

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  
**SINGAPORE**

**Final Year Project Final Report**

**Smart Non-Contact Thermometer**

**With ESP32 and Modular Components**

**Submitted by: Chng Wee Ping**

**Matriculation Number: U1722887D**

**Supervisor: Dr Ji-Jon Sit**

**Examiner: A/P Wang Hong**

School of Electrical & Electronic Engineering

A final year project report presented to the Nanyang Technological  
University

in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

**2021**

# Table of Content

Chapter 1	Introduction.....	1
1.1	Background .....	1
1.2	Motivations .....	2
1.3	Objectives and Scope .....	3
1.4	Organisation of Report.....	4
Chapter 2	Literature Review.....	5
2.1	Temp Pal .....	5
2.2	Kinsa Smart Thermometer .....	6
2.3	Best Smart Thermometers 2021 article piece .....	7
Chapter 3	Methodology .....	8
3.1	Hardware.....	8
3.1.1	The Microcontroller (LilyGo ESP32).....	9
3.1.2	The Barcode/QR Scanner (GM65) .....	10
3.1.3	The temperature sensor (MLX90614) .....	12
3.2	Software .....	13
3.2.1	Arduino IDE.....	13
3.2.2	phpMyAdmin/XAMPP .....	14
3.3	Implementation .....	15
3.3.1	The Display .....	15
3.3.2	Reading the data (Scanner and IR Sensor) .....	16
3.3.3	Sending the data (WiFi, php and HTTP request).....	18
3.3.4	Connecting all the components together.....	20
3.4	Final result of the prototype .....	22

3.4.1	Usage of the prototype .....	23
Chapter 4	Conclusion, Difficulties & Future Work .....	24
4.1	Difficulties .....	24
4.1.1	Choosing the microcontroller .....	24
4.1.2	The Barcode Scanner .....	25
4.1.3	Product Design.....	25
4.1.4	Battery and Power Source.....	26
4.2	Recommendation in Future Work.....	27
4.3	Final thoughts.....	28
References	.....	29
Appendix	.....	32

(End of Table of Content)

# Abstract

In light of the Covid-19 pandemic that have shown that fast responses in the detection of the virus is essential in slowing down the virality, there needs to be an efficient method of taking temperatures of many people to detect infections early.

This project aims to prototype a non-contact thermometer that can record the details of an individual via a barcode/QR scanner and is also able to send data wirelessly via Wifi or Bluetooth to a database. Portability is the key feature of this project, a feature which many existing products is missing at the time of writing.

This project is envisioned to be best utilized in hospitals without advanced temperature management, especially in makeshift hospitals that appear in pandemics. It could also be used to check-in and record the temperature of personnel within a building.

A microcontroller, ESP32, will be the heart of this project, which will be complemented with modular components – GM65 Barcode Scanner and MLX90614 Infrared Temperature Sensor.

The GitHub link for this project is: <https://github.com/DevilRulerEx/fyp-smart-thermometer>

# Acknowledgements

The author would like to express his deepest appreciation towards all that have provided any form of assistance and guidance on the completion of this project.

The author would like to give the biggest thanks to the project supervisor, Dr Ji-Jon Sit for the opportunity to work on this Final Year Project, as well as the constant advices and encouragements. This project has benefitted greatly from the help of Dr Ji-Jon Sit.

# Acronyms

BLE	Bluetooth Low Energy
ULP	Ultra-Low Power
UART	Universal Asynchronous Receiver-Transmitter
IDE	Integrated Development Environment
IR	Infrared
QR	Quick Response
ID	Identity Document
IPS	In-Plane Switching

# List of Figures

Figure 1 Temp Pal.....	5
Figure 2 Kinsa Smart Thermometer .....	6
Figure 3 Article piece on best smart thermometers .....	7
Figure 4 LilyGo ESP32.....	9
Figure 5 GM65 Barcode Reader .....	10
Figure 6 Singapore's national identity card with the 1D barcode. ....	10
Figure 7 Philippines Postal ID card with the 2D QR code. ....	10
Figure 8 Dimensions of GM65 .....	11
Figure 9 MLX90614 Temperature Sensor.....	12
Figure 10 Arduino IDE logo.....	13
Figure 11 XAMPP Logo.....	14
Figure 12 phpMyAdmin Logo.....	14
Figure 13 An early rendition of the UI .....	15
Figure 14 Prototype UI .....	15
Figure 15 Snippets of MLX code .....	16
Figure 16 Snippet code for initialising GM65 .....	16
Figure 17 Snippet Code of scanner data to string.....	16
Figure 18 Reset Factory Setting.....	17
Figure 19 UART Output .....	17
Figure 20 Continuous Mode .....	17
Figure 21 Snippet WiFi Setup code .....	18
Figure 22 Snippet code of HTTP request .....	19
Figure 23 Database Table .....	19
Figure 24 Records taken in the project. ....	19
Figure 25 Pin Diagram of LilyGo ESP32.....	20
Figure 26 Circuit Diagram of the project.....	20
Figure 27 Toggle/Push lock switch .....	21
Figure 28 Push switch.....	21

Figure 29 Input_Pullup for buttons.....	21
Figure 30 Inside workings of the prototype.....	22
Figure 31 Exterior look of the prototype .....	23



# List of Tables

Table 1 Simple product description of the LilyGo ESP32 .....	9
Table 2 Simple product description of GM65 Barcode Scanner .....	11
Table 3 Product Description of MLX90614 .....	12

# Chapter 1 Introduction

## 1.1 Background

The 21st century has shown that despite advances in science and medicine, the large population of the world are still susceptible to viral infectious diseases. Recent pandemics has proven that it is a huge challenge to monitor the health of the general populace. [1]

The Covid-19 pandemic has speeded the adoption of digital technologies for many business and government sectors [8]. In Singapore, technologies were used for contact tracing, ensuring compliances with stay-at-home notices and work-from-home implementations. [9]

Therefore, there are lot of potential of using technologies to solve a problem in a pandemic, such as a smart thermometer for this project.

## 1.2 Motivations

Today, digital thermometers are the most popular type of thermometer due to its' reliance and affordability. However, digital thermometers do not usually keep a record on the individual's temperature to keep track of the individual's health. [2]

At the time of writing, facial recognition scanner with a built-in temperature sensor technologies are deployed in workplaces and schools, and are endorsed by governments [3][7], but the technology has raised some concerns about privacy and data security. [4]

There are also no commercially available thermometers at the start of this project which can read an individual details, send data wirelessly and are also portable altogether.

The author has resided in Singapore during the Covid-19 pandemic, and has seen the implementation of a barcode scanner and temperature sensor for contact tracing purpose in temperature screening areas around malls and offices. Hence, an idea was formed to combine the barcode scanner and temperature sensor in a portable device.

## **1.3 Objectives and Scope**

This projects aims to design and develop a smart thermometer using the ESP32 platform. The smart thermometer would have an IR temperature sensor to take temperature readings, a 2D/3D scanner to read barcodes or QR codes of a person's identification number; A database would be created on phpMyAdmin to take in all records of the reading in this project; The Wifi feature of the ESP32 will be used to send the data wirelessly to the created database.

The target group of this project would be hospitals which does not have advanced temperature management, especially makeshift hospitals that are created during pandemics. This would help to facilitate the collection of temperature data of patients which can be analysed by medical experts.

## **1.4 Organisation of Report**

This report includes Chapter 1 which serves as an introduction of the project, which details the motivations, objectives and scope of this project.

Chapter 2 includes the literature review, which will detail market research for existing commercial products that are similar to this project.

Chapter 3 will discuss the software and hardware used in this project, the implementations, and how the prototype for the demonstration was conceived.

The last chapter of this report, Chapter 4, concludes this report by summarizing the work done by this author. It will also discuss further recommended work that could improve this project in the future.

# Chapter 2

## Literature Review

This chapter will discuss about commercially available products that are similar to this project, and how this project will be improving upon them.

### 2.1 Temp Pal



Figure 1 Temp Pal

Founded by Taiwan-based healthcare wearable device company, iWEECARE. This small smart thermometer weighs only 3 grams and can last 36 hours on a single full charge. It will be stuck onto the body skin; temperature readings will then be transmitted via BLE to a mobile app and cloud dashboard [10].

In the Covid-19 situation, it provided real-time temperatures of many patients in the hospital, while also reducing direct contact between hospital staff and patients. The Nanjing City of China has implemented this system in some residential districts to help with self-quarantine management [10].

The drawback to Temp Pal is that each device is only limited to one individual. For makeshift hospitals or budget-constrained hospitals, getting this technology into each and every one of their patients might be logistically challenging.

My project's prototype may still need manual reading of temperatures, but the barcode scanner features will record any patients' IDs along with their temperatures at no setup time so long as they have a unique ID.

## 2.2 Kinsa Smart Thermometer



Figure 2 Kinsa Smart Thermometer


This thermometer will need to be pair up with a mobile app, temperature readings will be sent to the mobile app wirelessly via Bluetooth, and the mobile app itself will have personalised guidance according to the temperature readings over time [16].

The limitation is that each individual must have an account set up to use the smart features, which means this is ideally meant only for a family each.

## 2.3 Best Smart Thermometers 2021 article piece

iMore
E

**Staff Pick**




Source: Withings

**Keep it smart**  
**Withings Thermo**

This smart thermometer uses 16 infrared sensors that take over 4000 measurements to provide highly accurate results. Data from these measurements and health advice automatically appears in the Withings Thermo app via Wi-Fi for iOS and Android users. Up to eight users can access their personal temperature histories with the ability to share with their respective health care professionals if they choose.


\$99 at Amazon



Source: iHealth

**Smart chip tech**  
**iHealth Forehead Thermometer**

iHealth no-touch thermometer features three ultra-sensitive sensors and the latest smart chip technology to deliver instant readings and ensure accurate performance. An sensitive infrared sensor collects more than 100 data points per second from the forehead. The additional distance and environmental sensors make necessary adjustments to give you accurate readings. The large LED screen displays readings in bright white light, and you get a gentle vibration notification instead of a loud beep when the reading is complete



Source: VAVA

**Keep the whole family healthy**  
**VAVA Smart Thermometer**

This smart thermometer features a clean design and comes with 10 adhesive strips for continuous temperature monitoring. It features a soft silicone patch designed to be affixed to your baby's armpit while they sleep. You can monitor the baby's temperature from anywhere in the house with the home charging unit. The monitor will instantly alert you if your little one's temperature rises over a preset number, allowing tired parents to get some much-needed rest.

\$80 at Amazon

Figure 3 Article piece on best smart thermometers

In an article that list out their list of the best smart thermometer, most of the listed thermometers are for personal use and none of them have a barcode scanner module to record identities [15].

Therefore, a prototype to use barcode scanners with a temperature reading sensor that is also portable is worth to research and work on.



# Chapter 3 Methodology

## 3.1 Hardware

This project consists of the following hardware that are essential to this project:

- LILYGO® TTGO T-Display ESP32 WiFi And Bluetooth Module  
Development Board 1.14 Inch LCD Control Board
- GM65 Bar Code Reader Module
- MLX90614 Non-Contact Infrared Temperature Sensor

Other components are used for the prototype but will not be discussed much:

- Push button
- Toggle button
- AA Batteries with holder
- Cardboard casing

### 3.1.1 The Microcontroller (LilyGo ESP32)



Figure 4 LilyGo ESP32

This neat piece of technology measures up to 51.5mm by 25mm with a thickness of only 8.5mm, and yet it has many features in the small size. The model used in this project has an integrated WiFi, Bluetooth and IPS display. It is cheaper and much powerful than most Arduino boards. Overall, the compact size, processing capacity and the compatibility with programming language used in the well-established Arduino, makes this ESP32 model a perfect choice as the heart for this project.

Table 1 Simple product description of the LilyGo ESP32

Chipset	ESPRESSIF-ESP32 240MHz Xtensa® single-/dual-core 32-bit LX6 microprocessor
Modular interface	UART、SPI、SDIO、I2C、LED PWM、TV PWM、I2S、IRGPIO、ADC、capacitor touch sensor DACLNA pre-amplifier
Working voltage	2.7V-4.2V
Wi-Fi	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC (esp32 chip)
Bluetooth	Meet bluetooth v4.2BR/EDR and BLE standard

### 3.1.2 The Barcode/QR Scanner (GM65)

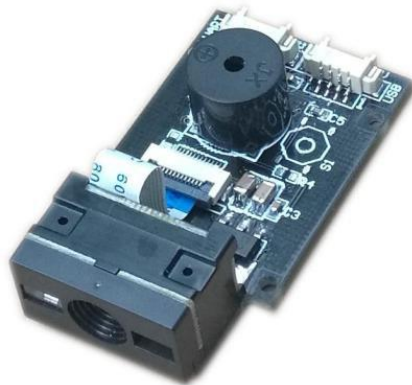


Figure 5 GM65 Barcode Reader

This scanner is able to read and translate 1D or 2D barcodes into eligible text formats. It will serve as the apparatus to record names or identity numbers from barcodes or QR codes found in a person's national identity card; in cases which the national identity card does not have 1D/2D barcodes, there are several free online tools that will convert a string of national identity number to barcodes/QR codes.



Figure 6 Singapore's national identity card with the 1D barcode.



Figure 7 Philippines Postal ID card with the 2D QR code.

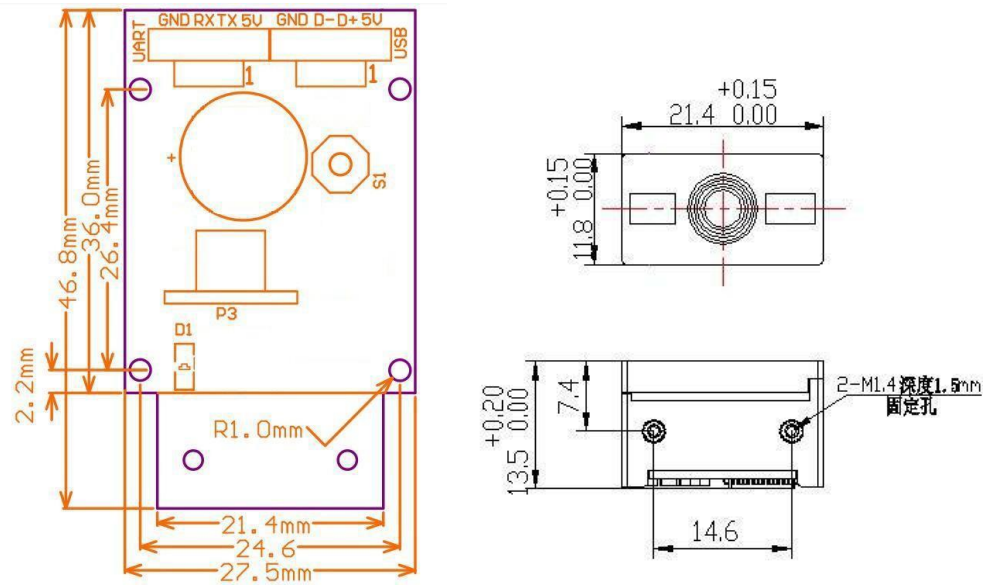


Figure 8 Dimensions of GM65

The GM65 barcode reader is one of the very few compact modules that works with the Arduino platform. The small size and programmability of this module is suitable to be embedded into this project.

Data can be received and transmitted through the UART connection. The scanner has several pre-set settings that can be configured by receiving low-level language machines code or by scanning certain command QR codes, information can be found on the datasheet of this module.

The main drawback of this module is the high operating voltage and the difficulty in understanding the machine code needed to program this module.

Table 2 Simple product description of GM65 Barcode Scanner

Interface	USB, UART, USB Vcom
Operating Voltage	DC 4.2-6.0V
Operating Current	160mA

### 3.1.3 The temperature sensor (MLX90614)



Figure 9 MLX90614 Temperature Sensor

This infrared temperature sensor came in highly recommended as the module to have for a non-contact thermometer in many Arduinos projects. This module measures 11.3mm by 16.9mm and is compact enough to fit into many Arduino/ESP32 projects. It is also easy to program due to the huge support by the manufacturer and users.

Table 3 Product Description of MLX90614

Operating Voltage	2.6V to 3.6V
Supply Current	1.5mA
Object Temperature Range	-70° C to 382.2°C
Accuracy	0.02°C
Reading Distance	2cm-5cm (approx..)

## 3.2 Software

This project consists of the following software that helped to integrate all the components together:

- Arduino IDE
- phpMyAdmin/XAMPP

### 3.2.1 Arduino IDE



Figure 10 Arduino IDE logo

With some initial setup, the ESP32 can be configured to work in the Arduino IDE with the same programming language as Arduino [13]. The ESP32 would be able to utilise the various Arduino libraries that exist, which helped to build this project.

### 3.2.2 phpMyAdmin/XAMPP



Figure 11 XAMPP Logo



Figure 12 phpMyAdmin Logo

For the purpose of data collection of people's ID and temperature, a database is needed.

Therefore, XAMPP, a simple and lightweight program that create a local web server on a computer is used for the project's demonstration. Built-in with the local web server by XAMPP is phpMyAdmin, which will actually store all the records of ID and temperatures in this project.

## 3.3 Implementation

This section will illustrate how the hardware and software are formed as an ecosystem methodically.

The full code used in this project can be found in the appendix section or be viewed in the GitHub link provided in the abstract section.

### 3.3.1 The Display



Figure 13 An early rendition of the UI

A third-party library, <TFT\_eSPI.h> is needed to configure the embedded display to create a thermometer UI. The UI displays the recorded name, the temperature taken and status message that shows the actions taken in this project.



Figure 14 Prototype UI



### 3.3.2 Reading the data (Scanner and IR Sensor)

<Adafruit\_MLX90614.h> is the library required to declare a custom mlx variable that can start sensing temperatures.

```
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

read_temperature = String(mlx.readObjectTempC());
```

Figure 15 Snippets of MLX code

As the barcode scanner uses UART connection to the ESP32, a new serial is allocated for the scanner.

```
Serial2.begin(9600, SERIAL_8N1, 25, 26); //25 is TX (Black), 26 is RX (Yellow)
```

Figure 16 Snippet code for initialising GM65

A function is created to convert what the scanner reads to string.

```
void serialScannerEvent() {
  //
  if (Serial2.available() > 0) {
    // get the new byte:
    char inChar = (char)Serial2.read();
    // add it to the inputString:
    inputString += inChar;
    countstr++;
    millisendstr = millis();
  }
  else {
    if (millis() - millisendstr > 1000 && countstr > 0) {
      stringComplete = true;
    }
  }
}
```

Figure 17 Snippet Code of scanner data to string

The barcode scanner to set to continuous scanning mode, where the scanner will scan automatically and continuously as long as power is supplied to the scanner.

The scanner scanned these series of QR codes to configure the settings needed for the project:



Figure 18 Reset Factory  
Setting



Figure 19 UART Output



Figure 20 Continuous Mode

### 3.3.3 Sending the data (WiFi, php and HTTP request)

WiFi from the ESP32 is a main mode of wireless data transfer for this project.

```
#include <WiFi.h>

const char* ssid      = "Chng Laptop";
const char* password = "XXXXXXXXXX";
char server[] = "192.168.1.1";

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
    status_message="Connecting to WiFi";
}
Serial.println("Connected to the WiFi network");
```

Figure 21 Snippet WiFi Setup code

After setting up a local web server in XAMPP, a php script was created to receive data sent by the ESP32 and be recorded in the phpMyAdmin database. (See Appendix 2) The php script is named as **get\_temperature.php**.

A HTTP request was created in a function to connect to the php script, sending over the recorded ID and temperature to the php script.

```
if (client.connect(server, 80)) {
    Serial.println("connected");
    //Make a HTTP request:
    String str="/get_temperature.php?inputName=";
    String str2=String(inputName);
    str+=str2;
    str+="&send_temperature="+send_temperature;
    Serial.print("str=");Serial.println(str);

    client.println("GET "+str+" HTTP/1.1");
    // client.println("Host: 192.168.1.100");
    client.println("Host: 192.168.1.100");
    client.println("Connection: close");
    client.println();
}
```

Figure 22 Snippet code of HTTP request

In phpMyAdmin, a table is created to record the data from the script.

#	Name	Type
<input type="checkbox"/> 1	<b>id</b>	int(10)
<input type="checkbox"/> 2	<b>personid</b>	text
<input type="checkbox"/> 3	<b>datetime</b>	datetime
<input type="checkbox"/> 4	<b>temperature</b>	decimal(4,2)

Figure 23 Database Table

	id	personid	datetime	temperature
<input type="checkbox"/> Edit Copy Delete	24	S93	2021-03-22 15:28:02	23.51
<input type="checkbox"/> Edit Copy Delete	23	S75	2021-03-22 15:06:54	32.35
<input type="checkbox"/> Edit Copy Delete	22	S93	2021-03-22 09:00:53	22.83
<input type="checkbox"/> Edit Copy Delete	21	Hello_World	2021-03-22 07:39:48	36.77

Figure 24 Records taken in the project.

### 3.3.4 Connecting all the components together

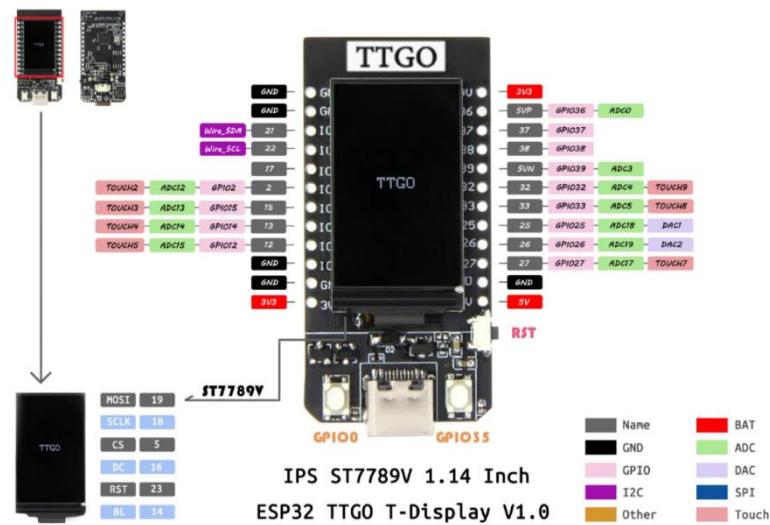


Figure 25 Pin Diagram of LilyGo ESP32

The barcode scanner, temperature sensor and buttons are allocated to certain pins, each pin has coded function to each the connected components.

### SKETCH OF CIRCUIT DIAGRAM

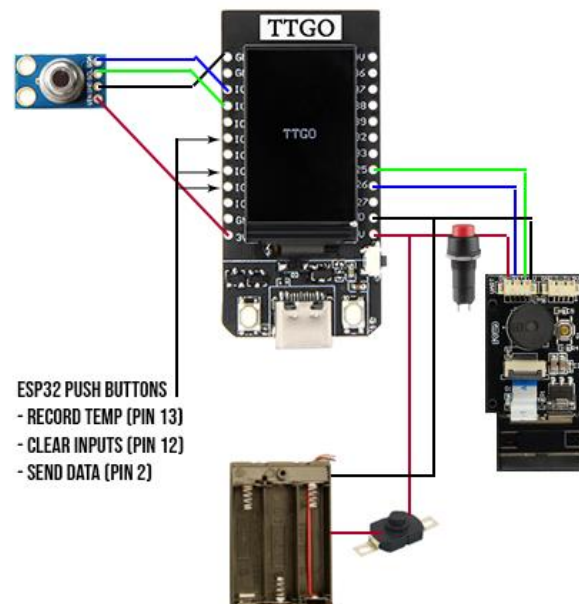


Figure 26 Circuit Diagram of the project

### 3.3.4.1 Battery

A 3-slot AA battery holder provide 4.5V voltage to the whole circuit, an additional AA battery is connected in series to provide a total voltage of 6V after testing of the first assembled prototype.

### 3.3.4.2 Buttons



Figure 27 Toggle/Push lock switch



Figure 28 Push switch

A toggle/push lock button is placed between the power source and 5V pin of the ESP32 to act as an on/off switch for the prototype.

A push button is placed between the 5V pin of the ESP32 and the Vin of the barcode scanner to act as a button to control the flow of power.

3 other push buttons initialised a function when they are pressed, they are used for sending the data to the database, recording the temperature, and clearing any input in the memory. The 3 push buttons use “*INPUT\_PULLUP*”, which result in button state to be HIGH when not pressed and LOW otherwise.

```
pinMode(buttonSend, INPUT_PULLUP);
pinMode(buttonRecord, INPUT_PULLUP);
pinMode(buttonClear, INPUT_PULLUP);
```

Figure 29 Input\_Pullup for buttons

### 3.4 Final result of the prototype

After many trails-and-errors in trying to figure out the best way to assemble all the components as a prototype, I ended up with using a cardboard box with holes cut out for the components, blue-tack and double-sided tapes to hold all the components in place, and I directly soldered all the connections with wires.

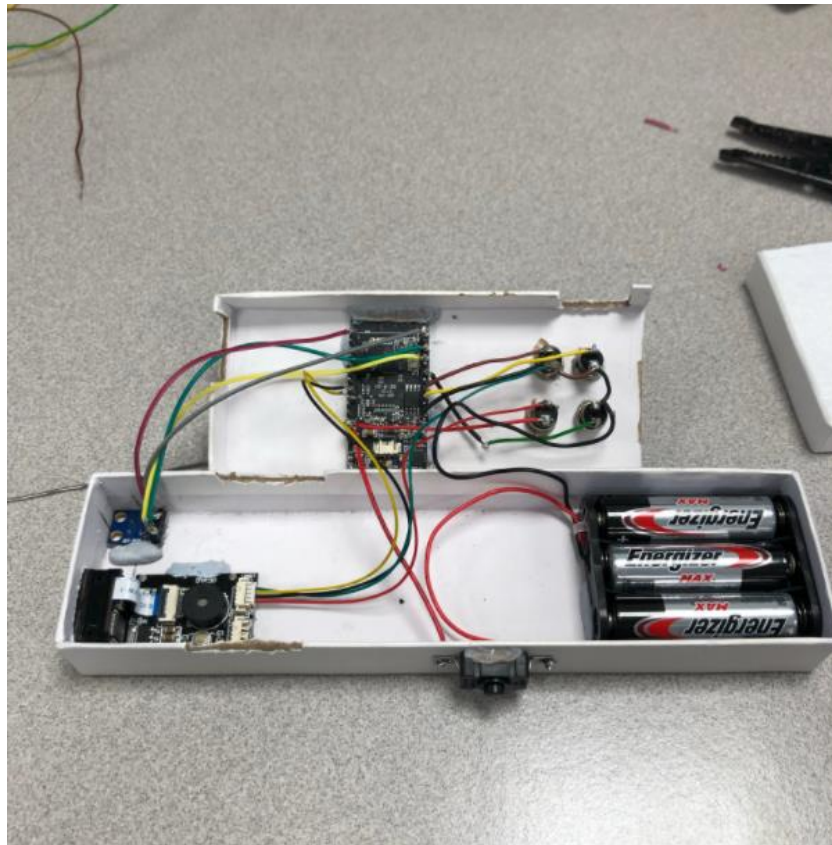


Figure 30 Inside workings of the prototype

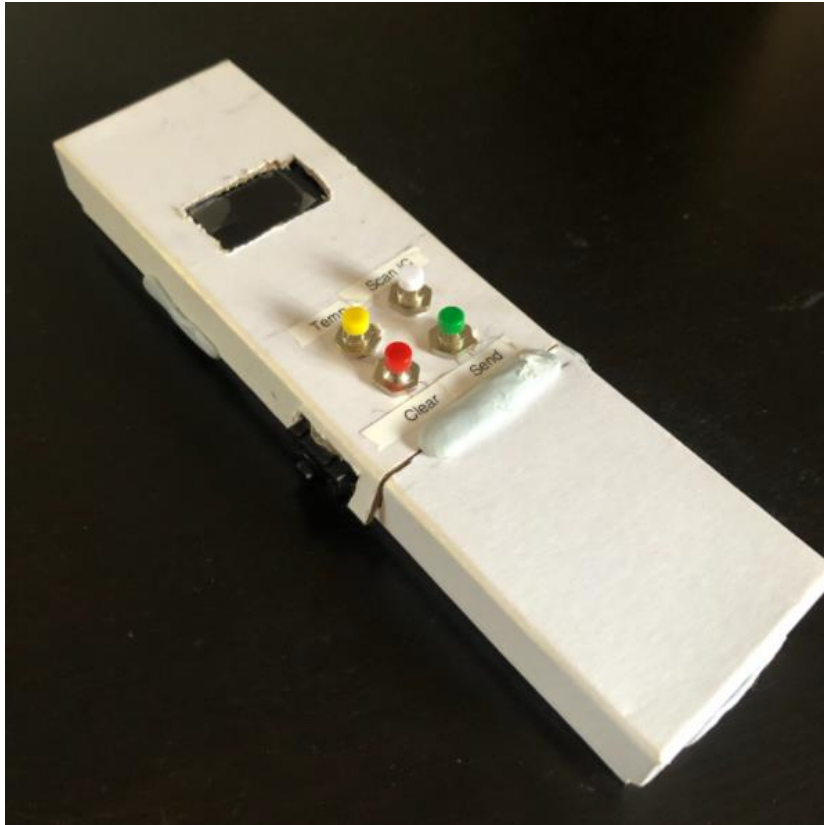


Figure 31 Exterior look of the prototype

### 3.4.1 Usage of the prototype

1. Turn on the power button.
2. Allow your prototype to connect to WiFi (hardcoded SSID and password)
3. Scan the temperature of a person with the press of a button.
4. Scan the 1D/2D code of a person with the press of another button.
5. Send the data wirelessly to the database with the press of another button.

Step 3-5 are repeatable, and it takes around 10 seconds to record and send data.



# **Chapter 4 Conclusion, Difficulties & Future Work**

## **4.1 Difficulties**

### **4.1.1 Choosing the microcontroller**

An Arduino was my first choice when I first embarked on this final year project, as I am very familiar with the architecture. However, I was very glad that when I submit my project plan to my supervisor, Dr Ji-Jon Sit, he recommended me to use an ESP32 instead, which turns out to be cheaper, more powerful, have more features, and is smaller than Arduino boards. My knowledge of Arduino was also transferable to using ESP32.

At the start of the project, many ESP32 boards do not have an integrated display, an IPS display embedded together with the ESP32 board is relatively new and thus documentations on the usage of the display are vague.

I actually have to look through the review section on the website where I buy my ESP32 from to find the solutions to get the display working.

### **4.1.2 The Barcode Scanner**

The datasheet document for the GM65 scanner is very confusing for an inexperienced user on sending machine codes for commands, and there are not many projects that use the barcode scanner, thus there are not many documentations for it, and it could have potentially ended up as a paperweight for this project.

Fortunately, the only project that uses the barcode scanner I could find at the start of the project suits my need and has helped me utilize the barcode scanner [14].

### **4.1.3 Product Design**

There are just so many ways to go about when it comes to making a product design choice for my prototype. I have little experience on product design, and yet I even had planned to learn 3D printing, but I dropped the idea later as I think it may not payoff in the end. I was also not good at soldering.

There were a lot of unforeseen problems when I was doing trails-and-errors during the assembling phase such as a certain non-essential component being too big for my selected casing, and the length and type of electrical wires used not suitable for the prototype.

Thanks to my supervisor, Dr Ji-Jon Sit, he gave me a lot of good pointers when it comes to assembling the prototype. The final prototype drew a lot of inspirations from his advices.

#### **4.1.4 Battery and Power Source**

This problem is attributed to my lack of experience with product designs. When I was testing all the components through a breadboard, a 4.5 voltage from a 3-slot AA batteries holder was deemed sufficient to power up the ESP32, temperature sensor and barcode scanner. Hence, the cardboard box used in this prototype was chosen as it was just the right size to fit the 3-slot AA batteries holder.

However, when the final prototype was assembled with the introduction of button switches and longer wirings, the electrical resistances of the whole project was found to be just enough for the prototype to power off randomly and reset, essentially making the 4.5V power source non-ideal for this project. Attempts have been made to include an additional AA battery in series as well as changing the 3-slot AA batteries holder to a 4-slot AAA batteries holder, but the cardboard box and the soldered wires did not leave enough space for further modifications.

Fortunately, the prototype was designed to be able to receive power from the USB-C slot in the ESP32.

## **4.2 Recommendation in Future Work**

The prototype's functionalities can be further upgraded and improved in the future.

Product design for the prototype could definitely be improved a lot more, the cardboard box can be replaced with a sturdier case which may require learning how to do laser cutting. The ways to hold the components in place and the wiring management also leave a lot to be desired.

The use of different temperature sensors can also be explored, as the accuracy of the current temperature sensor becomes lower the longer I use it (the sensor might be damaged during the assembling phase of my project).

The prototype could be pitched to medical experts and hospital management, and feedback can be gathered on whether the project's implementation of a smart thermometer can help with temperature management with many patients.

Alternatively, with how my project is being programmed, I could make a few changes to my code and replace the IR temperature sensor and barcode scanner with just a thinner temperature sensor such as the TMP117 which may be suitable for wearables. This new implementation will allow users to wear the temperature sensor as they continue about their daily life, while automatically send temperature records at interval timings, similar to the Temp Pal product in the literature review section.

## **4.3 Final thoughts**

Overall, I am very satisfied with how my prototype has turned out. I could foresee a more finalised version of my prototype be deployed in hospitals, where it could not only take the temperatures of many patients efficiently, but also record the patient ID along with the temperature reading to be send to a database for analysis if needed.

I have very glad to have experimented with relative new technology such as the ESP32 and barcode scanner. While I may not have mastered all the potential there is with the ESP32 and barcode scanner, I have learnt a lot on how to use them and really appreciate how these modular technology can help to improve the world.

I have also learnt to be better in the product design, and is more confident to embark on more difficult materials to work on more future projects.

In conclusion, this project required engineering requirements and knowledge. Hence, many valuable skillsets were gained throughout the journey of this final year project.

# References

- [1] World Health Organization, “Maintaining essential health services during the COVID-19 outbreak,” [Online]. Available: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/related-health-issues>. [Accessed 11 March 2021].
- [2] G. Connolly, “Different types of thermometer,” 12 December 2016. [Online]. Available: <https://www.ashtonshospitalpharmacy.com/blog/2016/12/different-types-of-thermometer/>. [Accessed 19 December 2020].
- [3] Y. C. Chee, “Lessons from SARS,” 2003/2004. [Online]. Available: [https://www.sma.org.sg/UploadedImg/files/Lessons%20from%20SARS%20\(Prof%20Chee%20Yam%20Cheng,%20SMA%20News%202003-2004\).pdf](https://www.sma.org.sg/UploadedImg/files/Lessons%20from%20SARS%20(Prof%20Chee%20Yam%20Cheng,%20SMA%20News%202003-2004).pdf). [Accessed 20 May 2020].
- [4] A. H. Min, “COVID-19: Demand for sanitisers, antibacterial soap and thermometers still high, say retailers,” Channel News Asia, 20 Feb 2020. [Online]. Available: <https://www.channelnewsasia.com/news/singapore/covid-19-demand-for-sanitisers-antibacterial-soap-and-12450608>. [Accessed 20 May 2020].
- [5] “Visitor-Management-(Temperature-Screening-Solution),” [Online]. Available: [https://www.ncss.gov.sg/Our-Initiatives/Tech-and-Go/Centre-Management-and-Covid-19-Solutions/Visitor-Management-\(Temperature-Screening-Solution\)](https://www.ncss.gov.sg/Our-Initiatives/Tech-and-Go/Centre-Management-and-Covid-19-Solutions/Visitor-Management-(Temperature-Screening-Solution)). [Accessed 15 March 2021].
- [6] “The Facial Recognition Debate,” [Online]. Available: <https://www.informationweek.com/big-data/ai-machine-learning/the-facial-recognition-debate/a/d-id/1336767?>. [Accessed 15 March 2021].

- [7] “Entering a Building May Soon Involve a Thermal Scan and Facial Recognition,” [Online]. Available: <https://spectrum.ieee.org/the-human-os/biomedical/devices/entering-a-building-may-soon-involve-a-thermal-scan-and-facial-recognition>. [Accessed 15 March 2021].
- [8] McKinsey & Company, “How COVID-19 has pushed companies over the technology tipping point—and transformed business forever,” 5 October 2020. [Online]. Available: <https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever>. [Accessed 15 March 2021].
- [9] J. Ong, “Technology is critical to Singapore's Covid-19 response: PM Lee,” The Straits Times, 17 November 2020. [Online]. Available: <https://www.straitstimes.com/singapore/politics/technology-is-critical-to-singapores-covid-19-response-pm-lee-hsien-loong-tells>. [Accessed 15 March 2021].
- [10] D. Koh, “Temp Pal smart thermometer helps reduce COVID-19 spread in hospitals,” 15 April 2020. [Online]. Available: <https://www.mobihealthnews.com/news/apac/temp-pal-smart-thermometer-helps-reduce-covid-19-spread-hospitals>. [Accessed 15 March 2021].
- [11] J. Teo, “Coronavirus: Singapore Expo to house first batch of mild cases tomorrow to free up hospitals,” 9 April 2020. [Online]. Available: <https://www.straitstimes.com/singapore/health/spore-expo-to-house-first-batch-of-mild-cases-tomorrow-to-free-up-hospitals>. [Accessed 15 March 2021].
- [12] “LilyGo Product View,” [Online]. Available: [http://www.lilygo.cn/prod\\_view.aspx?TypeId=50033&Id=1126&FId=t3:50033:3](http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=1126&FId=t3:50033:3). [Accessed 15 March 2021].

- [13] “Installing the ESP32 Board in Arduino IDE (Windows, Mac OS X, Linux),”  
[Online]. Available: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>. [Accessed 8 November 2020].
- [14] “Wireless Product Tracking (Arduino MKR & Barcode Scanner) © GPL3+,”  
[Online]. Available: <https://create.arduino.cc/projecthub/sanruskmv/wireless-product-tracking-arduino-mkr-barcode-scanner-3708d9>. [Accessed 5 December 2020].
- [15] N. R. Mick Symons, “Best Smart Thermometers 2021,” 5 December 2020.  
[Online]. Available: <https://www.imore.com/best-smart-thermometers>. [Accessed 15 March 2021].
- [16] R. Reader, “Smart thermometers could be the secret to reopening schools,”  
11 May 2020. [Online]. Available: <https://www.fastcompany.com/90485719/smart-thermometers-could-be-the-secret-to-re-opening-schools>. [Accessed 15 March 2021].



# Appendix

Appendix 1: The main code embedded in the prototype.

```

DisplayTemp_v2$
// New background colour
#define TFT_BROWN 0x38E0

// Pause in milliseconds between screens, change to 0 to time font rendering
#define WAIT 2000

// Wifi Connection
#include <WiFi.h>
//const char* ssid      = "Ch[REDACTED]";
//const char* password = "[REDACTED]";
const char* ssid      = "Ch[REDACTED]";
const char* password = "[REDACTED]";
WiFiClient client;
//char server[] = "192.[REDACTED]"; //My Home IP
char server[] = "192.[REDACTED]"; //NTU IP
unsigned long dataPreviousTime = 0;
const long dataInterval = 10000;

//OLED display
#include <TFT_eSPI.h> // Graphics and font library for ST7735 driver chip
#include <SPI.h>
TFT_eSPI tft = TFT_eSPI(); // Invoke library, pins defined in User_Setup.h
unsigned long targetTime = 0; // Used for testing draw times
const long interval = 2000; //The 'delay' time.
String status_message="";

```

```

const int buttonRecord = 13;
int recordState;
int lastRecordState = HIGH;
unsigned long lastRecordTime = 0; // the last time the output pin was toggled
unsigned long recordDelay = 50; // the debounce time; increase if the output flickers

const int buttonClear = 12;
int clearState;
int lastClearState = HIGH;
unsigned long lastClearTime = 0; // the last time the output pin was toggled
unsigned long clearDelay = 50; // the debounce time; increase if the output flickers

const int buttonScan = 15;
int scanState;
int lastScanState = HIGH;
unsigned long lastScanTime = 0; // the last time the output pin was toggled
unsigned long scanDelay = 50; // the debounce time; increase if the output flickers

//Temperature Sensor
#include <Wire.h>
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
String read_temperature="";
String send_temperature="";

//Barcode Scanner
//#include <Arduino.h>
String inputString = "";
String inputName = "";
String lastinputName = "";
String testName = "TestName";
boolean stringComplete = false;
int countstr=0;
unsigned long millisendstr=0;

//Buttons
const int buttonSend = 2;
int sendState;
int lastSendState = HIGH;
unsigned long lastSendTime = 0; // the last time the output pin was toggled
unsigned long sendDelay = 100; // the debounce time; increase if the output flickers

```

```

void setup(void) {
  Serial.begin(9600);
  Serial2.begin(9600, SERIAL_8N1, 25, 26); //25 is TX (Black), 26 is RX (Yellow)

  tft.init();
  tft.setRotation(45); //Set to 0 for portrait, 45 for landscape.

  mlx.begin(); //22 is SCL (Yellow), 21 is SDA (Green)

  inputString.reserve(20);

  pinMode(buttonSend, INPUT_PULLUP);
  pinMode(buttonRecord, INPUT_PULLUP);
  pinMode(buttonClear, INPUT_PULLUP);
  pinMode(buttonScan, INPUT_PULLUP);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
    status_message="Connecting to WiFi";
  }
  Serial.println("Connected to the WiFi network");
  status_message="WiFi connected!";
}

void loop() {
  int sendReading = digitalRead(buttonSend);
  if (sendReading != lastSendState) {
    // reset the debouncing timer
    lastSendTime = millis();
  }

  int recordReading = digitalRead(buttonRecord);
  if (recordReading != lastRecordState) {
    // reset the debouncing timer
    lastRecordTime = millis();
  }

  int clearReading = digitalRead(buttonClear);
  if (clearReading != lastClearState) {
    // reset the debouncing timer
    lastClearTime = millis();
  }

  int scanReading = digitalRead(buttonScan);
  if (scanReading != lastScanState) {
    // reset the debouncing timer
    lastScanTime = millis();
  }
}

```

```

serialScannerEvent();
sendState = digitalRead(buttonSend);
recordState = digitalRead(buttonRecord);
clearState = digitalRead(buttonClear);
scanState = digitalRead(buttonScan);

if (stringComplete) {
    Serial.println(inputString);
    Serial.println("String Complete!");
    inputName = String(inputString);
    inputName.replace(" ", "_");
    inputName.trim();
    status_message = "ID Recorded!";
    inputString = "";
    //send_temperature = read_temperature;
    stringComplete = false;
    countstr=0;
}

//Sending the data
if ((millis() - lastSendTime) > sendDelay) {
    if (sendReading != sendState) {
        sendState = sendReading;
        if (sendState == LOW && inputName != NULL && send_temperature != NULL) {
//            if (sendState == LOW) {
                Serial.println("Send button Pressed");
                status_message = "Sending data...";
                send_temp_to_server();
                clear_data();
            }
            if (sendState == LOW && inputName == NULL || send_temperature == NULL) {
                status_message = "Missing inputs!";
            }
        }
    }
}

//Reading the data
if ((millis() - lastRecordTime) > recordDelay) {
    if (recordReading != recordState) {
        recordState = recordReading;
        if (recordState == LOW) {
            Serial.println("Record button Pressed");
            status_message = "Temp Recorded!";
            send_temperature = read_temperature;
        }
    }
}
}

```

```

//Clearing the data
if ((millis() - lastClearTime) > clearDelay) {
    if (clearReading != clearState) {
        clearState = clearReading;
        if (clearState == LOW) {
            Serial.println("Clear button Pressed");
            status_message = "Data Cleared!";
            clear_data();
        }
    }
}

drawInformation();

lastSendState = sendReading;
lastRecordState = recordReading;
lastClearState = clearReading;

}

void serialScannerEvent() {
    //
    if (Serial2.available() > 0) {
        // get the new byte:
        char inChar = (char)Serial2.read();
        // add it to the inputString:
        inputString += inChar;
        countstr++;
        millisendstr = millis();
    }
    else {
        if (millis() - millisendstr > 1000 && countstr > 0) {
            stringComplete = true;
        }
    }
}

```

```

void drawInformation() {
    unsigned long currentMillis = millis();
    if (currentMillis - targetTime >= interval) {
        targetTime = currentMillis;
        tft.setTextSize(1);
        tft.fillScreen(TFT_BLACK);
        tft.setTextColor(TFT_GREEN, TFT_BLACK);

        tft.drawString("Name or ID:", 0, 0, 4);
        tft.drawString(inputName, 0, 24, 2);
        tft.drawString("Temperature: "+send_temperature+"°C", 0, 48, 4);
        read_temperature = String(mlx.readObjectTempC());
        tft.drawString("Reading: "+read_temperature+"°C", 0, 72, 4);
        //tft.drawString("Reading:", 0, 96, 4);
        tft.setTextColor(TFT_RED, TFT_BLACK);
        tft.drawString(status_message, 0, 120, 2);
    }
}

void send_temp_to_server() {
    client.stop();
    unsigned long dataCurrentTime = millis();

    if (client.connect(server, 80)) {
        Serial.println("connected");
        //Make a HTTP request:
        String str="/get_temperature.php?inputName=";
        String str2=String(inputName);
        str+=str2;
        str+="&send_temperature="+send_temperature;
        Serial.print("str=");Serial.println(str);

        client.println("GET "+str+" HTTP/1.1");
        // client.println("Host: 192.168.1.100"); //HOME
        client.println("Host: 192.168.1.100"); //NTU
        client.println("Connection: close");
        client.println();

        status_message = "Data sent!";

    } else {
        Serial.println("connection failed");
        status_message = "Connection failed!";
    }
}

```

```
void clear_data(){  
    inputName = "";  
    send_temperature = "";  
}
```

## Appendix 2: get\_temperature.php

```
get_temperature.php ●
get_temperature.php
1  <?php
2
3  date_default_timezone_set("Asia/Singapore");
4
5  $location="localhost";
6  $user="root";
7  $pass="";
8  $db_name="esp32_thermometer";
9
10 // connect db
11 $conn = new mysqli($location, $user, $pass, $db_name);
12
13 // Check connection
14 if ($conn->connect_error) {
15     die("Connection failed: " . $conn->connect_error);
16 }
17
18 $sql = "INSERT INTO readings SET
19     personid='".$$_GET['inputName']."',
20     temperature='".$$_GET['send_temperature']."',
21     datetime='".date('Y-m-d H:i:s')."' ";
22
23
24 if ($conn->query($sql) === TRUE) {
25     echo "New record created successfully";
26 } else {
27     echo "Error: " . $sql . "<br>" . $conn->error;
28 }
29
30 $conn->close();
31
32 ?>
```