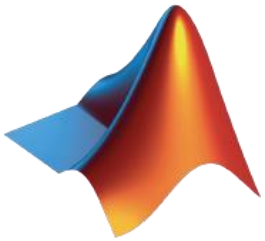

FACE DETECTION AND HUMAN TRACKING ROBOT

STUDENT NAME: AQEEL JARDINE

STREAM: MECHANICAL ENGINEERING

PROJECT NUMBER: 214

A detailed optimization project was submitted for the *MATLAB and Simulink Challenge Project*s.

February 12, 2024

EXECUTIVE SUMMARY

Interaction between humans and robots is essential in many computer vision applications, including surveillance, smart home security applications, activity recognition, and vehicle safety. The role of a robot is to assist people with work in an effective manner. Communicating and engaging with people must come naturally to the robot. Essentially, what this kind of robot needs is a human face-tracking system as its primary visual system. Face detection is one way to improve human-robot interaction in robotics. The intricacy of the system and the vast range of postures make it challenging to automatically detect and track persons using a sensor or algorithm. Ultimately, the issue is one of thorough optimization.

Thus, an Android smartphone and Deep Learning technology was used to create an affordable, user-friendly, and live autonomous human monitoring robot. A face detection model on an Android smartphone tracked people, while the Arduino Uno Controller handles the real-time control. The Simulink Deep Learning and Computer Vision Toolboxes was used to create the face detection model. The Android and Arduino devices were interfaced and deployed with Simulink models using hardware support packages. The face detection algorithm on the Android device needed to be able to process facial recognition data and provide the necessary data to the Arduino to command the robot's motions. Therefore, a robot for face identification and tracking based on deep learning was created.

To achieve the face detection Android-Arduino robot, the face detection algorithm had to be developed such that the valuable position of the detected face and image size can be extracted and used in controlling the required responses for the camera to keep the detected face at a predefined distance. The Arduino Uno robot had to tested for performance and operational errors in the hardware using C++ code before deploying a Simulink model that can read the input response signal and determine which motor pin signal configuration to move the robot in a specific direction.

The Arduino Uno robot could only communicate through the serial port. Thus, two ways for communication were established. A direct Serial connection between Android device and Arduino board and a TCP/IP connection between two Android devices while one of them is connected to the Arduino board via the Serial port. A button control application was developed for both communication lines. It was determined that the TCP/IP connection is best used for the button control application and the direct serial connection is best used for the face detection application because the face detection application requires the Android device to be attached to the Arduino Uno robot. Thus, using the direct serial connection between the Android device and Arduino board and the two Simulink models for face detection and response determination and Arduino Motor Control, a Face Detection Android-Arduino Robot was created.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Problem Description

Human-robot interaction is crucial in many computer vision applications, such as activity identification, automobile safety, smart home security applications, and surveillance. A robot's job is to be an efficient assistant to aid humans with their tasks. The robot must be proficient in communicating with and interacting with people. In this situation, the main requirement for the vision system in this type of robot is a human face-tracking system. Robotic human-robot interaction can be improved via face detection. Detecting and tracking humans automatically using a sensor or algorithm is a difficult problem due to the wide variety of positions, and system complexity. It is, ultimately, a detailed optimisation problem.

Therefore, a low-cost, user-friendly, and real-time autonomous human tracking robot using Deep Learning technology and an Android mobile is to be put into action. The human tracking is done by a face detection model done on an Android device while the real-time control is done by an Arduino Uno Controller. The face detection model should be designed using the Computer Vision Toolbox and Deep Learning Toolbox on Simulink. The hardware support packages are used to interface and deploy the Android and Arduino devices with Simulink models. The Android device should be able to run the face detection algorithm and send the required information to the Arduino to control the movements of the robot. Therefore, a deep learning-based face detection and tracking Robot will be developed.

Objectives

- Build a Simulink model that can detect and track a human face using an Android device.
- Build a Simulink model that can control the Arduino Uno Robot.
- Develop a communication line between the Android device and the Arduino robot.
- Build a robot that can move towards the detected human face and stop at a predefined distance.

Methodology

- Install the relevant support packages and software tools to develop the Simulink models for the face detection algorithm and Arduino uno control.
- Develop the face detection model for the Android Device using Simulink support packages.
- Track the detected face using MATLAB code and retrieve the position of the face on the image.
- Use the position of the face to determine the best possible response for the Arduino robot.
- Develop a Simulink model that communicates the response as an input signal and outputs signals to the motors to move the Arduino Uno robot in a specific direction.

2. FACE DETECTION APPLICATION ON ANDROID

The Simulink model that detects a face using an Android device's camera was obtained from the example that was developed in 2019. This model was then modified to account for the change in hardware (the use of different Android devices). The figure below shows the Simulink model.

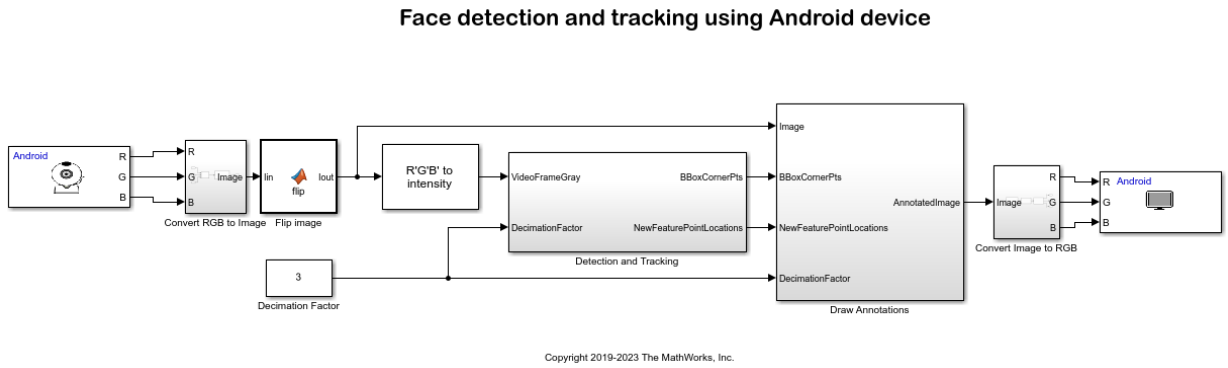

Figure 1: Face Detection Simulink Model [1]

An application on an Android device is built through Simulink, where the application contains the Simulink face detection model. This model is deployed, built, and sent to the Android device using the Simulink software and hardware drivers that allow the communication between Simulink and Android software.

This example uses the Android device's camera to get a video frame to detect and track the face on that frame, draw a box around the face and then display the image with the box being the detected face.

Detection and Tracking Block

The model uses the vision.CascadeObjectDetector object to detect the location of the face in a video frame. The Viola-Jones detection algorithm is a machine-learning technique to detect objects but was primarily used to detect faces [1]. This algorithm is used in the cascade object detector to detect faces on a single frame. However, a video contains multiple frames and computing the location of a face for each frame is computationally expensive. Therefore, the Kanade-Lucas-Tomasi (KLT) algorithm is used to detect the face over time and through multiple frames. This algorithm utilises a set of feature points across video frames. These feature points are reliably tracked on each frame[1].

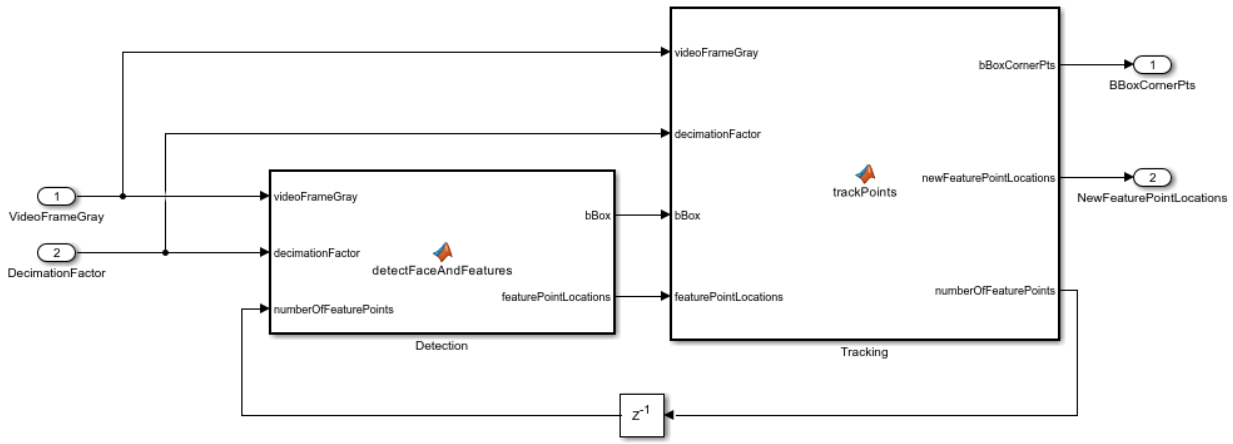

Figure 2: Detection and Tracking Simulink Block [1]

These feature points can allow the use of the vision.PointTracker System object to track the points. The point tracker attempts to determine the location of a point in a frame based on the corresponding location of a point in the previous frame [1]. Thereafter, the estimateGeometricTransform2D function is used to determine the rotation, translation and scaling between the new points and the old ones. This is applied to a bounding box that is created when the cascade object detector is used [1].

Draw Annotations Block

This Simulink block draws the bounding box from the corner points of the detected face. It also draws the feature points that show as plus signs inside the bounding box of the detected face. The figure below shows the draw annotations block.

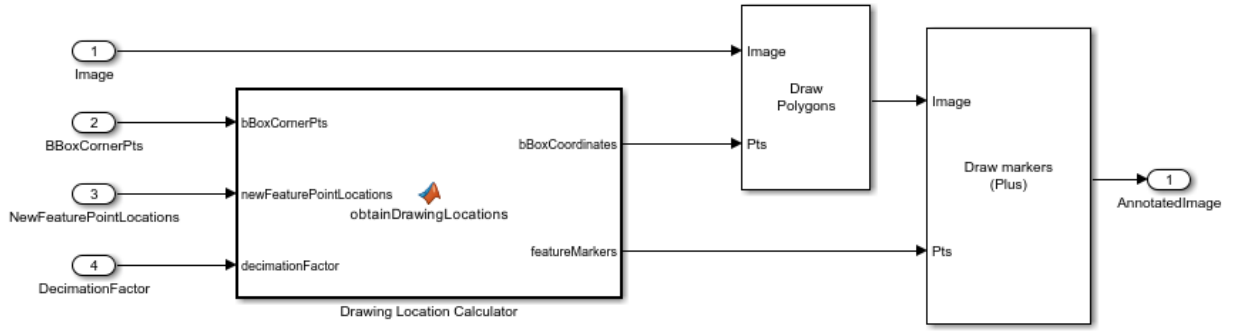

Figure 3: Draw Annotations Block [1]

Face Detection and Reference Simulink Model

The face detection and tracking algorithm is used as a basis for the face detection and reference Simulink model shown in the figure below. This application can display the image size of the video and the coordinates of the corner points of the bounding box.

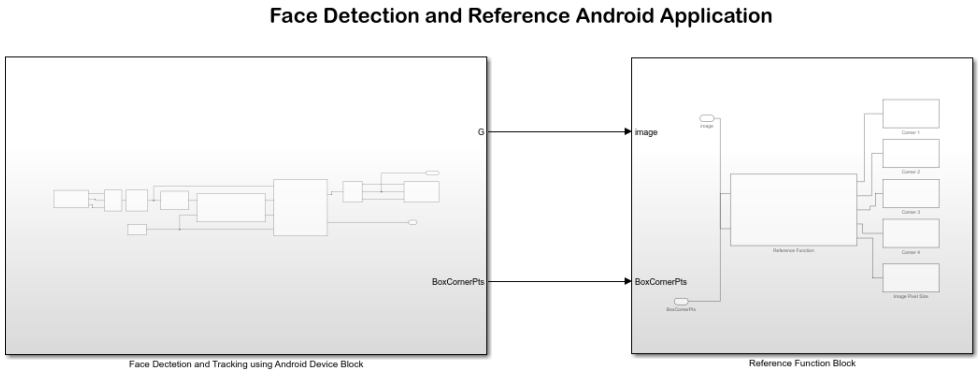

Figure 4: Face Detection and Reference Simulink Model

The Reference Function is used to extract the coordinates of the corner points of the bounding box using the BoxCornerPts matrix and determine the pixel image size of the video frame using one of the image colour matrices. The block is shown in the figure below.

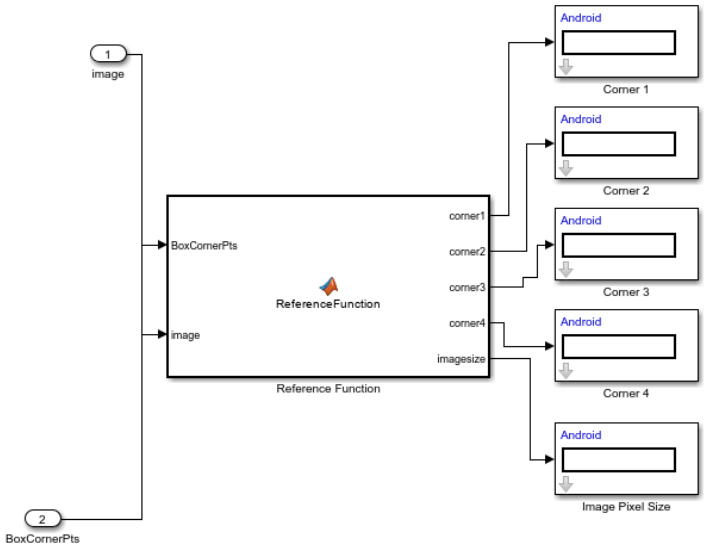

Figure 5: Reference Function Block

A screenshot of the application is shown in the figure below. The figure shows the coordinates of the bounding box with its feature points and the pixel size of the image. The corners are labelled from the top left to the bottom left going in a clockwise direction.

Figure 6: Screenshot of the Face Detection Application in use with corner points and image pixel size.

Face Detection and Response Simulink Model

Using the reference Simulink model, the origin of the coordinates used in the corner points can be determined and is determined to be the top left corner of the video frame. The bottom right corner of the video frame is then the maximum pixel size of the video frame. This is depicted in the figure below where the image pixel size was 640x480.

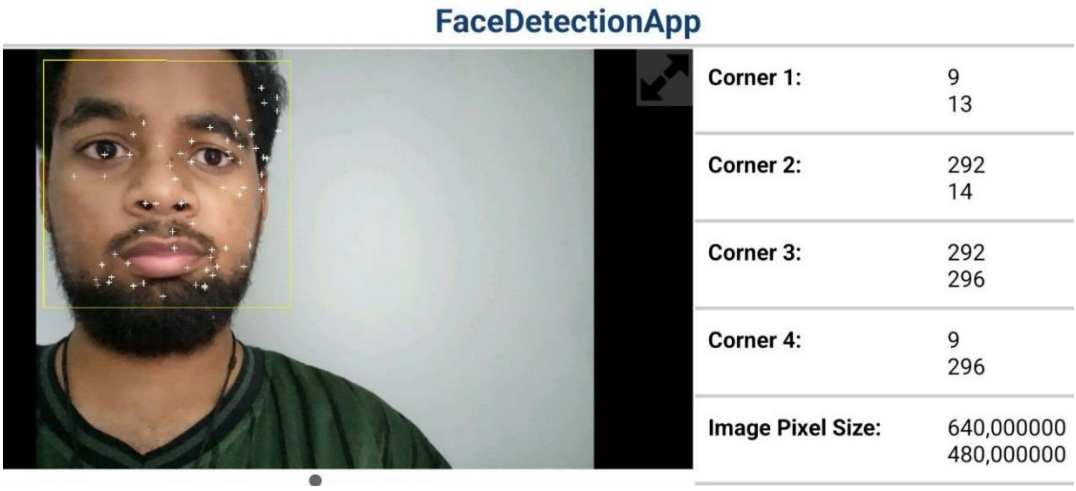

Figure 7: Screenshot of the Application showing the origin being the top left corner.

A controlling function is used to determine what response is required to move the camera in a specific direction to keep the detected face at a predefined distance and position from the camera. Using the corner points and image pixel size, the mid-points of the bounding box and video frame are used to determine if the camera should rotate left or right. If the x component of the midpoint of the bounding box (X1) is more

than the x component of the midpoint of the video frame (X2), then the camera should rotate to the right and if X1 is less than X2 then the camera should rotate left.

The area of the bounding box is used to determine the distance between the detected face and the camera. It was determined that the camera should be placed such that the area of the bounding box (Ab) should be about a third of the image pixel size. If Ab was more than two-thirds of the image size, the camera should move back. If Ab was less than a third of the image size and the difference between X1 and X2 was in a specific range, then the camera should move forward.

Limitations

The application is dependent on the type of Android device used and the image size of the video frame. The more quality type of video frame the better the accuracy of the face detection and tracking but it reduces the processing speed. This means that the device would require more processing power to detect a face if the camera is constantly moving. Thus, this limitation would affect the control of the Arduino robot. Therefore, there needs to be a balance between accurate face detection and less processing power. Thus, it was determined that the best pixel-size video camera for a Samsung A31+ Android device was a back camera with a pixel size of 384x384 pixels. Light also plays a huge role in face detection. Therefore, a well-lit area is used for face detection.

Final Simulink Model

The figure below shows the final Simulink model for the Android device face detection, tracking and response algorithm.

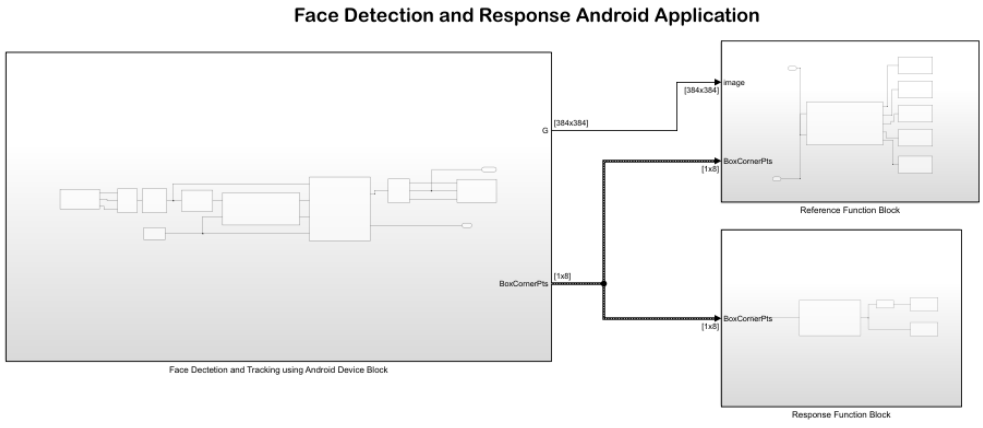

Figure 8: Face Detection and Response Application Model

3. CONTROL OF ARDUINO UNO VEHICLE

Hardware Components Details and Condition

The Arduino vehicle uses an Arduino Uno board as the main controller for all the sensors and actuators that can be used to control the vehicle's motion. The Arduino vehicle consists of the following components:

- 4 Gear Motors
- 4 Wheel Tyres
- 4 Motor Support Pieces
- Two 100 x 213 x 5 mm Plates
- 1 L298N Motor Driver Board
- 1 Arduino Uno R3 Board
- 1 Sensor Extension Board V5
- 1 Holder
- 1 SG90 Servo Motor
- 1 HC-SR04 Ultrasonic Sensor- 4 Pin Module
- 3 TCRT5000 Line Tracing Module
- 1 Infrared Receiving Sensor Module
- 1 18650 Battery Box
- 2 18650 Batteries
- 1 Bluetooth Module

The components of the vehicle were obtained and bought for a total price of ZAR1500 [2]. A figure of the assembled vehicle with all its components is shown below.

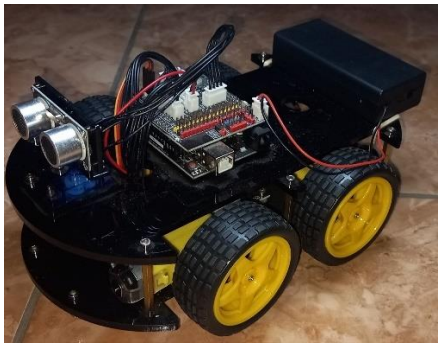

Figure 9: Arduino Uno Robot

The Arduino vehicle can be programmed through Simulink. The vehicle was programmed with various Arduino C++ codes to test the performance of the vehicle's components assembled vehicle. The vehicle can follow a line using the Follow Line Code. The vehicle can avoid obstacles in its path by using the Obstacle Avoidance Code. Thus, the vehicle is deemed acceptable to use and is fully operational.

The Arduino Uno is required to run using a Simulink model and run independently and not with the use of the code running on the computer in the background. Therefore, an Android device is used to control the motion of the vehicle. The Android device should send a specific response signal to the Arduino robot to move in a specific direction.

Due to the limitations of the Simulink support package for the Arduino Hardware, the only communication signal sent to the Arduino Uno robot is via the Serial Connection, where a USB cable is required to be connected to the Arduino Uno board to receive the input response signal. Therefore, a general Simulink model is deployed on the Arduino Uno board that can run independently from the computer.

Arduino Simulink Model

The Simulink model used to control the Arduino cannot run independently as it requires a signal input from either the Android device or computer to move the robot. In this case, the input signal would be the response required. The input response signals and descriptions are shown in the table below.

Table 1:Input Response Signals and their Descriptions

Description	Input Response Signal
Stop	0
Forward	1
Back	2
Right	3
Left	4

The speed can also be changed and vary depending on the setup of the Arduino vehicle. The Simulink model used is shown in the figure below.

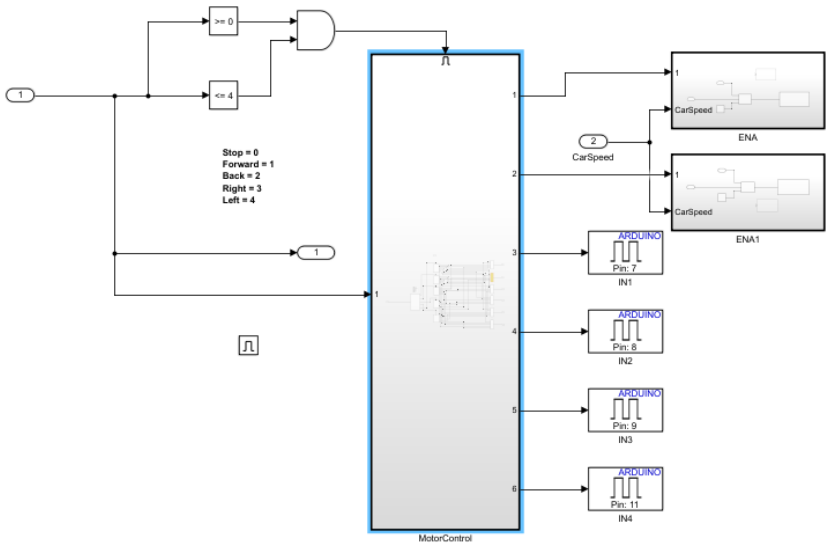

Figure 10: Arduino Simulink Model

For the MotorControl block, the response is used in a switch-case block, and this is used to determine which direction to move the vehicle. The Motor Control block is shown in the figure below.

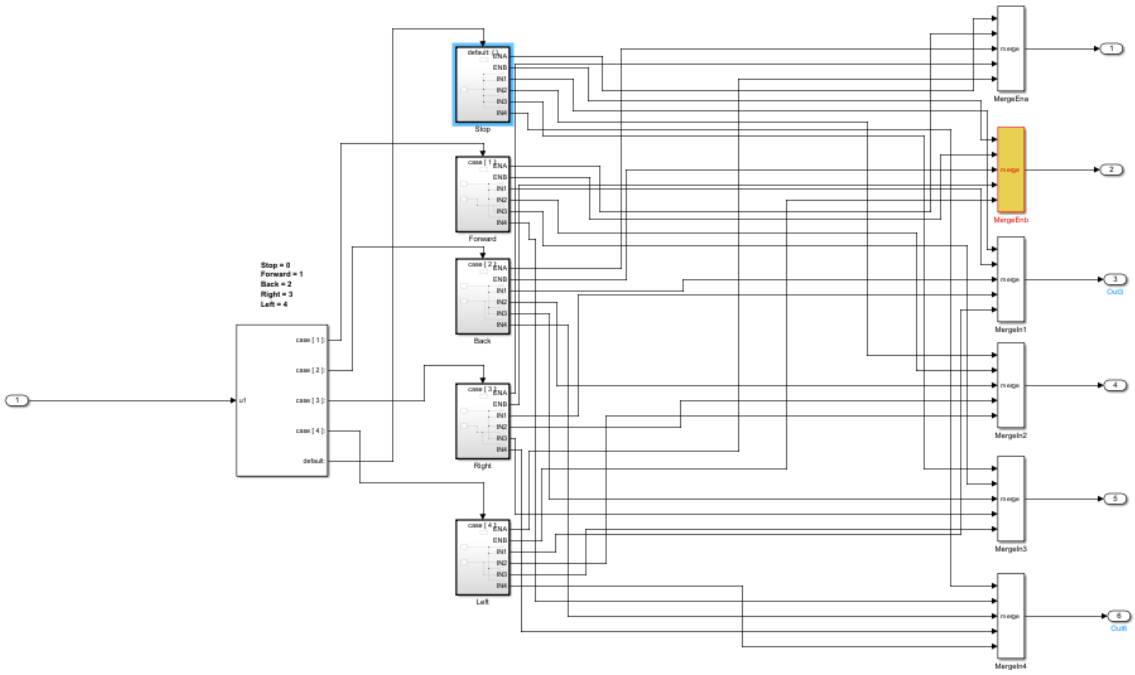

Figure 11: MotorControl Block model with Switch-Case for different responses.

Each response has a specific configuration of signals for each digital pin on the Robot. The configuration of each response is shown in the table below.

Table 2: Response Configuration of pin signals

ENA (D6 pin)	ENB (D5 pin)	IN1(D7 pin)	IN2(D8 pin)	IN3(D9 pin)	IN4(D11 pin)	Response
HIGH	HIGH	HIGH	LOW	LOW	HIGH	Forward
HIGH	HIGH	LOW	HIGH	HIGH	LOW	Back
HIGH	HIGH	LOW	HIGH	LOW	HIGH	Left
HIGH	HIGH	HIGH	LOW	HIGH	LOW	Right
LOW	LOW	LOW	LOW	LOW	LOW	Stop

Using the serial connection block for the input response signal, the final Simulink model used to deploy and build on the Arduino Uno board is shown in the figure below.

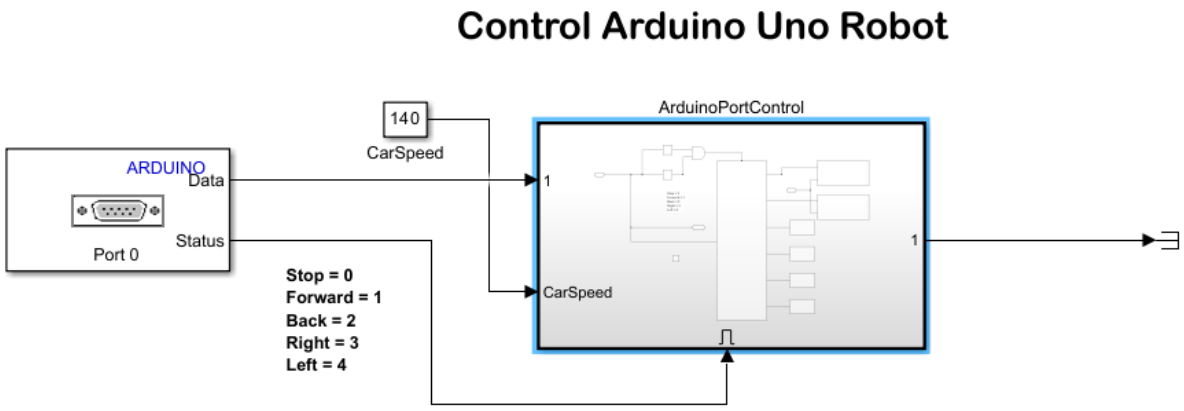

Figure 12: Control Arduino Uno Robot Simulink Model including the Serial port connection block.

4. COMMUNICATION BETWEEN ANDROID DEVICE AND ARDUINO ROBOT

Using the Simulink model for the Arduino Uno board, two ways were developed to communicate the information from the Android device to the Arduino Uno board.

i. A direct Serial Connection between the Android Device and the Arduino Uno board
ii. A TCP/IP (Transmission Control Protocol/Internet Protocol) connection between two Android devices, where one is connected to the Arduino Uno Robot.

Both ways use a ButtonControl function where the input signal comes from an Android Device button displayed on the applications. The function sends the required response signal corresponding to the button pressed. The function also ensures that when two buttons are pressed at the same time the robot stops until one of the buttons is unpressed. The code for the ButtonControl function is shown in the figure below.

```
%Function to control Arduino robot using the android device
%Robot Controls:
%response = 0 --> Stop
%response = 1 --> Forward
%response = 2 --> Reverse
%response = 3 --> Right
%response = 4 --> Left

function response = ButtonControl(stop,forward,back,right,left)
Buttons=[stop,forward,back,right,left];
if sum(Buttons)==1
    if stop == 1
        response = 0;
    elseif forward==1
        response =1;
    elseif back ==1
        response=2;
    elseif right ==1
        response=3;
    elseif left==1
        response=4;
    else
        response=0;
    end
else
    response=0;
end
```

Figure 13: Screenshot of code for the ButtonControl function.

Direct Serial Connection

A wired connection between the Android device and the Arduino Uno robot using a USB to type-c USB converter and a USB cable. The figure below shows the connection layout of the Arduino robot and Android Device.

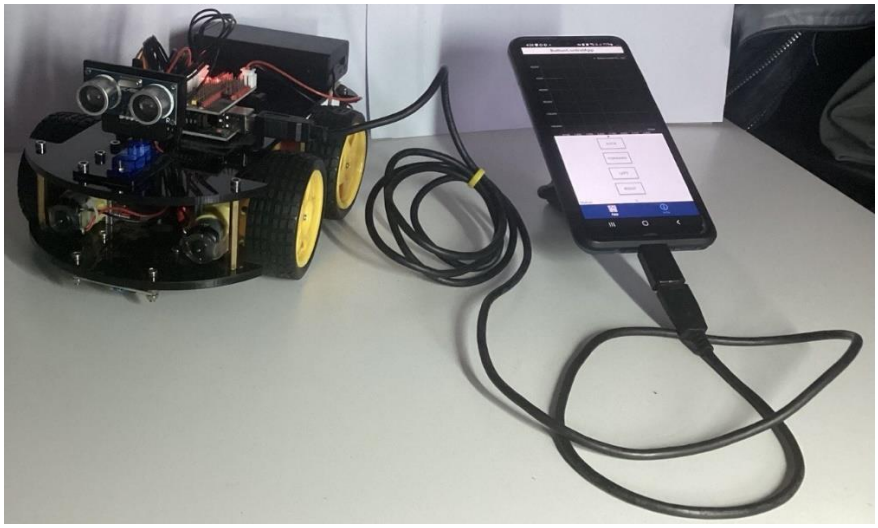

Figure 14: Image of Direct Serial Connection Layout

A Simulink model deployed on the Android device was developed and shown in the figure below. The model shows the Android serial send block that transfers the input response to the Arduino board directly.

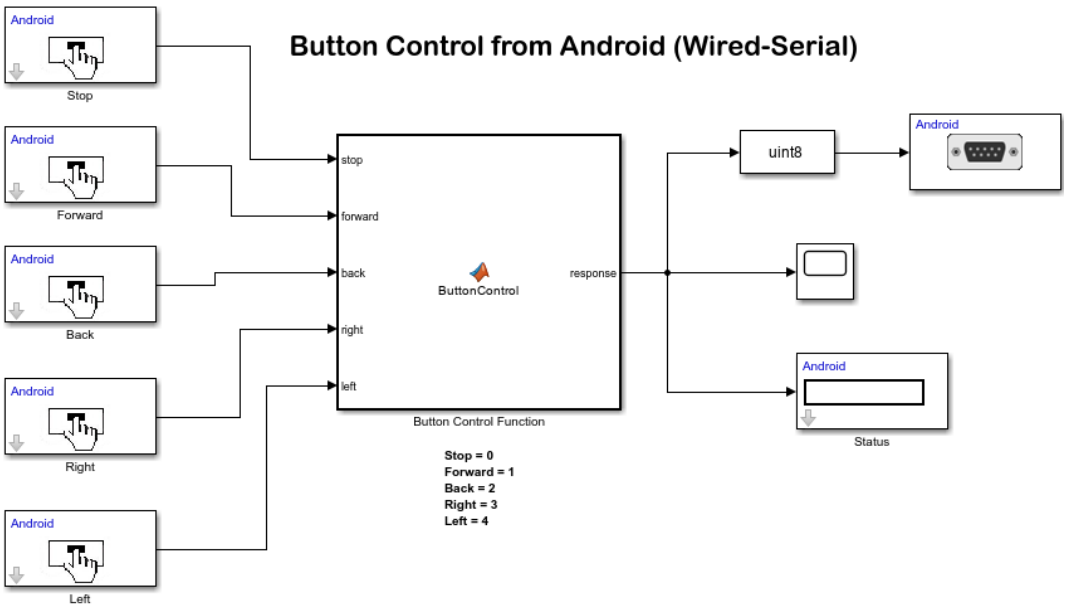

Figure 15:Direct Serial Connection Simulink Model

TCP/IP Connection

A TCP/IP connection is established when an android device (server) sends out an input response signal to a specific port using a specific IP address then a second android device (client), which is connected to the Arduino Uno board via a serial connection, receives the signal when using the same IP address and port. This connection layout was developed due to the robot having no wireless connection components that can be connected to one Android device directly. Therefore, the second Android device acts as a wireless component for the Arduino robot. This connection works if both Android devices are connected to the same Wi-Fi network. The figure below shows the connection layout of the Arduino Robot and the Android devices.

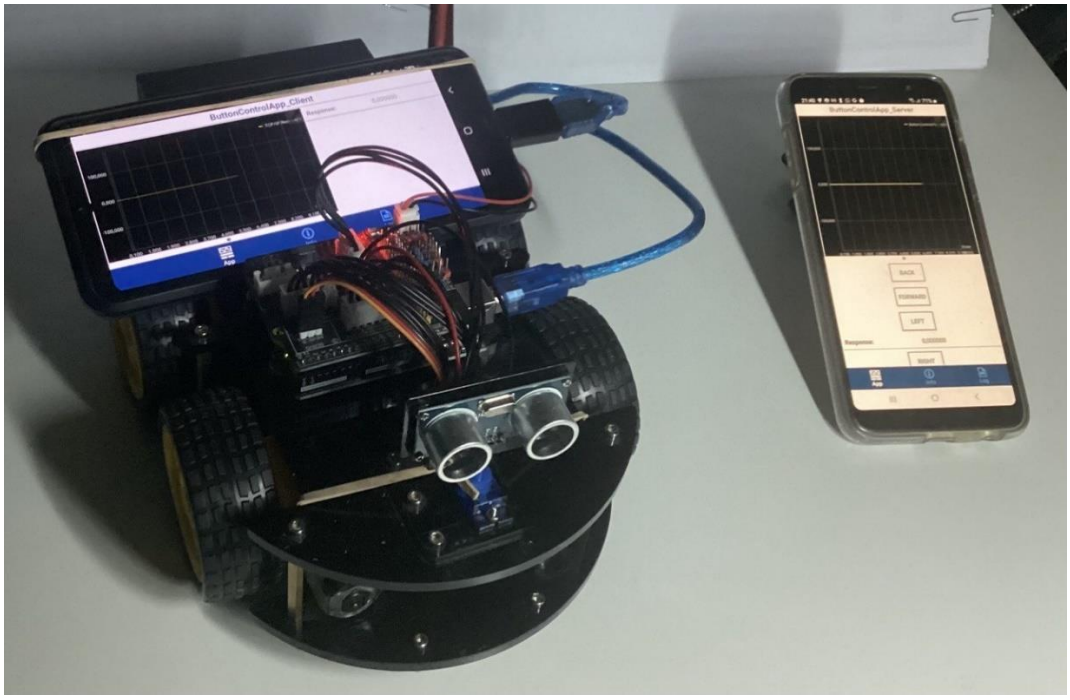

Figure 16: Image of TCP/IP Connection layout between Android devices.

The Simulink model used for the first Android device (server) which is like the model used for the direct serial connection is shown in the figure below.

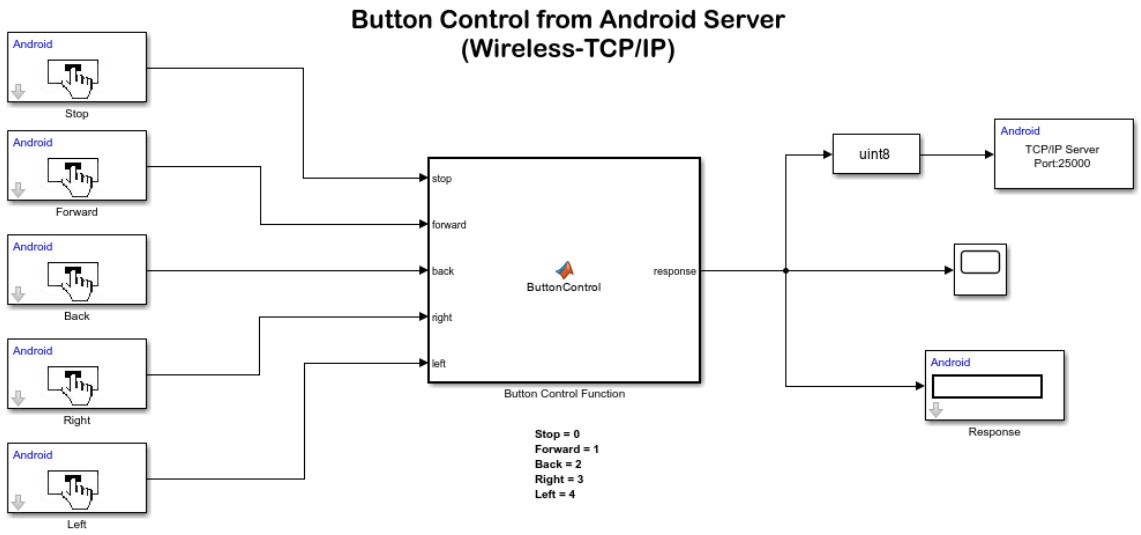

Figure 17: First Android device (server) Simulink model.

The Simulink model used for the second Android device (client) which is a direct connection between the Android TCP/IP receive block and the Android serial send block is shown in the figure below.

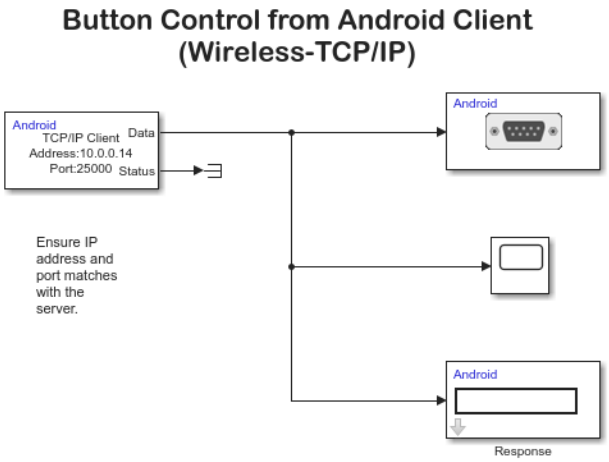

Figure 18: Second Android Device (client) Simulink model.

Best Connection Type

Both connection types can respond with minimal lagging. However, each has its advantages and disadvantages. Thus, the optimal connection type would depend on the application and environment the robot would be in. The optimal connection is decided for the ButtonControl application.

For the serial connection, there is a direct link between the Android and Arduino. This minimizes the interference that could occur between devices. However, it would limit the range the robot would have to move due to the small length of the cable.

For the TCP/IP connection, the Arduino can move in a larger range of motion than the serial connection. However, both devices are required to be on the same Wi-Fi network for the connection to work. If using a busy Wi-Fi network, the connection would severely deteriorate, and this would cause longer time response lags to occur. Therefore, it is suggested to use a less busy and private network for Android devices.

The optimal connection for the ButtonControl application was determined to be the TCP/IP connection as this connection allows a remote-controlled robot to be in action. However, for the face detection application, a direct serial connection would be more favourable. This is due to the reliable connection between the Android device and the Arduino robot. Also, the Android device would be fixed to the Arduino robot which does not require a large range of motion between the Android device and the Arduino robot.

5. ARDUINO ROBOT WITH FACE DETECTION AND TRACKING

Now that the main programming for the Android device and Arduino robot is complete and the communication type is established. The Android device would need to be attached to the Arduino Robot such that the responses required from the face detection follow the same direction as the Arduino robot.

It was determined that a selfie stick attached to the robot would be used to hold the Android phone in a specific position with some height to allow a face to be detected and control the Arduino robot from the ground. The selfie stick and phone create a lever weight that is accounted for by adding a ballast that is made from a container of coins and a small power bank. The container of coins would be used to modify the weight of the ballast if need be.

The selfie stick and phone lever are inclined and facing forward to allow the vehicle to stabilise when turning. The ballast allows the centre of mass to be closer to the middle of the vehicle but still allows some weight to be closer to the front. This allows the back wheels to have lower friction than the front wheels which allows the vehicle to turn at a more controlled pace. The layout of the Android-Arduino Uno Robot is shown in the figure below.

Figure 19: Image of the Face Detection Android-Arduino Uno Robot

Final Android Simulink Model

The final Simulink model deployed in the Android device is shown in the figure below. The model can detect and track a human face capture on the Android device and use the position of the face to determine the required input response signal and send the response signal to a serial port connection with the Arduino robot.

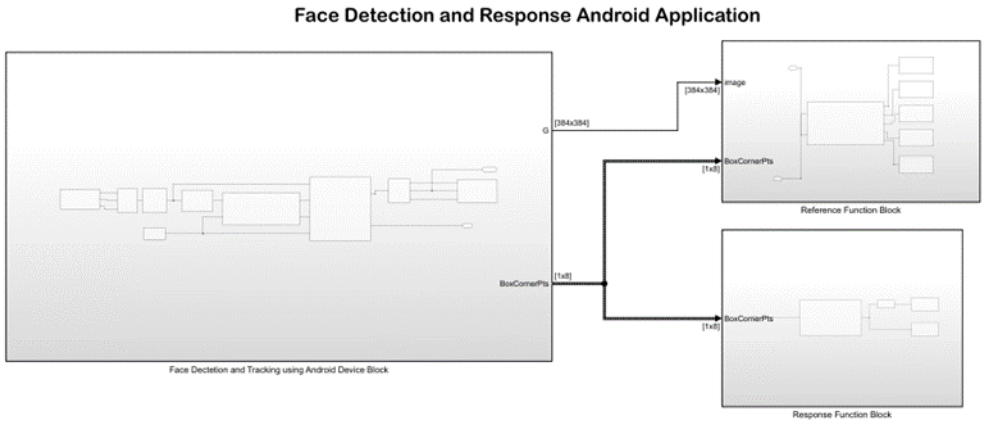

Figure 20: Final Simulink Model for the Android Device.

Final Arduino Simulink Model

The final Simulink Model deployed in the Arduino robot is shown in the figure below. The model can receive the input response signal sent from the Android device and the model can determine which pin signal configuration matches the response signal and sends the pin signal configuration to the digital pins that contain the motors for moving the robot in the required direction.

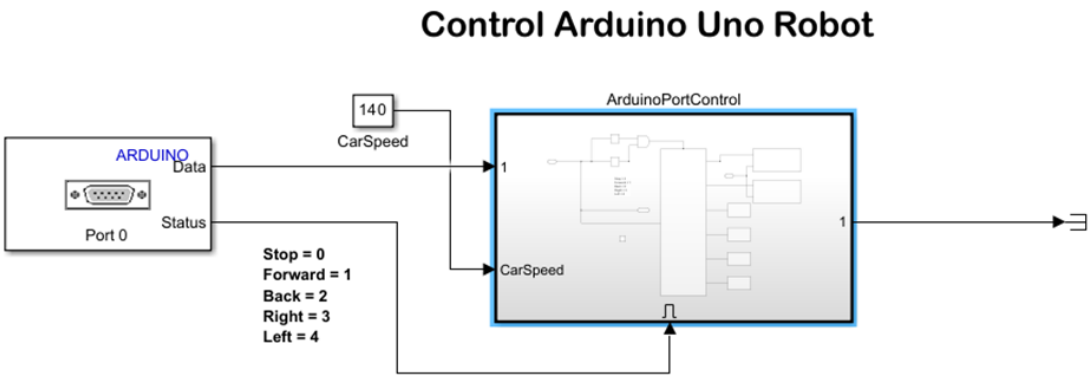

Figure 21: Final Simulink Model for Arduino robot.

.

6. CONCLUSION AND RECOMMENDATIONS

The solution was able to successfully track and detect a face using an Android device. The solution was able to extract the position of the detected face on the video frame and determine and send the required response on an Android device to the Arduino Uno robot that would be able to move in the required direction. Other means of connection were also successfully developed for better user-defined interface applications.

Due to the Arduino Uno robot being small, the mounting for the Android device was then long to allow face detection to occur closer to the human's face. However, the robot was also too short lengthwise, and this caused major instabilities when the robot began to move and started to stop due to inertia acting on a small device with a long lever arm. Thus, it is recommended that a bigger and longer Arduino controller be used to reduce the instability of the Android device's mounting.

The pixel size was constrained due to the Android device's processing power. This would reduce the quality of video frames that are captured for face detection. The quality of the video frames has a strong positive correlation with the accuracy of face detection. Thus, an Android device that can handle more processing power would be able to capture and use high-quality video frames for face detection without any lags or slow responses.

The use of better mounting equipment or resources can improve the control of the response. For example, instead of a selfie stick, a thin and small pipe with a motor and phone holder attached on its end would allow the phone to rotate and follow the detected face while the robot remains still until the human stops for a period then the robot would move in the direction of the human while the phone remains on the detected face. This would improve the instabilities that are found in the current design.

.

7. DEMO VIDEOS AND GITHUB REPOSITORY

The Simulink models and demo video can be found on the GitHub repository page. Here is the URL link: https://github.com/AqeelJar/Android-Arduino_Face_Detection_Robot.

8. REFERENCES

[1] 'Detect and Track Face on Android Device - MATLAB & Simulink'. Accessed: Feb. 10, 2024. [Online]. Available: https://www.mathworks.com/help/supportpkg/android/ref/detect-and-track-face-on-an-android-device.html

[2] 'Arduino Smart Robot Car Kit | Shop Today. Get it Tomorrow! | takealot.com'. Accessed: Feb. 10, 2024. [Online]. Available: https://www.takealot.com/arduino-smart-robot-car-kit/PLID72722636