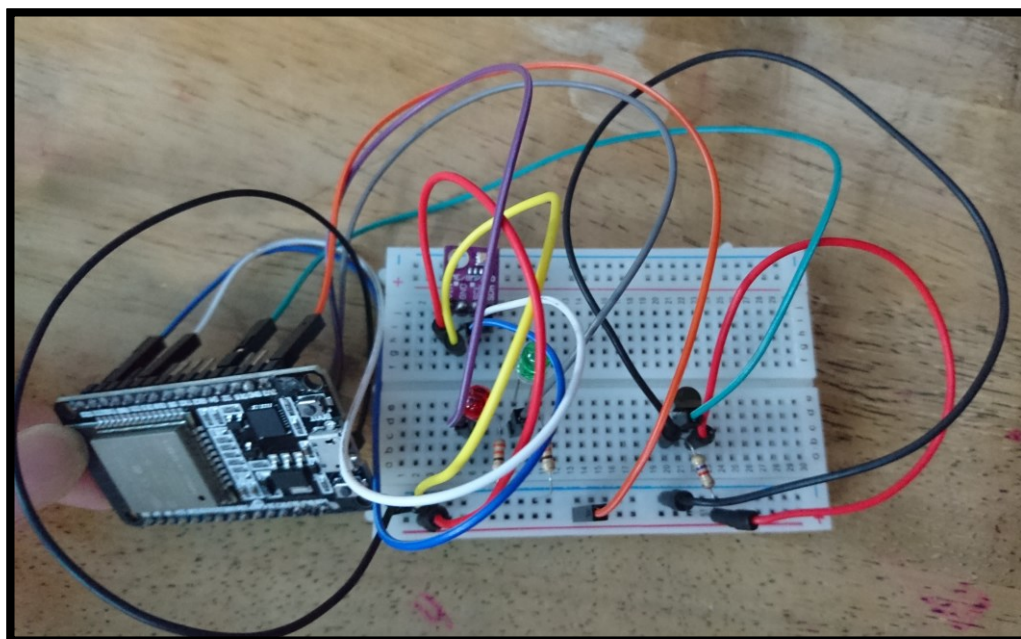


Figure 30. Circuit breadboard

5.2.1 Programmed in the Arduino Software

For the Arduino software file name “second_project.ino” to deliver data from the ESP32 to the web server domain, all these steps must be followed. First of all, the network user id and password required to be given in the code for the ESP32 to connect to the Internet and then used the array pointers to put the name of the website which is “https://danhthesis.000webhostapp.com/thesis/post-esp-data.php” for the ESP32 to send all its readings to. Secondly, this code would provide a random and characteristic string variable apiKeyValue which was the security key to protect the main database from being read and written by other anonymous users. Thirdly, the protocol to send the HTTP request to store the data from the ESP32 to the specific columns of the web’s database was indicated in figure 31 below.

```

secondproject.ino | Arduino 1.8.13
File Edit Sketch Tools Help

secondproject.ino 6
//Check WiFi connection status
if(WiFi.status() == WL_CONNECTED){
  HTTPClient http;

  // Your Domain name with URL path or IP address with path
  http.begin(serverName);

  // Specify content-type header
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  // Prepare your HTTP POST request data
  String httpRequestData = "api_key=" + apiKeyValue + "&sensor=" + sensorName
    + "&location=" + sensorLocation + "&temperature1=" +
    String(bme.readTemperature()) + "&humidity=" +
    String(bme.readHumidity()) + "&pressure=" +
    String(bme.readPressure()/100.0F) +
    "&temperature2="+String(temperature)
    + "";

  // Send HTTP POST request
  int httpResponseCode = http.POST(httpRequestData);
}

Done Saving.

117 DOIT ESP32 DEVKIT V1, 80MHz, 115200, None on COM5

```

Check Wifi Status

The server name is
 const char* serverName =
 "https://danhthesis.000webhostapp.com/post-esp-data.php";

Send the data to the database

Figure 31. Request to store the data in the second project.

5.2.2 Programmed by PHP Scripts.

To design the writer's domain provided by the web host service, three PHP script files are required to store in the web's file managers. In particular, one file is to connect and store the data sending from the Arduino software (which are connected to the ESP32) into the specific database of the server, and the others are visualizing those data in the table and line chart forms. The guides to design these PHP scripts are summarized apparently in figure 32 below.

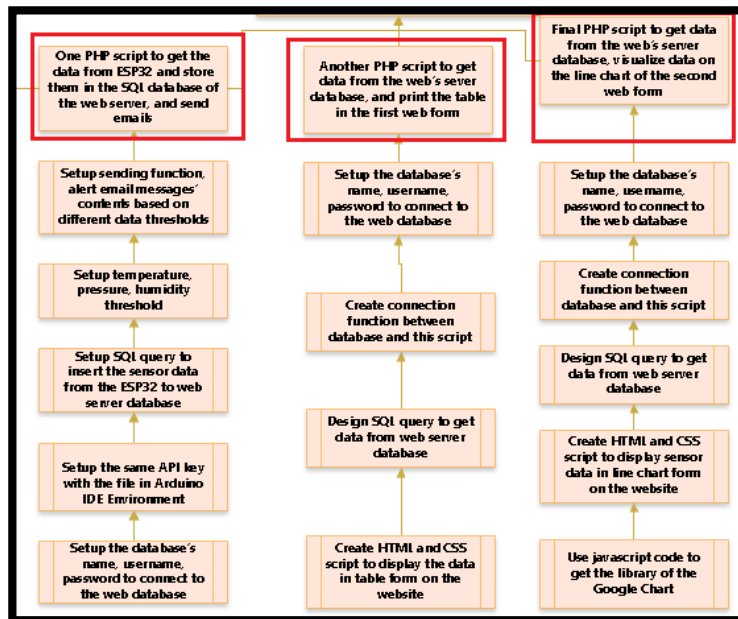


Figure 32. PHP scripts in brief.

The beginning file's name is `post-esp-data.php` whose main functions are accumulating the data from ESP32 to the server's database and sending emails. There are numerous defined variables such as the database's name, the username, and the password of the server to connect to the database. In addition, the API key value is the most essential one to accomplish the requirement of acquiring data from the Arduino platform. The stage to check if the API key and all the required parameters are similar to the others in the Arduino platform is demonstrated in figure 33 below.

```

3  $servername = "localhost";
4
5  // REPLACE with your Database name
6  $dbname = " ";
7  // REPLACE with Database user
8  $username = " ";
9  // REPLACE with Database user password
10 $password = " ";
11
12 // Keep this API Key value to be compatible with the ESP32 code provided in the project page.
13 // If you change this value, the ESP32 sketch needs to match
14 $api_key value = "tPmAT5Ab3j7F9";
15 $email_address = "caodanh1802@gmail.com";
16 $api_key= $sensor = $location = $temperature1 = $humidity = $pressure = $temperature2 = "";
17
18 if ($ _SERVER["REQUEST_METHOD"] == "POST") {
19     $api_key = test_input($_POST["api_key"]);
20     if($api_key == $api_key value) {
21         $sensor = test_input($_POST["sensor"]);
22         $location = test_input($_POST["location"]);
23         $temperature1 = Test_input($_POST["temperature1"]);
24         $humidity = test_input($_POST["humidity"]);
25         $pressure = test_input($_POST["pressure"]);
26         $temperature2 = Test_input($_POST["temperature2"]);
  
```

Check if the API key value is as same as the other in Arduino IDE

Check if the column name is as same as the other one in Arduino IDE to save the data from the ESP32 to the database

Figure 33. Save the data from the ESP32 to the web's database.

After that, the stage to use the SQL query INSERT INTO to send all the data from the ESP32 to the web database is indicated in figure 34 below.

```

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO SensorData1 (sensor, location, temperature1, humidity, pressure, temperature2)
VALUES ('" . $sensor . "', '" . $location . "', '" . $temperature1 . "', '"
. $humidity . "', '"
. $pressure . "', '" . $temperature2 . "')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
}
else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();

```

Check the connection with web's database.

Using SQL query to insert data to the web's database.

Figure 34. SQL query to store data from the ESP32 to the web's database.

In addition, this PHP also includes other functions and conditions to send the email alerts, based on defined temperature, pressure, and humidity threshold. This characteristic is indicated in figure 35 below.

```

27 // Create email message
28 // Email message
29 $email_msg = "Temperature in the house is: " . $temperature1 . " degree Celsius. " . "This is below Threshold\n";
30 $email_msg1 = "Temperature in the house is: " . $temperature1 . " degree Celsius. " . "This is above threshold\n";
31
32 $email_msg2 = "Temperature in the garden is: " . $temperature2 . " degree Celsius. " . "This is below Threshold\n";
33 $email_msg3 = "Temperature in the garden is: " . $temperature2 . " degree Celsius. " . "This is above threshold\n";
34
35 $email_msg4 = "Pressure in the house is: " . $pressure . " hPa. " . "This is below Threshold\n";
36 $email_msg5 = "Pressure in the house is: " . $pressure . " hPa. " . "This is above threshold\n";
37
38 $email_msg6 = "Humidity in the house is: " . $humidity . " m. " . "This is below Threshold\n";
39 $email_msg7 = "Humidity in the house is: " . $humidity . " m. " . "this is above threshold\n";
40
41 // Use wordwrap() if lines are longer than 70 characters
42 $email_msg = wordwrap($email_msg, 70);
43
44 // Uncomment the next if statement to set a threshold
45 // ($value1 = temperature, $value2 = humidity, $value3 = pressure)
46 // temperature1
47 if ($temperature1 < 25.0) { Define minimum temperature threshold
48     mail($email_address, "Alert Temperature in the house is below Threshold", $email_msg);
49
50     echo "Email sent";
51
52 Sending email protocol in PHP
53 if ($temperature1 > 31.0) { Define maximum temperature threshold
54     mail($email_address, "Alert Temperature in the house is Above Threshold", $email_msg1);
55
56     echo "Email sent";

```

Figure 35. Email conditions in PHP script.

The second file name is esp-data.php to present the data from the server's database in table form using the <table> tag in HTML and CSS code. The table header is defined as <th> in the code which texts are bold and center, while the <td> tags are the name of each column in the table cell whose texts are left-aligned. At first, there are some HTML and CSS codes to design the table. This protocol is demonstrated in figure 36.

```

echo '<table cellpadding="5" cellspacing="5"
      style="
        background-color: rgba(240, 246, 240, 0.883);
        border: 1px solid black;
      " >>
  <tr>
    <td>ID</td>
    <td>Sensor</td>
    <td>Location</td>
    <td>Temperature 1</td>
    <td>Humidity</td>
    <td>Pressure</td>
    <td>Temperature 2</td>
    <td>Timestamp</td>
  </tr>';

```

Figure 36. Design table form.

After that, in the "esp-data.php", there are also some SQL queries to insert data from the web's database to the table and organize the data based on descending increment ID numbers. Therefore, the latest data which has the highest ID number is on the top row of the table. These tasks are indicated in figure 37.

```

49 $sql = "SELECT id, sensor, location, temperature1,
50 humidity, pressure, temperature2, reading_time
51 FROM SensorData1 ORDER BY id DESC";
52
53 if ($result = $conn->query($sql)) {
54     while ($row = $result->fetch_assoc()) {
55         $row_id = $row["id"];
56         $row_sensor = $row["sensor"];
57         $row_location = $row["location"];
58         $row_temperature1 = $row["temperature1"];
59         $row_humidity = $row["humidity"];
60         $row_pressure = $row["pressure"];
61         $row_temperature2 = $row["temperature2"];
62         $row_reading_time = $row["reading_time"];
63
64         $row_reading_time = date("Y-m-d H:i:s", strtotime("$row_reading_time + 7 hours"));
65
66         echo '<tr>
67             <td>'. $row_id . '</td>
68             <td>'. $row_sensor . '</td>
69             <td>'. $row_location . '</td>
70             <td>'. $row_temperature1 . '</td>
71             <td>'. $row_humidity . '</td>
72             <td>'. $row_pressure . '</td>
73             <td>'. $row_temperature2 . '</td>
74             <td>'. $row_reading_time . '</td>
75         </tr>';
76     }
77     $result->free();

```

SQL query to organize the data based on descending increment ID

Insert data from the database to the PHP's variables

Display the value of the PHP variables in table format of the web server

Figure 37. Insert Data into the table of the web server.

The last PHP script file is esp32_line_chart.php to represent the data from the ESP32 in the form of line charts using the Google Chart library. Firstly, to use this library, some Javascript codes are required to be embedded in this PHP file which demonstrates in figure 38.

```

<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
  google.charts.load('current', {'packages':['corechart']});
  google.charts.setOnLoadCallback(drawChart);

```

Figure 38. Sample code to use the Google Chart's libraries.

After that, the function “drawChart” is used three times to visualize the sensor data in the line chart forms. Figure 39 is the sample code to visualize humidity sensor data on the line chart.

```

58     chart.draw(data, options);
59   }
60   google.charts.setOnLoadCallback(drawChart1); // A callback function to create and populate data, instantiate the line chart, and store data in
61   function drawChart1() {
62     var data1 = google.visualization.arrayToDataTable([
63       ['reading_time', 'humidity'],
64     // PHP Code to get the humidity from your table sensordata
65     <?php
66       $query="select * from SensorData1";
67       $res=mysqli_query($conn,$query);
68       while($data=mysqli_fetch_array($res){
69         $reading_time=$data['reading_time'];
70         $humidity=$data['humidity'];
71       }
72       ?>
73       ['<?php echo $reading_time;?>','<?php echo $humidity;?>'],
74     <?php
75     )
76     ?>
77   ]);
78   var options1 = {
79     title: 'ESP32 Humidity Control Chart',
80     curveType: 'function',
81     legend: ( position: 'bottom' )
82   };
83   // Draw the Line chart which get use the SQL query Select* from.
84   // reading_time is the horizontal data, humidity is the vertical data
85   var chart1 = new google.visualization.LineChart
86   (document.getElementById('curve_chart1'));
87   chart1.draw(data1, options1);
88   // Draw new Google line which based on the data1,
89   // options1 and others inside the function drawChart1

```

Figure 39. Sample code for the line chart of humidity sensor data.

All in all, these files above are stored in the file managers of the web domain which indicates in figure 40.

Name	Size	Date	Permissions
.htaccess	0.5 kB	2020-09-02 03:13:00	-rwxr-xr-x
.htpasswd	0.1 kB	2020-09-02 03:13:00	-rwxr-xr-x
db.php	0.3 kB	2020-09-02 03:11:00	-rwxr-xr-x
esp-data.php	2.4 kB	2020-09-02 03:05:00	-rwxr-xr-x
esp32_line_chart.php	4.0 kB	2020-09-02 03:11:00	-rwxr-xr-x
post-esp-data.php	4.7 kB	2020-09-06 14:52:00	-rwxr-xr-x

Figure 40. File Manager on Web Domain.

To protect two web pages that visualize sensor data in table and line chart forms, it is essential to enable the secure protection of the web server. Particularly, two files “.htaccess” and “.htpasswd” are required to add some additional codes which are demonstrated in figure 41 below.

The image shows two code snippets. The top snippet is the content of a .htaccess file, and the bottom snippet is the content of a .htpasswd file. Red lines and arrows point from the .htpasswd file paths in the .htaccess file to the .htpasswd file content, and from the user names in the .htpasswd file to the explanatory text on the right.

```

/public_html/thesis/.htaccess
1
2 # HTID:15208041: DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES BELOW
3 AuthType Basic
4 AuthName "Restricted Access"
5 AuthUserFile "/storage/ssd3/645/14240645/public_html/thesis/.htpasswd"
6 Require user caodanh
7 # DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES ABOVE HTID:15208041:
8
9 # HTID:15208091: DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES BELOW
10 AuthType Basic
11 AuthName "Restricted Access"
12 AuthUserFile "/storage/ssd3/645/14240645/public_html/thesis/.htpasswd"
13 Require user danh

/public_html/thesis/.htpasswd
1 caodanh:$apr1$/xHIYvwt$TP766iC1xwFkQ/AdI.YF3.
2 danh:$apr1$1UYJ7qiU$NadubbUxI4/VwaKm8XMx0.
3
  
```

Links to .htpasswdfile to find passwords of users'names: caodanh and danh

Password written in ASCII language for secure protection

Figure 41. Programming for web's secure protection

6 Results

6.1 Local Web Server Control for the First Project.

6.1.1 Description

In this part, a local web server has been designed as demonstrated in figure 42. This web server will display the current temperature data from the DS18B20 and allow users to input an appropriate email address for alerts. Furthermore, two relays will be turned on if the temperature goes out the max and min temperature threshold. For example, the red led is turned on when the temperature is too high or the green one will be turned on when the temperature is too low, and both will be turned off just after the temperature becomes normal. The last function is sending an email which content includes the dangerous current temperature, and a message indicates that this data is upper or lower the control threshold which is dedicated in figure 43.

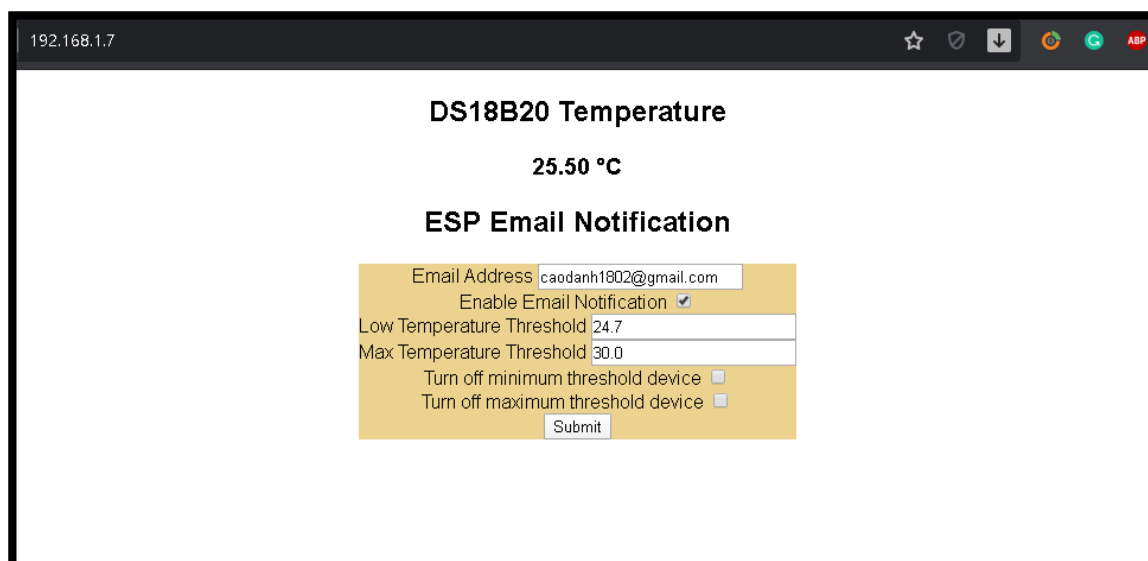


Figure 42. First project's local web server.

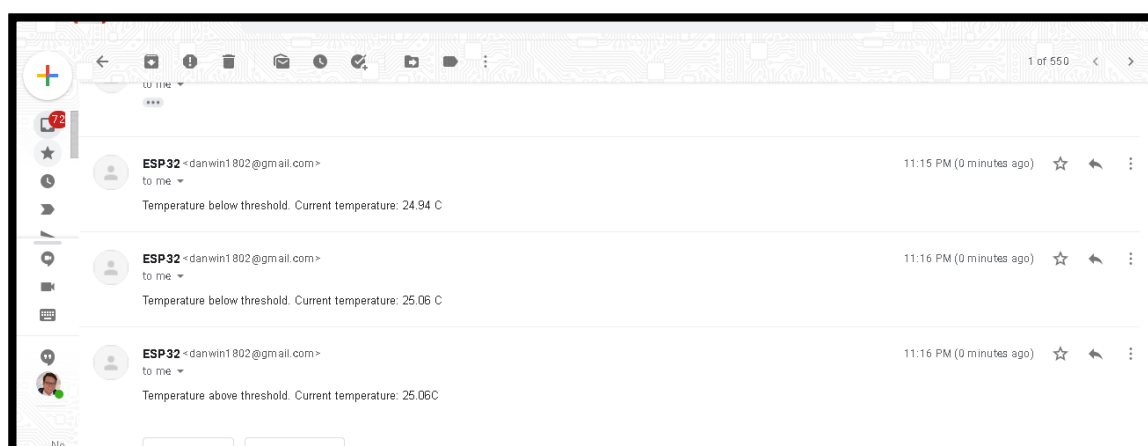


Figure 43. Emails were sent when the temperature went out the normal range.

6.1.2 Flow Specification

There are several steps to design this local web server, and they can be divided into three main essential tasks. The first task is to insert some mandatory libraries into the Arduino IDE environment, and the next one is to create and program the “first_project.ino” inside the coding IDE. The final goal is to flash the program file into the ESP32. Figure 44 below demonstrates the step-by-step flow chart of these processes.

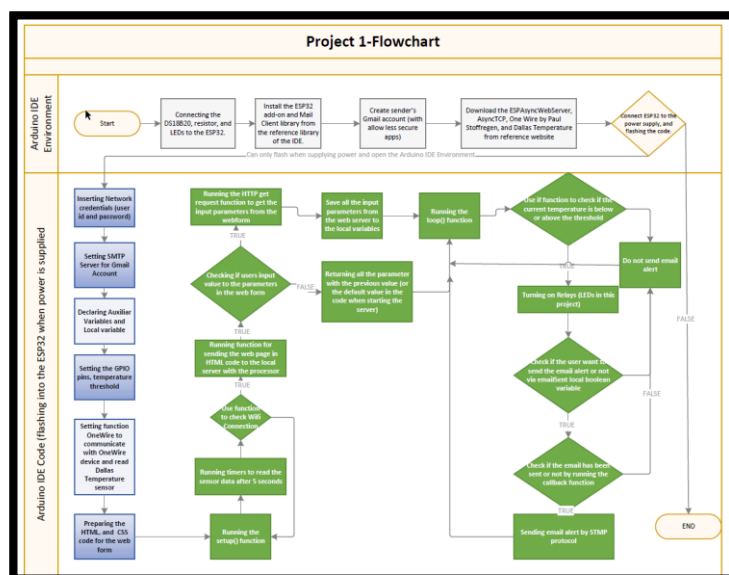


Figure 44. Programming flow chart of the first project.

After flashing the software into the ESP32, this system can run continuously to check whether the temperature is in the normal range or not, turn on specific relays defined by the thresholds, and send email alerts. In addition, the system stops working only when disconnecting the ESP32 from the power supply, and the web server, and alert function will not work when ESP32 cannot connect to the wifi. Therefore, in the smart home application, when powering and flashing the ESP32, users can know when the temperature in a specific area is too high or too low without the need to stay inside this region thanks to this system.

6.2 The Final Web Server with Data Visualization

6.2.1 Description

The earlier project has two issues in the wireless network and data storage. Specifically, the first issue is that it works only in the local domain which means that the ESP32 and the computer need to connect to the same wireless network. Secondly, the temperature information will always update every five second and the old information was deleted. Therefore, to connect to the website through the global Internet network and save all sensor's data, a global web domain has been used where the writer can upload all the

sensor data from the ESP32 to its database and visualize these statistics in table or line charts form from two different links which are demonstrated in figure 45 and 46 respectively.

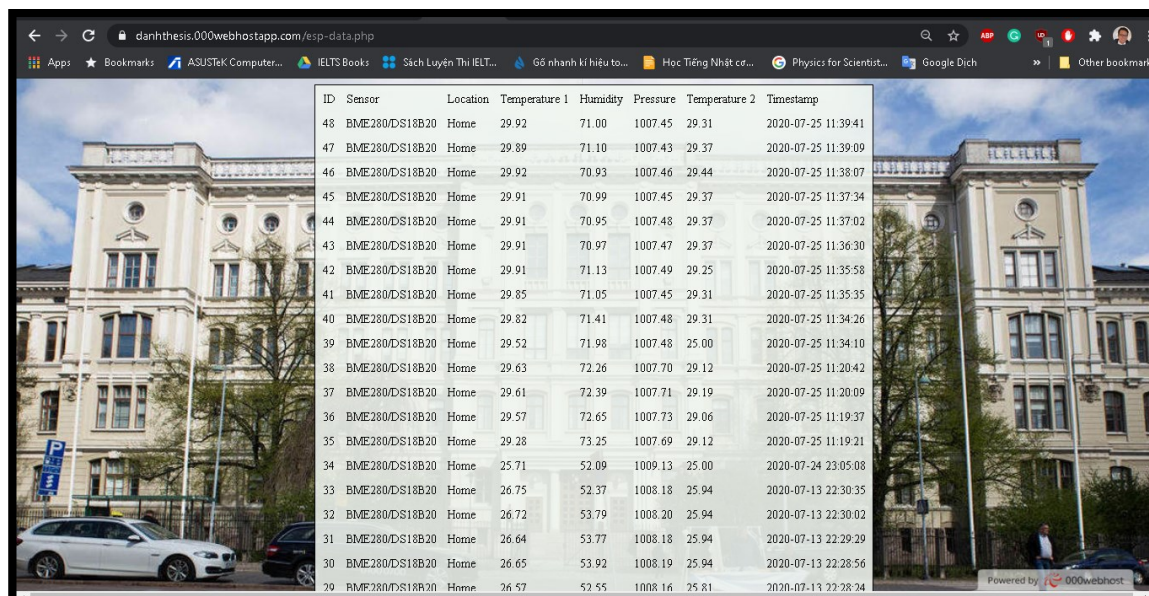


Figure 45. Data visualization in table form.

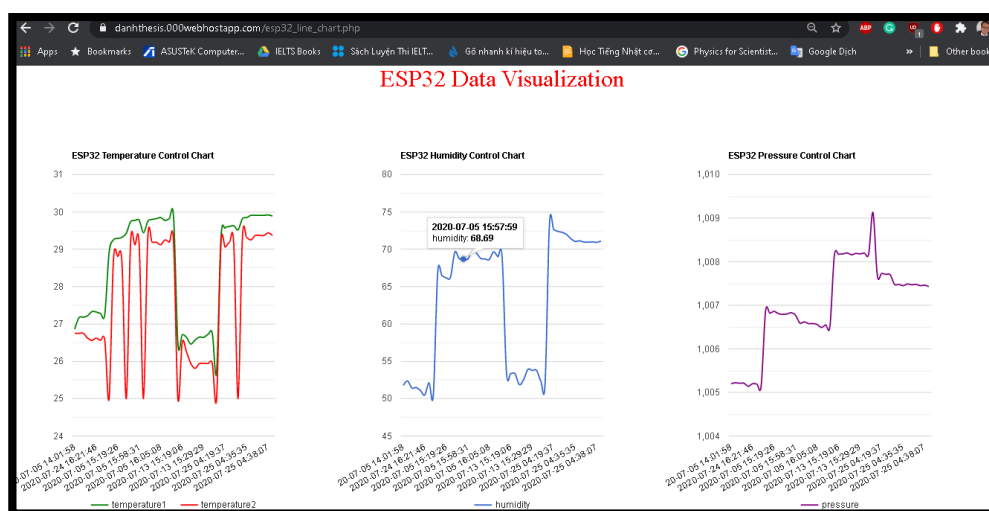


Figure 46. Data Visualization in Line Chart form.

However, for data privacy secure protection, to access to the above web pages, it is essential for the user to input the appropriate user name and password. If wrong credential information is inputted, the secure authentication will present continuously which is demonstrated in figure 47 below.

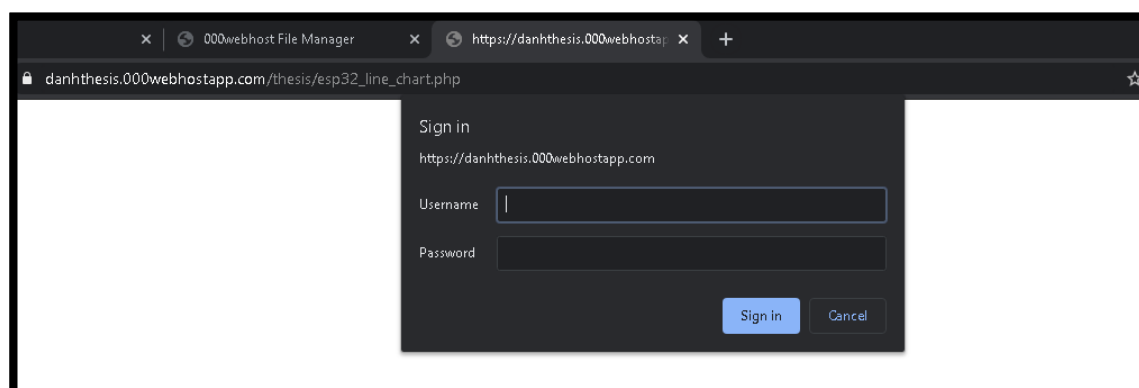


Figure 47. Web page's authentication

There are numerous reasons why storing the sensor data of the ESP32 to the web server database is essential. First of all, if only the latest temperature information is acquired like what already did in the first project, users cannot check if the previous data is wrong or not without the alert email function. Secondly, when all the information is saved, they can be used to predict different trends for further maintenance service or to handle some problems related to the sensors' data in the industrial environment. Lastly, when using in a big system or manufacturing, every single data needs to be stored and used for further analysis and research to improve the efficiencies and outcomes.

The table form helps users quickly observe the data from the sensor, as the data has been programmed to display the latest data on the first row of the table, with a specific ID. Besides, the line chart form improves the visualization of the user to control data, especially in the wide range of data. For example, based on the graph, users can know if the temperature rises consistently or unsteadily, and find out the solution for a specific purpose. Therefore, these forms are quite useful and convenient for the industrial company to manage the tendency of their sensors' data and determine appropriate predictive maintenance.

Particularly, based on the defined threshold of different parameters and data in the web's database, this design can send the alarm messages to an appropriate person when the temperature, humidity, and pressure exceed the normal range by using the PHP scripts. Figure 48 demonstrates the data which got from the ESP32 on the web server's database management software-PHPMyAdmin.

	id	sensor	location	temperature1	humidity	pressure	temperature2	reading_time
<input type="checkbox"/>	9	BME280/DS18B20	Home	28.91	66.73	1006.85	25.00	2020-07-05 15:18:19
<input type="checkbox"/>	10	BME280/DS18B20	Home	29.25	66.47	1006.82	28.69	2020-07-05 15:18:53
<input type="checkbox"/>	11	BME280/DS18B20	Home	29.29	66.17	1006.86	28.81	2020-07-05 15:19:26
<input type="checkbox"/>	12	BME280/DS18B20	Home	29.32	66.28	1006.81	28.62	2020-07-05 15:19:59
<input type="checkbox"/>	13	BME280/DS18B20	Home	29.43	69.63	1006.79	25.00	2020-07-05 15:56:02
<input type="checkbox"/>	14	BME280/DS18B20	Home	29.74	68.76	1006.80	29.19	2020-07-05 15:57:27
<input type="checkbox"/>	15	BME280/DS18B20	Home	29.77	68.69	1006.83	29.12	2020-07-05 15:57:59
<input type="checkbox"/>	16	BME280/DS18B20	Home	29.78	68.67	1006.77	29.12	2020-07-05 15:58:31
<input type="checkbox"/>	17	BME280/DS18B20	Home	29.44	69.68	1006.59	25.00	2020-07-05 16:03:18
<input type="checkbox"/>	18	BME280/DS18B20	Home	29.75	69.49	1006.62	29.31	2020-07-05 16:03:30
<input type="checkbox"/>	19	BME280/DS18B20	Home	29.80	68.78	1006.58	29.19	2020-07-05 16:04:03
<input type="checkbox"/>	20	BME280/DS18B20	Home	29.82	68.70	1006.58	29.19	2020-07-05 16:04:36
<input type="checkbox"/>	21	BME280/DS18B20	Home	29.85	68.62	1006.56	29.12	2020-07-05 16:05:08
<input type="checkbox"/>	22	BME280/DS18B20	Home	29.77	69.66	1006.49	29.25	2020-07-05 16:09:01
<input type="checkbox"/>	23	BME280/DS18B20	Home	29.82	69.02	1006.55	29.19	2020-07-05 16:09:33
<input type="checkbox"/>	24	BME280/DS18B20	Home	29.85	68.92	1006.53	29.19	2020-07-05 16:10:05
<input type="checkbox"/>	25	BME280/DS18B20	Home	26.48	53.58	1008.13	25.00	2020-07-13 15:18:34

Figure 48. Web server's database management

Figure 49 below indicates the email alerts sent from the web server with the help of php scrip file "post-esp-data.php".

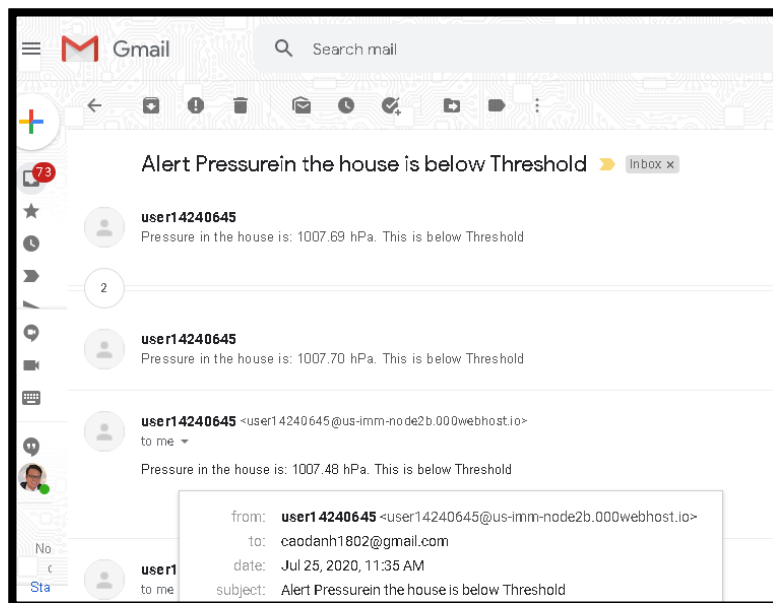


Figure 49. Email Alert from Web Server using PHP script

6.2.2 Flow Specification

Compared to the first project, this web server requires more steps which are summarized into four main tasks. The first and the second tasks are to prepare the breadboard of the ESP32 and the code "second_project.ino" in the Arduino IDE environment. The third step is working with the webserver domain to create a database, table, and including three PHP scripts to save the data from the Arduino code to the database, and visualize all the sensor information in the table and line chart form. The final step is to flash the program in the IDE to the main breadboard and start the IoT system. Figure 50 below is the descriptive diagram for the second project's programming flow.

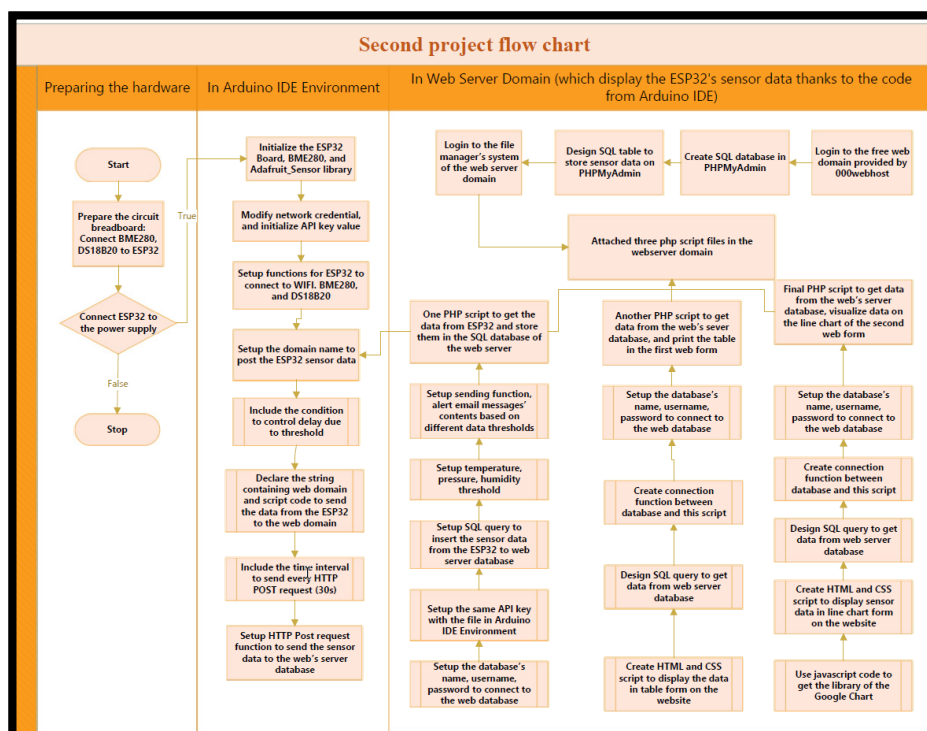


Figure 50. Programming flow chart of the second project

After flashing the software to the ESP32, this system will work continuously to send the data from the ESP32 to the web database, and if the ESP32 cannot connect to the Wifi or the power supply, the system will stop. In addition, while the web server of the first project can be accessed via the devices connecting to the local internet, this project allows users to open the web pages via any internet connection.

7 Conclusion

The main goal of this thesis work was to introduce the overview of the development of the industry from the very beginning to the Industry 4.0 revolution in different aspects, especially in the field of the Internet of Things with various evidence such as the intelligent car, automatic home, self-regulating supply chain, electronic industrial alert system. Although this evolution brought numerous significant requirements and challenges when designing the IoT and embedded system, the benefits of it are undeniable in humans' lives.

There are two practical parts in this thesis to demonstrate the evolution of the IoT and embedded software in the field of smart home and industrial management. These two parts all use the low-power SoC design of the ESP32 connected to the industrial sensors such as BME280, and DS18B20 (non-water-resistant version). The first prototype is to create a local web server to control the temperature data and send email alerts. Then the second project has been developed to visualize sensor data and send emails based on specific thresholds by the global web server domain.

In general, the first project is a stepping stone for the second one because of two main reasons. Firstly, the ESP32's old data required to be deleted before using the HTTP GET request to prevent memory shortage. Therefore, it is not possible to save the data from the ESP32 to its memory or visualize them on the local web server. Secondly, it is more convenient for the users to access the database even when they are outside or living very far from their local server. Consequently, the second project has been established to maintain all the data from the ESP32 by saving them to the SQL database on the free web domain and visualize these statistics for further analysis. As a result of visualizing data on the graphs, it is more comfortable for the engineer, managers, or employees to predict the tendency of the process and execute some immediate actions to handle the serious problems or generating some innovative steps for predictive maintenance. Besides, using free domain made the statistics accessible all over the world, and it is extremely helpful in the smart home applications.

Lastly, there are still some improvements needed in these projects. First of all, it still uses a free web domain from the third service, so it is not securable for data protection. In the

future, it can be easily changed when a new domain is bought, as they still have the same setup as this project. The user just needs to attach all the coding files below to their own web server database and execute some modifications such as network credentials, database's username, passwords, and the API key. Secondly, in the real industry, these embedded systems are always used in their local web server for security reasons. However, it requires a lot of competences to build a full embedded local security web server, because it needs to make a lot of encryptions which dedicates in the above sections, and huge data will need to store and visualize. Therefore, Power BI has been dedicated in this project to demonstrate one of the most powerful tools for the factory to visualize the data, execute filtering, and calculate the total revenue in the business field, because it is quite simple to use with many incredible available functions and sample code, compared to program hardly to get various sensors' data in the factory. However, the Power BI tool was not free and costed a lot for some small factories, and individuals to use. Therefore, this project is quite useful for some small businesses and personal users to implement. Finally, the project can add more modern and convenient components and devices, such as sensors, and relays or use with other IoT kits, such as Raspberry BI, Mongoose OS IoT kit, so it can give more precise data from different electrical equipment in numerous areas.

References

References

- [1] "Industry 4.0: the fourth industrial revolution – a guide to Industry 4.0," [Online]. Available: <https://www.i-scoop.eu/industry-4-0/>. [Accessed 10 June 2020].
- [2] K. Pouspourika, "The Four Industrial Revolutions," Institute of Entrepreneurship Development, 30 June 2019. [Online]. Available: <https://ied.eu/project-updates/the-4-industrial-revolutions/>. [Accessed 12 August 2020].
- [3] "Why is IoT so important," [Online]. Available: <https://www.oracle.com/internet-of-things/what-is-iot.html>. [Accessed 12 June 2020].
- [4] L. Columbus, "Seven Things You Need To Know About IIoT In Manufacturing," 9 June 2019. [Online]. Available: <https://www.enterpriseirregulars.com/138763/seven-things-you-need-to-know-about-iiot-in-manufacturing/>. [Accessed 12 June 2020].
- [5] M. Sharma, "Stages of "Internet of Things" Architecture," 19 September 2019. [Online]. Available: <https://www.marlabs.com/blog-stages-of-iot-architecture/>. [Accessed 12 June 2020].
- [6] Rajiv, "What are the major components of the Internet of Things," 10 January 2018. [Online]. Available: <https://www.rfpage.com/what-are-the-major-components-of-internet-of-things/>. [Accessed 16 June 2020].
- [7] "What is IoT Architecture," 11 May 2020. [Online]. Available: <https://www.avsystem.com/blog/what-is-iot-architecture/>. [Accessed 12 August 2020].
- [8] "IOTArchitecture," 2020. [Online]. Available: <https://www.javatpoint.com/iot-architecture-models>. [Accessed 13 August 2020].
- [9] A. O'neill, "What is IoT Edge," 16 April 2020. [Online]. Available: <https://www.c-sharpcorner.com/article/what-is-iot-edge/>. [Accessed 13 August 2020].

- [10] "Choosing the best edge platform – IoT edge platform selection criteria and advice," I-Scoop, [Online]. Available: <https://www.i-scoop.eu/internet-of-things-guide/iot-platform-market-2017-2025/iot-edge-platform-selection/>. [Accessed 9 June 2020].
- [11] A. S. P. A. D. G. Jayant D.Bokefode, "Developing A Secure Cloud Storage System for Storing IoT Data by Applying Role Based Encryption," 22 August 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050916310729>. [Accessed 17 June 2020].
- [12] "Powerful Data Visualization Tools for IoT Applications," 5 September 2019. [Online]. Available: <https://www.iotforall.com/data-visualization-tools-iot-applications/>. [Accessed 16 June 2020].
- [13] V. Padghan, "Power BI Tutorial: Visualizing Data Like Never Before With Power BI Desktop," Edureka, 19 May 2020. [Online]. Available: <https://www.edureka.co/blog/power-bi-tutorial/>. [Accessed 1 June 2020].
- [14] K. T. L. Trinh, "Tap Chi Lap Trinh," Kien Thuc Lap Trinh, 13 February 2020. [Online]. Available: <https://tapchilaptrinh.vn/2020/02/13/ve-bieu-do-chart-cho-trang-web-bang-html-va-google-charts/>. [Accessed 19 July 2020].
- [15] "Using Google Charts," Google, [Online]. Available: <https://developers.google.com/chart/interactive/docs>. [Accessed 19 July 2020].
- [16] S. C. Mukhopadhyay, Internet of Things: Challenges and Opportunities, 2014.
- [17] E. Systems, "ESP32 Series Datasheet," 2020. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed 25 July 2020].
- [18] E. Systems, "ESP32-WROOM-32 Datasheet," 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [Accessed 20 July 2020].
- [19] "SoftAP," Wikipedia, [Online]. Available: <https://en.wikipedia.org/wiki/SoftAP>. [Accessed 13 August 2020].

- [20] "RAM, ROM, and Flash Memory," [Online]. Available: <https://www.dummies.com/computers/computer-networking/networking-components/ram-rom-and-flash-memory/>. [Accessed 13 August 2020].
- [21] S. v. D. K. t. Difference. [Online]. Available: <https://www.guru99.com/sram-vs-dram-difference.html>. [Accessed 13 August 2020].
- [22] Zerynth, "DOIT ESP32 DEVKITv1," [Online]. Available: https://docs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html. [Accessed 22 July 2020].
- [23] R. Santos, "Getting started with ESP32," [Online]. Available: <https://randomnerdtutorials.com/getting-started-with-esp32/>. [Accessed 22 July 2020].
- [24] L. M. Engineers, "Interface BME280 Temperature, Humidity & Pressure Sensor with Arduino," [Online]. Available: <https://lastminuteengineers.com/bme280-arduino-tutorial/>. [Accessed 20 July 2020].
- [25] "BASICS OF THE SPI COMMUNICATION PROTOCOL," Circuit Basics, [Online]. Available: <https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/>. [Accessed 1 July 2020].
- [26] "BASICS OF THE I2C COMMUNICATION PROTOCOL," Circuit Basics, [Online]. Available: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>. [Accessed 2 July 2020].
- [27] R. Santos, "Random Nerd Tutorials," [Online]. Available: <https://randomnerdtutorials.com/esp32-email-alert-temperature-threshold/>. [Accessed 19 June 2020].
- [28] "DS18B20 Temperature Sensor," 7 May 2018. [Online]. Available: <https://components101.com/sensors/ds18b20-temperature-sensor>. [Accessed 19 July 2020].
- [29] D. Semiconductor, "DS18B20 Programmable Resolution One Wire Digital Thermometer Datasheet," [Online]. Available:

<https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html>.

[Accessed 20 July 2020].

[30] "000webhost," Hostinger, [Online]. Available: <https://www.000webhost.com/#feature-table>. [Accessed 20 July 2020].

[31] PHPMYAdmin, "PHPMYAdmin," [Online]. Available: <https://www.phpmyadmin.net/>. [Accessed 20 July 2020].

Appendix 1. Local webserver prototype for the first project whose name is “first_project.ino”

```

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "ESP32_MailClient.h"

//NETWORK USER ID AND PASSWORD
const char* ssid = ""; // writer's confidential information
const char* password = ""; // writer's confidential information

// Output state will store in these variable
String defaultStateoutput26 = "off";
String defaultStateoutput27 = "off";

// Define pins for output parts
const int relayoutput1 = 26;
const int relayoutput2 = 27;

// Using Gmail Account
#define emailSenderAccount    "danwin1802@gmail.com"
#define emailSenderPassword   "danh18021991."
#define smtpServer            "smtp.gmail.com"
#define smtpServerPort        465
#define emailSubject          "[Urgent Alert] Alert ESP32's temperature is
outside the threshold "

// Default Responsible Email Alert Address
String EmailAddress = "caodanh1802@gmail.com";
String enableAlertEmail = "checked";
String EnableInput = "true";

// Default Temperature threshold data
String lowthreshold = "26.0"; //low threshold
String maxthreshold = "30.0"; //max threshold
String lastTemperature;

//Control threshold device
String lowthresholddevice = "false";
String maxthresholddevice = "false";

// HTML web page
const char web_index_code[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html><head>
  <title>Email Notification with Temperature</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<style>html { font-family: Helvetica;display: inline-block; margin: 0px
auto; text-align: center;}
  form {
    display: inline-block;
    width: fit-content;
    height: fit-content;
    background-color: rgb(218, 165, 32, 0.5);
  }
</style>

```

```

<body>
<h2>First Project in Temperature Control</h2>
<h3>%temperaturedata% &deg;C</h3>
<h2>ESP32 Alert Email</h2>
<form action="/get">
  Input Receiver's Address <input type="email" name="email_input"
value="%EMAIL_INPUT%" required><br>
  Allow Email Alert <input type="checkbox" name="enable_email_input"
value="true" %ENABLE_EMAIL%><br>
  Input Low Threshold <input type="number" step="0.1" name="in-
put_low_threshold" value="%LOW%" required><br>
  Input Max Threshold <input type="number" step="0.1" name="in-
put_max_threshold" value="%MAX%" required><br>
  Turn off minimum threshold device <input type="checkbox" name="low_thresh-
old_device" value="true" %TURN_OFF_LOWTHRESHOLD%><br>
  Turn off maximum threshold device <input type="checkbox" name="max_thresh-
old_device" value="true" %TURN_OFF_MAXTHRESHOLD%><br>
  <input type="submit" value="Submit">
</form>
</body></html>)rawliteral";

void notFound(AsyncWebServerRequest *request) {
  request->send(404, "text/plain", "Not found");
}

AsyncWebServer server(80);

// Save the variable of the web server to the data of the ESP32
String processor(const String& var){

  if(var == "temperaturedata"){
    return lastestemperature;
  }
  else if(var == "EMAIL_INPUT"){
    return InputEmailAddress;
  }
  else if(var == "ENABLE_EMAIL"){
    return enableAlertEmail;
  }
  else if(var == "LOW"){
    return lowthreshold;
  }
  else if(var == "MAX"){
    return maxthreshold;
  }
  else if (var == "TURN_OFF_LOWTHRESHOLD"){
    return lowthresholddevice;
  }
  else if (var == "TURN_OFF_MAXTHRESHOLD"){
    return maxthresholddevice;
  }
  return String();
}

// Using flag variable to keep track if email notification was sent or not,
and set the output state
bool emailSent = false;
String minimum = "false";
String maximum = "false";

const char* paraminput1 = "emailaddress_input";
const char* paraminput2 = "email_alert_enable";
const char* paraminput3 = "input_low_threshold";

```

```

const char* paraminput4 = "input_max_threshold";
const char* paraminput5 = "low_threshold_device";
const char* paraminput6 = "max_threshold_device";

// Time to read sensor data
unsigned long starttime = 0;
const long timelength = 5000;

// PIN OF DS18B20
const int oneWireBus = 4;

OneWire oneWire(oneWireBus);

DallasTemperature sensors(&oneWire);

SMTPData smtpData;

void setup() {
  Serial.begin(115200);

  pinMode(relayoutput1, OUTPUT);
  pinMode(relayoutput2, OUTPUT);

  digitalWrite(relayoutput1, LOW);
  digitalWrite(relayoutput2, LOW);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  if (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("WiFi Failed!"); // Check if the wifi is connected on the
    local machine
    return;
  }
  Serial.println();
  Serial.print("ESP IP Address: http://");
  Serial.println(WiFi.localIP());

  // Start reading the data from DS18B20 sensor
  sensors.begin();

  // Send web page to client
  server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
    request->send_P(200, "text/html", web_index_code, processor);
  });

  // Receive an HTTP GET request at <ESP_IP>/get?emailaddress_input=<In-
  putEmailAddress>&email_alert_enable=<EnableInput>&threshold_input=<lowthresh-
  old>
  server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {

    if (request->hasParam(paraminput1)) {
      InputEmailAddress = request->getParam(paraminput1)->value();

      if (request->hasParam(paraminput2)) {
        EnableInput = request->getParam(paraminput2)->value();
        enableAlertEmail = "checked";
      }
      else {
        EnableInput = "false";
        enableAlertEmail = "";
      }
    }
  });
}

```

```

    if (request->hasParam(paraminput3)) {
        lowthreshold = request->getParam(paraminput3)->value();
    }
    if (request->hasParam(paraminput4)) {
        maxthreshold = request->getParam(paraminput4)->value();
    }
    if (request->hasParam(paraminput5)) {
        lowthresholddevice = request->getParam(paraminput5)->value();
    }
    if (request->hasParam(paraminput6)) {
        maxthresholddevice = request->getParam(paraminput6)->value();
    }
}
else {
    InputEmailAddress = "Cannot send the Message";
}
Serial.println("Admin input to web-page");
Serial.println(InputEmailAddress);
Serial.println(EnableInput);
Serial.println(lowthreshold);
Serial.println(maxthreshold);
Serial.println(lowthresholddevice);
Serial.println(maxthresholddevice);
if (lowthresholddevice=="true") {minimum = "true";} else minimum="false";
if (maxthresholddevice=="true") {maximum = "true";} else maximum="false";

    request->send(200, "text/html", "HTTP GET request sent to your ESP.<br><a href=\"/\>Return to Home Page</a>");

});
server.onNotFound(notFound);
server.begin();
}

void loop() {
    unsigned long currentMillis = millis();
    if (currentMillis - starttime >= timelength) {
        starttime = currentMillis;
        sensors.requestTemperatures();

        float temperature = sensors.getTempCByIndex(0);
        Serial.println("Current Temperature is");
        Serial.print(temperature);
        Serial.println(" *C");

        lastestemperature = String(temperature);

        // Check if temperature is above threshold and if it needs to send the
        Email alert
        if(temperature > maxthreshold.toFloat() && !emailSent ){
            String emailMessage = String("Temperature above threshold. Current tem-
            pera-ture: ") +
                String(temperature) + String("C");
            Serial.println("Current temperature is above threshold");
            digitalWrite(relayoutput1, HIGH);
            digitalWrite(relayoutput2, LOW);
            Serial.println(maximum);
            if (maximum == "true") {
                digitalWrite(relayoutput1, LOW);
            }
        }
        if( EnableInput == "true"){
            if(sendEmailNotification(emailMessage)) {

```

```

        Serial.println(emailMessage);
        emailSent = true;
    }
    else {
        Serial.println("Email failed to send");
    }
}

}

// Check if temperature is below threshold and if it needs to send the
Email alert
else if((temperature < lowthreshold.toFloat()) && !emailSent ) {
    String emailMessage = String("Temperature below threshold. Current tem-
pera-ture: ") +
                                String(temperature) + String(" C");
    Serial.println("Current temperature is below threshold");
    digitalWrite(relayoutput2, HIGH);
    digitalWrite(relayoutput1, LOW);
    Serial.println(minimum);
    if(minimum == "true"){
        digitalWrite(relayoutput2, LOW);
    }
    if (EnableInput == "true"){
        if(sendEmailNotification(emailMessage)) {
            Serial.println(emailMessage);
            emailSent = true;
        }
    }
    else {
        Serial.println("Email failed to send");
    }
}

}

// In Normal range input 3 is low threshold, input 4 is high threshold
else if((temperature > lowthreshold.toFloat()) && (temperature < inputMes-
sage4.toFloat())) {
    Serial.println("In Normal range");
    digitalWrite(relayoutput2, LOW);
    digitalWrite(relayoutput1, LOW);
}
else {
    emailSent= false;
}
}
lowthresholddevice = "false";
maxthresholddevice = "false";
}

bool sendEmailNotification(String emailMessage){
    // Set the SMTP Server Email host, port, account and password
    smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount,
emailSenderPassword);

    // Set the sender name and Email
    smtpData.setSender("ESP32", emailSenderAccount);

    // Set Email priority or importance High, Normal, Low or 1 to 5 (1 is high-
est)
    smtpData.setPriority("High");
}

```

```
// Set the subject
smtpData.setSubject(emailSubject);

// Set the message with HTML format
smtpData.setMessage(emailMessage, true);

// Add recipients
smtpData.addRecipient(InputEmailAddress);

smtpData.setSendCallback(sendCallback);

// Start sending Email
if (!MailClient.sendMail(smtpData)) {
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());
    return false;
}
// Clear all data from Email object to free memory
smtpData.empty();
return true;
}

// Callback function to get the Email sending status
void sendCallback(SendStatus msg) {
    // Print the current status of email sending
    Serial.println(msg.info());

    // Do something when complete sending the email
    if (msg.success()) {
        Serial.println("-----");
    }
}
```


Appendix 2. Second project prototypes for the web's server domain.

File in the Arduino CC Software: "second-project.ino".

```

#include <Wire.h>
#include <HTTPClient.h>
#include <WiFi.h>
#include <OneWire.h>
#include <Adafruit_BME280.h>
#include <Adafruit_Sensor.h>
#include <DallasTemperature.h>

// User ID and Password of the Wifi
const char* ssid = "";
const char* password = "";

// Writer's domain name
const char* serverName = "https://danhthesis.000webhostapp.com/thesis/post-esp-data.php";

// Key to protect the data
String apiKeyValue = ""; //Own API key must be filled

String sensorName = "BME280/DS18B20";
String sensorLocation = "Home";

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C

// Variables to store the current output state
String Staterelay26 = "off";
String Staterelay27 = "off";

// Relay variable pins
const int relay26 = 26;
const int relay27 = 27;

const int oneWireBus = 4;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(115200);
  pinMode(relay26, OUTPUT);
  pinMode(relay27, OUTPUT);

  digitalWrite(relay26, LOW);
  digitalWrite(relay27, LOW);

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}

```

```

sensors.begin();
bool status = bme.begin(0x76);
  if (!status) {
    Serial.println("BME280 could not be found or wrong address could be provided");
    while (1);
  }
}

void loop() {

  unsigned long currentMillis = millis();
  sensors.requestTemperatures();
  // DS18B20 Temperature in Celsius Degree
  float temperature = sensors.getTempCByIndex(0);
  Serial.print(temperature);
  Serial.println(" *C");
  // Below threshold
  if (temperature < 30){
    digitalWrite(relay26, HIGH);
    digitalWrite(relay27, LOW);
  }
  else if (temperature >35){
    digitalWrite(relay27, HIGH);
    digitalWrite(relay26, LOW);
  }
  else{
    digitalWrite(relay26, LOW);
    digitalWrite(relay27, LOW);
  }

  if(WiFi.status()== WL_CONNECTED){
    HTTPClient http;

    // Begin HTTPClient Server
    http.begin(serverName);

    // Header is defined
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    // HTTP Post is defined to send the data to the web server's database
    String httpRequestData = "api_key=" + apiKeyValue + "&sensor=" + sensorName
        + "&location=" + sensorLocation + "&temperature1=" +
String(bme.readTemperature())+ "&humidity=" + String(bme.readHumidity()) +
"&pressure=" + String(bme.readPressure()/100.0F)+ "&temperature2="+String(temperature)
        + "";
    Serial.print("httpRequestData: ");
    Serial.println(httpRequestData);

    // Request is being sent to post the data from the ESP32 to the database
    int httpResponseCode = http.POST(httpRequestData);

    if (httpResponseCode>0) {
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
    }
    else {

```

```

        Serial.print("There are some errors, the codes are ");
        Serial.println(httpResponseCode);
    }
    // Resources of the HTTPClient is being free
    http.end();
}
else {
    Serial.println("There is no connected WIFI")
}
//HTTP POST is sent every 30 seconds
delay(30000);
}

```

File on the web's file manager: "post-esp-data.php".

```

<?php

$servername = "localhost";

// Own Database's confidential
$dbname = "";
$username = "";
$password = "";

$api_key_value = ""; //Key value must be the same with Arduino code

$email_address = "caodanh1802@gmail.com"; // email address to send alerts
$api_key= $sensor = $location = $temperature1 = $humidity = $pressure = $tem-
pera-
ture2 = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $api_key = test_input($_POST["api_key"]);
    if($api_key == $api_key_value) {
        $sensor = test_input($_POST["sensor"]);
        $location = test_input($_POST["location"]);
        $temperature1 = test_input($_POST["temperature1"]);
        $humidity = test_input($_POST["humidity"]);
        $pressure = test_input($_POST["pressure"]);
        $temperature2 = test_input($_POST["temperature2"]);

        // Email message
        $email_msg = "Temperature in the house is: " . $temperature1 . " de-
gree Celsius. " . "This is below Threshold\n";
        $email_msg1 = "Temperature in the house is: " . $temperature1 . "de-
gree Celsius. " . "This is above threshold\n";

        $email_msg2 = "Temperature in the garden is: " . $temperature2 . " de-
gree Celsius. " . "This is below Threshold\n";
        $email_msg3 = "Temperature in the garden is: " . $temperature2 . "de-
gree Celsius. " . "This is above threshold\n";

        $email_msg4 = "Pressure in the house is: " . $pressure . " hPa. " .
"This is below Threshold\n";
        $email_msg5 = "Pressure in the house is: " . $pressure . " hPa. "
."This is above threshold\n";

        $email_msg6 = "Humidity in the house is: " . $humidity . " m. " . "This
is below Threshold\n";
        $email_msg7 = "Humidity in the house is: " . $humidity . " m. " . "this
is above threshold\n";
    }
}

```

```
$email_msg = wordwrap($email_msg, 70);

//Setting temperature threshold for the BME280, as it places inside
the house

if($temperature1 < 25.0){
    mail($email_address, "Alert Temperature in the house is below
Threshold", $email_msg);

    echo "Email sent";
}
if($temperature1 > 31.0){
    mail($email_address, "Alert Temperature in the house is Above
Threshold", $email_msg1);

    echo "Email sent";
}

//Setting temperature threshold for the DS18B20, as it places outside
the house as it also has a water-proof version

if($temperature2 < 30.0){
    mail($email_address, "Alert Temperature in the garden is below
Threshold", $email_msg2);

    echo "Email sent";
}
if($temperature2 > 34.0){
    mail($email_address, "Alert Temperature in the garden is Above
Threshold", $email_msg3);

    echo "Email sent";
}
//Setting pressure threshold for the BME280

if($pressure < 1008.0){
    mail($email_address, "Alert Pressurein the house is below Thresh-
old", $email_msg4);

    echo "Email sent";
}
if($pressure > 1010.0){
    mail($email_address, "Alert Pressure in the house is Above Thresh-
old", $email_msg5);

    echo "Email sent";
}
// Setting humidity threshold for the BME280
if($humidity < 70.0){
    mail($email_address, "Alert Humidity in the house is below Thresh-
old", $email_msg6);

    echo "Email sent";
}
if($humidity > 77.0){
    mail($email_address, "Alert Humidity in the house is Above Thresh-
old", $email_msg7);

    echo "Email sent";
}
```

```

// send emails with mail(receiver email address, email subject, email mes-
sage)

// Connect with the mysql server database
$conn = new mysqli($servername, $username, $password, $dbname);
// Condition to ensure that the database has been connected
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO ██████████ (sensor, location, temperature1, hu-
midity, pressure, temperature2) // Replace with your table in the database
VALUES ('" . $sensor . "', '" . $location . "', '" . $temperature1 .
"', '" . $humidity . "', '" . $pressure . "', '" . $temperature2 . "')";

if ($conn->query($sql) === TRUE) {
    echo "Data from the BME280 and DS18B20 are stored successfully";
}
else {
    echo "Connection Error with specific code: " . $sql . "<br>" .
$conn->error;
}

$conn->close();
}
else {
    echo "API key needs to be provided as same as the own user's key";
}
}
else {
    echo "There is no data in the server database";
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

```

File on the web's file manager: "esp-data.php"(data in table format).

```

<!DOCTYPE html>
<html><body style="
    background-image: url('https://metropoliawebsitetalks.files.word-
press.com/2019/04/1-6.jpg');
    background-size: 100% 100%;
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
">

<?php

$servername = "localhost";

Own database's confidential
$dbname = "";

```

```

$username = "";
$password = "";

// Design the connection between the database and the table chart form
$conn = new mysqli($servername, $username, $password, $dbname);

// Condition to ensure that the connection is successful
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Replace with your table's information such as name of the column, id.

$sql = "SELECT id, sensor, location, temperature1, humidity, pressure, tem-
pera-
ture2, reading_time FROM ██████████ ORDER BY id DESC";

echo '<table cellpadding="5" cellspacing="5"
style="
background-color: rgba(240, 246, 240, 0.883);
border: 1px solid black;
" >>
<tr>
<td>ID</td>
<td>Sensor</td>
<td>Location</td>
<td>Temperature 1</td>
<td>Humidity</td>
<td>Pressure</td>
<td>Temperature 2</td>
<td>Timestamp</td>
</tr>';

if ($result = $conn->query($sql)) {
    while ($row = $result->fetch_assoc()) {
        $row_id = $row["id"];
        $row_sensor = $row["sensor"];
        $row_location = $row["location"];
        $row_temperature1 = $row["temperature1"];
        $row_humidity = $row["humidity"];
        $row_pressure = $row["pressure"];
        $row_temperature2 = $row["temperature2"];
        $row_reading_time = $row["reading_time"];

        //Define specific time zone for each region
        $row_reading_time = date("Y-m-d H:i:s", strtotime("$row_reading_time +
7 hours"));

        echo '<tr>
<td>' . $row_id . '</td>
<td>' . $row_sensor . '</td>
<td>' . $row_location . '</td>
<td>' . $row_temperature1 . '</td>
<td>' . $row_humidity . '</td>
<td>' . $row_pressure . '</td>
<td>' . $row_temperature2 . '</td>
<td>' . $row_reading_time . '</td>
</tr>';
    }
    $result->free();
}

$conn->close();

```

```
?>
</table>
</body>
</html>
```

File on the web's file manager: "esp32_line_chart.php"(data drawn in three-line charts).

```
<?php
$servername="localhost";

// Own database's confidential
$username="";
$password="";
$dbname="";

$conn=new mysqli("$servername","$username","$password","$dbname");

if($conn){

}else{
    echo "Connection Failed";
}
?>

<html>
    <head>

        <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
        <script type="text/javascript">
            google.charts.load('current', {'packages':['corechart']});
            google.charts.setOnLoadCallback(drawChart);

            function drawChart() {
                var data = google.visualization.arrayToDataTable([
                    ['reading_time', 'temperature1', 'temperature2'],

                    //PHP Code to get the temperature data from your table's name
                    <?php
                        $query="select * from ██████████";
                        $res=mysqli_query($conn,$query);
                        while($data=mysqli_fetch_array($res)){
                            $reading_time=$data['reading_time'];
                            $temperature1=$data['temperature1'];
                            $temperature2=$data['temperature2'];
                        }
                    ?>
                    ['<?php echo $reading_time;?>','<?php echo $temperature1;?>',
                    <?php echo $temperature2;?>],
                    <?php
                    }
                ]);

                var options = {
                    title: 'ESP32 Temperature Chart',
                    curveType: 'function',
                    legend: { position: 'bottom' },
                    colors:['green','red'],
                };
```

```

        var chart = new google.visualization.LineChart
(document.getElementById('curve_chart'));

        chart.draw(data, options);
    }

google.charts.setOnLoadCallback(drawChart1);
function drawChart1() {
    var data1 = google.visualization.arrayToDataTable([
        ['reading_time', 'humidity'],

        //PHP Code to get the humidity from your table sensordata
        <?php
            $query="select * from ██████████";
            $res=mysqli_query($conn,$query);
            while($data=mysqli_fetch_array($res)){
                $reading_time=$data['reading_time'];
                $humidity=$data['humidity'];

            ?>
            ['<?php echo $reading_time;?>','<?php echo $humidity;?>'],
        <?php
            }

        ?>

    ]);

    var options1 = {
        title: 'ESP32 Humidity Chart',
        curveType: 'function',
        legend: { position: 'bottom' }
    };

    var chart1 = new google.visualization.LineChart
(document.getElementById('curve_chart1'));

    chart1.draw(data1, options1);
}

google.charts.setOnLoadCallback(drawChart2);
function drawChart2() {
    var data2 = google.visualization.arrayToDataTable([
        ['reading_time', 'pressure'],

        //PHP Code to get the sensor data from your table
        <?php
            $query="select * from ██████████";
            $res=mysqli_query($conn,$query);
            while($data=mysqli_fetch_array($res)){
                $reading_time=$data['reading_time'];
                $pressure=$data['pressure'];

            ?>
            ['<?php echo $reading_time;?>','<?php echo $pressure;?>'],
        <?php
            }

        ?>

    ]);

```



```
var options2 = {
  title: 'ESP32 Pressure Chart',
  curveType: 'function',
  legend: { position: 'bottom' },
  colors:['purple']
};

var chart2 = new google.visualization.LineChart
(document.getElementById('curve_chart2'));

chart2.draw(data2, options2);
}

</script>
</head>
<body style="display:flex; flex-direction: column; margin: 0; padding: 0;
width: 97%; height:97%">
  <div style="display: flex; justify-content: center; align-items:center;
font-style: bold;font-size: 35px; color:red;padding-bottom: 0!important">ESP32
Data Visualization</div>
  <div
    style="
margin: 0;
padding: 0;
display: flex;
justify-content: flex-start;
align-items: center;
width: 100%;
height: 100%;
"
  >

<div id="curve_chart" style="width: 100%; height: 100%"></div>
<div id="curve_chart1" style="width: 100%; height: 100%"></div>
<div id="curve_chart2" style="width: 100%; height: 100%"></div>

</div>
</body>
</html>
```