Autonomous Temperature Tracking (A.T.T)

Final Report

ECE 499

JULY 28, 2023

Group ID: Group #20

Faculty supervisor: Dr. Mihai Sima

Team Information:

No.	Name	Student Number
1.	Ashwin Muruganandam	V00927283
2.	Aidan MacNichol	V00928936
3.	Will Bowen	V00907600
4.	Serafin IV Villaraza	V00943474

Table of Contents

List of Figures and Tables

Introduction

With the emergence of SARS-CoV-2 in the latter half of 2019, populated locations have been adopting a safety first approach in managing the spread of contagious diseases.

The emergence of this highly contagious virus has exposed the vulnerabilities in our existing public health infrastructure and underscored the importance of proactive measures to manage infectious diseases effectively. As densely populated areas continue to be hubs for social and economic activities, ensuring the safety and well-being of communities in these spaces has become a paramount concern. Traditional methods of manual temperature checks and symptom screenings are often labor-intensive, time-consuming, and prone to human error.

This project aims to help address this issue by implementing a simple, yet effective technology to help identify sick individuals. This seamless integration of facial tracking and temperature monitoring will enable a swift and accurate assessment of individuals as they pass through designated checkpoints or areas of interest.

Our aim is to offer an additional layer of protection to populated locations, effectively improving their ability to manage and mitigate contagious disease outbreaks. Early detection of individuals with elevated temperatures can help prevent potential disease transmission, allowing for timely isolation and medical intervention when necessary. Moreover, the system's non-invasive and discreet nature ensures a user-friendly experience, avoiding unnecessary discomfort for the public while fostering compliance with health and safety protocols.

The final product will be made accordingly in consideration with the monetary and time constraints of this course. Through this endeavor, we aim to provide communities with an easy-to-deploy and reliable tool to enhance public health surveillance, and foster a safer environment for all.

The targeted demographic and user base includes locations of high and constantly moving population density that can include hospitals, airports, schools, and public libraries. This technology would also add value to any closed indoor space but is generally more effective for places that experience a constant stream of people. With social distancing being highly improbable and ineffective to properly follow in cramped spaces such as airplanes and public transport, accurate temperature tracking can help mitigate the spread of diseases.

For the purposes of ECE 499, on the hardware side of things, a physical contraption will be built consisting of a MEMS Thermal Sensor, a few servo motors, a 1080p Webcam, a Raspberry Pi Model 4B, a distance sensor, speakers, and a Liquid Crystal Display. For the software implementation, Python3 along with the OpenCV library will be used to control the apparatus.

Social Impact

The social impact of this project includes the improved health and welfare of communities due to better disease protection. False negatives may lead to infected individuals going undetected, posing a risk to the broader population, while false positives might result in unnecessary panic and disruptions. Therefore, it is imperative that the proposed design can maintain a certain level of reliability.
Some relevant principles from the code of ethics of EGBC for this design are the first and sixth points:

The first principle, which emphasizes holding paramount the safety and health of the public, guides the design philosophy of the project. We acknowledge that any technology intended for public use must meet rigorous safety standards and accuracy thresholds to minimize potential risks. To uphold these principles, the information on human temperature must be correct so as to maintain the public's safety.

Objectives

- Conceptualize the final design and how each component will be integrated with each other.
- Program the functionalities of each component (i.e servos, LCD, distance sensor, camera and thermometer).
- Design and construct the mechanical framework for the device and ensure each component can securely fit within the framework.
- Ensure accurate and feasible readings that correspond to real temperatures.
- Make the final assembly as small as possible to help with the deployability of the finished product.

Design Specifications

To ensure the feasibility of the proposed design, the following design specifications were necessary.

- **Temperature Sensor Accuracy between 0.1°C and 0.5°C .**
  To prevent false positives and negatives, accurate and real-time temperature readings are necessary to be produced at a fast rate. Since the range of normal human body temperature is quite small, it is imperative to maintain a higher precision reading that can indicate if someone is displaying an abnormal temperature. This is by far the most important criteria for this design and has the utmost of priority.

- **Servo Motor Duty Cycle ~ 2 Degrees.**

The webcam mounted onto the enclosure has to be able to view the user's face at all times during temperature tracking. For the webcam to be centered on the user's face, the servos have to maintain a certain speed and range of movement to help move the assembly of sensors.

- **LCD contrast adjusted for human readability.**
  Upon initial connection to the power rail, the LCD exhibits nearly white blocks covering the entire first row, indicating an issue with the voltage settings. Consequently, when programmed to display text on this row, readability becomes compromised for the user. To rectify this problem and improve visibility, a potentiometer is introduced to enable adjustments to the voltage supplied to Vo, thus ensuring optimal display performance.

- **Distance Sensor (HC-SR04) capable of accurately measuring up to 1 meter.**
  The HC-SR04 sensor is used to track distances usually up to a maximum of 400 cm. For the scope of this project, the user is positioned directly in front of the array of sensors. Hence, the distance sensor only needs to be able to track up to 100 cm.

- **Accurately keep track of the user's face to get a more accurate reading.**
  Being able to accurately keep track of the user's face means a more consistent temperature reading centered on the user's face. This also improves usability.

- **General Useability**
  This is a qualitative specification which will measure how intuitive and easy to use the contraption is. More specifically can people use it without instructions, do they understand what to do and how to use it quickly.

Proposed Design Schematic

The schematic for this design is extremely simple. The individual sensors and components are connected to the Raspberry Pi through its GPIO ports. A notable addition is the two pull-up resistors on the I2C lines of the Temperature Sensor (SDA/SCL) in order to pull-up the voltage to ~3V, as well as a potentiometer to supply power to the LCD. The figure below shows the Wiring Schematic.

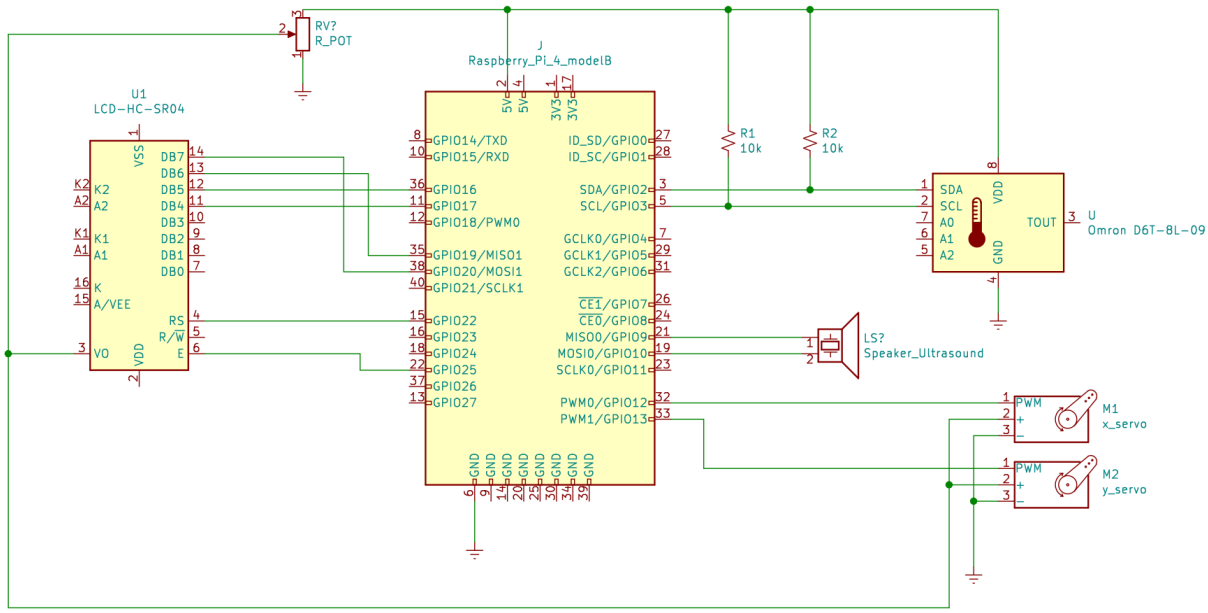

Figure 1: Wiring Schematic

Literature Survey

As technology advances further, leading researchers and scientists have continually developed new technologies to counter emerging viral diseases and illnesses. These technologies have been fundamental in preventing the spread of viral illnesses especially in densely populated areas by providing a means of detecting carriers before they are able to come into contact with other individuals. The following section of this report will focus on discussing various technologies that have been implemented in the past and how they compare to the project developed for ECE 499 in terms of various metrics such as cost, ease of implementation, etc.

DNA sequencing was discovered in the late 70's and it can be defined by the process of determining the order of bases in a human genome and has been used in technologies to detect viral diseases such as MinION [1]. The MinION device (seen in the figure below) is a small portable tool the size of a flash drive that has the capabilities to extract DNA from blood samples from a suspected carrier of a disease [2]. The device then runs the DNA sequencing procedure in real time and can generate results within an hour. After the device has finished gathering data from the gathered DNA from suspected carriers, a specialist is then needed to analyze the results in a computer by inspecting the constructed DNA sequences by the MinION to determine if the suspected carrier does indeed possess a viral disease. In recent years, this device has been deployed in West Africa, specifically in Guinea during the EBOV epidemic in which the deployed devices were able to successfully test approximately 140 suspected carriers in a span of 24 hours.

Figure 2: MinION Device

A major advantage of using this device is that it uses DNA sequencing, which is a highly advanced medical tactic that can result in a 99% accuracy [3]. However, a disadvantage of this device is that it required a relatively long time to gather the data (approximately 1 hour), which would make it difficult to deploy in an area with high population density. Furthermore, this device is relatively costly to manufacture and the price for each unit can go upwards of $1000 [4]. All of these factors when compared to the project developed for ECE 499, it can be seen that the ECE 499 project is estimated to cost around $250, much lower than the MinION.

Furthermore, the ECE 499 project is relatively easy to implement in public areas and it does not require a specialist to transcribe the gathered data, as it is easy for the average person to understand a temperature reading and determine whether or not they carry a viral illness.

Another widely used device to detect viral diseases is the Rapid Antigen Detection Test (or more commonly known as a rapid test) seen in the figure below [5]. This technology has been continually researched since the early 1980's during the HIV epidemic, but this technology has risen in popularity during the Covid-19 pandemic [6].

This device is a small compact plastic strip with two indicators, one indicator is bound with a SARS-CoV-2 protein and the second indicator is bound to a control antigen to serve as a control indicator. A swab is then taken from the suspected carrier and the sample collected is passed through a tiny slot in the rapid test strip which allows the liquid containing the antigen to flow through the device, binding to the indicators and displaying whether a person is indeed a carrier of the viral disease [7]. This was a popular choice among establishments and governments to identify potential carriers due to a number of factors. First, it is relatively easy to use and does not require the assistance of a medical or technical professional [8]. Second, it can provide high accuracy results in a short amount of time (approximately 10 to 15 minutes) [8]. Lastly, it requires no electronics and costs around $2 - $10 to manufacture each test kit [8].

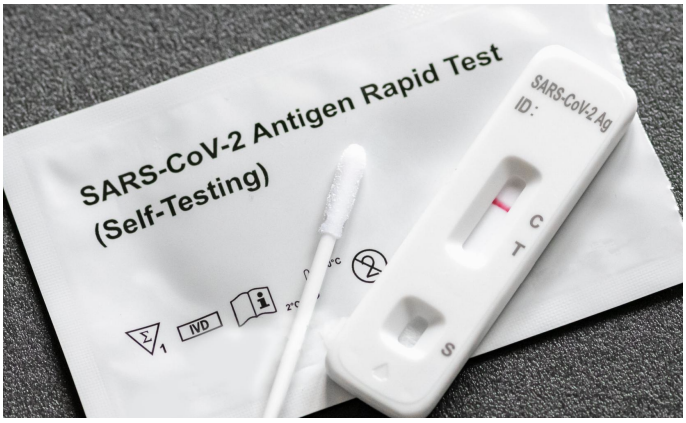

Figure 3: Rapid Antigen Test

Comparing this technology to the ECE 499 project, the cost of one testing kit is significantly lower, but it is important to note that the testing kits are single-use and the cost of using this device grows over time and will eventually surpass the cost of the ECE 499 project. Another area to compare the two devices is the time it takes to process the data and provide a result.

While the rapid test is generally fast, providing the result within 10-15 minutes, the ECE 499 project can provide a result almost instantaneously after successfully tracking a face and determining the location of the forehead. In terms of result accuracy, as stated beforehand, the rapid test kits uses antigen technology to accurately determine if SARS-CoV-2 proteins are present in a suspected carrier, which is a very accurate

During the height of the pandemic airports used non contact infrared thermometers (NCIT's) to check for travelers with a fever. For this project an infrared thermometer is going to be used; so looking at FDA guidelines for NCIT's is extremely useful. The first and most important FDA guidelines is the acceptable error of 0.5℃ [9]. This is the baseline accuracy as a higher error rate results in indeterminate results for detecting fevers. The FDA notes that although better than contact based thermometers, NCIT's still have a high risk of spreading diseases between the person handling the device and the persons who temperature is being taken [10].

Additionally there is some human error introduced as the NCIT has to be used directly perpendicular to the subjects forehead and within a certain range specified by the device's manufacturer (generally around 6-12 inches away) [10]. In addition to this there are a few more factors that may impact the accuracy of NCIT's [10]:

- Temperature of environment should be between 16℃ - 40℃ with a relative humidity below 85%
- Should be used out of sunlight.
- Headcovers or cosmetic wipes may increase/decrease the forehead temperature.

To summarize, compared to the available market solutions such as the MinION and the Rapid Antigen Detection Test, it is evident that the A.T.T has the potential to perform better in several aspects. One notable advantage of the A.T.T is its ability to produce instantaneous results compared to other market solutions, as it only requires the user to stand within view of the camera and the device uses an infrared laser to capture the user's temperature.

Another overall advantage this device has is its ease of use, since this device functions autonomously and does not require a complex procedure to be operated. This is a significant advantage over the MinION and the rapid test since the MinION requires a trained operator to be used, and although the rapid test can be operated by any individual, it still requires a precise set of steps that must be followed. A final advantage that the A.T.T has over other solutions is its low cost to produce, since the device consists mainly of readily available components in the market such as a Raspberry Pi, servos and other sensors. Due to the aforementioned factors, this justifies that the A.T.T has the potential to provide a better solution than what is currently offered in the market.

Team Duties and Project Planning

The following table details how the workload is split between the individual team members.

Table 1: Work Split Between Team Members

Team Member	Deliverable
Aidan	OpenCV Facial Recognition/Forehead Tracking
Aidan	Controlling/Moving Servos
Aidan	Infrared Thermometer
Ashwin	LCD
Ashwin	Distance Sensor
Will	Acquisition (i.e how the servos move to center the thermometer on target)
Serafin	Enclosure
Ashwin	Website
Serafin/Ashwin	Poster

Design Methodology and Analysis

Temperature Sensor

The **Omron D6T-8L-09** digital infrared thermometer was chosen for this project. This sensor works by continuously detecting radiant infrared energy and computing a temperature value from this. This is much better for our application than other non-contact thermometers like Pyroelectric sensors as they can only detect temperature changes and will not output a continuous and accurate temperature reading [11].

Another favorable aspect of this sensor is that it gathers temperature readings in 8 separate “elements” as shown in the figure below.

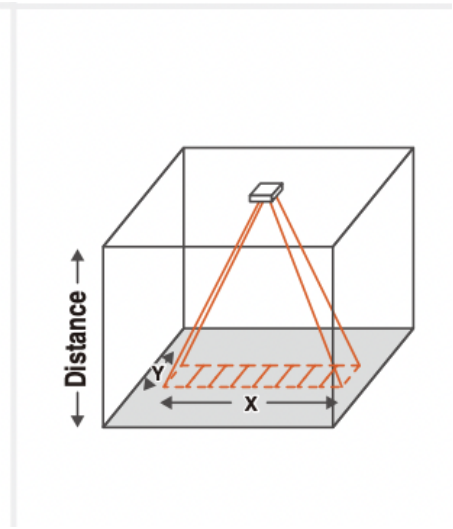

Figure 4: Temperature Sensor Visual

The viewing angle of the sensor is as follows [11]:

- x-direction: 54.4°
- Y-direction 5.5°

This gives a variable element size based on the distance between the sensor and the target. The distances and corresponding element size are shown in the table below:

Table 2: Element Size vs Distance

Distance from Target	Window size (of individual elements)
10 cm	1.0 x 1.0 cm
20 cm	2.0 x 2.0 cm

30 cm	3.0 x 3.0 cm
40 cm	4.0 x 4.0 cm

Upon initial testing of the sensor the most accurate readings were found to be at a distance < 30cm. This means that all of the infrared rays from just a 3x3cm area are read and converted into a temperature value which significantly reduces infrared radiation from other sources which may skew temperature readings.

The basic block diagram for the internal workings of the D6T is shown in the figure below [11]:

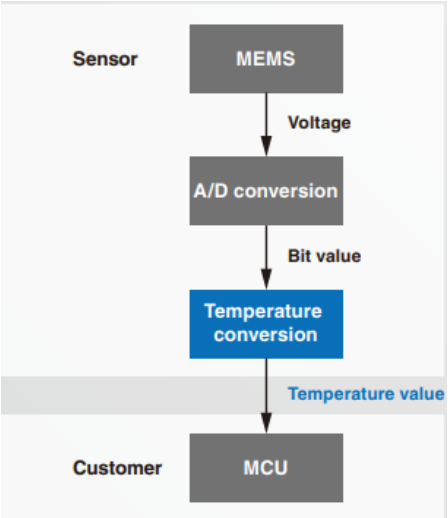

Figure 5: Rapid Block diagram of the Temperature Sensor

To output the data the D6T uses the I2C communications protocol with an external 10k pullup resistor on both the SDA and SLC output. From the software side a byte array of 19 size is read from the thermometer with the corresponding values:

- [0-1] PTAT temperature values
- [2-3] element 1 temperature values
- [4-5] element 2 temperature values
- [6-7] element 3 temperature values
- [8-9] element 4 temperature values
- [10-11] element 5 temperature values
- [12-13] element 6 temperature values
- [14-15] element 7 temperature values
- [16-17] element 8 temperature values
- [19] CRC checksum (used to detect errors in data transmission)

Each raw temperature data is converted to a Celcius as follows:

$$Temp\ ℃\ = \frac{tempArray[i+1]*256 + TempArray[i]}{constant}$$

The constant is suggested as 10 but can be adjusted through trial and error to get a much more accurate value.

The PTAT is the voltage difference between two p-n junctions which generate a current **proportional to absolute temperature (PTAT)** This value is inconsequential to our project.

Facial Recognition

There were 2 main challenges with this section:

1. Optimization
2. Forehead location relative to distance from camera

The facial recognition was implemented using OpenCV, an open source image processing library. This library included a method that returns the points of certain “face landmarks” [12] i.e left/right eye, lips, nose. This method implements two pretrained facial recognition models. One is a lighter weight which only returns the location of the nose tip, left eye and right eye. This model was used as it ran much faster than the larger model.

To get the location of the center of the forehead first the center point between the left and right eye was found. This is used as the x-coordinate of the center of the forehead. This was done using a simple midpoint formula:

$$midpoint\ = \frac{left\ eye_{x-coordinate}. + right\ eye_{x-coordinate}}{2}$$

To get the y-coordinate of the forehead a constant could not just be added to the y-coordinate of the eye locations. This is because if the subject is far away from the camera the forehead location would be very low on the face and conversely when the subject is close the location would be extremely high on the face or even above the head in some cases. To circumvent this problem the distance between the eyes was taken and divided by a constant. This is because the distance between the eyes changes based on subject distance which is used to scale the y offset of the location of the forehead from the eyes. The constant used was 2; which was found by trial and error testing it at different distances on different people. The figure below shows what the computer sees with red dots on the eyes and a red dot on the final calculated forehead location.

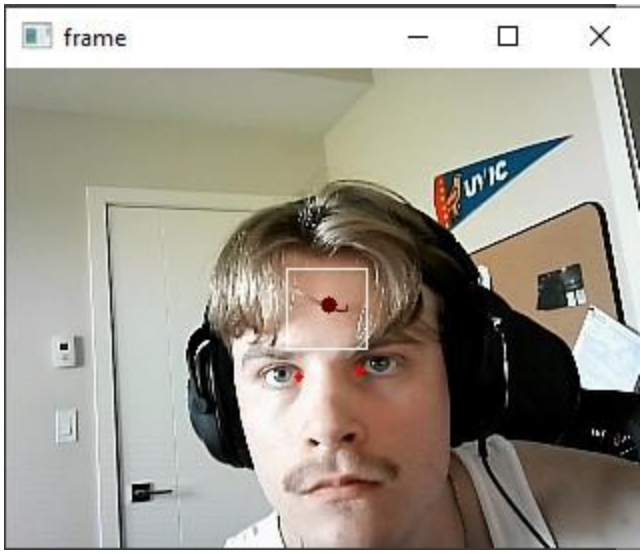

Figure 6: View of the USB Camera

Servos

The next big challenge is given an x,y coordinate to move two servos; one for the x-axis and one for the y-axis to center the temperature sensor on that given point.

The way the coordinates worked from the camera side is that the frame is divided by the number of pixels in it with these rules:

- The upper left corner is point (0,0)
- The lower right corner is point (320,240)
- The center of the frame is point (160,120)

The way the servos were implemented is that they are moved based off of a given duty cycle. The duty cycle values are arbitrary with a value of 500 being at an angle of 0 (all the way left) by these rules:

- A value of 500 is an angle of 0° (all the way left)
- A value of 2500 is an angle of 180° (all the way right)
- A value of 1500 is an angle of 90° (centered)

When the program is started the servos are centered at 1500. A calculation is done to find the distance from the forehead coordinate to the center of the frame in both the x and y direction:

$$x_{distance} = forehead_{x-coordinate} - 160$$
$$y_{distance} = forehead_{y-coordinate} - 120$$

These absolute values for both x and y are checked if they are then within a margin of error. Through trial and error we found about a 10x10 box that gave consistent results. This step is crucial as if there was no margin of error the servos would constantly overshoot the centerpoint and oscillate back and forth over it.

If $x_{distance}$ or $y_{distance}$ is greater than the margin of error the servos are then moved either left/right or up/down in the direction of the centerpoint. The amount they moved was arbitrarily set to be a value of 5 which corresponds to ~5°. This process is then repeated until the centerpoint is reached. **Figure xx** shows a visual representation of the frame and error box and **Figure xx** shows a flowchart for the code.

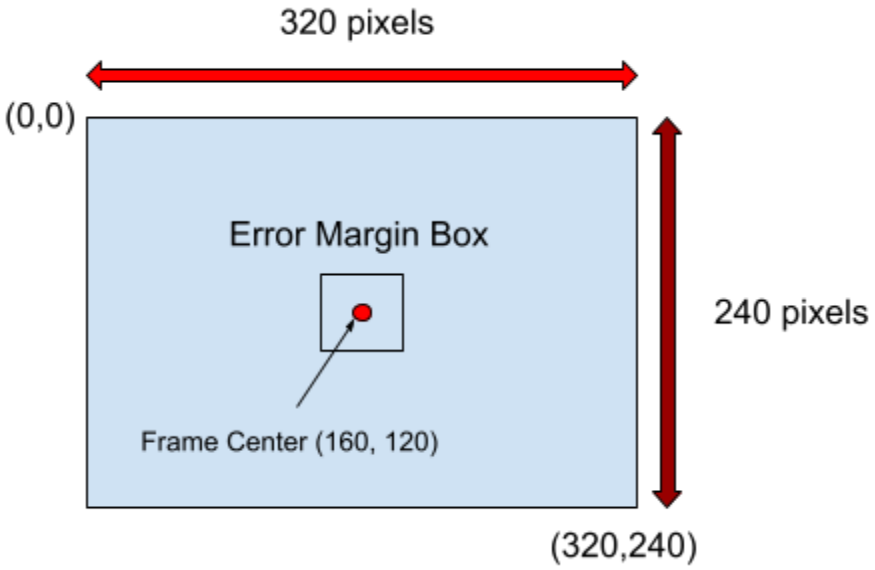

Figure 7: Targeting Diagram

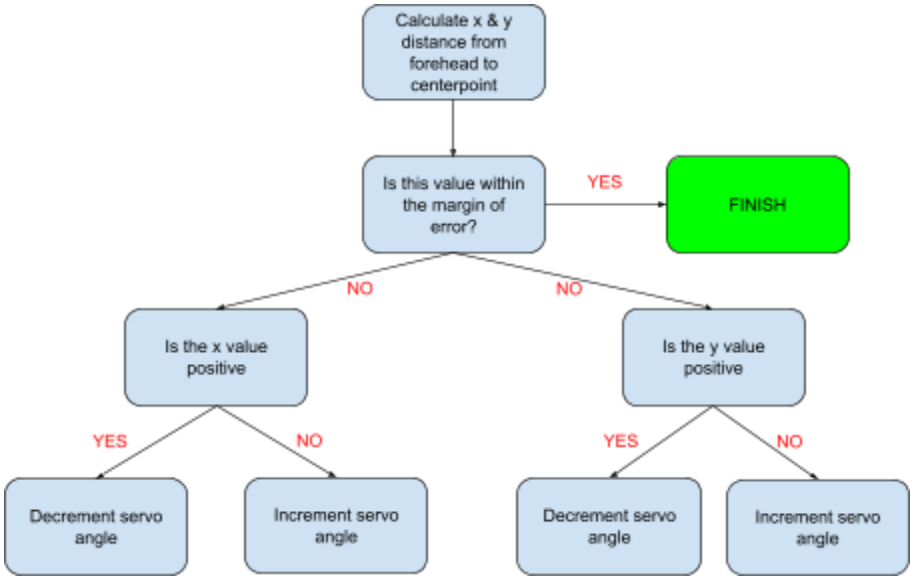

Figure 8: Servo Control Flowchart

Ultrasonic sensor

The **HC-SR04** ultrasonic sensor was chosen to act as a distance sensor for A.T.T. It provides distance measurements between the ranges of 2cm to 400cm. The way the module works is by sending eight 40kHz signals and waits for a pulse signal back to determine the distance.

$$Test\ Distance = (High\ Level\ Time * Velocity\ of\ sound\ /\ 2)$$

The timing diagram for the sensor depicts the trigger input required and the echo pulse that's received by the module. Below is an image that showcases the signals.

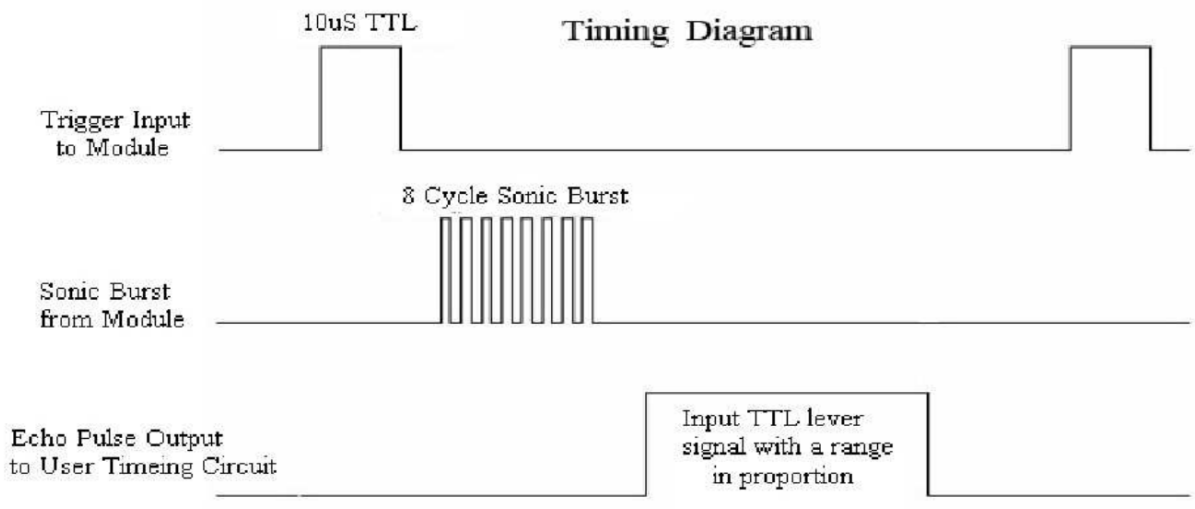

Figure 9: HC-SR04 Timing Diagram [16]

A short 10uS pulse is supplied to the trigger input to start the ranging. Subsequently, a short 8 cycle burst of 40kHz ultrasound is sent out with the range being calculated between the sending of the trigger signal and the receiving of the echo signal.

Liquid Crystal Display (LCD)

The 2x16 Liquid Crystal Display was chosen to help guide the user to an optimal temperature tracking distance with live feedback displaying the distance from the sensor, and after measurement, their recorded temperature.

The LCD was connected to the Raspberry Pi to receive signals that carried the temperature and distance data and fed to the display. The connections for the LCD were as follows:

- RS (Reset) connected to GPIO 22
- EN (Enable) connected to GPIO 17
- D4 - D7 connected to GPIO 18, 23, 24, 25
- 5V Vcc Input

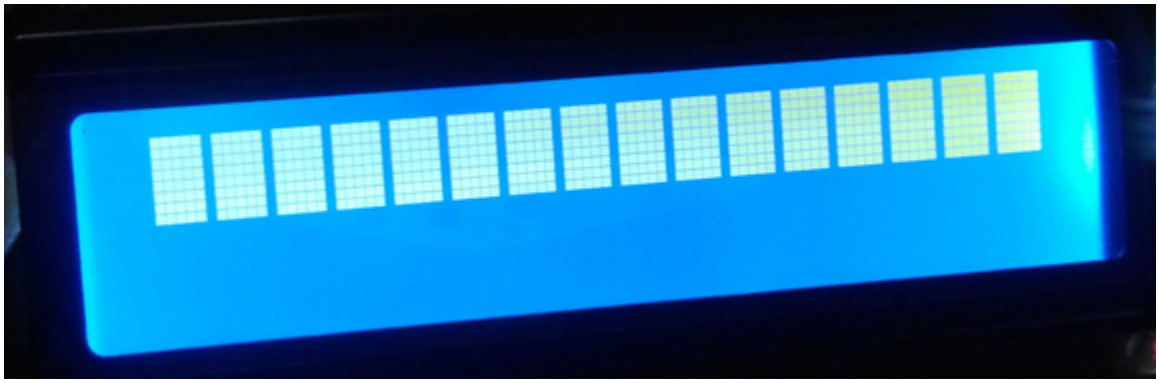

Figure 10: LCD when Powered On

To adjust the contrast ratio of the LCD, a precision potentiometer was used to control the voltage of the signal (V_o) supplied to the display helping improve readability for the user.

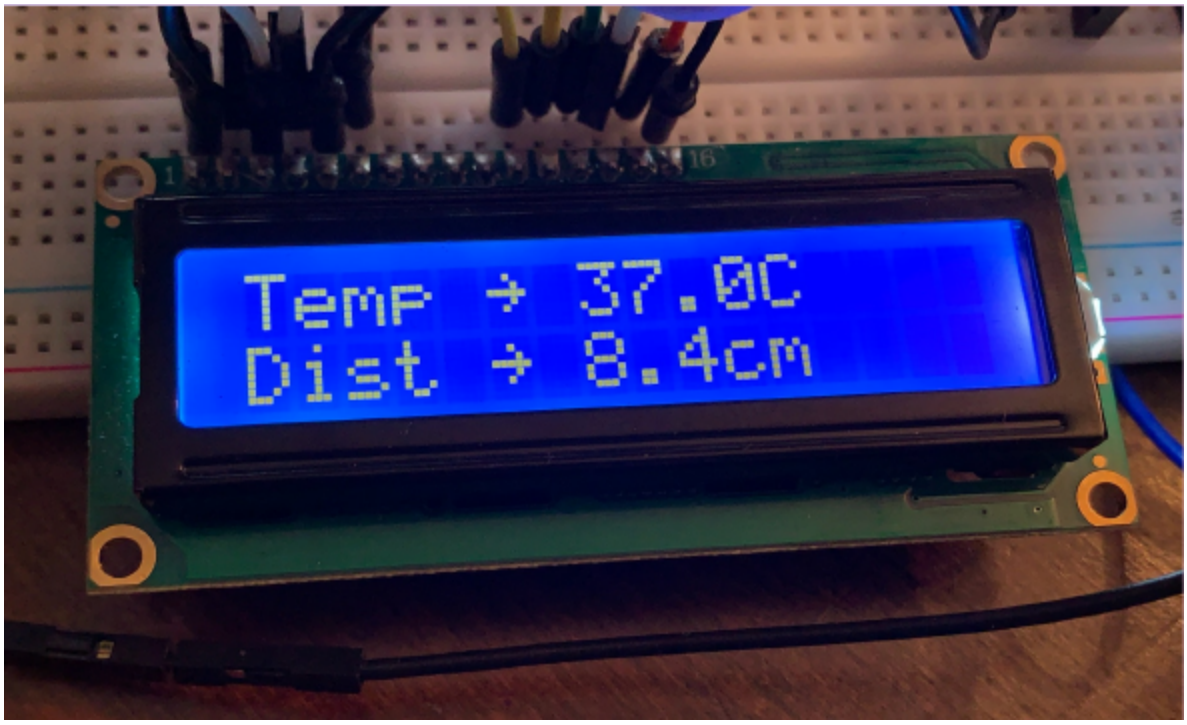

Figure 11: LCD with Adjusted Contrast

Mechanical Casing

Another major part of the project was designing the enclosure for the components and designing the mechanical framework that would secure all the sensors and servos into place so they may perform their intended function. Each part was designed using Solidworks, along with each component's appropriate measurements to ensure that each component will fit properly in their respective places.

The first component designed was the main box module seen in the image below where the main components such as the Raspberry Pi and the breadboard will be located, along with all the wiring connecting to the sensors and other components. This is to ensure that they will remain intact and provide an aesthetically pleasing outside appearance by avoiding all the complex wiring to be seen by the users. Located in the front of the box are two exterior cuts that will allow the ultrasonic sensor and the LCD displaying the user's temperature and distance away from the device to pass through.

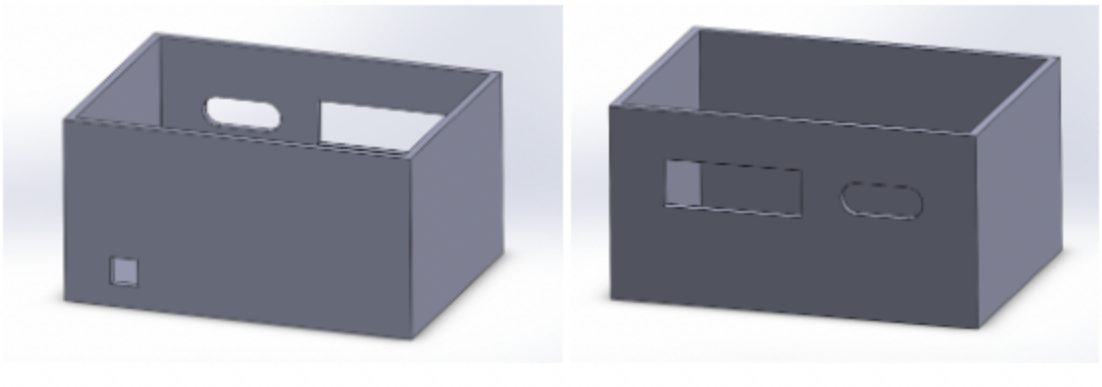

Figure 12: Front (right) and Back (left) of the Box Module

The image below displays all the necessary dimensions of the box that contains all the boards. The box's dimensions (190mm X 100mm X 100mm) were carefully chosen so that the Raspberry Pi and the breadboard (85mm by 56mm and 65mm by 50mm respectively) can easily fit inside the container while having excess space for wiring for all the components.

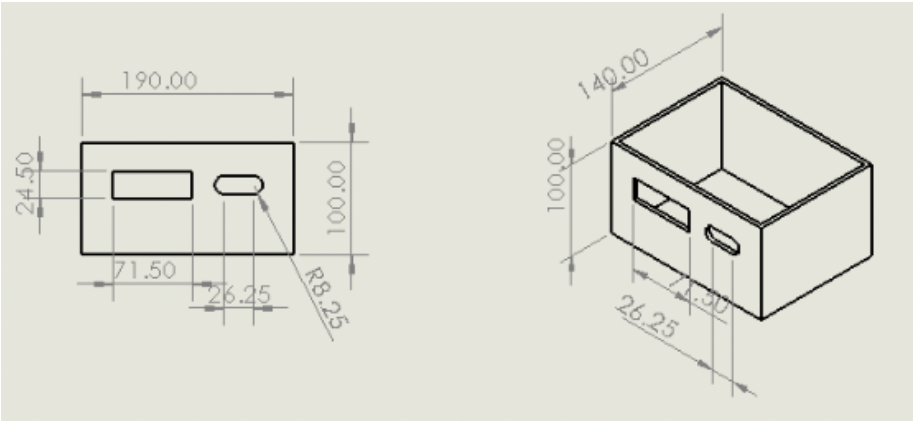

Figure 13: Front and Isometric View with Measurements

The next part to be considered is the head module seen in the image below. The head module is responsible for securing the laser thermometer and camera into place. This design features a cylindrical container attached to a rectangular base with two protruding prongs to help secure the head module onto the serve controlling the y-positioning. This design also includes two extruded cuts in the front to secure the thermometer and camera.

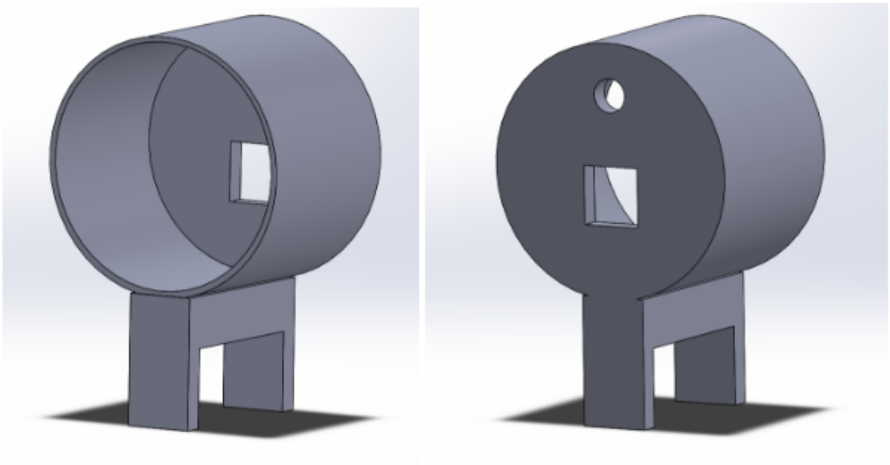

Figure 14: Top Module

Seen in the image below are the dimensions and measurements for the head module. Notice how the two prongs are separated by a distance of 40.5mm, this is to ensure that the servo (40mm wide) can secure the head module in place. Furthermore, the rectangular base where the cylinder is connected to is 20mm in height, this is to ensure that the cylinder would not wedge against the wheel on the servo.

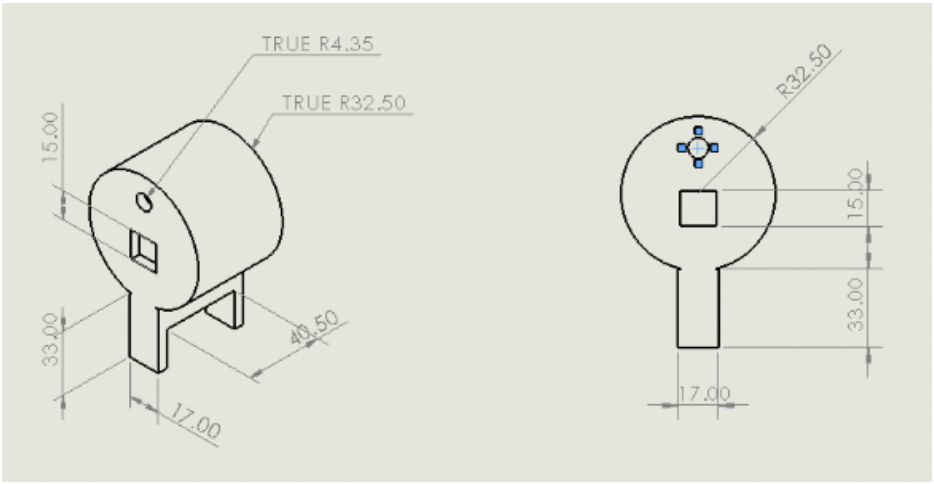

Figure 15: Top Module with Measurements

The next part to be discussed is the XY connector seen in the image below along with the corresponding dimensions and measurements. This part is what allows the head module to move in the X-direction and in the Y-direction. The servo controlling the movement in the X-direction would be attached to the bottom, while the servo controlling the movement in the Y-direction would be attached to the upright face of the connector. The base and the upright face are 30mm wide to ensure that the servos can

securely be drilled into the connector. There are also triangular support pieces to assist with stability and to maintain structural integrity of the device.

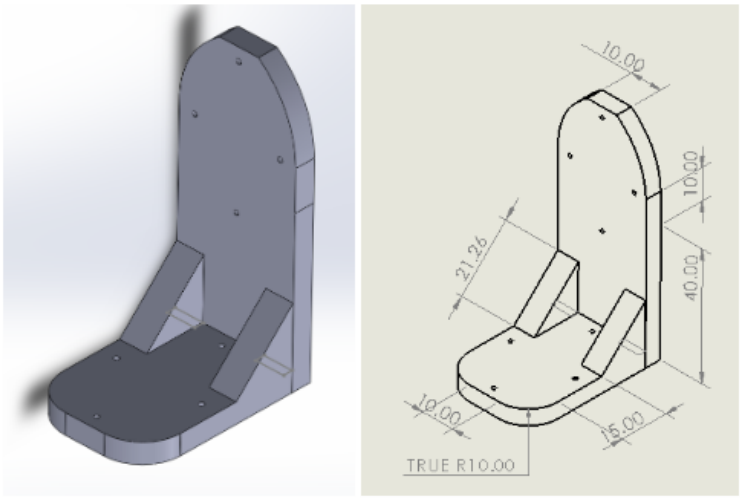

Figure 16: XY Connector with Measurements

The final part to be discussed in this section is the lid of the box module seen in the image below with the measurements. The lid is 190mm X 140mm with a groove on the edges to securely fit inside the box. On the surface, there is a curved cut-out so that wires connecting to the sensors, servos and camera can connect to the Raspberry Pi and the breadboard. Located in the center of the box is a rectangular engraving which serves as an indicator as to where the servo controlling the X-direction will be mounted to.

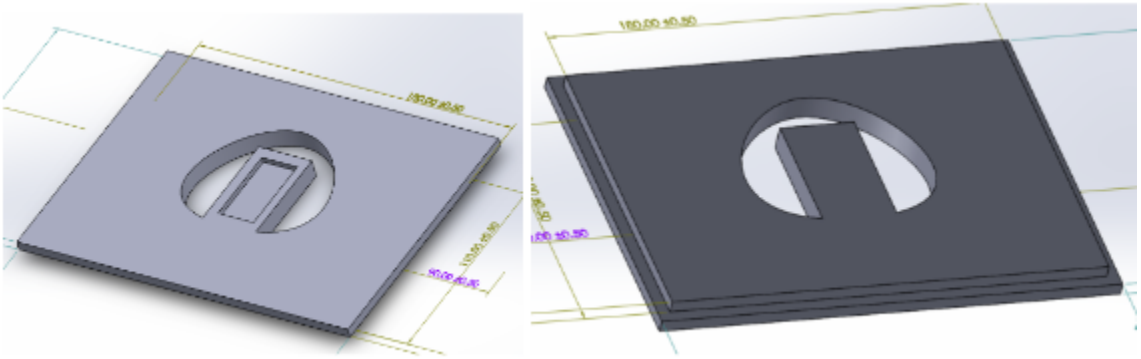

Figure 17: Lid Module

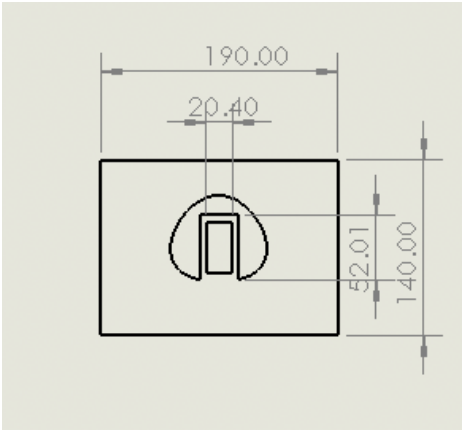

Figure 18: Lid Module with Measurements

Finally, the image seen below is a representation of how all the components will be fitted together, excluding all the other electrical components.

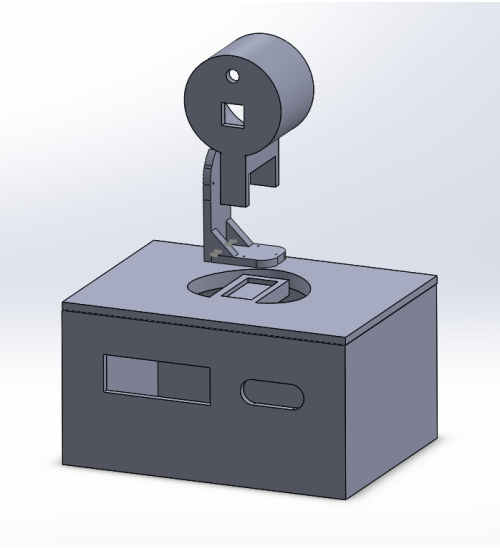

Figure 19: Finished Module

Prototyping and Testing / Evaluation

This project employed a bottom-up design methodology, which emphasized the development of individual elements before their integration into a larger sub-system. The process began with creating a basic design and using already available components for testing. By developing and testing smaller, independent elements first, the project can achieve incremental progress. This allows for early validation of individual functionalities and avoids investing significant effort in a design that may later prove unsuitable or difficult to modify.

The hardware components were ordered online and work began on implementing each component individually. The software portion of this project employed object oriented programming. Each of the individual components: Servo, Temperature Sensor, Distance Sensor, Facial recognition and LCD where individually coded and made into a class. This allowed each sub component to be individually tested and verified before integrating into the final structure.

To validate the initial structure, a prototype was constructed using a Pi microcontroller, two servo motors, a webcam, and a rough 3D printed enclosure. This early version successfully demonstrated the capability to locate and navigate to a person's forehead, but it lacked a thermometer to measure temperature readings. The prototype's software included the development of the facial detection code.

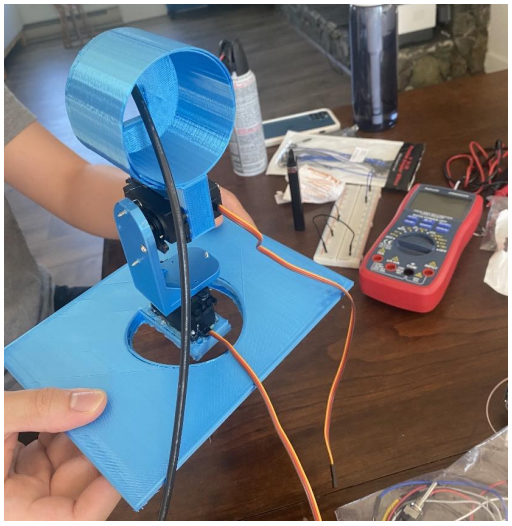

Figure 20: Integration of the Motors, Enclosure, and Camera

Once the enclosure was designed and 3D printed, its parts were assembled with the x and y servo motors by screws. The camera was then installed, and an algorithm was developed to enable the motors to move based on the forehead coordinates. The LCD display was set up and programmed to display the user distance from the sensor and their recorded temperature. For ease-of-use, the LCD was also programmed to show the user instructions such as: to face the sensor, to move closer, and to hold still while their temperature is being recorded. Finally, the LCD, temperature sensor, and ultrasonic distance sensor were integrated into the system. The code for collecting the temperature data was made and the thermometer was calibrated according to the manual instructions. This data could now be sent to the LCD, where it was displayed for the user's convenience.

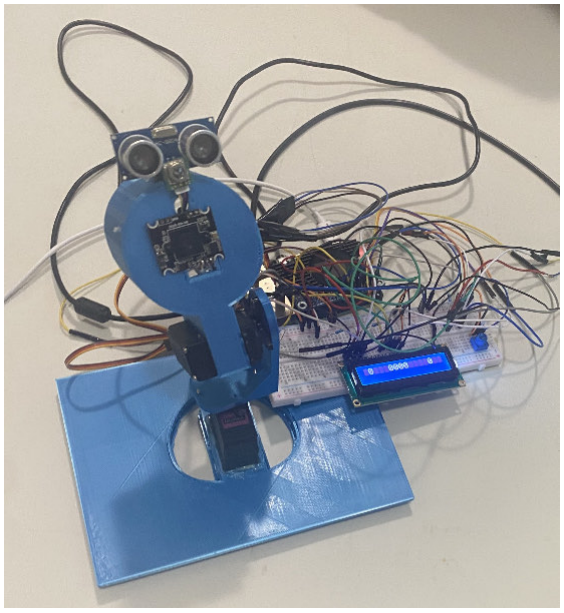

Figure 21: Completed Design

Throughout the entire design process, ensuring temperature accuracy remained the primary concern. Given the potential consequences of incorrect readings, such as the spread of diseases, the safety and health of the public were deemed paramount. Adherence to the EGBC (Engineers and Geoscientists British Columbia) code of ethics further underscored the importance of accurate body temperature information to safeguard public well-being.

To ensure the final system's reliability, extensive testing was conducted. Rigorous evaluations were carried out to verify the accuracy of temperature readings, which were then meticulously compared against readings obtained from an oral thermometer. This comprehensive testing process aimed to provide confidence in the device's ability to function correctly and safely, in line with its intended purpose and the welfare of the public it served. It revealed that the thermometer would only occasionally give a correct reading among several incorrect readings. After some analysis, it was clear that sweat and distance could impact the data. To solve this problem, the code was modified to only save a reading within a distance of 30 cm. The final temperature is printed as the average of ten of these readings. This change significantly increased the accuracy of the thermometer.

Test & Evaluation plan

This section details how the individual components will be tested along with the final testing plan:

Table 3: Individual Component Test Plan

Component	Test Plan
Facial Recognition	Use OpenCV tools to draw a dot on the calculated

	forehead location. Use a live feed of the video to confirm this dot is in the right location with 4 different test subjects at varying distances and locations
Servos	Confirm that given a duty cycle instruction the servo moves an appropriate distance and does not “jiggle” i.e stays still.
LCD	Confirm given a text input that it is properly displayed on the LCD
Distance Sensor	Use a tape measure to confirm the readings given by the distance sensor are accurate up to 1 meter.
Temperature Sensor	Do 50 trials on 4 different test subjects. Use an oral thermometer as a baseline value then immediately get a reading from the sensor at a distance of 30cm. Use a hot rag placed on the forehead to simulate high temperatures in the fever range.
Integrated Component Final Test Plan	Confirm the robot tracks various persons faces at various distances from 20 cm to 100 cm away at different starting locations. Confirm the correct on screen instructions are printed to the LCD: 1. **Cannot locate face** - when no face is in view 2. **Move Closer + Current Distance** - When the test subject is greater than 30cm away 3. **Hold Still** - When temperature readings are being taken and averaged. 4. **Temp:** - Displaying the final temperature reading.

Alternate Designs

Throughout the testing and implementation of this project, several features, components, and ideas were discarded due to various reasons such as being costly, too complex, or too time demanding. This section will discuss such features and components that were omitted from the final design of the A.T.T.

One of the challenges of building this device was constructing the appropriate enclosure with the correct measurements to secure all the components. The design seen in the figure below was the first version of

the top module before eventually being redesigned to the version seen in Figure 22. This design served a similar purpose to the final design with a rectangular cutout to secure the servo controlling the movement in the X-direction. The main difference this design has with the final design is that this has a cylindrical shaft roughly 100 mm in height and 30 mm in diameter, with a curved cutout at the top of the shaft to ensure that the XY connector and the Y-direction servo would not wedge against the shaft.

The main function of the shaft was for aesthetic purposes so that all the wires connecting the breadboard/Raspberry Pi to the various sensors and components would not be visible to the users. As for the reason why this design was discarded, it was determined that this part would take several days and a lot of material to print due to the size and complexity of this part. Furthermore, since 3D printing poses a lot of uncertainty, it was determined that this design would not be fabricated correctly due to its complex structure. The other issue is the 3D printer used was of lower quality and would often fail larger prints. This resulted in many day long prints failing which wasted a considerable amount of time and materials.

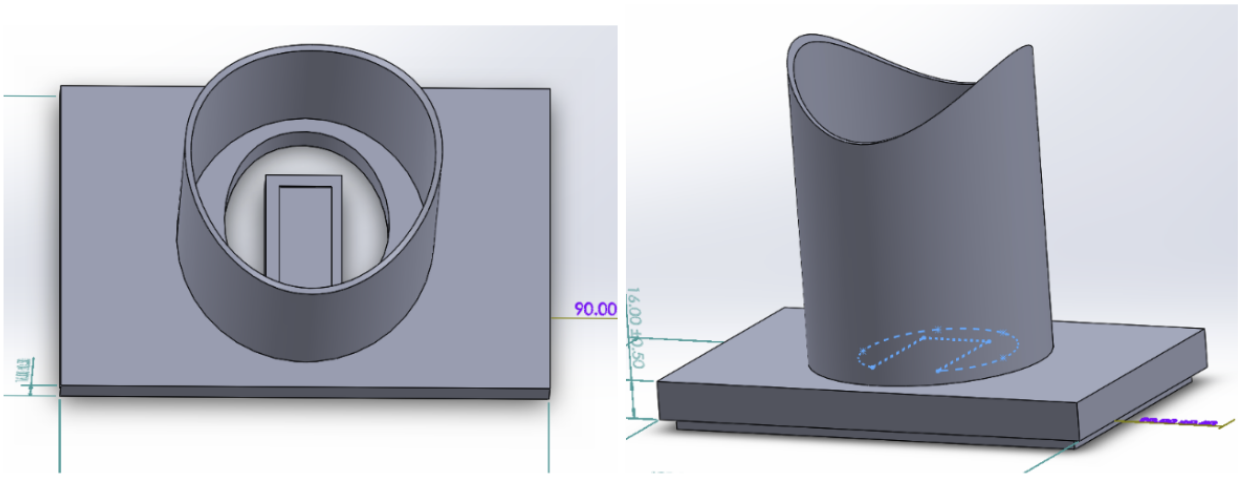

Figure 22: Alternate Design Module

Another feature of the A.T.T that was later discarded was the planned implementation of a speaker in the main box module. The goal of this speaker was to increase accessibility of the A.T.T for those who may be visually impaired. This function was of the lowest priority for the project and did not get implemented due to time constraints as well as the extra cost it would incur.

Temperature Sensor Testing

The temperature sensor selected measures temperature using infrared rays hitting a thermopile. One of the specific reasons this temperature sensor was chosen is that upon startup it measures all of the latent infrared rays to gather a baseline “room temperature” which is then used to calibrate the sensor. This circumvents a lot of issues with infrared radiation from other sources impacting the temperature reading. This technology is proprietary and as such Omron has detailed very little information on the inner workings on how it works. Omron states an accuracy of ±1.5°C at all far ranges with a ±0.5°C

accuracy out of the box at “close distances (they provide no further information at what a close distance is)” with up to a ±0.1°C with user calibration [11]. This calibration comes in the form of a user-found constant to divide the raw bit-temperature value to a human readable temperature value.

To calibrate and test the sensor we did **50 trials** total on 4 different people at a distance of **30 cm** away from the thermometer. The method for testing was to gather a baseline temperature using a standard oral thermometer; the Physio Logic Acuflex thermometer which claims to have an accuracy of ±0.1°C [13]. After an oral reading was gathered and immediate reading from the Omron sensor was taken. The way a final temperature value is taken by the Omron sensor is that 10 readings are taken and then averaged to get the final value. Any obvious outlier temperate readings (35°C < T < 41°C) are discarded in this averaging process to increase accuracy. These values were chosen as anything less than 35°C is considered hypothermic and anything greater than 41°C is considered a medical emergency and a person with these temperatures would not be able to walk around [14]. To simulate fever-like temperatures, (38℃ and above) a rag soaked in hot water was placed on the test subjects forehead for 30 seconds, the temperature of the rag was taken using the oral thermometer then an experimental value was gathered using the A.T.T. A graph of the Real temp vs the Measured temp is shown below: A table of the values obtained is shown in the appendix.

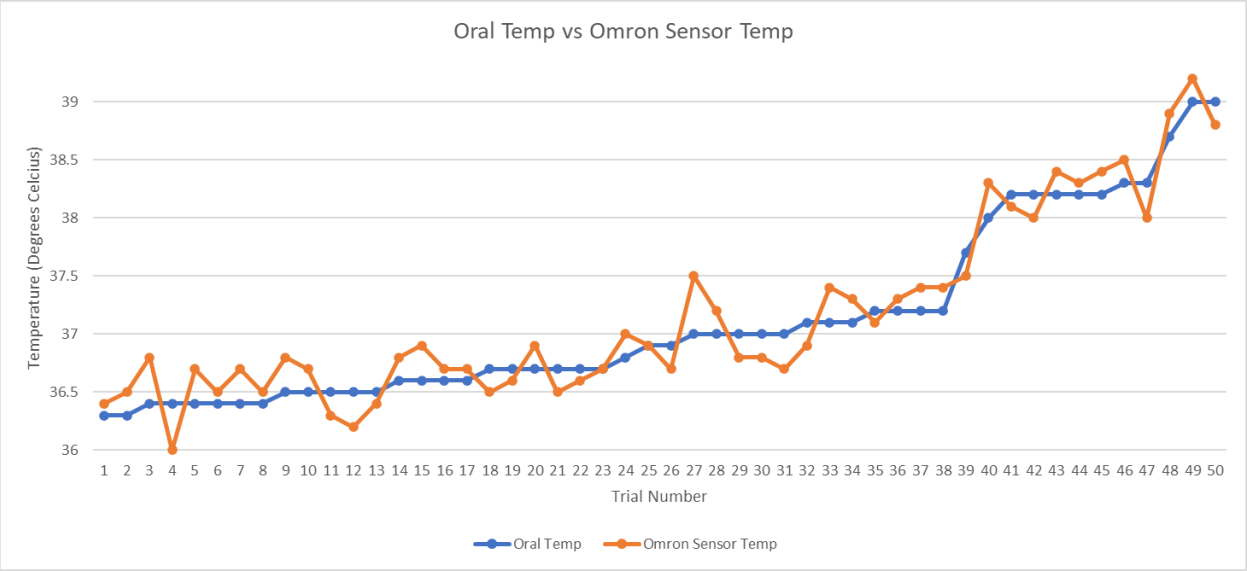

Figure 23: Oral Temperature vs Omron Sensor Temperature

Calculating the average difference between the Real temp and measured temp as well as the total % error between the two value gave the following results:

- **Average Error: 0.201** ℃
- **Percent Error: 0.538 %**

These results are surprisingly accurate. A notable thing about the Omron sensor readings is that although they are fairly accurate they are not consistent; meaning it can give a reading 0.1-0.2 degrees

above or below the oral temperature in consecutive trials on the same test subject. This can be particularly problematic if a person is near the temperature as a fever as one measurement may indicate they have a fever then a re-measure may say they don't.

Cost Analysis

Table 4: Product Expenses

Product Expenses				
Item Name	Description	Units	$/Unit	Total Cost
MG996R metal Gear Servo Motor	High-torque metal geared servo motor	2	$4.54	$9.08
Raspberry Pi 4	Low-cost microprocessor with I/O ports	1	$60.75	$60.75
Omron D6T-8L-09	Infrared Temperature Sensor	1	35.51	$35.51
Hc-SR04 Ultrasonic Sensor	Distance sensor	1	$9.53	$9.53
DFR0063 LCD Display	LCD display	1	$19.12	$19.12
Camera	USB Webcam	1	$18.00	$18.00
Final Cost				**$151.99**

Table 5: Labour Expenses

Labour Expenses				
Task Name	Number of Working Engineers	Hours	$/Hour (Minimum wage in BC for engineers)	Total
Researching design	4	3	$27.44	$329.28
Building prototype	4	4	$27.44	$439.04
Working on individual components	4	5	$27.44	$548.80
Integrating Components	4	8	$27.44	$878.08
Testing	4	2	$27.44	$219.52
Final Cost				**$2,414.72**

Based on the data presented in the tables, the total cost of purchasing all of the components amounts to $152. By selling each unit for $303.99 at a 50% profit margin, a total of 16 units need to be sold to break even from the labor costs. In the case of a successful business model, this device holds potential across various industries, including travel and entertainment. With 1500 units sold, the projected return on investment would be an impressive $225,568.00. To further economize, strategic measures can be implemented. For instance, opting for more cost-effective components could substantially reduce expenses. Specifically, identifying economical alternatives to the Raspberry Pi can lead to significant savings. Moreover, taking advantage of bulk purchasing offers discounts, further contributing to cost reduction.

Discussion & Recommendations

Facial Acquisition

One of the important considerations in facial acquisition is the Field of View (FOV) of the camera used to capture facial images. The FOV refers to the extent of the scene that the camera can capture and is typically measured in degrees. One common issue with a narrow FOV camera is that it may lose faces when they are too close to the lens. As faces approach the camera, they may go beyond the boundaries

of the camera's FOV, leading to incomplete or cropped facial images. This can be problematic for this project since the only way the software can find the forehead is if a full face is in frame. To address this limitation, one recommendation is to use a wide-angle lens for facial acquisition. A wide-angle lens has a larger FOV compared to standard or telephoto lenses. By using a wide-angle lens, the camera can capture a broader scene, ensuring that faces remain fully within the frame even when they are in close proximity to the lens.

Temperature Readings

Our final testing results yielded an average temperature error of ±0.201℃ when compared to an oral thermometer. This greatly outperformed the conservative goal of ±0.5℃. Our results are within industry standards which are generally between ±0.1℃ to ±0.3℃ depending on the thermometer cost.

Our data size was only 50 readings. This is a relatively small sample size and much more readings should be done in the future to verify accuracy. In addition to this Oral thermometers are not entirely effective at identifying fevers and a much more accurate baseline for comparison would be to use rectal temperatures which are the most accurate.

A trend that was noticed during testing was that darker skin colors tended to produce a lower infrared temperature reading vs the oral reading. This should not be the case as skin tone does not have a significant impact on infrared emissivity [15]. A more systematic test with more samples across a range of skin tones should be done to confirm this in the future.

Ease of Use

During the presentation a huge discrepancy between how different people interact with the A.T.T. was noticed. When given the instruction to “move closer” by the LCD some people moved very far, very quickly towards the robot which in the worst cases they got too close and the camera was no longer able to detect their face and as such could not get a reading. Other people moved slower in smaller increments which produced much better results. This is extremely undesirable as the goal of this solution is to quickly and easily take lots of peoples temperatures and vague instructions like “move closer” should not be left up to an individual's interpretation. One possible solution to this is to add a chin rest (similar to what is used in an optometrist's office) at an ideal distance so there is no movement required on the user end. This solution is less than ideal as one of the objectives of this project was to be completely non-contact so that disease transmission is limited.

Cost & Performance

The A.T.T has a significant cost in parts alone. The prototype was only feasible as many of the parts were already owned and did not have to be purchased. For the hypothetical situation where it is used in an airport at the security line a standard handheld digital infrared thermometer costs ~$100. This is about

50% of the cost of the A.T.T prototype which came to $155.99 Significant returns could be made in the cost of labor as an hourly wage does not need to be paid if the A.T.T is used.

Overall the temperature readings given by the A.T.T were very accurate. This was the most important objective of the project and was successfully achieved. Significant improvements to the useability and speed of the machine need to be made. Like mentioned in the “Ease of Use” section there was significant variance and confusion when people interacted with the machine which caused it to fail in worst case scenarios. The programming was all done in Python and left largely unoptimized which significantly slowed down its operation. This was compounded with the fact that the Raspberry Pi used only has a fraction of the computing power of a full sized computer. These issues all accumulated in a few ms delay in between servos movements which resulted in a very slow time for the robot to center on a face. Future upgrades to both hardware and software could greatly improve the speed of the A.T.T which would result in a much smoother operation for the end user.

References

[1] DNA Sequencing, *genome.gov,* [online] Available: https://www.genome.gov/genetics-glossary/DNA-Sequencing#:~:text=DNA%20sequencing%20refers%20to%20the,use%20to%20develop%20and%20operate [Accessed: june 20, 2023]

[2] Emerging technologies for the detection of viral infections, *Future virology,* [online] Available:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6956246/#:~:text=DNA%20sequencing%20has%20long%20been,foundational%20information%20for%20molecular%20diagnosis. [Accessed: June 20, 2023] s%20the%20inherent,short%20reads%20and%20HiFi%20reads. [Accessed: June 20, 2023]

[3] Sequencing 101: understanding accuracy in DNA sequencing, *PacBio,* [online] Available: https://www.pacb.com/blog/understanding-accuracy-in-dna-sequencing/#:~:text=Read%20accuracy%20

[4] MinION Comparison, *Oxford Nanopore Technologies,* [online] Available: https://nanoporetech.com/products/minion-comparison [Accessed: June, 20, 2023]

[5] P. H. A. of Canada, “Government of Canada,” Canada.ca, https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/symptoms/testing/diagnosing.html (accessed Jul. 28, 2023).

[6]“The evolution of HIV testing,” The Evolution of HIV Testing | AACC.org, https://www.aacc.org/cln/articles/2017/september/the-evolution-of-hiv-testing (accessed Jul. 28, 2023).

[7] “How scientists test for covid-19,” Fred Hutch, https://www.fredhutch.org/en/research/diseases/coronavirus/serology-testing.html#:~:text=Testing%20

for%20Infection%3A%20Rapid%20Antigen,to%20return%20an%20inaccurate%20result. (accessed Jul. 28, 2023).

[8] “Benefits of rapid covid testing,” Drug Testing, https://drugtestingsupplies.com/blog/benefits-of-rapid-covid-testing (accessed Jul. 28, 2023).

[9] “CFR - Code of Federal Regulations Title 21,” accessdata.fda.gov, https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/cfrsearch.cfm?fr=113.40#:~:text=(iv)%20A%20temperature%2Dindicating,shall%20not%20exceed%2017%20deg. (accessed Jun. 23, 2023).

[10] Center for Devices and Radiological Health, “Non-contact infrared thermometers,” U.S. Food and Drug Administration, https://www.fda.gov/medical-devices/general-hospital-devices-and-supplies/non-contact-infrared-thermometers (accessed Jun. 23, 2023).

[11] Omron, https://omronfs.omron.com/en_US/ecb/products/pdf/en_D6T_users_manual.pdf (accessed Jul. 28, 2023).

[12] “Face_recognition package” face_recognition package - Face Recognition 1.4.0 documentation, https://face-recognition.readthedocs.io/en/latest/face_recognition.html (accessed Jul. 28, 2023).

[13] “Accuflex Pro, Oral Digital Thermometer,” Physio Logic®, https://www.amgphysiologic.com/product/accuflex-pro-oral-digital-thermometer/ (accessed Jul. 28, 2023).

[14] “Human body temperature,” Wikipedia, https://en.wikipedia.org/wiki/Human_body_temperature#:~:text=July%202014)-,Hot,C%20(115.7%20%C2%B0F). (accessed Jul. 28, 2023).

[15] M. Charlton et al., “The effect of constitutive pigmentation on the measured emissivity of human skin,” PloS one, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7688144/#:~:text=These%20data%20show%20that%20skin,on%20an%20individual’s%20skin%20tone. (accessed Jul. 28, 2023).

[16] Ultrasonic ranging module HC - SR04 - SparkFun Electronics, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf (accessed Jul. 28, 2023).

Appendix

Table 6: Mouth Thermometer vs. Omron Sensor Readings

Real Temp - Mouth Thermometer (℃)	Measured Temp Omron Sensor (℃)
36.3	36.4
36.3	36.5
36.4	36.8
36.4	36
36.4	36.7
36.4	36.5
36.4	36.7
36.4	36.5
36.5	36.8
36.5	36.7
36.5	36.3
36.5	36.2
36.5	36.4
36.6	36.8
36.6	36.9
36.6	36.7
36.6	36.7
36.7	36.5
36.7	36.6
36.7	36.9
36.7	36.5
36.7	36.6
36.7	36.7
36.8	37
36.9	36.9

36.9	36.7
37	37.5
37	37.2
37	36.8
37	36.8
37	36.7
37.1	36.9
37.1	37.4
37.1	37.3
37.2	37.1
37.2	37.3
37.2	37.4
37.2	37.4
37.7	37.5
38	38.3
38.2	38.1
38.2	38
38.2	38.4
38.2	38.3
38.2	38.4
38.3	38.5
38.3	38
38.7	38.9
39	39.2
39	38.8

Code Listing

forehead_track.py

```
import cv2
import numpy as np
import face_recognition
import serial
import math
import time
from gpiozero import DistanceSensor
# Our imports
from servo_motors import servo_motors
from temp_sens import temp_sens
from LCD import LCDController

# Aspect ratio of camera
height, width = 320, 240

vid = cv2.VideoCapture(0) #Pi is 0
vid.set(3,height)
vid.set(4,width)
assert vid.isOpened()
face_locations = []

# Create a servo Object
motors = servo_motors()
print("init servos done")

# Create temp sens object
temp_sensor = temp_sens()
print("init temp sens done")

# Create LCD object
lcd = LCDController()
print("init lcd done")

# Initialize distance sensor
dist = DistanceSensor(echo=10, trigger=9, max_distance=300)

# Find forehead given eye coordinates
```

```
def findForehead(p1, p2):
        x1 = p1[1][0]
        x2 = p2[1][0]
        y1 = p1[1][1]
        y2 = p2[1][1]
        midpoint = [int(((x1 + x2) / 2)), int(((y1 + y2)/2))]
        # midpoint[1] = midpoint[1] - int(math.dist(p1[0], p2[0]) / 2)

        # Draw dot on forehead ***For Testing***
        #cv2.circle(frame, midpoint ,5,(0,0,255), -1)

        return midpoint

# array for averaging temp
temp_avg = []
temp_4_bool = False

while True:
        ret, frame = vid.read()

        # convert to HSV colorspace
        img = frame[:, :, ::-1]

        # get facial landmarks
        face_landmarks = face_recognition.face_landmarks(img, model='small')

        # Give instructions on LCD
        lcd.cursor(0, 0)
        lcd.display_text('Please face')
        lcd.cursor(0, 1)
        lcd.display_text('the sensor!')


        # Make sure a face is in frame
        if len(face_landmarks) > 0:
        # grab inner eye coordinates for both eyes
        landmark_dict = face_landmarks[0]
        left_eye = landmark_dict["left_eye"]
        right_eye = landmark_dict["right_eye"]

        # Add a dot to frame **FOR TESTING***
        # cv2.circle(frame, right_eye[1],2,(0,0,255), -1)
```

```
# cv2.circle(frame, left_eye[1],2,(0,0,255), -1)

forehead = findForehead(left_eye, right_eye)

# Move Servos
motors.moveServos(forehead)

# Print temp data to LCD
lcd.clear_screen()


dista = round(dist.distance * 100, 1)

# dont show anything > 100cm away
if (dista < 100):
lcd.clear_screen()
lcd.cursor(0, 1)
lcd.display_text('Dist ~ ' + str(round(dista, 1)) + 'cm')

temp, ptat = temp_sensor.readData()

# '''
# take temp reading when:
#         1. distance <= 31cm
#         2. 35 degrees > temp < 41 degree
# '''
if(dista <= 31):
#print(temp)
lcd.cursor(0, 0)
lcd.display_text('Hold Still!')
if(temp[4] > 35):
        if(temp[4] < 41):

        temp_4_bool = True
        forehead_temp = temp[4]
        temp_avg.append(forehead_temp)

        # wait for 5 readings to average
        if(len(temp_avg) >= 5):
        forehead_temp_final = np.mean(temp_avg)
        #round(forehead_temp_final, 2)
        lcd.cursor(0, 0)
```

```
                lcd.display_text('Temp ~ ' + str("%.1f" % forehead_temp_final) + 'C')
                time.sleep(5)
                temp_avg = []
                temp_4_bool = False

        if(temp[5] > 35 and temp_4_bool == False):
                if(temp[5] < 41 and temp_4_bool == False):
                forehead_temp = temp[5]
                temp_avg.append(forehead_temp)
                print(temp_avg)

                # wait for 5 readings to average
                if(len(temp_avg) >= 3):
                forehead_temp_final = np.mean(temp_avg)
                #round(forehead_temp_final, 2)
                lcd.cursor(0, 0)
                lcd.display_text('Temp ~ ' + str("%.1f" % forehead_temp_final) + 'C')
                time.sleep(5)
                temp_avg = []

        else:
        lcd.cursor(0, 0)
        lcd.display_text('Move closer!')



        # show frame ** *FOR TESTING ***
        # cv2.imshow('frame', frame)
        #press q to stop running
        if cv2.waitKey(25) & 0xFF == ord('q'):
        break

vid.release()
cv2.destroy()
```

LCD.py

```
import Adafruit_CharLCD as LCDC
import time
from time import sleep
#from gpiozero import DistanceSensor
```

```
class LCDController:
        def __init__(self):
        # Define LCD pin connections
        self.lcd_rs = 22  # Raspberry Pi GPIO pin connected to LCD RS
        self.lcd_en = 17  # Raspberry Pi GPIO pin connected to LCD EN
        self.lcd_d4 = 25  # Raspberry Pi GPIO pin connected to LCD D4
        self.lcd_d5 = 24  # Raspberry Pi GPIO pin connected to LCD D5
        self.lcd_d6 = 23  # Raspberry Pi GPIO pin connected to LCD D6
        self.lcd_d7 = 18  # Raspberry Pi GPIO pin connected to LCD D7

        # Define LCD column and row size
        self.lcd_columns = 16
        self.lcd_rows = 2

        # Initialize the LCD module
        self.lcd = LCDC.Adafruit_CharLCD(
        self.lcd_rs, self.lcd_en, self.lcd_d4, self.lcd_d5, self.lcd_d6, self.lcd_d7,
        self.lcd_columns, self.lcd_rows
        )

        def clear_screen(self):
        self.lcd.clear()

        def display_text(self, text):
        self.lcd.message(text)

        def cursor(self, col, row):
        self.lcd.set_cursor(col, row)
```

servo_motors.py

```
import RPi.GPIO as GPIO
import time
from RaspberryMotors.motors import servos
import pigpio

#Class for moving servo motors
class servo_motors:
```

```
xservo = 11 #p in # of xservo
yservo = 5 #pin # of y servo
centerx = 160
centery = 120
duty_x = 1500
duty_y = 1900
pwm = pigpio.pi()


duty = 12

box_size = 20 #range of coordinates it tries to center within

def __init__(self):

# Center servos upon start
self.pwm.set_mode(self.xservo, pigpio.OUTPUT)
self.pwm.set_mode(self.yservo, pigpio.OUTPUT)

self.pwm.set_PWM_frequency(self.xservo, 50)
self.pwm.set_PWM_frequency(self.yservo, 50)

self.pwm.set_servo_pulsewidth(self.xservo, self.duty_x)
self.pwm.set_servo_pulsewidth(self.yservo, self.duty_y)


# Cleanup PWM pins
def cleanup(self):
self.pwm.set_PWM_dutycycle(self.xservo, 0 )
self.pwm.set_PWM_frequency(self.yservo, 0 )


def moveServos(self, cord):
# Calculate difference between the center of frame and current coordinates
x_diff = cord[0] - self.centerx
y_diff = cord[1] - self.centery

# Check if we are outside margin of error on x plane
if abs(x_diff) > self.box_size:
# move left or right depending on servo location and forehead coordinates
# also check we are not outside range of servos
if x_diff > 0:
```

```
        self.duty_x -= self.duty
        if self.duty_x < 500:
                self.duty_x = 500
        elif x_diff < self.box_size:
        self.duty_x += self.duty
        if self.duty_x > 2500:
                self.duty_x = 2500

        # Check if we are outside margin of error on y plane
        if abs(y_diff) > self.box_size:
        # move up or down depending on servo location and forehead coordinates
        # also check we are not outside range of servos
        if y_diff > 0:
        self.duty_y -= self.duty
        if self.duty_y < 500:
                self.duty_y = 500
        elif y_diff < 0:
        self.duty_y += self.duty
        if self.duty_y > 2500:
                self.duty_y = 2500
        else:
        return

        # Move servos to new coordinates
        self.pwm.set_mode(self.xservo, pigpio.OUTPUT)
        self.pwm.set_mode(self.yservo, pigpio.OUTPUT)

        self.pwm.set_PWM_frequency(self.xservo, 50)
        self.pwm.set_PWM_frequency(self.yservo, 50)

        self.pwm.set_servo_pulsewidth(self.xservo, self.duty_x)
        self.pwm.set_servo_pulsewidth(self.yservo, self.duty_y)


if __name__ == '__main__':
        xMotor = servo_motors()
        xMotor.cleanup()
```

temp_sens.py

```
import smbus
import sys
import getopt
import time
import pigpio

"""
        The sensor has eight pixels

        [       0       |       1       |       2       |       3       | ...OTHER TEMP PIXELS... |
18      ]
        [ PTAT Lo  | PTAT Hi   |   P0 Lo   |   P0 Hi   | ...OTHER TEMP PIXELS... |   PEC       ]

        SDA: (temp blue) pin 3  --THESE PINS ARE I2c channel 1
        SCL: (temp yellow) pin 5
        rn have a 10k pullup resistor
        To calculate a temperature value for a pixel we take first byte + second byte * 256 / divisor

"""

class temp_sens:
  pi = pigpio.pi()
  divisor = 15.6


  def __init__(self, address=0x0a):
        # open i2c connection
        try:
        self.handle = self.pi.i2c_open(1, 0x0a)
        except AttributeError:
        print('run "sudo pigpiod" please')
        raise


  def readData(self):
        self.pi.i2c_write_device(self.handle, [0x4c])

        # Data is a tupule with first item being size of byte array (19 heres)
        data = self.pi.i2c_read_device(self.handle, 19) #19

        # get temp data array
        self.temp = []
```

```
            # ptat value
            self.ptat = (data[1][1] *256 + data[1][0]) / self.divisor


            # Calculating the 8 temperature pixels
            self.temp.append((data[1][3] * 256 + data[1][2]) / self.divisor)
            self.temp.append((data[1][5] * 256 + data[1][4]) / self.divisor)
            self.temp.append((data[1][7] * 256 + data[1][6]) / self.divisor)
            self.temp.append((data[1][9] * 256 + data[1][8]) / self.divisor)
            self.temp.append((data[1][11] * 256 + data[1][10]) / self.divisor)
            self.temp.append((data[1][13] * 256 + data[1][12]) / self.divisor)
            self.temp.append((data[1][15] * 256 + data[1][14]) / self.divisor)
            self.temp.append((data[1][17] * 256 + data[1][16]) / self.divisor)

            return self.temp, self.ptat


if __name__ == '__main__':
  sensor = temp_sens()

  while(True):
            temp, ptat = sensor.readData()
            print(temp)
            print(f"ptat: {ptat}")
            time.sleep(1)
```