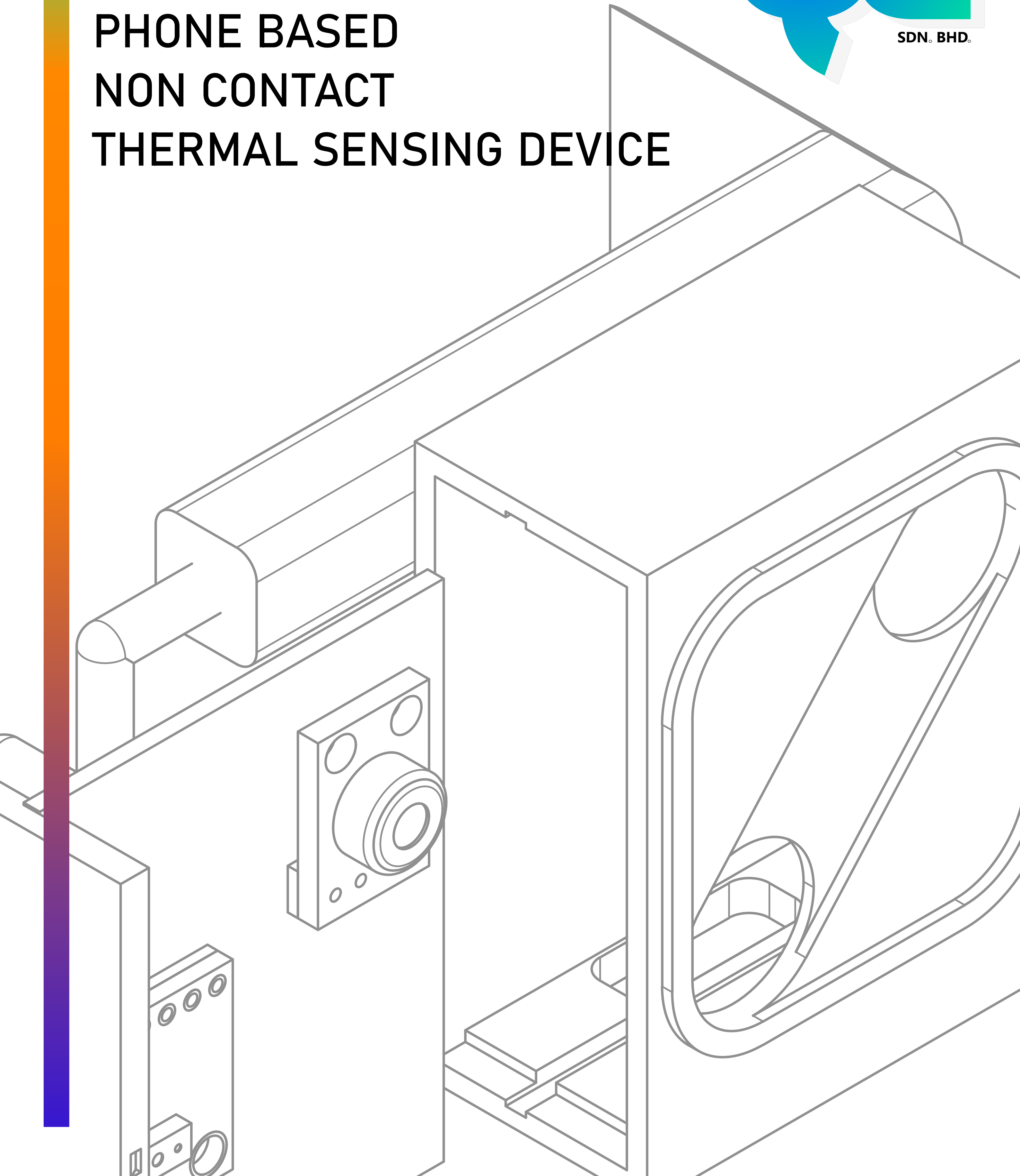
FINAL REPORT

14/08/2020

ID: 0002

TITLE:

PHONE BASED NON CONTACT THERMAL SENSING DEVICE



Contents

1. Introduction

1.1. Team Members

Project Engineer: Lim Yee Teng

Software Engineer: Ibrahim Izdhan

CAD Engineer: Choy Jun Juet

Test Engineer: Wong Wai Kiat

PCB Engineer: Low Wai Sing

1.2. Market Research / Existing Product

The need for thermal screening devices has increased in demand in the wake of the recent Covid-19 pandemic due to its ability to aid frontline workers and governments in determining early signs of infection at entry screening points [1]. The need for effective thermal monitoring solution has caused an untimely gap in the supply and demand of conventional devices to achieve this solution – either causing existing solutions to become unable to deliver or causing spike in prices [2].

One of the market similar smartphone based non-contact thermometers is the thermometer produced by 'Comper'. Its application is likely used in the medical field. Its measurement method is through infrared ray to measure the body temperature of people by pointing their forehead. It can be connected to the phone through Bluetooth to show the value of temperature on the phone. Its temperature range is between 32 ^{o}C (89.6 ^{o}F) and 43 ^{o}C (109.4 ^{o}F). The Comper Smart Infrared Thermometer is designed to read the body temperature as easy as possible. Besides, their Comper Health App can visualize the body temperature ranges and trends. The smart infrared thermometer can provide maximum precision with results that are accurate within 0.2 ^{o}C and it also can provide results in less than a second. Other than that, Comper Smart Infrared Thermometer also includes a convenient smart app with features like sharable readings and real time analytics, and it also supports an unlimited number of users. The thermometer can automatically record all the temperature and save them to the app for tracking purposes.

Figure 1 Similar Existing Product - Comper Thermal Screening Device [3]

1.3. Problem Statement

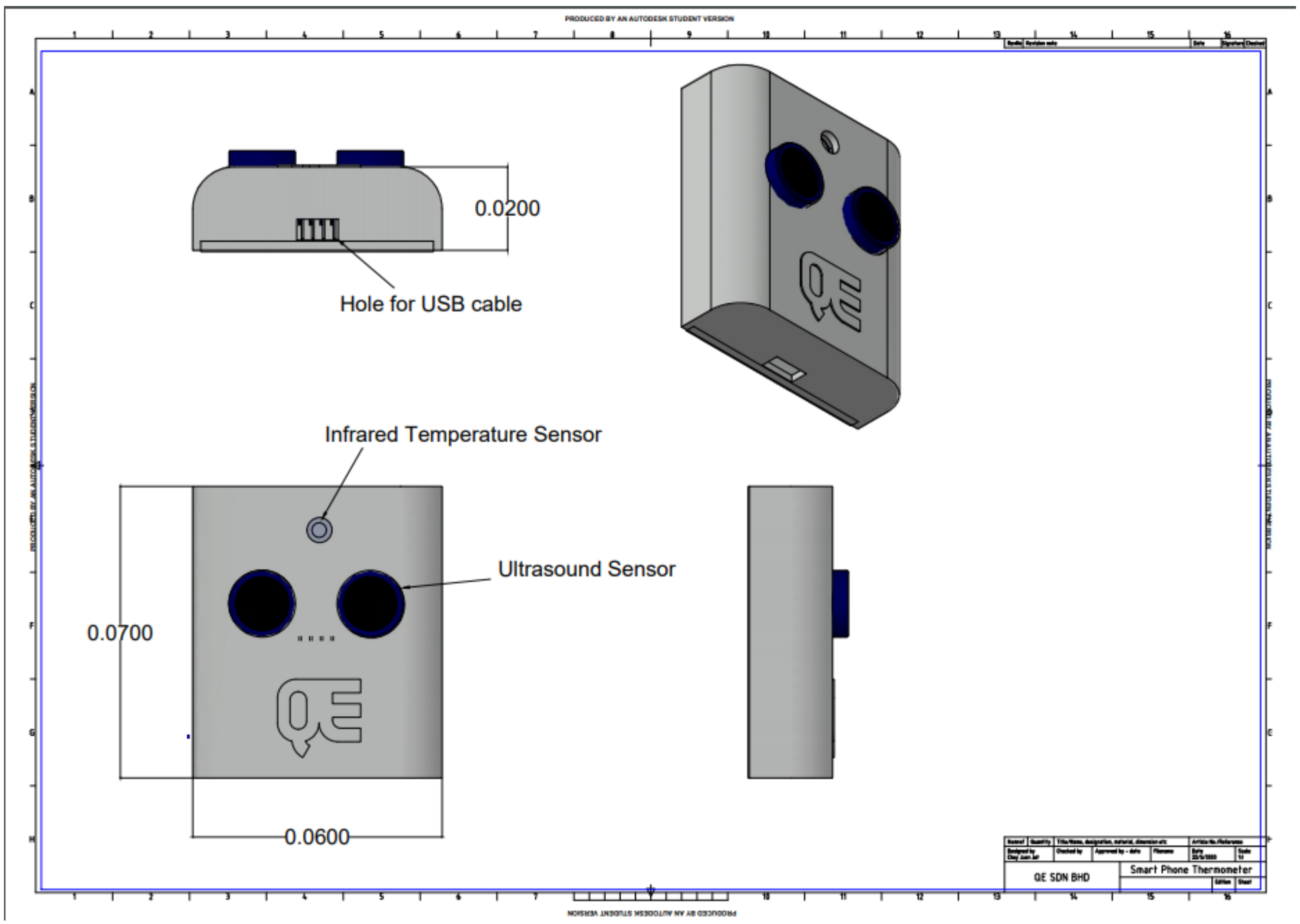
The increase in Covid-19 cases across the country has sparked a need for proper, easily accessible and cheap screening procedures to be developed in order to accommodate the rising demand for such devices in places of high traffic. Such a device should be able to screen for fevers, which is a common symptom of Covid-19, without implicating the persons using the screening device, via non-contact measurement of body temperature. The device should be easy to operate and should be powered with a power source that is readily available for most individuals – with clear indication of remaining power status and ability to recharge.

1.4. Objective

To develop a cheap Arduino Based non-contact thermometer that interfaces with the user’s phone to provide clear and precise (to within an acceptable margin of error) temperature of a person being screened. The device should be powered by the phone’s serial port as well as provide data to the phone via the port to be processed and displayed on the phone.

2. Methodology

2.1. Draft of Product in Proposal



The casing of the design will be attached to the back of the smart phone via a simple clamping mechanism found in most cheap phone holders – allowing for the placement of the thermal device to be agnostic to the dimensions of the phone. This draft outlook implements the Ultrasound sensor to detect proximity of the forehead from the sensor – the effectiveness of detection and option to switch to IR proximity will be tested by test engineer and the design will be further revised if necessary.

2.2. Changed Design due to Hardware Change (1)

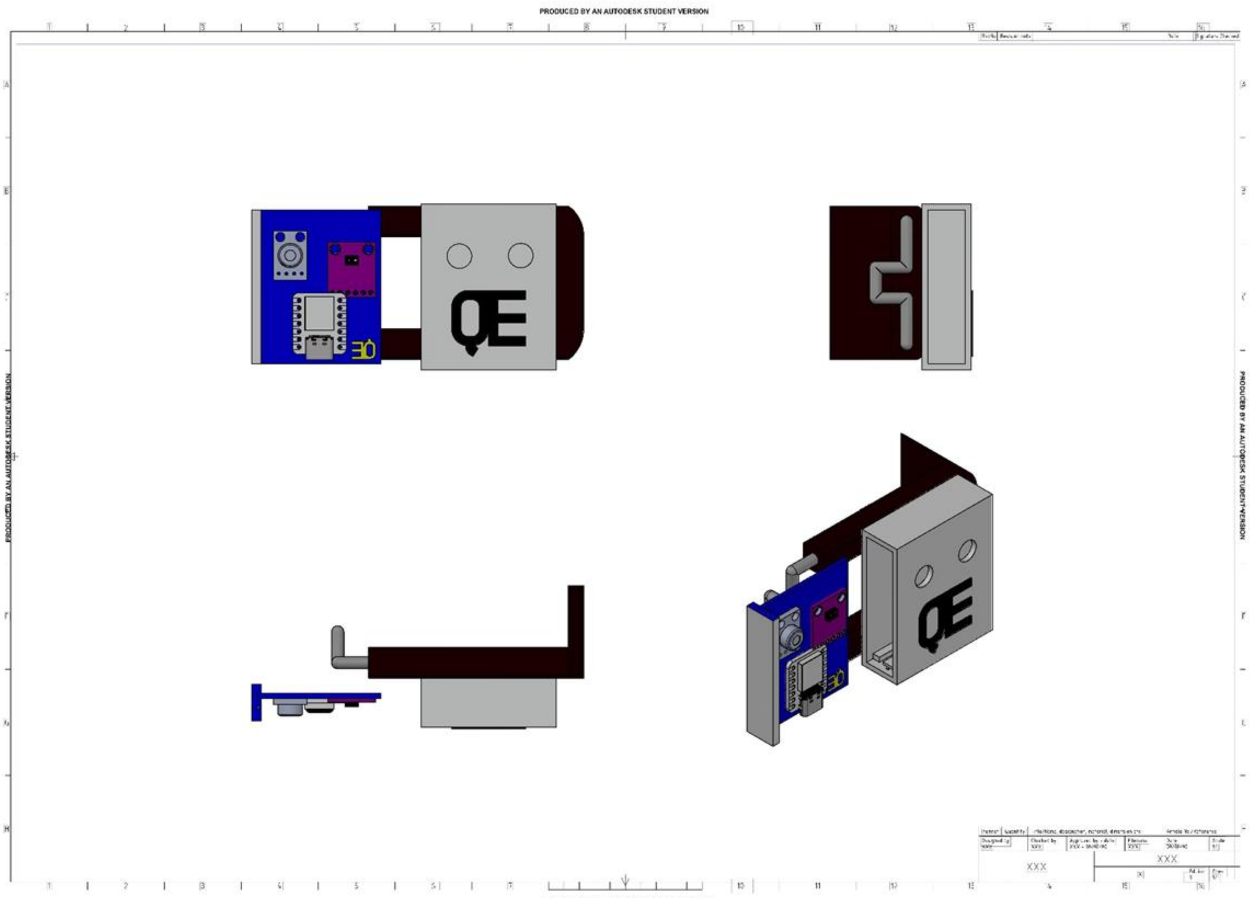


Figure 2 Initial Design of Device

The design is based on the PCB board we first drafted. The PCB board has 3 important components which is the Seeeduino Xiao, temperature sensor and the proximity sensor. There are 3 holes designed specifically for these components. The first one is a rectangular hole with round edges at the bottom of the casing which enables the USB cable connection with the Seeeduino. The second and third hole in front of the casing is for the temperature sensor and proximity sensor.

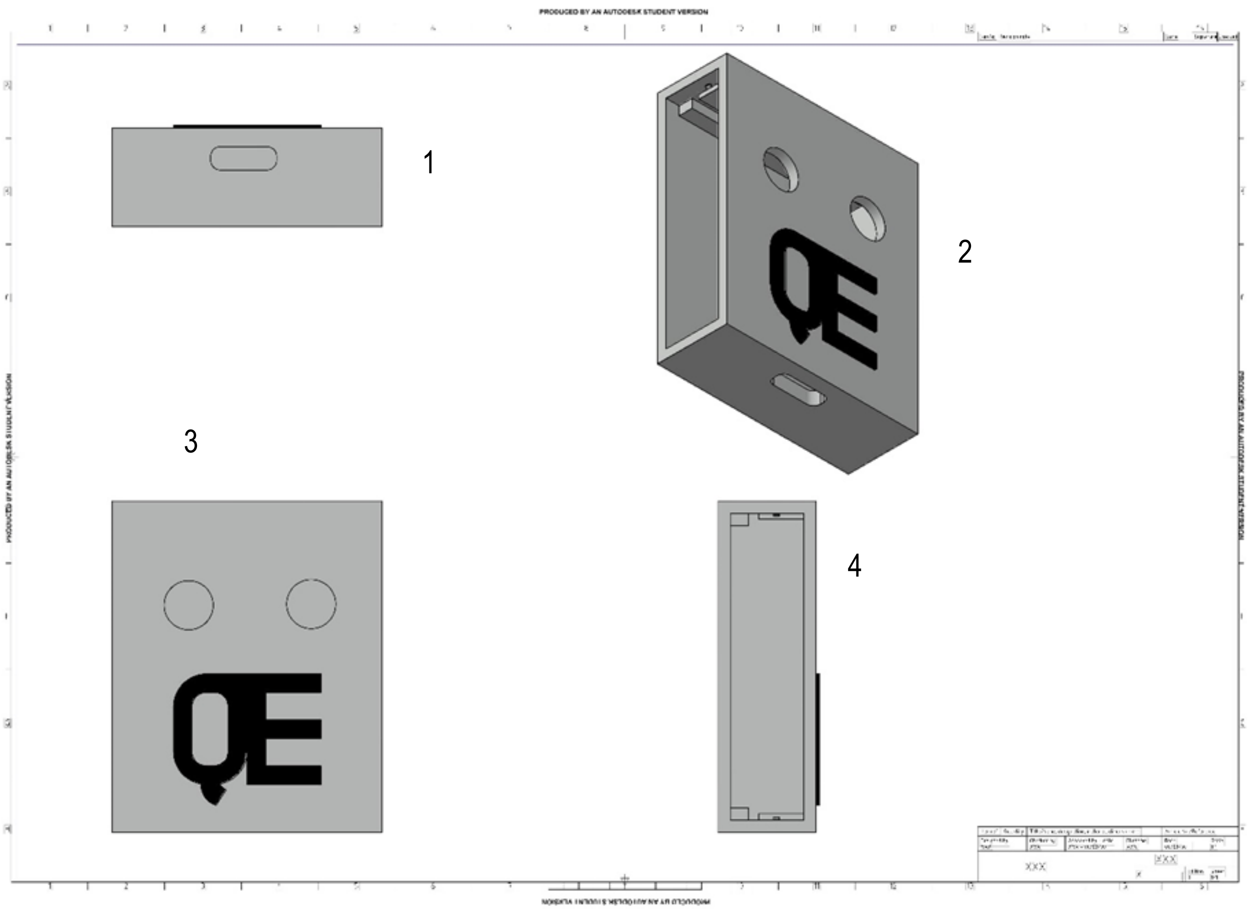


Figure 3 Initial Design without components nor attached support

The holes are needed for the temperature sensor and proximity sensor since the casing will get in the way of the sensors if the holes are not there. The holes are aligned to make the casing tidier. The logo of our company "QE" is added below the 2 holes in front of the casing to make it more aesthetic.

Features:

The hole at the bottom as shown in **1** is used to connect the USB cable.

The 2 holes in front of the casing as shown in **3** is for the temperature sensor and proximity sensor.

The side as shown in **2** and **4** have a place holder for the PCB board. The PCB will be attached to the cover at the side and then slide into the casing.

2.3. Final Casing Design after Design Restrictions

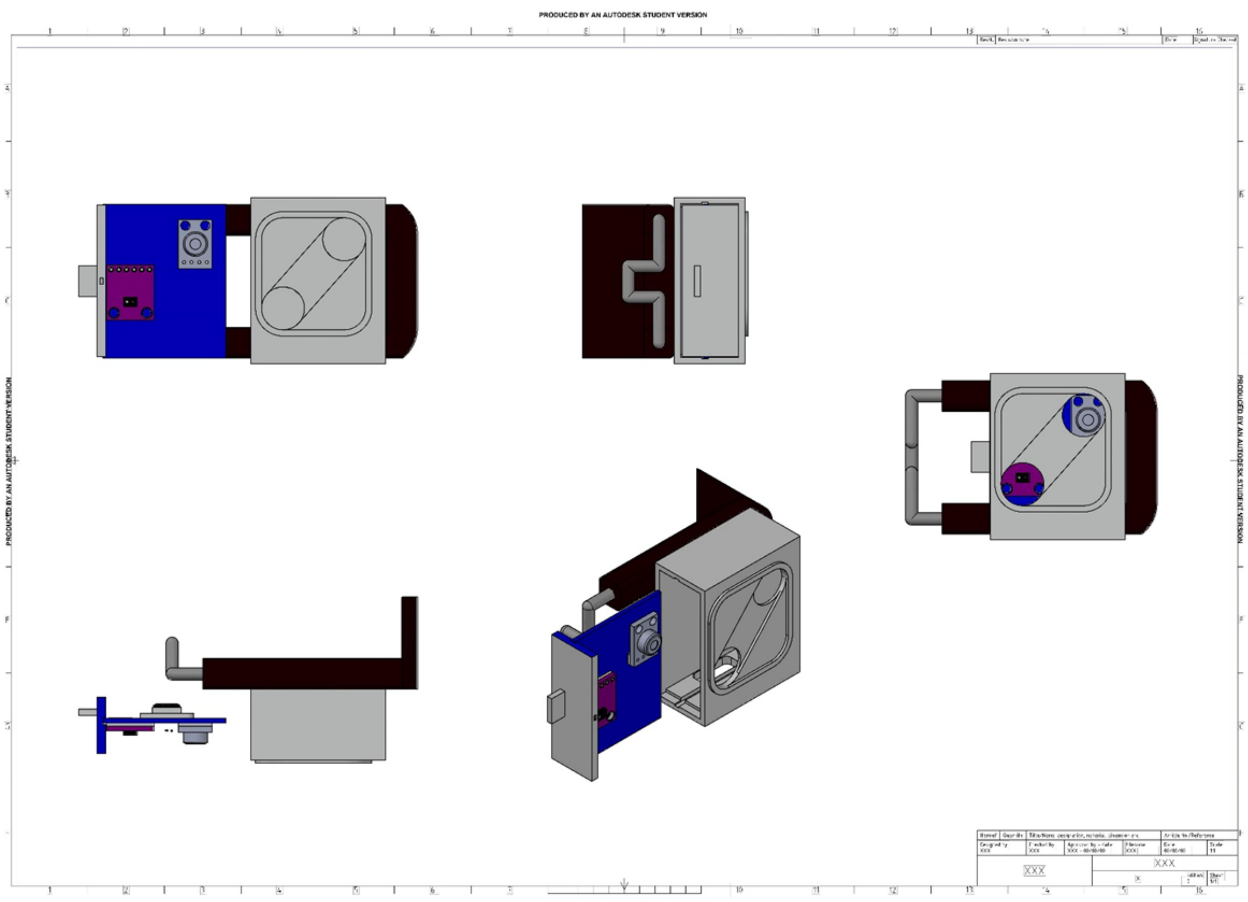


Figure 4 Final Casing with Support

When we solder the components onto the PCB, we had some problems with the placement of the components. To resolve the problem, the holes for the components are moved to their respective components to make the device work. The new design is based on the first design with the sliding PCB mechanism and holes for sensors. A locking mechanism is added because the previous design has a problem with the slide sliding off when in use. A small extension piece at the side casing is there because the side casing couldn't be open once the casing is closed. The small piece at the side making sure the casing can be opened after that. The round edge rectangular hole for the USB

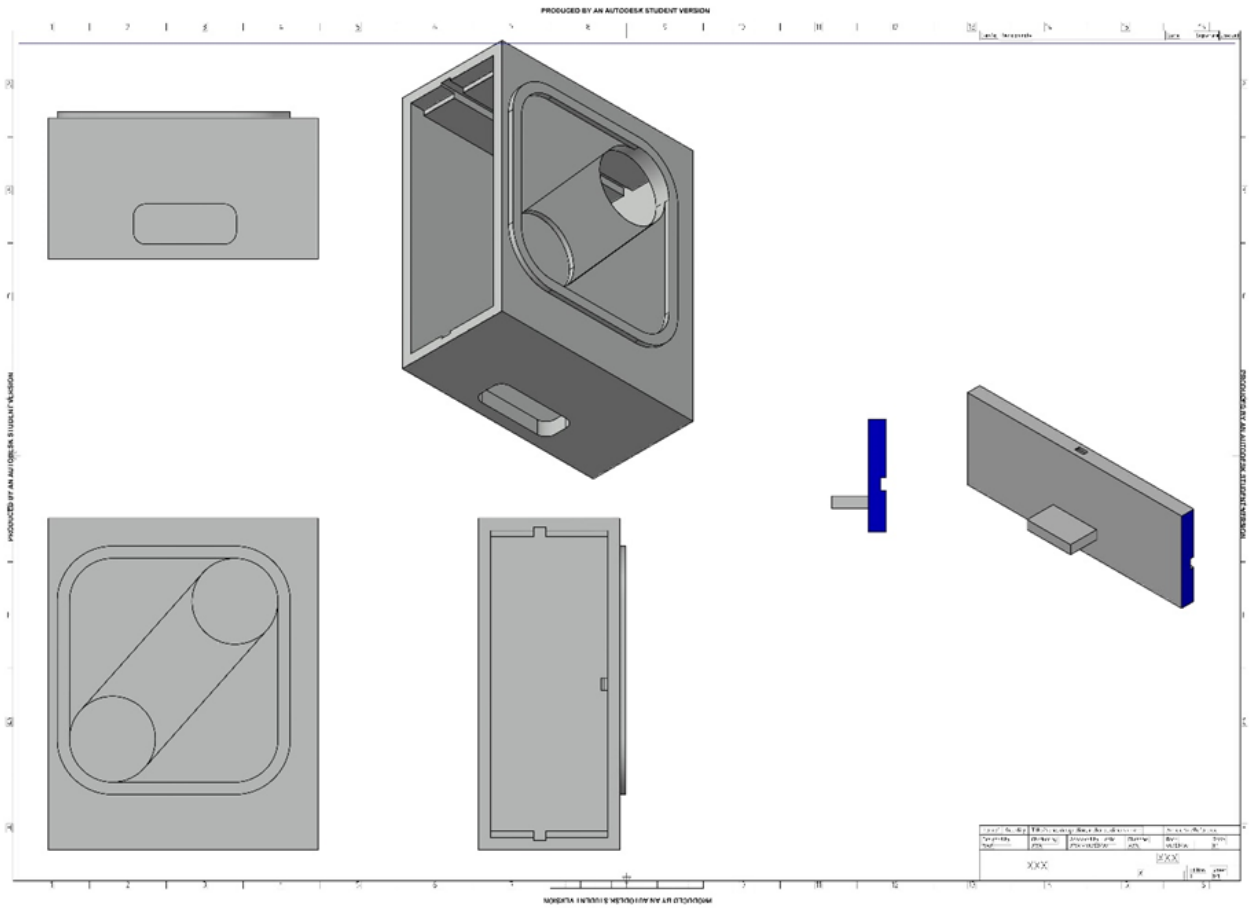


Figure 5 Final Casing Design without supports and PCB

USB didn't change much except for the position and the increased size. The holes for the sensors are also made bigger to make sure there is no interference from the casing. Since the holes are repositioned, the "QE" logo was removed as there is not enough space for it. To replace the logo design, a bridge like structure is made between the holes where it makes the holes look like its connected. There is also a square with round edges around the holes to make it look tidy rather than holes randomly placed on the casing.

Features different with second draft:

Hole placement changed to reflect the PCB.

Side Mounted wall thinner to allow easier insertion of PCB into case.

2.4. Proposed Hardware

Arduino Nano	Serve as the Microcontroller to receive data from the thermal sensor (via I2C) as well as the ultrasound sensor (via digital pins) and send the data through serial to the connected android device. The Arduino will route power supplied to it by the phone through its 5V (out) pin to supply power to the attached sensors as well. Any intermediate data processing to be done before transmission of data to the phone will also be done on the Arduino
MLX90614 Thermal Sensor (GY90614)	The thermal sensor (PCB module) used to detect temperature values. The libraries of the sensor are provided by the manufacturer and thus can be used to detect temperature at 2- 5 cm with minute manual tuning required in software.
HC-SR04 Ultrasound Sensor Module	The Ultra sound sensor will be used to ensure that the distance from the temperature sensor to the forehead of the individual is within the effective range of 2 – 5 cm. Processing of distance done within Arduino

2.5. Changed Hardware from Proposal Stage

Seeeduino Xiao	Replaces the Arduino Nano as it has a smaller from factor. It is still coded in Arduino thus poses no additional development requirements.
AP3216 Proximity Sensor (CJMCU - 3216)	Replaces the HC-SR04 as it is more compact and due to the sequential nature of both reading distance and taking temperature (by the temperature sensor), no interference is expected due to delay inserted in software.

2.6. Proposed Software

CAD DESIGN	The proposed software for CAD design of the casing of the non-contact thermometer is Auto CAD due to the simple nature of the casing design (no mechanically moving components) as well as prior experience with Auto CAD
PCB DESIGN	The proposed software for PCB Design is EasyEDA due to its simple UI and low learning curve. In addition, the said software supports placing order for custom made PCBs through vendors such as JLC PCB (easy output to Gerber file)
App Development (Android)	The proposed software for developing the phone interface to be used with the non-contact thermometer is Android Studio as it will allow the creation of a highly customizable and free (in terms of subscriptions to be paid for the use of said app or specific allowances required for the design of the app) android application that can be provided free of charge alongside the device. Android Studio allows for virtual debugging, though this will not be as useful for a serial-based application, the layout can be tested before being uploaded onto a real test device
Arduino Coding	The base Arduino IDE is the only available method and will be used to code the Arduino Nano that will be used in the non-contact thermometer

2.7. Changed Software from Proposal Stage

App Development	Production switched from JAVA to using the Flutter framework by google, which implements the use of the DART programming language.
3D Printing	Since it was decided to 3D print the casing for the thermal sensing device, Ultimaker Cura was used as it is a free and eeasy to use 3D printing/splicing solution

3. Circuit Design

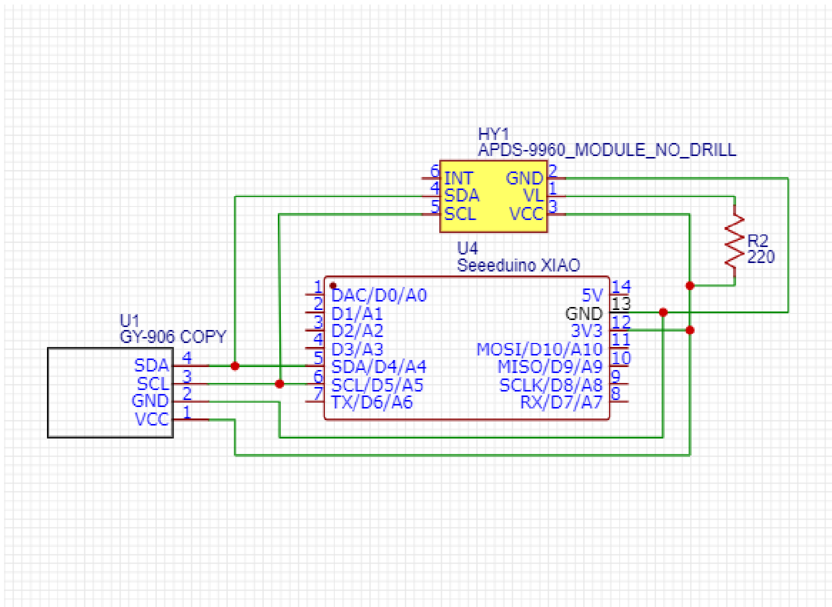


Figure 6 Circuit Schematic of the project

The operational voltage of both the GY-906 and the AP3216 Proximity Sensor is between 3.3 – 5 V, thus the lower option was preferred for safety of the components. Both of these sensors are designed to communicate using the I^2C protocol and as such have been connected in a manner that their SCL (Clock) and their SDA (Data) are interconnected with each other. A 220-ohm resistor is used to bias the IR blasting mechanism in the AP3216 Proximity Sensor. All grounds in this system are common (Seeeduino) and are interconnected together.

The circuit as seen in Figure 6 was designed in EasyEDA to streamline the PCB manufacturing process later on. The schematic connections were chosen from the EDA library and then compared to ensure that the models were the same. As noticeable in Figure 6, the model for AP3216 Proximity Sensor, is used interchangeably with the APDS9960 as the connection schematic is the same.

4. Breadboard Testing

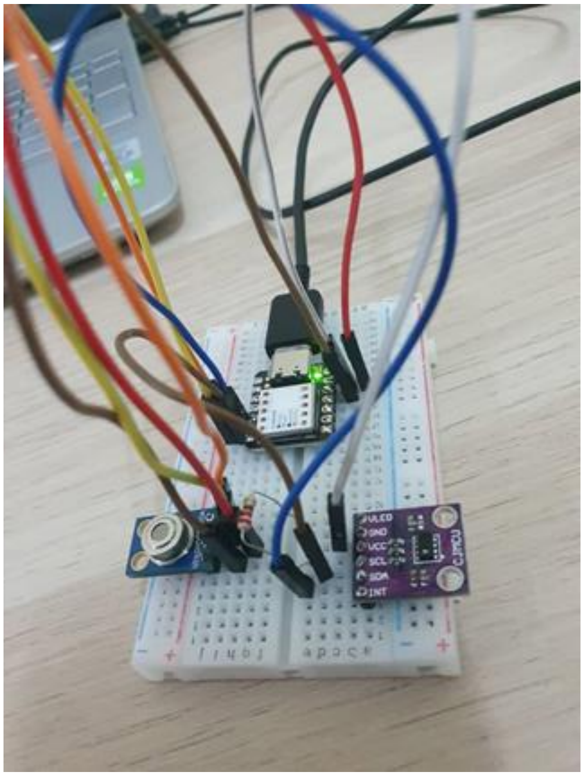


Figure 7 Breadboard placement of components

Before doing the breadboard testing, the MLX90614 thermal sensor and proximity sensor were soldered with the pins provided to ensure that they have good connection with the breadboard while testing on it – so that the signal passed using I2C is delivered properly.

Next, the code written by software engineer was transmitted to the Seeeduino XIAO for the test engineer to test the functionality of thermal sensor and proximity sensor individually using the standard test code provided by the manufacturer. Afterwards, the test engineer placed all the components on the breadboard based on the circuit designed by the PCB engineer as shown in Figure 7 and check whether the components were connected correctly to prevent short circuit. In our first testing on breadboard, we did not get the results that we wanted. Then, after further testing, the test engineer figures out that it is the problem of breadboard and replaced it with another new breadboard. After replacing the breadboard, we observe the results that we wanted. This means that all the components are in good condition and the circuit connection is correct. Afterwards, the tested components were handed to the software engineer for further calibration and tuning.

5. PCB Design

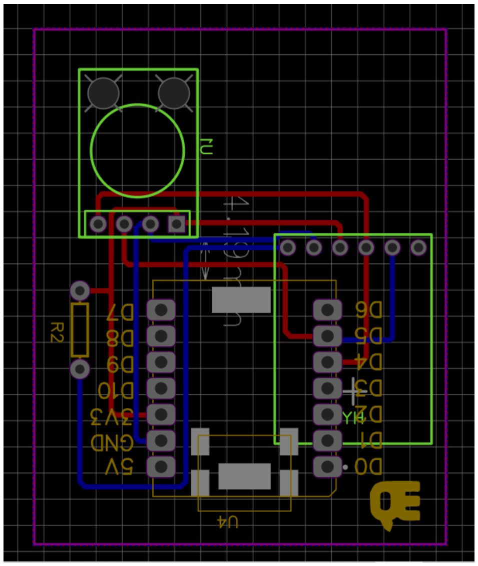


Figure 8 Schematic of PCB

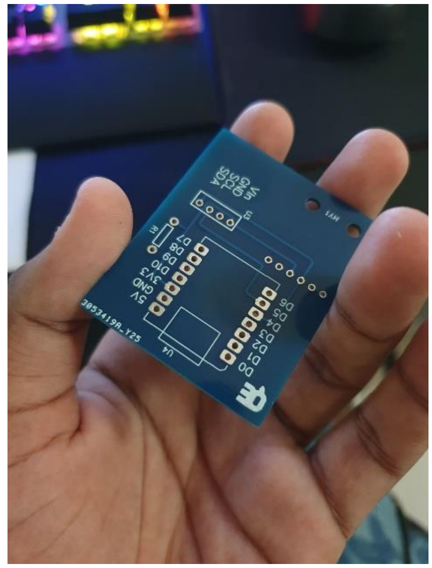


Figure 9 Final PCB

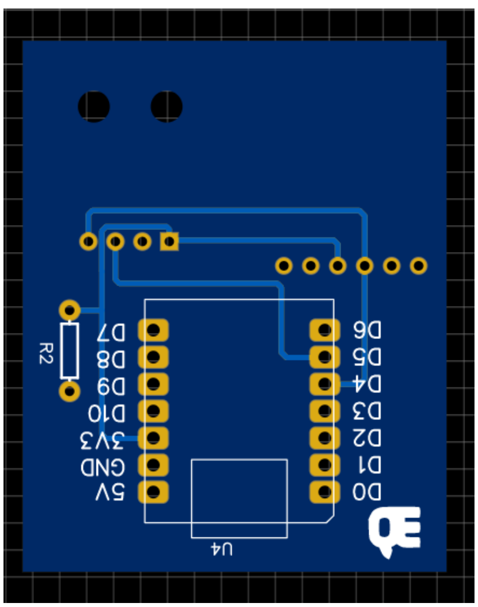


Figure 10 Top PCB VIEW 3D

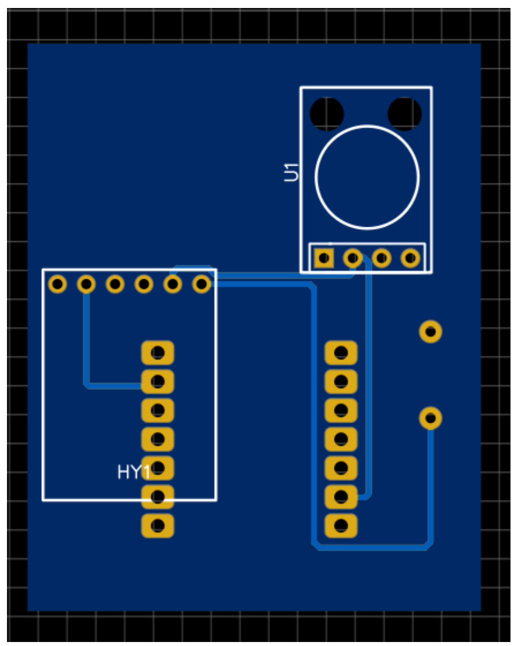


Figure 11 Bottom PCB VIEW 3D

PROCEDURE:

1. After designing the circuit, converted it to PCB and the footprint of the electronic parts were checked
2. Set the PCB board outline as diameter equal to (50mm*40mm).
3. Dragged all the electronic parts at the default position inside the PCB board.
4. Changed the proximity sensor and infrared thermometer sensor from top silk layer to bottom silk layer.
5. Set the width of trace as 20 mil at the setting and start connecting the components. Changed to trace to bottom layer when the trace overlaps each other.
6. Selected image at ‘PCB tools’ to insert our company logo at top silk layer of our PCB design.
7. After finishing and saving the PCB design, click the ‘Design Manager’ to check for any errors
8. Gerber file for our PCB design was then generated once everything was set up correctly and ordered our PCB through JLCPCB.

When converting the circuit design to PCB, ensure all the components have the correct footprint with the real components that are used. For the connection of traces in the PCB, avoid using 90-degree trace angles, opt for 45 degrees as the edge of the 90 degrees tracks will be etched smaller than the trace width set by designer or those tracks will not be fully etched and might lead to a short circuit. For the ideal PCB design, the turning point of each track should be curved but due to limitation, we used the smaller 45 degrees gaps. Next, enough space should be left between all the tracks and the components so as to avoid overlapping as the PCB manufacturing process might not be entirely precise. As we know, the higher the resistance, the lower the power dissipated as the resistance is inversely proportional with current, and current powered by two is directly proportional to power consumption. Last but not least, PCB design should always be printed on a blank paper to double check whether the footprint is exactly similar to the parts that are being used to ensure that the specific component matches with the footprint in EasyEDA.

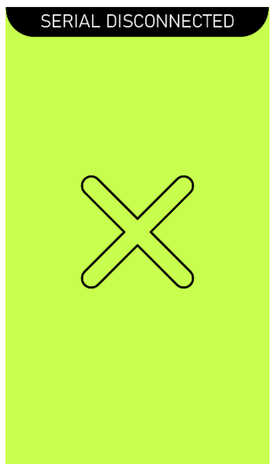
6. Software Development

6.1. Android App Draft Plan

Illustrated below is the draft plan of the layouts (views[1]) that comprise the accompanying android application to be used with the thermal sensing device. Design is subject to change based on further debugging or usage testing.

View when serial device is properly transmitting data and temperature is within accepted range

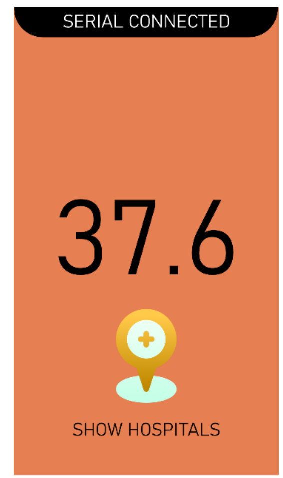
View when serial device is disconnected or there is an issue with data received



View when the data received indicates that the forehead is too far from the optimal range

View when the temperature is close to the upper threshold for a fever

View when fever detected – button to view hospitals on maps (google extension)



[1] Android applications are comprised of "views" that contain fundamental elements such as images, text and other placeholders – whose attributes can be updated via functions

6.2. Android App Finalized Widgets

The draft design of our application was designed with the software intended to be coded in JAVA. However, during production the team decided that Flutter would be a more interactive solution to designing the app hence the design was slightly altered while maintaining key design features from the previous design and implementing additional features which will be discussed in detail in the following sections.

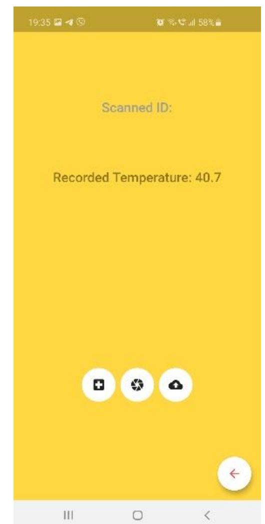
Initial Startup Screen of Application, Seeeduino not detected thus showing Disconnected

Status shows connected but the request to get temperature reading not set

Screen that shows temperature received if it is within a non-fever region. Color changes.

When fever is detected a button appears with more options as seen in the bottom right.

More options screen once fever detected, to scan QR and upload details to server, call Hospital



6.3. Main Screens in Detail

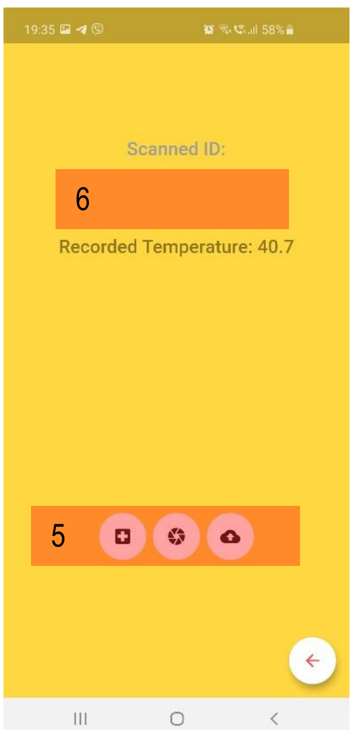
1 – The Serial Connection Status between the Seeeduino and the Phone. It will update the message when the Seeeduino is connected/disconnected and will also update if the Android program detects if there are any problems with the connection, such that a device is connected but data is unable to send properly.

2 – The entire region serves as a button which initiates the requestTemperature() function, which will prompt the Seeeduino to send data over to the phone.

3 – If the user clicks this region they will be taken to a widget(page in Flutter) where the information of the compay is displayed and they may contact us via Instagram.

These 3 Regions are **present in all screens** (besides the QR scanning page and company info page)– and serve the same function.

4 – Once the average temperature taken over 3 successful consecutive readings average out to be greater than the standard fever temperature (37.5° C), the region is made active. Clicking on the button in this region will direct the users to a widget (screen below) where they will be able to deal with such cases more effectively.



5 – This region consists of 3 buttons with 3 distinct functions. From the left, the button opens up any mapping software installed on your device such as google maps and automatically searches for hospitals nearby. The second button, when pressed opens a scanning window where the user can scan the ID of the affected Employee. This data is automatically displayed on (6) so that they may double check the reading. The last button in this row automatically updates the data of the ID, as well as the temperature recorded to a Google Sheets database that the company HR may keep track of – no manual entry required.

6.4. Installation of IDEs

ARDUINO IDE

The Arduino IDE can be installed through the windows store or following instructions located at the official Arduino Website https://www.arduino.cc/en/main/software. The Seeeduino board used should be enabled in the IDE by following the instructions found at the official Seeeduino website https://wiki.seeedstudio.com/Seeed_Arduino_Boards/

ANDROID STUDIO (FLUTTER)

Android Studio can be installed using the link https://developer.android.com/studio/install. Once the IDE has been installed, enable Flutter development on Android Studio using the instructions on the Flutter website https://flutter.dev/docs/get-started/editor.

In your Flutter project, inside android > app > build.gradle set the minSdkVersion 21 so that the QR code scanning features work.

6.5. Arduino Dependencies

The project in question requires the Arduino to take in data from 2 sensor modules - a thermal sensing module and an ALS distance measuring module. Code would collect the data and then process it before passing data through connected serial to an accompanying Android device.

AP3216 Proximity Sensor (CJMCU - 3216)

The proximity sensor used in the project is a CJMCU IR and Light-based proximity sensor. The sensor was programmed using the library by **Wollewad,** which can be accessed at https://github.com/wollewald/AP3216_WE , and tweaked for single pulse detection mode of proximity following the instructions on his repo. The values received for proximity are compared in a conditional that was calibrated for our specific sensor and if it was within threshold values (**distance**

of roughly 2 - 4 cm) from forehead the reading from the thermal sensor is triggered - else the Arduino returns **0** in the serial indicating that the forehead is not at an optimal distance.

MLX90614 Thermal Sensor (GY90614)

Once the thermal sensor is triggered, the **getObjectTemperatureC** method is invoked for the mlx object and it returns the temperature of the object. if it is between a validated range the Arduino prints the temperature to 1 dp to the Serial - else 0 is returned to indicate erroneous range for a human. The code uses the **Adafruit Library for the Module** https://github.com/adafruit/Adafruit-MLX90614-Library to initialize the object and invoke the methods.

6.6. Flutter Dependencies

The dependencies to be placed in the pubspec.yaml file in your Flutter project are bolded below with a short description of what each package does.

usb_serial: ^0.2.4

The package is used to establish serial connection with any connected devices and the android phone.

Documentation: https://pub.dev/packages/usb_serial

maps_launcher : ^1.2.0

The package provides a quick method to launch any maps software installed on your android phone with a specific search query.

Documentation: https://pub.dev/packages/maps_launcher

path_provider : ^1.6.11

This package provides commonly used locations on the filesystem. Supports iOS, Android, Linux and MacOS. Not all methods are supported on all platforms.

Documentation: https://pub.dev/packages/path_provider

audioplayers: ^0.15.1

This package is used in conjunction with path_provider to play audio files to alert the fever status before temperature is read out.

Documentation: https://pub.dev/packages/audioplayers

flutter_tts: ^1.2.3

This package is used to convert text (temperature value) into speech, which will be read out in your program.

Documentation: https://pub.dev/packages/flutter_tts

flutter_barcode_scanner: ^1.0.1

This package is to scan and return QR codes data before sending it to the GOOGLE SHEETS DATABASE using http.

Documentation: https://pub.dev/packages/flutter_barcode_scanner

http:

This package facilitates the GET request to store the data of individuals with a fever into a GOOGLE SHEETS, which is deployed as a Web App.

Documentation: https://pub.dev/packages/http

6.7. Flow Chart Seeeduino (Arduino)

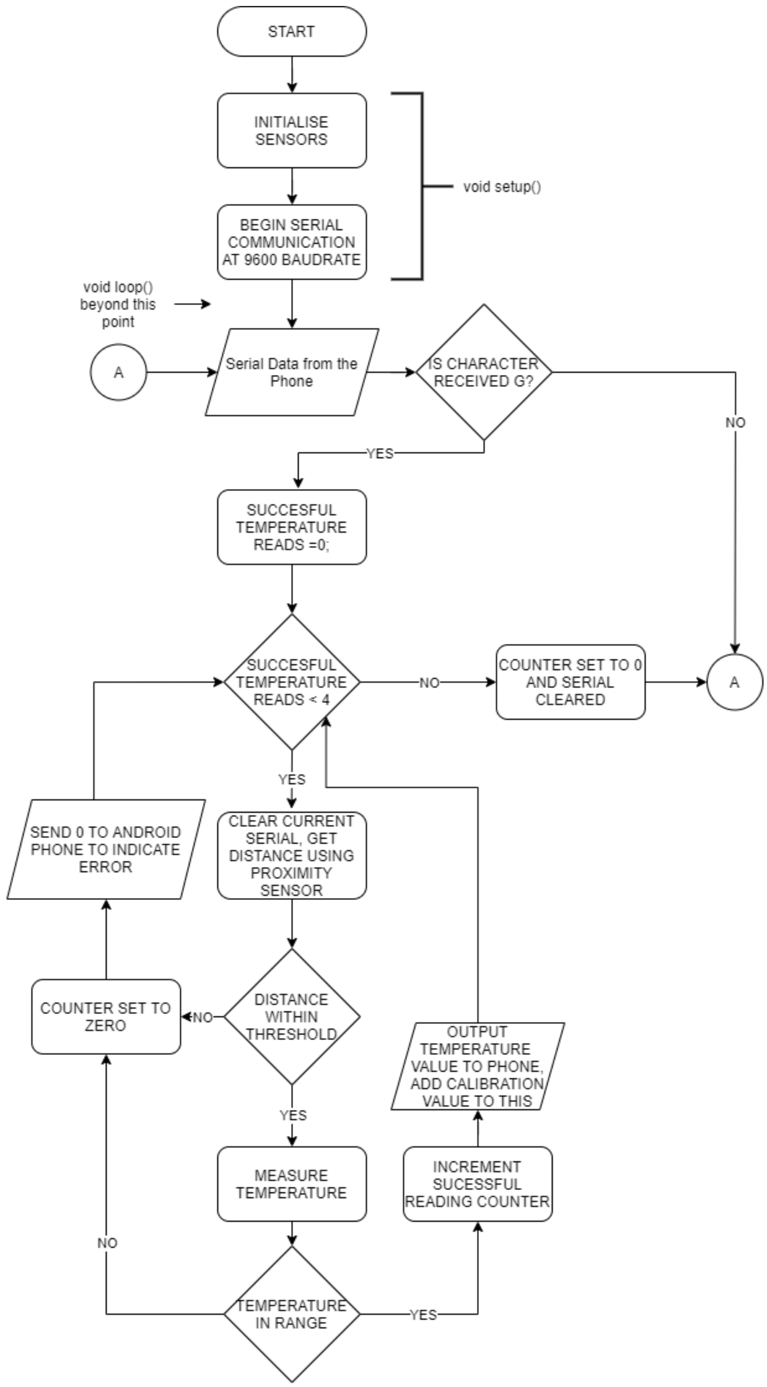


Figure 12 Flow Chart of Arduino Code

6.8. Flow Chart Android App

Some actions are always running in the background hence it is indicated with a *.

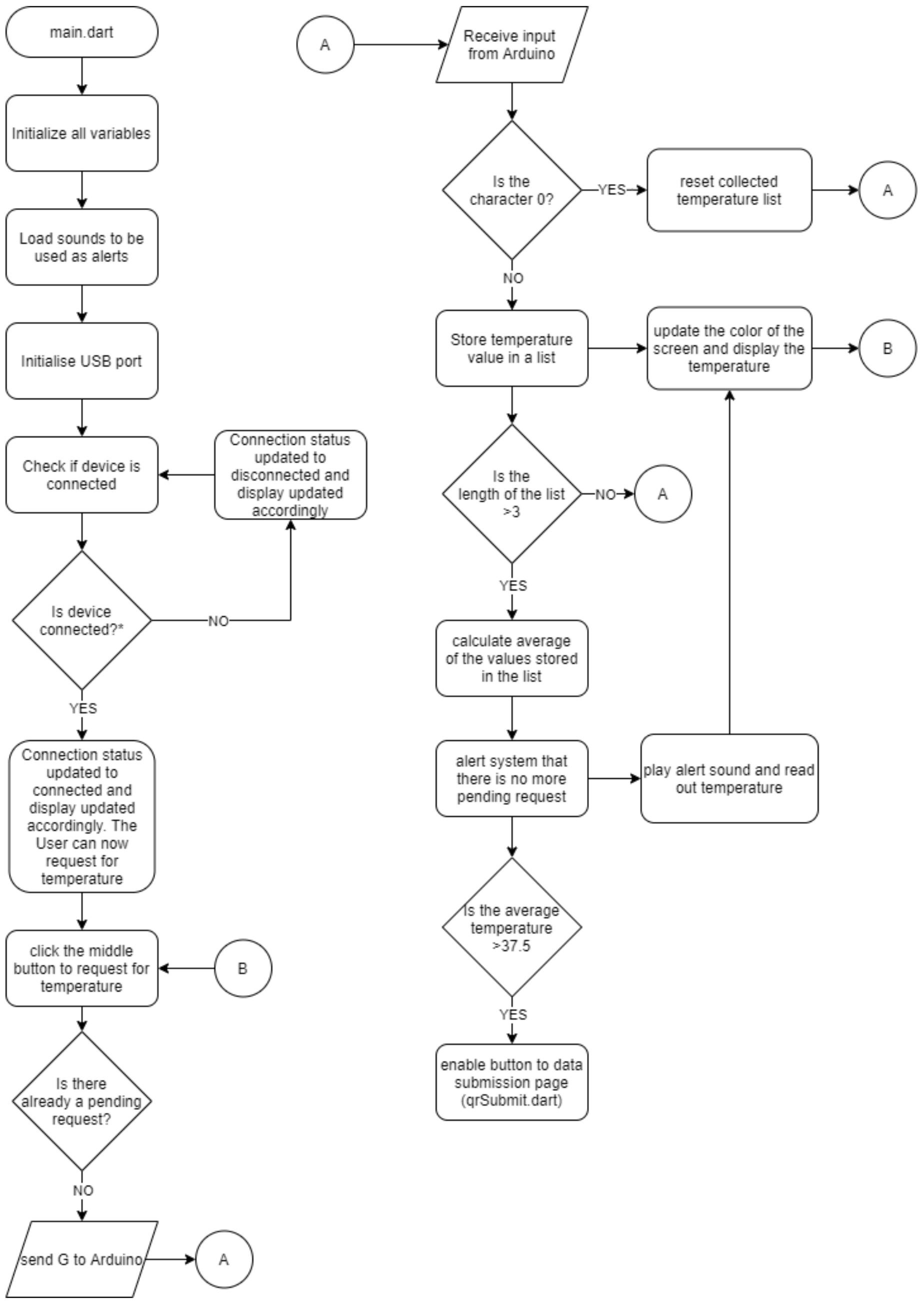


Figure 13 Flow chart of home screen

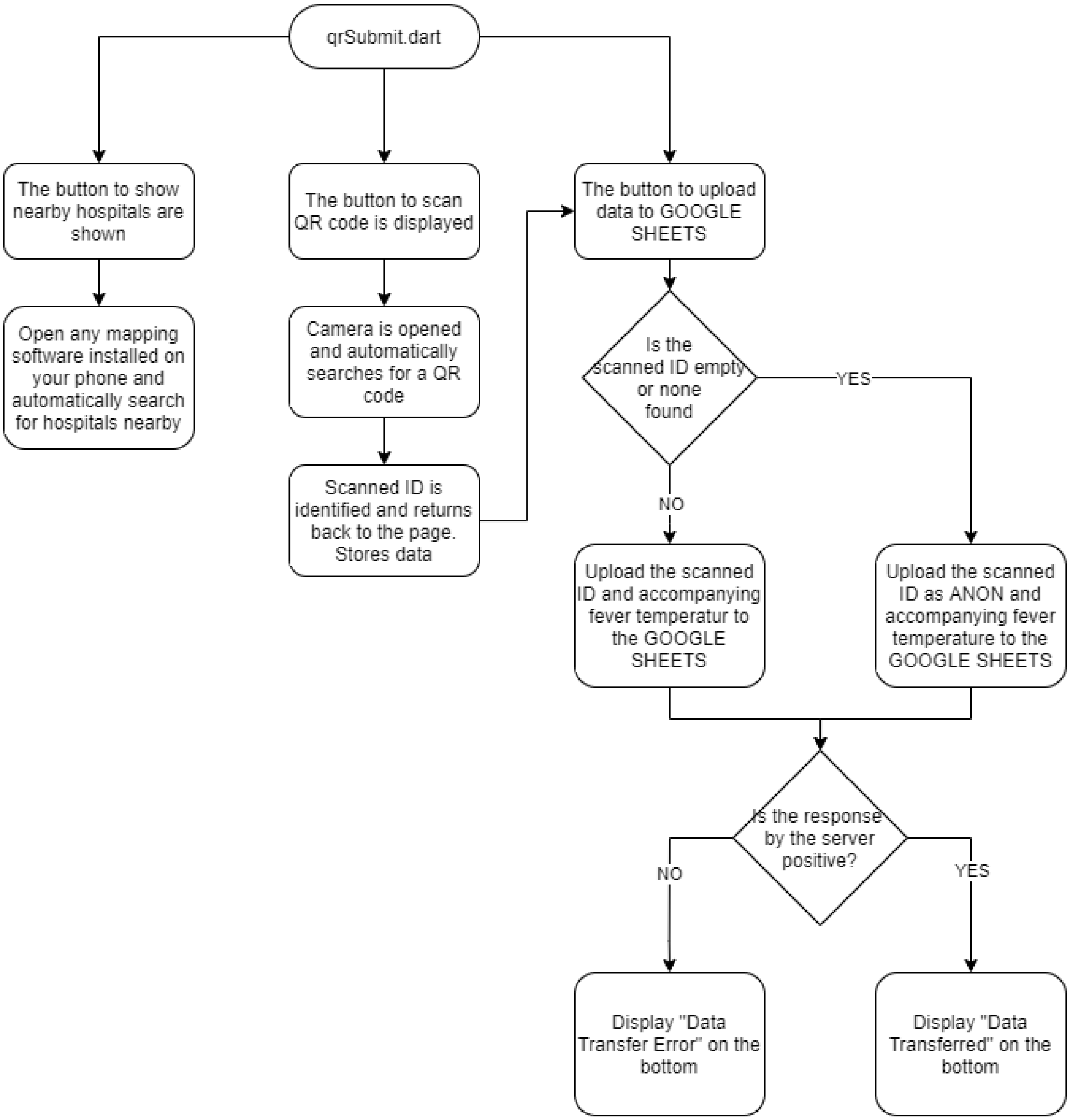


Figure 14 Flow Chart of Fever response page

6.9. Explanation of Key Functions (Arduino)

getProximityArduino() This function when called first sets the proximity sensor to a single pulse mode. Then after a delay of roughly 0.6 seconds, it measures the distance using the proximity sensor. If the retrieved distance value is between a range of 1000 of 40 (which is roughly between 2-4 cm), the function goes to **getTemperature().** If the distance is greater the counter of successful readings and sends 0 to the Android phone as mentioned in Figure 12.

getTemperature() Read the temperature and store it. Checks if temperature is between 10 and 50 degree Celsius and if its is out of this range, resets the counter of successful readings and sends 0 to the Android phone as mentioned in Figure 12. If it is within the accepted range the temperature that was recorded to the Android Phone.

6.10. Explanation of Key Functions (Android)

main.dart – HomeScreen

_speak() This function makes your code read out the given text. Which in our case, includes the temperature displayed on the Screen (Final Average Temperature).

playSoundFever() and playSoundNormal() These Functions play a warning sound (playSoundFever()) when the temperature of the user averages out to be more than 37.5, or an okay sound effect (playSoundNormal()) when the temperature is lower than 37.5. These functions are triggered whenever the final average temperature is calculated.

_loadSounds()	Prepare the sounds that are used in the above portion.
_connectTo(device)	This function connects to the device passed into it and sets the app so that it automatically updates the status of the connection and once the data is received from the Arduino, it automatically updates the display in the middle of the app.
_getPorts()	This function runs when a device is connected into the Android phone, and checks whether it can establish a communication to the device using the above function.
requestTemperature()	This function sends a letter G to the Arduino triggering the Arduino to start its reading process. This function is programmed in a way so that it does not allow multiple requests – one request sent to the Arduino at a time.
updateColorsAndOpacity()	This function is run each time the phone receives a temperature reading from the Arduino. It tries and takes 3 successful consecutive readings from the Arduino and calculates the average. If this calculation is interrupted by the 0 message from the Arduino, the phone displays FAR to indicate that the value is false and repeats the average procedure from scratch again. Once the average temperature is calculated, appropriate sounds are played and the color of the background changes. If a fever is detected, it allows the user to navigate to another page (by showing a button) that would allow them to store the temperature in a database, or search for nearby medical assistance.

qrSubmit.dart – The screen that pops Up after you click the button to log temperature when fever is detected

scanEmployeeID()	This function opens the camera and allows you to scan QR codes. Returns the QR code value and is displayed on the screen.
submitData()	This function when invoked submits the data to the GOOGLE SHEETS web app that will be created upon the request of the company. For now it sends the scanned ID and the temperature that was recorded to the web app in the URL inside the function.
_showSnackbar(String message)	This function shows a little notification on the bottom of the app with whatever message is passed into it. It is used inside submitData() to show the status of the data upload.

7. Source Code and All Accompanying files

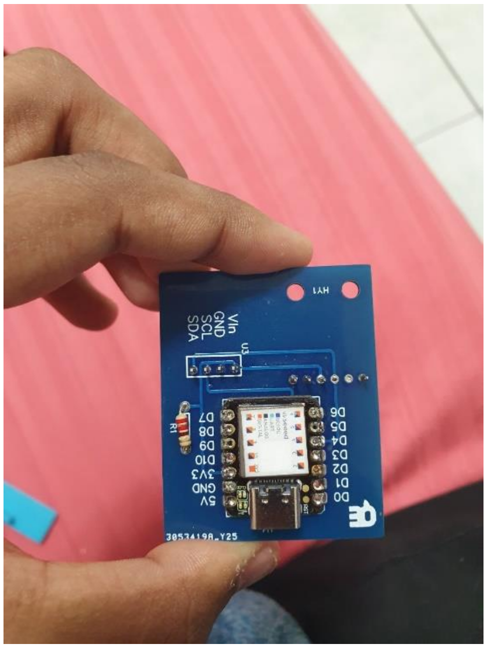
The source code of the project can be found at our software engineer's GitHub account and is distributed under the MIT License of fair usage. https://github.com/Prof-Iz/EMI-CODE-Thermal-Sensing-Android-peripheral. This source code further explains in detail of how the code works through comments. The GitHub repository also contains the STL files of the CAD model for 3d printing as well as the Gerber files of the PCB. If anyone wished to reproduce our results they may do so using the resources available at this repository.

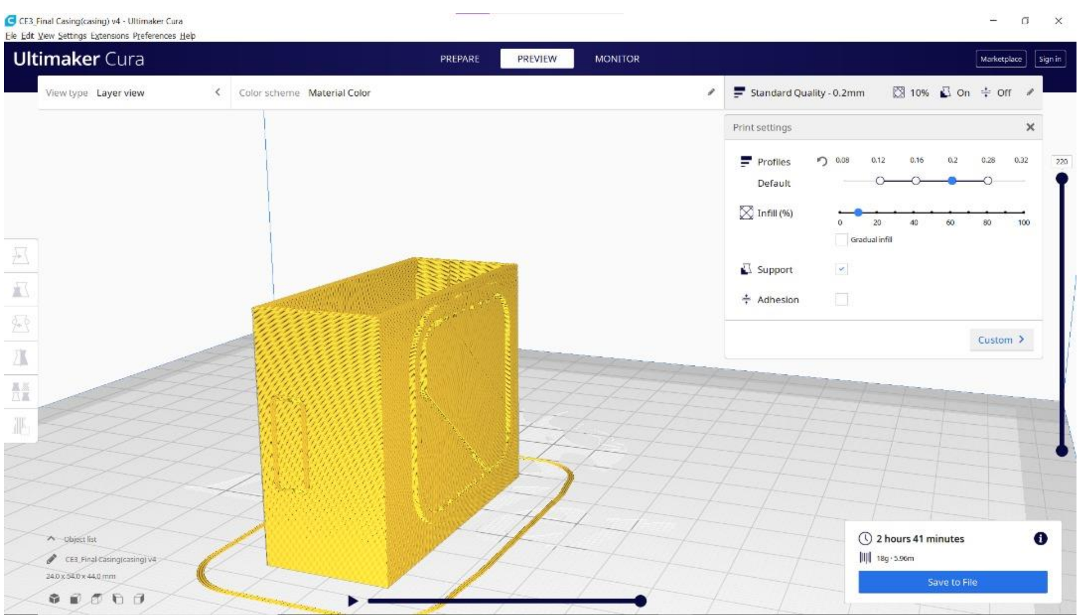
8. Warning

Due to the nature of some dependencies having issues in Flutter, the app was built and is distributed in Debug mode. It offers the same functionality as the release version of the app with the exception of the file size being larger, though this does not pose much of a challenge as the max size is still approximately 50MB.

Since the components need to be soldered using a soldering iron onto the PCB, care must be taken to ensure that the thermal sensor is not exposed to high heat during the soldering process.

9. Finished casing and Product





Slicing the Casing in a 3D Slicing Program to prepare for printing

Figure 15 Final Product

10.Costing

10.1. Projected Costing

Item	Approx. Unit Price		Quantity	Subtotal
MLX9061 GY-906 Module	MYR	100.00[2]	1	MYR 100.00
Arduino Uno	MYR	14.00	1	MYR 14.00
Ultrasound Sensor Module HC-SR04	MYR	5.00	1	MYR 5.00
Casing	MYR	10.00	1	MYR 10.00
PCB Manufacturing	MYR	20.00[3]	5[4]	MYR 20.00
			Total	MYR 149.00

10.2. Actual Costing

Item	Approx. Unit Price		Quantity	Subtotal
MLX9061 GY-906 Module	MYR	132.00	1	MYR 132.00
Seeeduino Xiao	MYR	34.90	1	MYR 34.90
AP3216 Proximity Sensor	MYR	12.00	1	MYR 12.00
Casing (3D print)[5]	MYR	0.90	1	MYR 0.90
PCB Manufacturing + Shipping	MYR	28.37	5	MYR 28.37
			Total	MYR 208.17

The high price in making of the functional prototype is mainly attributed to the inflated price of the MLX9061 GY-906 Module due to shortages in supply and increased demand due to the onset of COVID -19. This sensor tends to retail as low as MYR 20 s thus we anticipate that with the increase of supply and decrease

[2] Sensor price currently high due to increased demand during Covid-19 Peak season
[3] Rough conversion from USD to MYR and approximation of shipping fees for 5 PCB
[4] PCB printing service provides 5 PCBs as standard package – only one used per thermal device
[5] Assuming 1kg reel at RM50

in demand stabilizes the price back to its pre COVID-19 levels, the price of the product will become more affordable.

11. Special Thanks

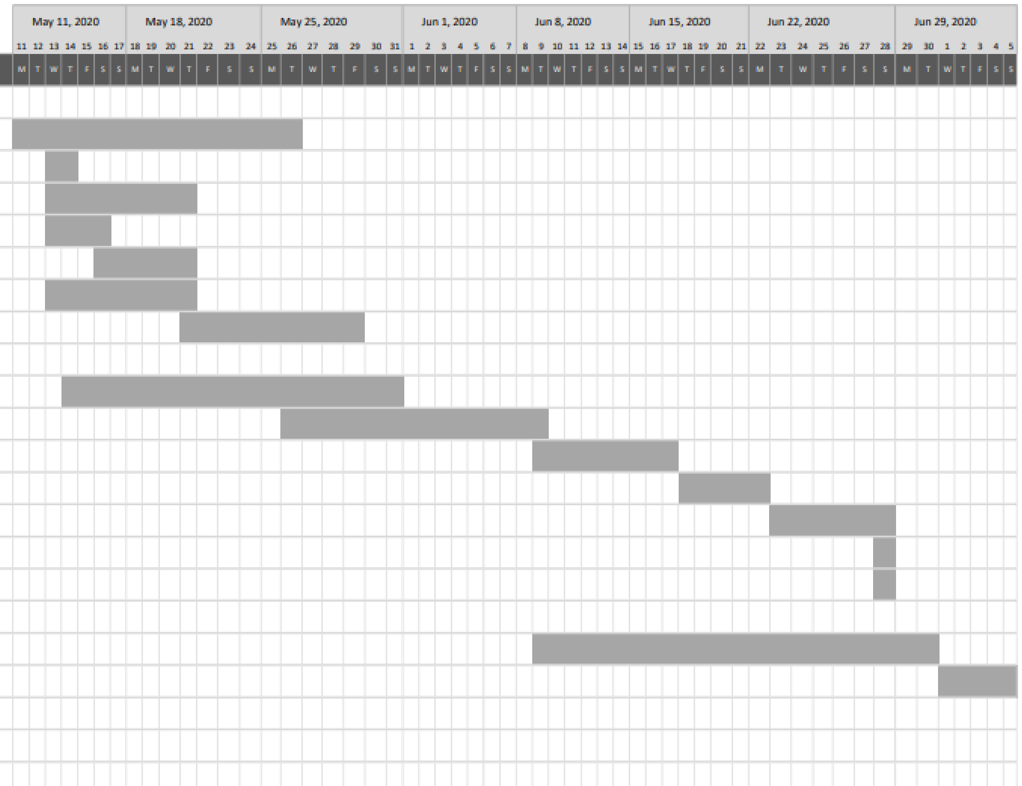
TMOM Sdn Bhd and QE Sdn Bhd worked closely and shared development ideas regarding the software thus we would like to express our gratitude to them.

12.Project Schedule

Project Start: Mon, 5/11/2020

Display Week: 1

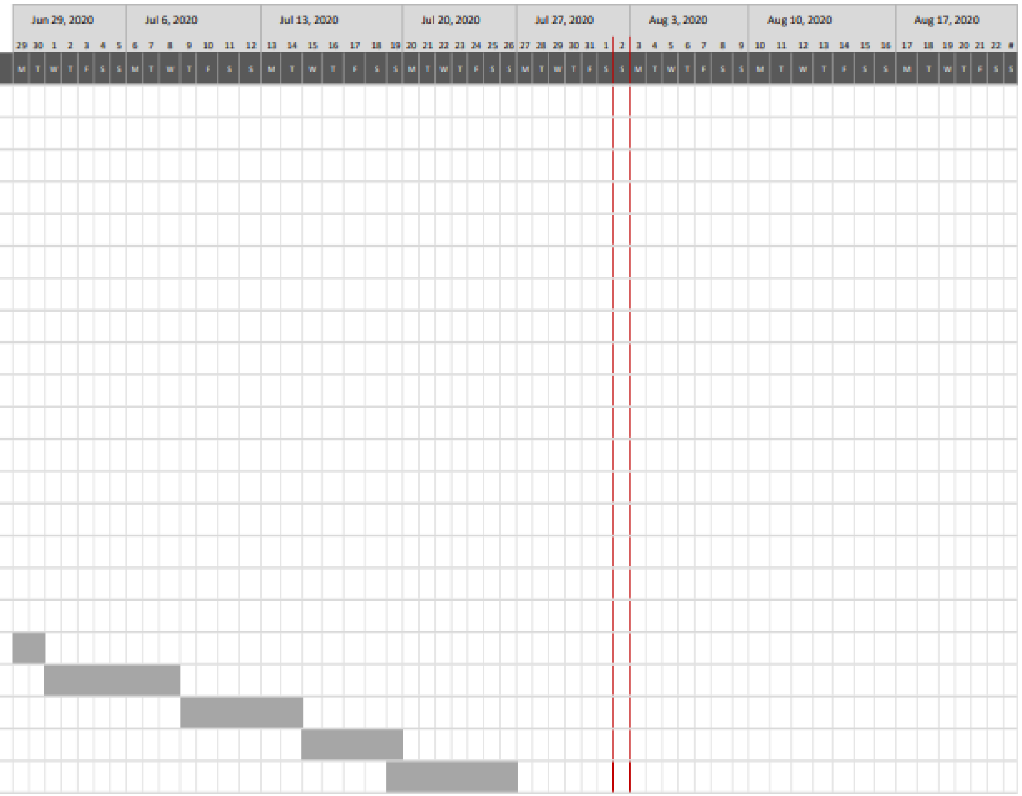
TASK	ASSIGNED TO	PROGRESS	START	END
Preparation Work				
Waiting for components' arrival	WaiSing	100%	5/11/20	5/26/20
Measurement of some components	Ibrahim Izdhan	100%	5/13/20	5/14/20
Research on existing product	All members	100%	5/13/20	5/21/20
First draft of product	All members	100%	5/13/20	5/16/20
Autocad drawing of product	Juen Jet	100%	5/16/20	5/21/20
First draft of proposal without marketing resear	All Members	100%	5/13/20	5/21/20
Finalise of proposal	All Members	100%	5/21/20	5/29/20
Project manufacturing				
Research on suitable android apps	Ibrahim Izdhan	100%	5/14/20	5/31/20
Working on product	All members	100%	5/26/20	6/9/20
Testing/ Amendment of the product	All members	100%	6/9/20	6/17/20
Software development	Ibrahim Izdhan	100%	6/18/20	6/22/20
Product Casing	All members	100%	6/23/20	6/28/20
Hardware final look	All members	100%	6/28/20	6/28/20
Software final look	Ibrahim Izdhan	100%	6/28/20	6/28/20
Video Shooting				
Work on video shooting idea	All members	100%	6/9/20	6/30/20
Video shooting day	All members	100%	7/1/20	7/8/20
Video editing	Ibrahim Izdhan	100%	7/9/20	7/14/20
Finalise of video	All members	100%	7/15/20	7/19/20
Final report checking	All members	100%	7/19/20	7/26/20



Project Start: Mon, 5/11/2020

Display Week: 8

TASK	ASSIGNED TO	PROGRESS	START	END
Preparation Work				
Waiting for components' arrival	WaiSing	100%	5/11/20	5/26/20
Measurement of some components	Ibrahim Izdhan	100%	5/13/20	5/14/20
Research on existing product	All members	100%	5/13/20	5/21/20
First draft of product	All members	100%	5/13/20	5/16/20
Autocad drawing of product	Juen Jet	100%	5/16/20	5/21/20
First draft of proposal without marketing researc	All Members	100%	5/13/20	5/21/20
Finalise of proposal	All Members	100%	5/21/20	5/29/20
Project manufacturing				
Research on suitable android apps	Ibrahim Izdhan	100%	5/14/20	5/31/20
Working on product	All members	100%	5/26/20	6/9/20
Testing/ Amendment of the product	All members	100%	6/9/20	6/17/20
Software development	Ibrahim Izdhan	100%	6/18/20	6/22/20
Product Casing	All members	100%	6/23/20	6/28/20
Hardware final look	All members	100%	6/28/20	6/28/20
Software final look	Ibrahim Izdhan	100%	6/28/20	6/28/20
Video Shooting				
Work on video shooting idea	All members	100%	6/9/20	6/30/20
Video shooting day	All members	100%	7/1/20	7/8/20
Video editing	Ibrahim Izdhan	100%	7/9/20	7/14/20
Finalise of video	All members	100%	7/15/20	7/19/20
Final report checking	All members	100%	7/19/20	7/26/20



13. References

[1] WHO, "Updated WHO advice for international traffic in relation to the outbreak of the novel coronavirus 2019-nCoV," 2020. [Online]. Available: https://www.who.int/news-room/articles-detail/updated-who-advice-for-international-traffic-in-relation-to-the-outbreak-of-the-novel-coronavirus-2019-ncov-24-jan/.

[2] J. Bogaisky, "Tech That Scans People For Fever In Big Demand Amid Coronavirus Crisis, Boosting Wuhan Company," Forbes, 2020. [Online]. Available: https://www.forbes.com/sites/jeremybogaisky/2020/02/10/we-are-running-as-fast-as-we-can-coronavirus-sparks-surge-in-demand-for-infrared-fever-detection-equipment/#191e675d60c9.

[3] Medical Expo, "MEDICAL THERMOMETER / INFRARED / NON-CONTACT / FOREHEAD," [Online]. [Accessed 2020].