Palestine Polytechnic University

College of Information Technology and Computer Engineering

Department of Computer System Engineering

Gestures Controlled Robotics (GCR)

Project Team:

Abdelkhalik Aljuneidi

Fares AbuIram

Supervisors:

Dr. Alaa Halawani

Dr. Zein Salah

December 2021

Acknowledgment

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project.

This project gave us valuable experience in embedded systems and machine learning. For that, we want to thank all the people who helped us through the project.

We would like to express our gratitude to our graduation project supervisors Dr. Alaa Halawani, and Dr. Zein Salah for their guidance, support, and encouragement throughout the project.

Furthermore, we have to show our gratitude to Eng. Wael Takrouri for answering our questions and providing us with valuable suggestions and tips that helped us in our project, especially in the hardware problems.

Moreover, we must thank our families for their generous encouragement, and continuous support throughout our life. For our friends, we are truly grateful for all of your support and help during the project and throughout the educational stage.

Finally, we would like to thank all the people who helped and supported us even with their feelings.

Abstract

Each year, we lose many lives due to forcing humans to function in hazardous places. In this project, we will focus on providing an alternative to putting human lives at risk. In essence, we propose an embedded system for hand gesture based control of a robot system.

Our system utilizes deep learning and computer vision algorithms for real-time hand gestures detection in order to achieve a more natural approach of controlling robotic systems, which will help minimize the distance between the digital world and the physical world.

In this project, we designed, implemented, and trained an intelligent small-scale gestures controlled car. The system was able to achieve great dynamic gesture detection as it was tested with different people’s hands.

Table of Contents

List of figures

List of Tables

List of Listing

Chapter 1: Introduction

1.1 Overview and motivation

In 2020 in California's seasonal wildfires, four firefighters and 29 other people were devoured by the flames [1]. And in Australia, at least 33 people died, including three volunteer firefighters during the start of the fire season in late 2019 [2]. Many such accidents repeat each and every year, where the humans are set to stay and function in hazardous places, where they put their life at risk.

Having a robotic alternative to such risky human missions will save lives, and allow us to discover more and more of the unreachable places on earth or outer space by using them instead of people. And at the same time having a more naturalized way of controlling such robotic systems, will allow us to expand their use to everyone, including disabled individuals, who face huge challenges in their day-to-day tasks.

Although gesture detection is not an easy task, once a high-quality signal could be captured with good accuracy, it can principally be used to control multiple systems, not only hardware robotic systems, but also software systems, like opening an email or sending a text message.

In this project, we aim to build a robotic system that can be controlled by hand gestures that are detected using deep learning techniques. The system is implemented using a microcontroller as a proof of concept of the power of gestures in controlling certain types of systems.

1.2 System Objectives

In this project we aim to achieve the following objectives:

- Design a system based on deep learning and computer vision algorithms for real-time hand gestures detection and control of a robotic system with those detected gestures.
- Propose a more natural approach to control systems, which will help minimize the distance between the digital world and the physical world
- Conduct experiments to provide accuracy and performance reports of the system.

1.3 General System Diagram

Our system will capture images using a digital camera which is going to be stable on a remote desktop. The camera will continuously capture images of the person's hand gestures and process them in real-time to control the robot's movement. Also, there is another camera installed on the front interface of the robot. It is a real-time live camera that sends what the robot sees to the desktop machine. When the user changes his hand gesture, the system will determine the meaning of the gesture using computer vision and deep learning techniques, the captured signal will be sent to the robot to change its movement as intended, as shown in figure 1.1.

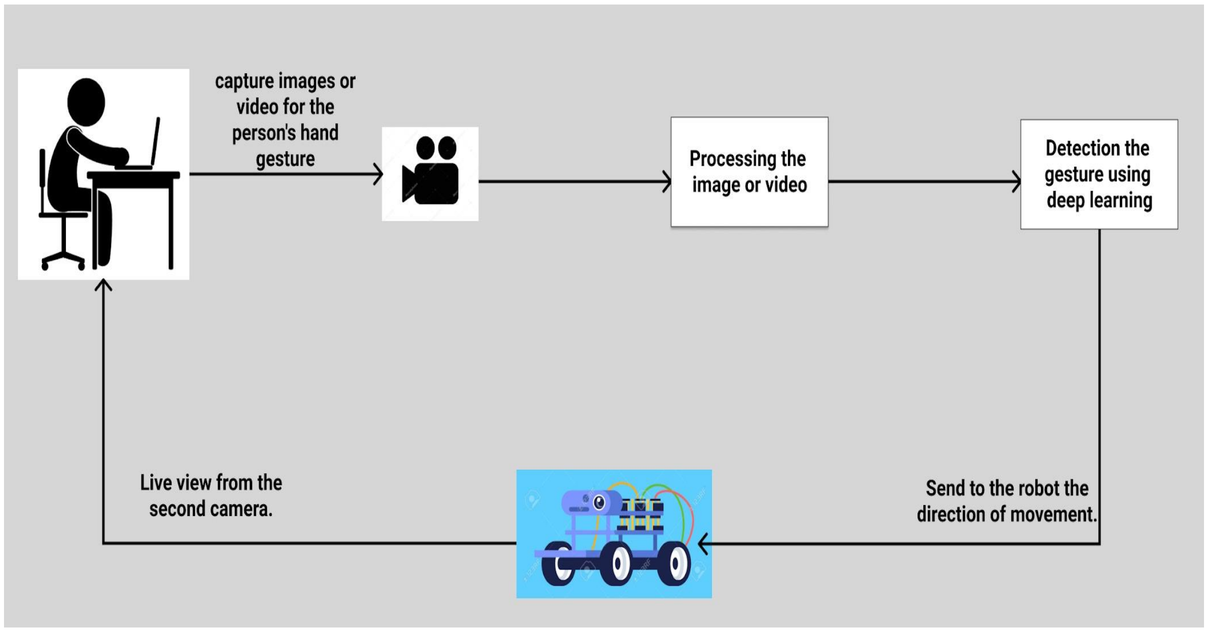

capture images or video for the person's hand gesture
Processing the image or video
Detection the gesture using deep learning
Send to the robot the direction of movement.
Live view from the second camera.

Figure 1.1: Block diagram of the system

1.4 Problem Statement

Each year, too many lives are to either die or get severely injured when sending them to hazardous places like wildfires, outer space, and so on. Our suggestion is to use robotic machines instead of people to save their lives. In this project, we will build a simple application to control the movements of a robot using human hand gestures to achieve man-machine interaction that feels as natural as possible, without overcomplicating such interactions by wearing any sensors or wearable systems that cause some inconvenience to the end-user.

1.5 List of Requirements

In a nutshell, the system we present within this project should achieve the following requirements:

1. Detect the controlling person's hand gesture in real-time.
2. Live camera viewing the area that is in the front of the robot.
3. Move the robot based on the detected hand gesture.

1.6 Expected Results

1. A fully functioning model that can detect human hand gestures to control a robot in real-time.
2. Decrease the distance between the digital world and the physical world, by using human gestures to control electronic devices, and using such systems to reach places that are unreachable to humans.

1.7 Constraints

In the currently implemented version, we should note that:

- This system responds only to 5 human hand gestures.
- Good lighting towards the user’s hand is important in order for the camera to detect the user's hand gesture.

Chapter 2: Background

2.1 Overview

This chapter mainly provides a quick background about the important components in our project. First, we will talk about the theoretical background, we will explain general terms like machine learning, deep learning, and computer vision. Secondly, we will present a literature review to show how previous studies achieved such objectives. Thirdly, we will look over the main hardware devices we need and why we need them. Finally, we will talk about the software background, generally about the programming languages and the algorithms we will use.

2.2 Theoretical background

In this section, we will provide some fundamental information about the main concepts on which our system is based.

2.2.1 Machine learning

Machine learning is a branch of Artificial Intelligence concerned with building applications that learn from the data and improve their accuracy over time without needing programmers to do that. In traditional programming, the algorithm is a sequence of statistical processing steps, but in machine learning, the algorithm is trained from the data features and patterns to make a decision[3].

There are three main types of machine learning, Supervised, Unsupervised, and Semi-supervised machine learning. Supervised machine learning is trained by using a labeled data set and the new data is classified based on this data. For example, if you need to classify apples and bananas, you should have labeled data with apples and bananas, that's like if you teach a kid, you hold an apple and tell him this is an apple, and so on. On the other hand in unsupervised training no labels are used. The program aggregates the common features between them and classifies them based on those common features. Semi-supervised is a medium between supervised and unsupervised. It uses small label data for classification and large unlabeled data for features extraction. Figure 2.1 explains the difference between them [4].

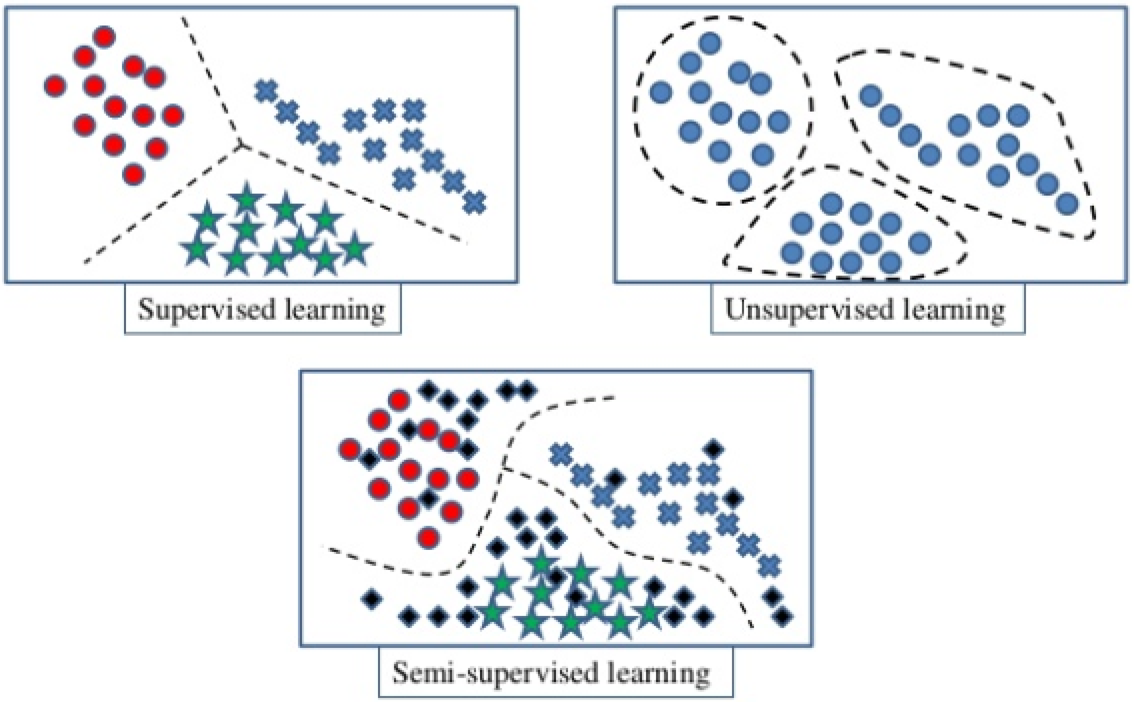


Figure 2.1: ML methods [4]

Another really important type of machine learning is reinforcement machine learning which is the training of machine learning models to make a sequence of decisions. The agent or in this case is a robot that learns to achieve a goal in an uncertain environment. So, the artificial intelligence faces a game-like situation. Where the computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer needs, the AI gets either rewards or penalties for the actions it performs. The goal is to maximize the total reward.

2.2.1 Deep learning

Deep learning is a subset of machine learning that enables the machine to learn without human supervision. Neural networks are the basic building block of deep learning. It contains nodes that work together mimicking the human brain [5]. In terms of architecture, a neural network consists of an input layer, an output layer, and hidden layers. Each layer consists of one or more connected nodes, each node called a **neuron** or a perceptron. Each neuron has a weight and an activation function. If the output of the neuron is above a specific threshold the neuron fires the output data to the input of the next neuron. Otherwise, no data passes to the next neuron.

The deep learning models need a huge amount of data passed through their layers to apply tuning for the weights and biases for each layer to get the best outcome. The Convolutional Neural Networks (CNNs) is a great implementation of that, and it is used primarily for computer vision applications, Which can detect patterns, features and recognize objects from images or videos [6].

When it comes to the difference between machine learning and deep learning in terms of processing Figure 2.2 explains that [7]. The feature extraction in machine learning is done by feature engineering, then the classification happens depending on these features, But in deep learning, you don't need to extract any features. It's done automatically inside the model.

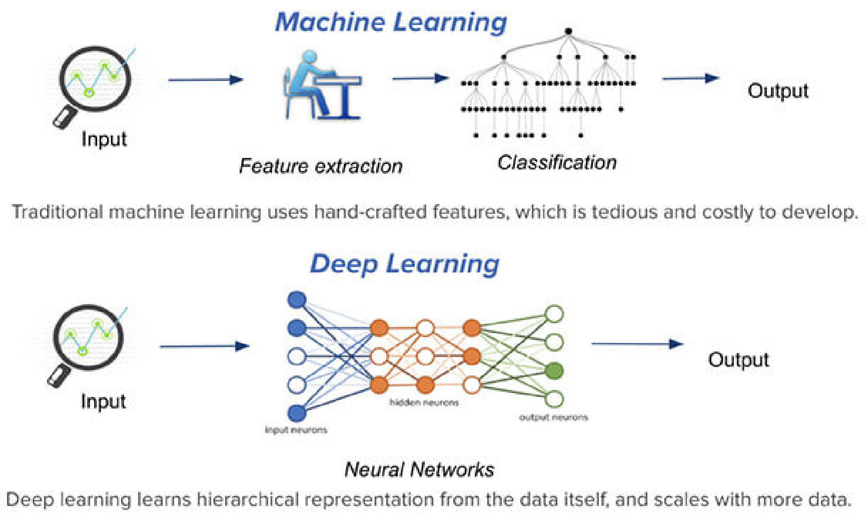

Machine Learning
Input
Feature extraction
Classification
Output
Traditional machine learning uses hand-crafted features, which is tedious and costly to develop.
Deep Learning
Input
input neurons
hidden neurons
output neurons
Output
Neural Networks
Deep learning learns hierarchical representation from the data itself, and scales with more data.

Figure 2.2: Machine learning and Deep learning processing [7]

2.2.3 Computer vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to extract information from digital images, videos, and other visual inputs, then make an action depending on such information. So, we can say that the system can “think” based on what the system “sees”.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions.

In the last five years, computer vision has witnessed continuous developments. The best performance came with the introduction of CNN. Today, deep learning with CNNs is the crux of most computer vision applications, such as self-driving cars, facial security features, gesture recognition, etc.

When it comes to sequence-based recognition, Recurrent Neural Networks (RNN) are used. RNN is a class of neural networks that allow previous outputs to be used as inputs while having hidden states. A drawback of RNN is the difficulty of accessing information from a long time ago, here where long Short-Term Memory (LSTM) shine, a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber [8].

2.3 Literature review

Many studies have examined gestures recognition. Looking at those studies with a critical eye helped us choose the techniques that suit the needs of our project.

Some of those studies focused on capturing the hand gestures using a wearable device that is based on an accelerometer [9], although this technique had an impressive hand gesture capturing accuracy, it is still far a little bit from the way we intended to capture those signals since we are focused to have a more natural way of controlling the digital world. Other studies concentrated on real-time video capturing of the hand gestures and then analyzing them to output the proper signals [10], we choose to focus on this type of recognition for our project.

Real-time recognition of dynamic hand gestures from video streams is challenging since there is no indication of when a gesture starts and ends in a video. Also, the performed gestures should be only recognized once. Such challenging components were resolved by proposing a hierarchical structure enabling offline-working CNN architectures to operate online efficiently by using the sliding window approach [11]

Another study used a hidden Markov model (HMM) which takes a continuous stream as an input and can automatically segment and recognize human gestures, here the HMM's gesture recognizer performs both segmentation and recognition of the human gesture simultaneously [12]. Although this study did design the system to recognize the entire movements of an individual, which is not detail-focused as the system we want to achieve. But their results were really promising and that is why we decided to consider their research.

To deal with those challenges in dynamic hand gestures recognition, another research group focused on coarticulation. Moreover, a new set of novel features in the feature extraction stage was used such as the position of the hand, self co-articulated features, ratio and distance features. The Artificial Neural Network (ANN) and Support Vector Machine (SVM) was used to develop two independent models using a new set of features as input [13].

Although most of the researchers focused on architecting an intelligent system that can adapt its input and improve its accuracy over time, some researchers tried to improve and add on to common image processing techniques like Finger Segmentation [14]. In such a method the hand region is extracted from the background with the background subtraction method. Then, the palm and fingers are segmented to detect and recognize the fingers. Finally, a rule classifier is applied to predict the labels of hand gestures.

2.4 Hardware components

In this section, we will present the main hardware components of the project and some detailed descriptions of them. For most components, we actually present different options that are available and explain the reason behind our choice.

2.4.1 Logitech camera C905 [15]

The C905 Logitech camera (see Figure 2.3) is a USB camera 2 MP Portable Webcam, Ultra-smooth AutoFocus with Carl Zeiss optics, and 2-megapixel resolution for high-clarity video and vibrant photos up to 8 MP, RightLight2 Technology light correction for the best images, even in dim and backlit conditions.

Figure 2.3: Logitech camera [15]

2.4.2 Raspberry Pi Camera[16]

The Raspberry Pi camera module v1.3 (Figure 2.4) is an official product of the Raspberry Pi foundation. This means it has software and hardware support from the original developers of the credit-card-sized computer itself. For that matter, even enthusiasts are active in developing beginner to advanced libraries for this camera module.

Table 2.1 shows the difference between the two cameras:

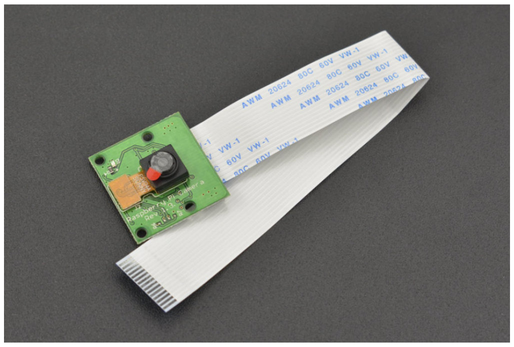


Figure 2.4: Raspberry pi camera [16]

Raspberry Pi camera	**Logitech C905**
Not autofocus	Autofocus
Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording	Up to 30-frames-per-second video
5MP Omnivision 5647 Camera Module	Up to 8-megapixel photos (enhanced from the native 2-MP sensor)
Picture Resolution: 2592 x 1944	720 p widescreen mode with recommended system High-definition video(up to 1600 X 1200)
15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board	Hi-Speed USB 2.0 certified

Table 2.1 Difference between Logitech C905 and Raspberry Pi camera

We choose the Logitech C905 to capture the live feed from the robot and send it to the remote desktop due to of these reasons:

1. Logitech is available for us.
2. When using the Raspberry Pi with 64 bit, you need to download the Libcamera driver to work but the USB camera works directly.

2.4.3 Laptop or PC Camera

We choose to use the camera of the laptop or PC to capture the gesture from the user then we transfer the picture to the respray pi to determine the intended direction of the robot movement.

2.4.4 Raspberry Pi Model B

Raspberry Pi 4 Model B (as shown in figure 2.5) is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+ while retaining backward compatibility and similar power consumption. For the end-user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems [17].

This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

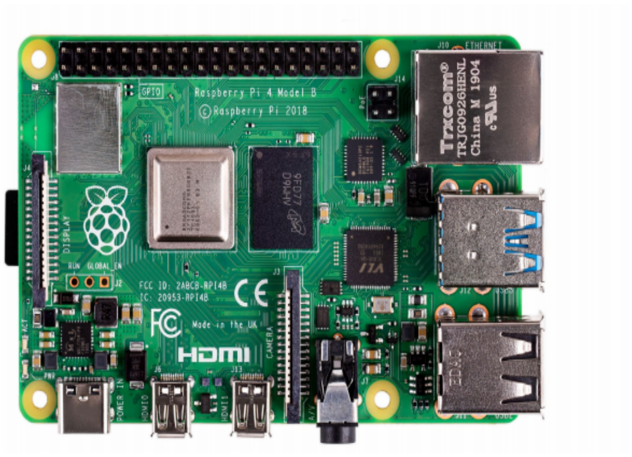


Figure 2.5: Raspberry Pi 4 Model B [17]

2.4.5 Jetson Nano Developer Kit

The NVIDIA® Jetson Nano™ Developer Kit (as shown in figure 2.6) delivers the compute performance to run modern AI workloads at unprecedented size, power, and cost. Developers, learners, and makers can now run AI frameworks and models for applications like image classification, object detection, segmentation, and speech processing [18].

The developer kit can be powered by micro-USB and comes with extensive I/Os, ranging from GPIO to CSI. This makes it simple for developers to connect a diverse set of new sensors to enable a variety of AI applications. And it is incredibly power-efficient, consuming as little as 5 watts.

Jetson Nano is also supported by NVIDIA JetPack, which includes a board support package (BSP), Linux OS, NVIDIA CUDA®, cuDNN, and TensorRT™ software libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more. The software is even available using an easy-to-flash SD card image, making it fast and easy to get started.

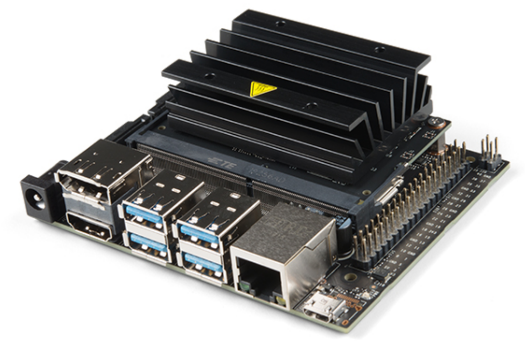


Figure 2.6 Jetson Nano Developer Kit [18]

As you see from the below list of initial specs of both boards, there certainly are some similarities. Let’s take a look at comparing the specifications of both the Raspberry Pi 4 and NVIDIA Jetson Nano Developer Kit. look at table 2.2

	Raspberry Pi 4	**Nvidia Jetson Nano**
CPU	Quad- core ARM cortex-A72 64-bit @ 1.5 Ghz	Quad- core ARM cortex-A57 64-bit @ 1.42 Ghz
GPU	Broadcom VideoCore VI (32-bit)	NVIDIA Maxwell w/128 CUDA @ 921 MHZ
Memory	4 GB LPDDR4	4 GB LPDDR(2GB version also available
Networking	Gigabit Ethernet/ wifi 802.11ac	Gigabit Ethernet / M.2 key E (for wifi support) (2 GB version includes USB Wireless adaptor)
Display	2x micro- HDMI (up to 4Kp60)	HDMI 2.0 and eDP 1.4
USB	2x USB 3.0 2x USB	4x USB 3.0 USB 2.0 Micro-B
Other	40-pin GPIO	40-pin GPIO
Video Encode	H264(1080p30)	H264/H265 (4kp30)
Video Decode	H265(4Kp60), H264(1080p60)	H264/H265 (4k p60, 2x 4kp30)
Camera	MIPI CSI port	MIPI CSI port
Storage	Micro-Sd	Micro-Sd
Price	55$ USD	99$ USD

Table 2.2: Difference between Raspberry Pi 4 and Jetson Nano.

The two kits have similar specifications, but we choose the **Raspberry pi model B 64bit 8GB** because it is the best version of raspberry pi available in our hands. Another reason, prior experience of colleagues revealed several problems with the versioning for some libraries and the set up of the Jetson Nano Kit, which discouraged us from choosing it.

2.4.6 Motor Driver L293D

It is an IC that contains 16 pins. It is capable of running two dc motors at the same time and controlling their direction independently. We will use it to enable the dc motors.

2.4.7 DC Motors

It is a device that converts direct current electrical energy into mechanical energy and uses the electricity generated to produce motion for wheels.

2.4.8 Arduino Uno[19]

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. Look at figure 2.7.

Figure 2.7: Arduino Uno [19]

The reason to choose Arduino in our system is that the response time for the low-level hardware components like ultrasonic and motors in Arduino is less than Raspberry Pi. So, the system response speed will increase.

2.4.9 Ultrasonic Distance Sensor - HC-SR04[20]

This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module (Figure 2.8) includes an ultrasonic transmitter, a receiver, and a control circuit.

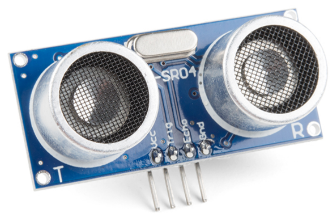


Figure 2.8: Ultrasonic Sensor [20]

There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground).

2.5 Software background

Software is a really important part of our system. In this section, we give an overview of the programming languages, frameworks, and libraries that have been used. In addition, we present a brief description of the adopted algorithms.

2.5.1 Programming languages

Primarily, Python is used as our programming language since it deals with the complex algorithms in machine learning efficiently and it is also widely used by the machine learning community, in addition, most of the research work we saw uses python as a primary tool to build the model that was responsible for the gestures detection.

2.5.2 Frameworks and Libraries

MediaPipe

MediaPipe [21] is a cross-platform framework for building multimodal applied machine learning pipelines. One of the most useful models for our project is the Hands model which is a high-fidelity hand and finger tracking solution for both right and left hand. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame and in real-time.

MediaPipe Hands solution utilizes an ML pipeline consisting of several models working together. The first one is a palm detector model called BlazePalm which is a single-shot detector model optimized for real-time uses. A single-shot detector model is a model that puts a detected object in a bounding box, that is why the MediaPipe team decided to detect the palm first because it is easier to put the palm in a bounding box than to figure out all the finger's details.

Then the result of that BlazePalm detector will run into a hand landmark model that performs precise landmark localization of 21 2.5D coordinates inside the detected
hand regions via regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands.

The model has three outputs (see Figure 2.9):

1. 21 hand landmarks consisting of x, y, and relative depth.
2. A hand flag indicates the probability of hand presence in the input image.
3. Binary classification of handedness e.g. left or right hand.

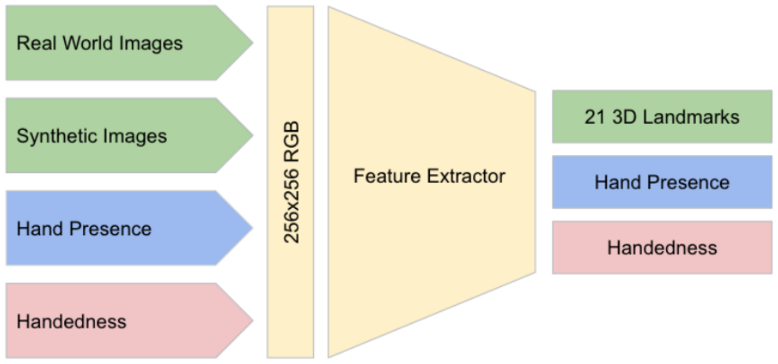


Figure 2.9: Mediapipe Hands model outputs [22]

Hand landmark Coordinates are presented in the below figure. Here

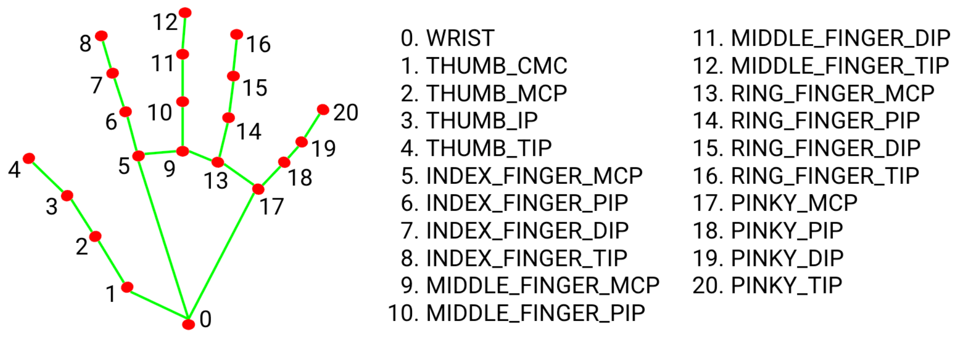


Figure 2.10: Hand landmark Coordinates details [21]

PyTorch

PyTorch [23] is an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab. It is free and open-source software released under the Modified BSD license. Such libraries allow tensor computing with the ability for accelerated processing via GPU, an important point also that this library has good support for neural networks, since they are built on a tape-based auto diff system.

TensorFlow

TensorFlow [24] is an open-source platform that was developed by Google's Brain team, that contains a comprehensive, flexible ecosystem of tools, libraries, and community resources. This tool is widely used within the community, where huge contributions to make such models and algorithms be more and more accurate and efficient will help us out during our work on this side of the project.

We will be using TensorFlow instead of PyTorch since it has a much bigger community behind it than PyTorch, which means that it is easier to find resources to learn Tensorflow, this point is important for us since installing such libraries into microcontrollers like Raspberry pi is mostly bug full and having such community will be a great resource for us. Also, it was proven that TensorFlow is much better for production models and scalability since it was built to be production-ready.

OpenCV

OpenCV, short for open-source Computer Vision, is a cross-platform open-source programming library that includes a numerous number of Computer Vision algorithms and Functions [25].

NumPy

NumPy is an open-source python library that includes powerful functions and mechanisms for handling matrix arithmetic as well as functions in Linear Algebra, Fourier transform, and others [26].

2.5.3 Algorithms

This section describes the algorithm needed to implement the gestures recognition function of our system. Where we looked at multiple intelligent and classical algorithms and compared them to choose the best approach.

2.5.3.1 Recurrent Neural Networks

A recurrent neural network is a type of artificial neural network commonly used in speech and video recognition and natural language processing. Recurrent neural networks recognize data's sequential characteristics and use patterns to predict the next likely scenario. They are networks with loops in them, allowing information to persist. Figure 2.11 explains a single node of RNN.

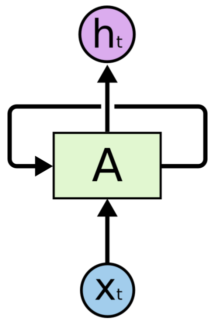


Figure 2.11: RNN node [27]

This loop-like structure can be unfolded as well to form the regular shape RNN as in Figure 3.12

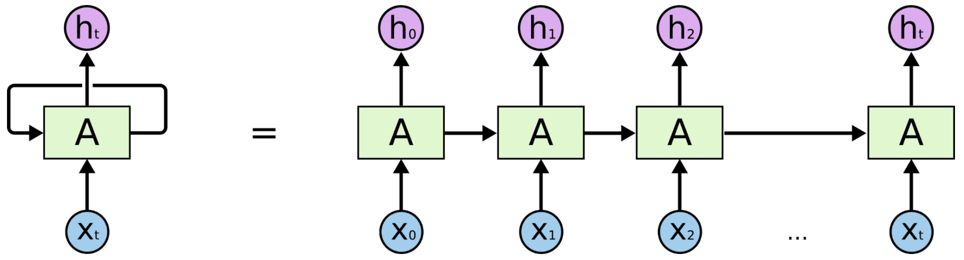


Figure 2.12: RNN unrolled [27]

The problem with RNN is Long-Term Dependencies where it is hard for the network to consider far-away data into its prediction, and limiting its prediction to short-term data span. That is where Long Short-Term Memory comes in use.

2.5.3.2 Long Short-Term Memory

Long Short-Term Memory Networks (LSTM) is a special kind of RNN, capable of learning long-term dependencies. They are explicitly designed to avoid long-term dependency problems. Remembering information for long periods of time is particularly their default behavior.

LSTMs also have this chain-like structure as you can see in Figure 2.13, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

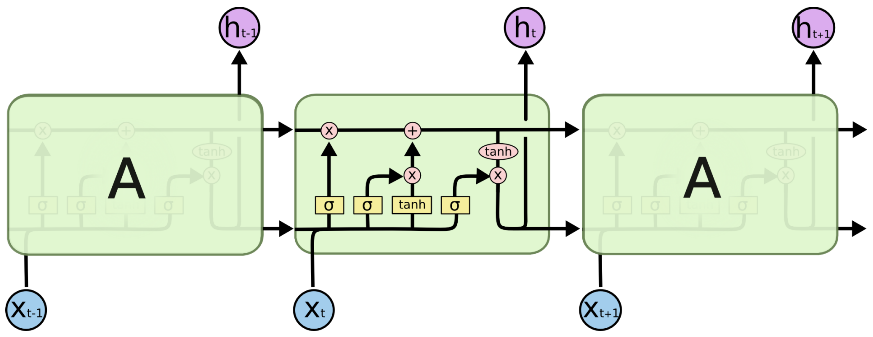


Figure 2.13: LSTM internal structure [27]

We will use LSTMs as a basis for our gestures sequence detection deep neural network model since it is more able to remember inputs over a long period of time better than the vanilla RNN.

2.5.3.3 Finite State Machine

A Finite State Machine is used to model four qualitatively distinct phases of a generic gesture [28]. Fingertips are tracked in multiple frames to compute motion trajectories, which are then used for finding the start and stop position of the gesture. Gestures are represented as a list of vectors and are then matched to stored gesture vector models using table lookup based on vector displacements. Such a method seems extremely limited to certain motion trajectories, and we see that the LSTM architecture is more helpful in our case than the Finite State Machine.

2.5.3.4 Finger Segmentation

Since the palm is static, while the fingers are not, detecting their movement is a huge part of the recognition process[14], a way of doing that is to start by detecting the hand using the background subtraction method and then the result of this process is transformed to a binary image. Then, the fingers and palm are segmented to facilitate finger recognition. Moreover, the fingers are detected and recognized. Finally, hand gestures are recognized using a simple rule classifier.

In the segmentation image of the fingers, the labeling algorithm is applied to mark where the fingers are. Only the regions of enough pixels are regarded as fingers. For each remaining region, that is, a finger, the minimal bounding box is used to enclose the finger. Then, the center of the minimal bounding box is used to represent the center point of the finger.

The type of recognition resulting from such a method is limited to certain hand color, and background color compared to other intelligent methods, where the limitation was minimized resulting in a better gestures recognizer. So a combination of Mediapipe for hands key points extractor and an LSTM Neural Network where we can detect those dynamic gestures. Will be the best compared to the proposed solutions, since it will have the ability to detect dynamic gestures, and will not be limited by any kind of background or any other elements within the frame other than the user’s hands.

Chapter 3: Design

3.1 Overview

This chapter discusses the overall design of the gestures controlled robotic embedded system and the way its components are integrated together, showing the block diagram and schematic diagram for the design, in addition to some details about the software gestures recognition side of our project.

3.2 Detailed Design

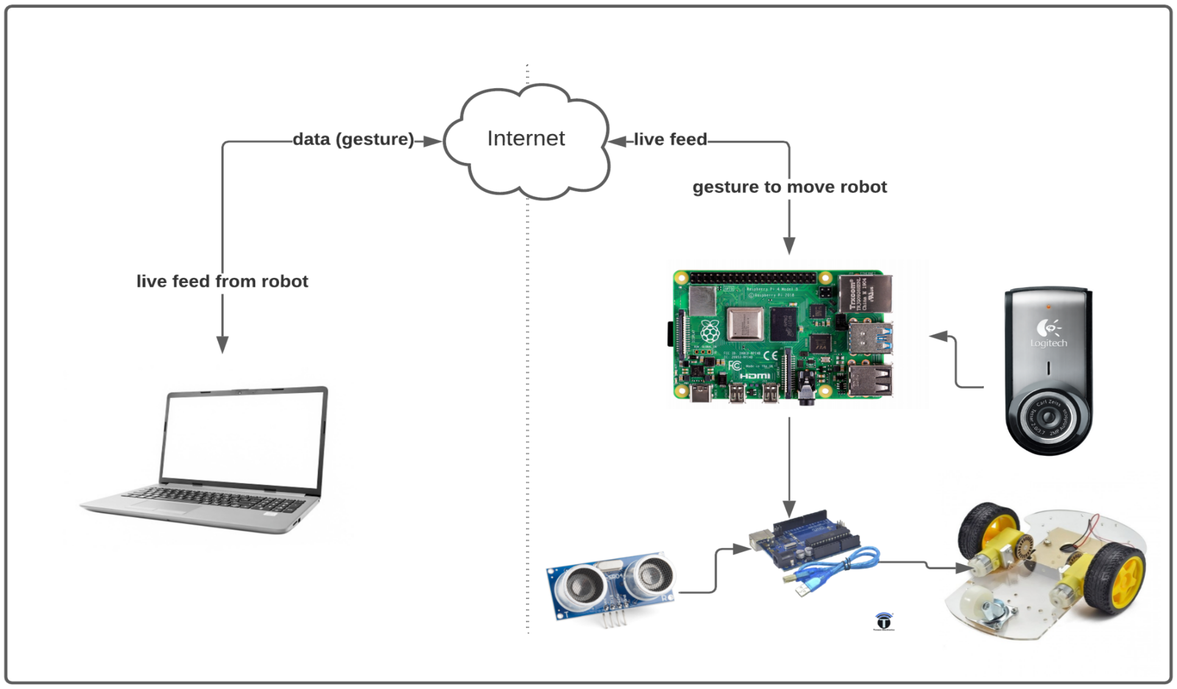


Figure 3.1:System Design

The system design, shown in Figure 3.1, explains the general overview of the system's main components and the connection between them. On the left part, the user uses the laptop's camera to capture his hand gesture and send it over the internet to Raspberry Pi on the right. The gesture is passed into our model to determine the movement. After that, we notify Arduino to change the car movement. On the other hand, we send a live video from the robot to the remote desktop to see what the robot sees.

3.3 Block Diagram

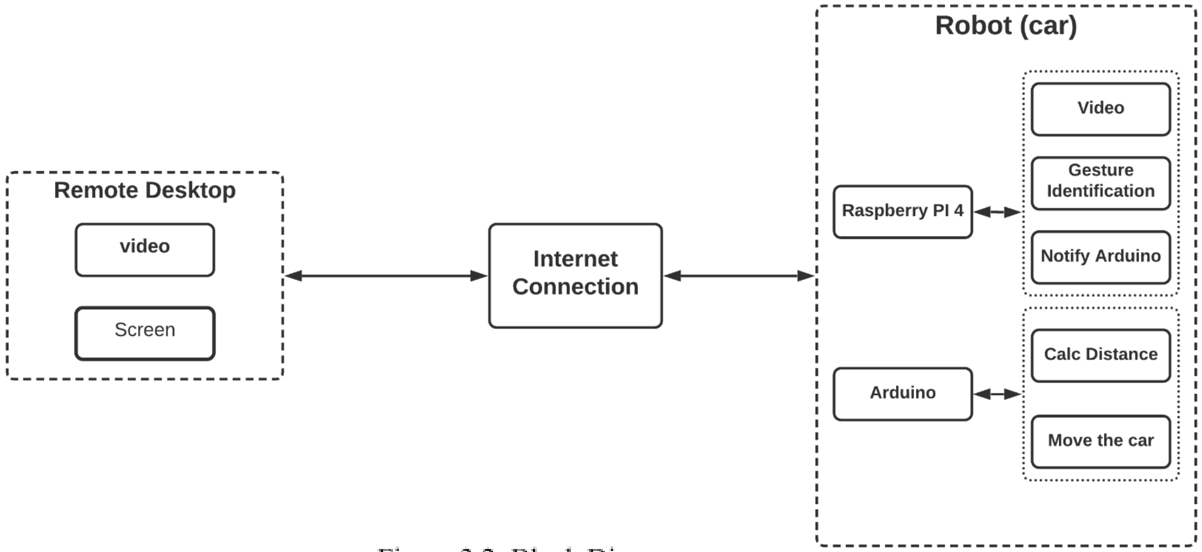


Figure 3.2: Block Diagram

The system diagram in figure 3.2 shows how the communication happened between the system sides (laptop and robot). We are using WebRTC [29] and Firebase to create a channel between these sides in real-time, that allows both sides to see the video from the other side.

Also, Arduino calculates the distance between the car and anything in front or back of it using Ultrasonic sensors. The car stops if the distance is less than 30 cm.

3.4 Pseudo-Code

The following pseudo-codes describe the whole project process and how the components are connected using WebRTC and Firebase. The first pseudo-code explains how the remote desktop connects with the WebRTC and FireBase channel and sends the gesture to the other side.

```
* Remote Desktop *

 ** Setup Part **

INITIALIZE network connection
Connect the laptop or the PC with the internet
Connect with the WebRTC and FireBase
Create Group on the site to connect it with Raspberry P4i

 ** Remote side **
While the program running do
        Send the gesture to the raspberry pi
        Display the received video from Raspberry Pi on the screen
end while
```

Listing 3.1: Pseudocode for remote desktop

The second pseudo-code explains how the raspberry pi connects with the WebRTC and FireBase channel and sends the live video to the remote side. Also, it describes how to move the robot based on the result of the detection model over various movements.

```
* Raspberry PI *

** Setup Part **

INITIALIZE network connection
Connect the Raspberry Pi with the internet
Connect with the WebRTC and FireBase
Connect to the same group that Remote Desktop has created

** Raspberry pi side **
While the program running  do
        Send the live video to the remote desktop
        Get the gesture from remote desktop
        Detect the movement from the gesture
        Move the car depending on the detection movement
end while
```

Listing 3.2: Pseudocode for Raspberry Pi

3.4 Schematic Diagram

Figure 3.3 shows the schematic diagram of the hardware components. It shows how the L293D IC is connected with the DC motors and Arduino to control the directions and speed of the car. Also, it shows how the ultrasonic sensors are connected with Aurdino to calculate the distances between the car and any object on the front of the movement.

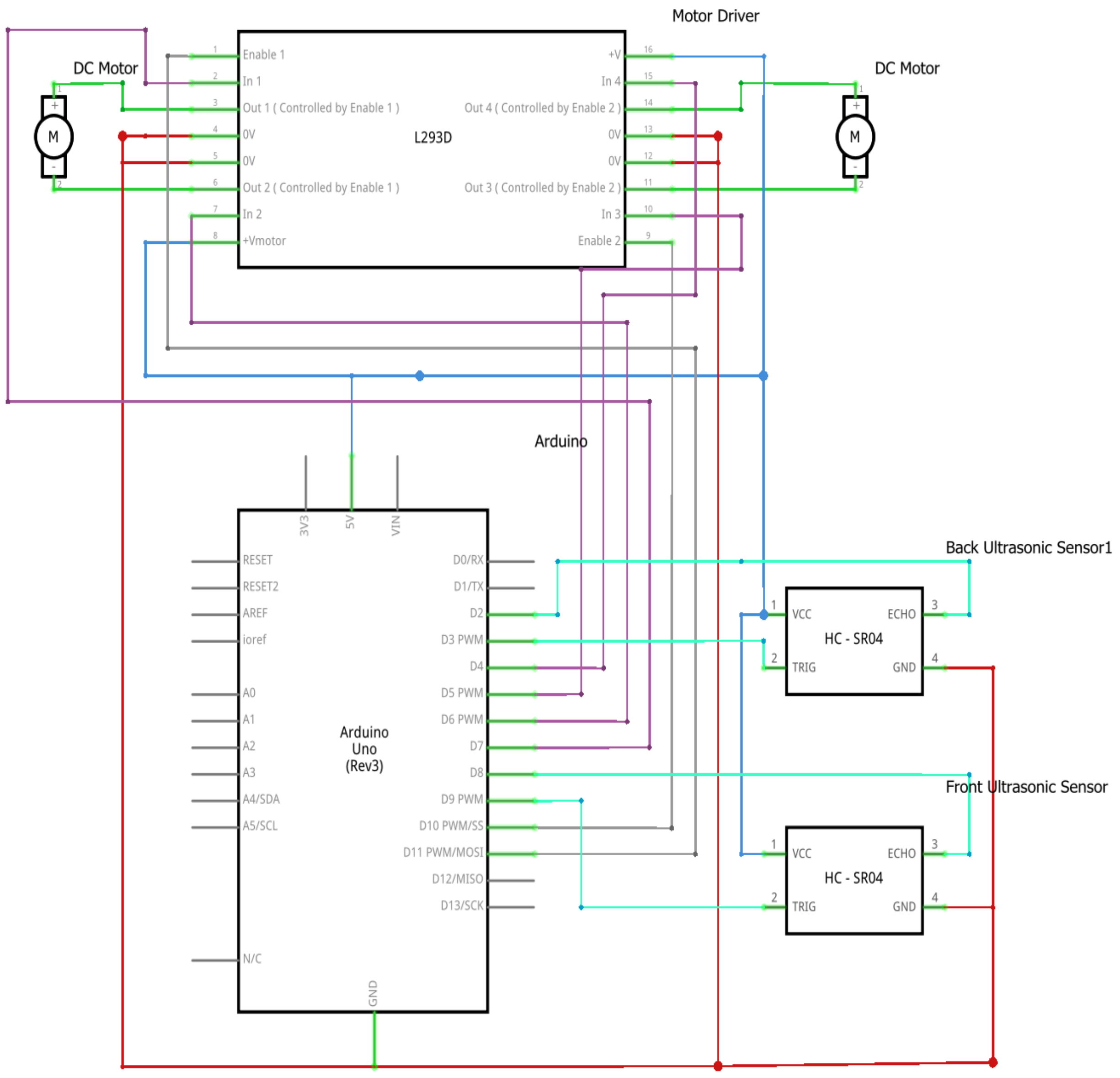


Figure 3.3: Schematic Diagram

We've connected the Arduino with Raspberry pi using TTL to USB capel to make serial communications between them. Look at 3.4 Figure.

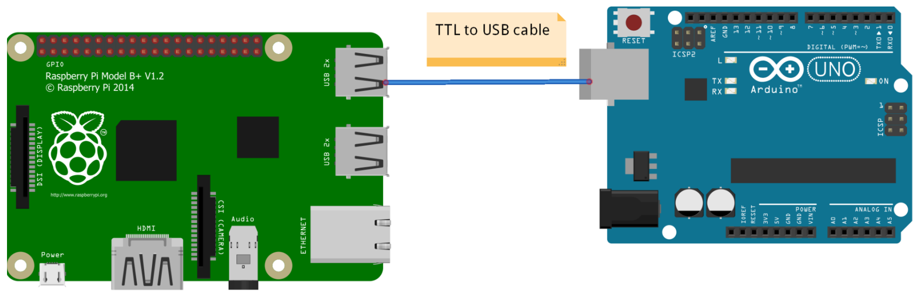


Figure 3.4: TTL to USB Cable between Raspberry Pi and Arduino

3.5 Detailed System Software Design

This section describes the software logic of the system. It briefly explains how to employ some of the techniques discussed in the previous chapter of this document.

LSTM Neural Network Specifications

Figure 3.4 describes the LSTM Neural Network architecture used in our main detection model. The input is a 30 × 126 2D vector, where each frame of the 30 frames that represent the input video clip will consist of 126 key points value, where for each hand we will have 21 key points, and each key point consist of 3 direction, [x, y, z] represents the positions of each key point in a 3D space. Those key points directions from both hands will be flattened into a 126 1D vector.
The first layer consists of 64 LSTM units, the second layer consists of 128 LSTM units, the third layer consists of 64 LSTM units to shape a kind of symmetrical shape. The next layer is fully connected and contains 64 units. The next layer is also a fully connected layer with 32 units. The rectified linear unit (ReLU) is used as an activation function over all the layers except for the last Softmax layer.

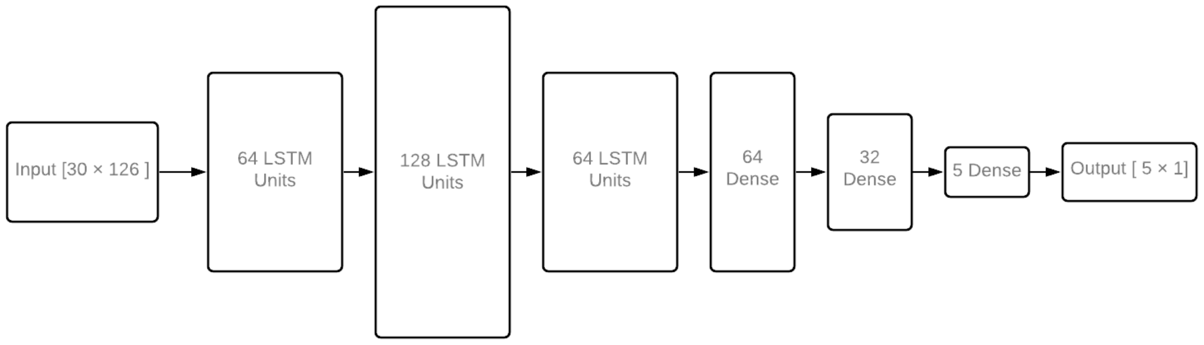


Figure 3.5: LSTM Neural Network Architecture

Chapter 4: Implementation

4.1 Overview

This chapter explains the implementation part for the hardware components and the software algorithms. It dives into more details in the project overall different hardware components and software modules.

4.2 Hardware Implementation

Figure 4.1 shows the hardware components we used to implement the subsystem responsible for moving the robot. The figure was generated using an online simulator called Tinkercad which demonstrates the actual design of each component as well as the connections between them.

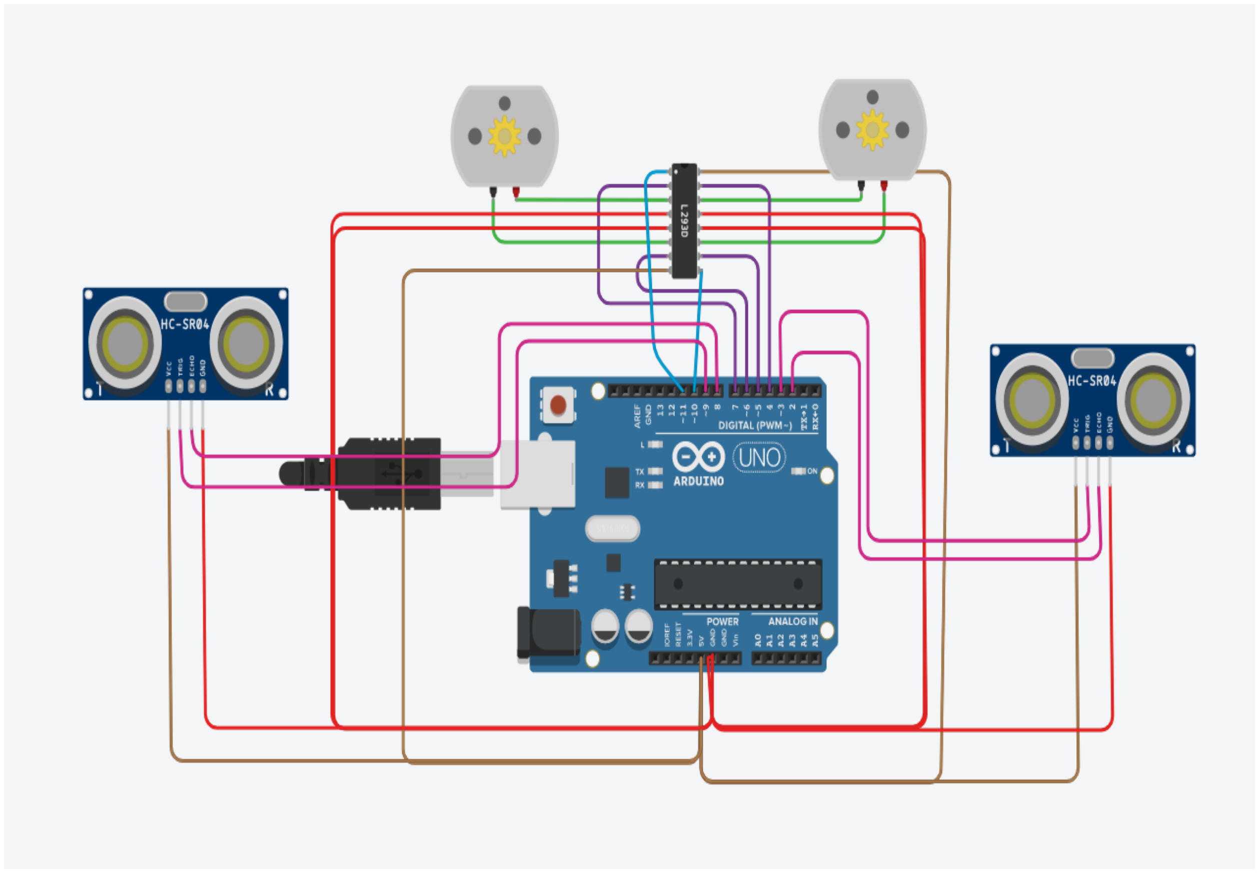


Figure 4.1: Tinkercad Simulation

To send the detected movement from the Raspberry Pi to the Arduino after detecting the gesture, we connect them using TTL to USP capel to make serial communications between them using the library in Python called serial. Look at Figure 4.2.

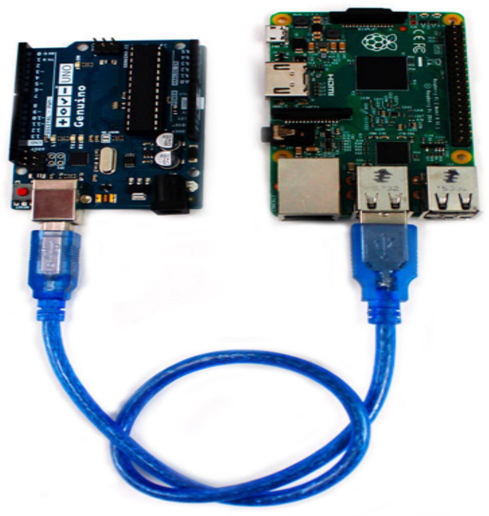


Figure 4.2: Serial Communications between Raspberry Pi and Arduino [30]

The steps below show the hardware implementation for the robot:

1. Connect the DC motors with the L293D driver, the following steps describe the process:
    a. The first DC motor connects with the pins (3,6) on the IC board to set it as output.
    b. The second DC motor connects with the pins (11,14) on the IC board to set it as output.
2. Connect the L293D driver with the Arduino, follow the steps below:
    a. The (2,7) pins from the IC connect with the pins (7,6) from Arduino respectively to control the direction of the first motor.
    b. The (10,15) pins from the IC connect with the pins (5,4) from Arduino respectively to control the direction of the second motor.
    c. The **first** pin from the IC connects with the **eleventh** pin from Arduino to control the speed of the first motor.
    d. The **ninth** pin from the IC connects with the **tenth** pin from Arduino to control the speed of the second motor.
3. The pins (8,16) connect to the VCC and The pins (4,5,13,14) connect to the ground.
4. Connect the Ultrasonic sensors with the Arduino:
    a. Every sensor has 4 pins (VCC, Trigger, Echo, Ground).
    b. The **trigger** pin from the first sensor connects with the **ninth** pin from the Arduino, and the echo pin connects with the **eighth** pin.
    c. The **trigger** pin from the second sensor connects with the **third** pin from the Arduino, and the echo pin connects with the **second** pin.

d. The VCC and Ground pins from the sensors connect with the VCC and Ground pins from the Arduino.

The USB Camera connects with any USB port of the raspberry pi. We've enabled the camera from the Interfacing Options of Raspberry PI.

Figure 4.3 shows the final look of the robot after connecting its parts.

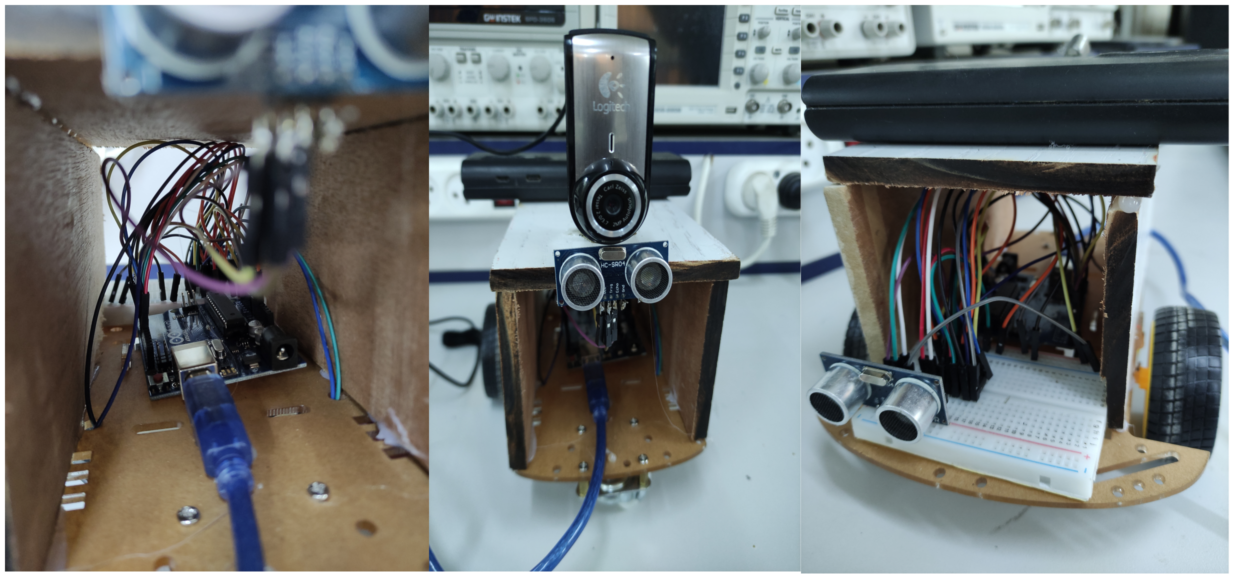


Figure 4.3: Final Look for the Robot

4.3 Software Implementation

This section describes the implementation details of the various software components of the system. It also explains the choice of different parameters and the features and functions of each software component in the system.

4.3.1 Packages installation

A crucial step of this system is to install a few software components such as OpenCV, TensorFlow, and Mediapipe. We used the pip package manager to install all the packages we used on different platforms. We faced multiple compatibility issues during installation, but we were able to overcome them.

4.3.2 Detection module

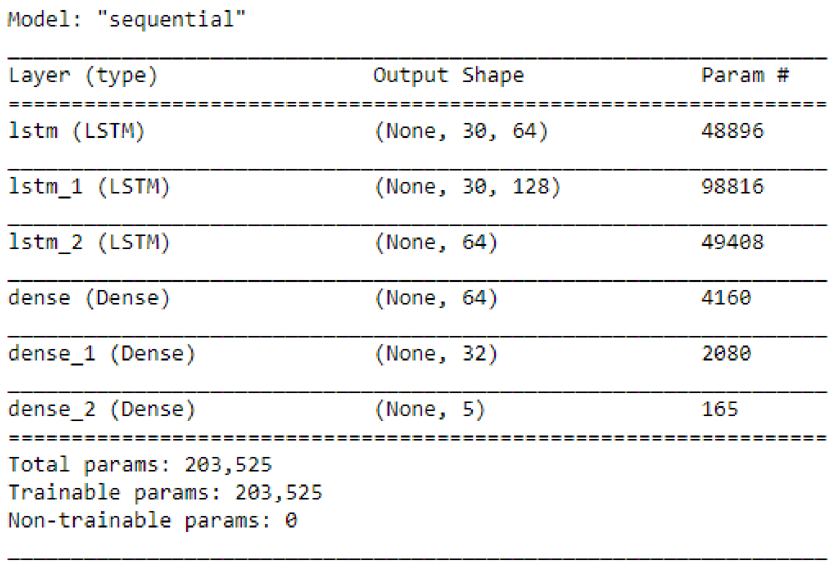
The detection module is responsible for detecting the gestures received from the front camera, there are multiple functionalities within the model, such as extracting the key points and the logic for the final detection.

This module will receive the real-time video stream of the user gestures, each frame is proceeded via the Mediapipe Hands model and ended up as a set of 42 key points values (21 left-hand landmarks and 21 right-hand landmarks) and each one of those key points is in a 3D space containing the dimensions of x, y, and z. If there is no left or right hand detected, the non-detected hand will be represented as a zero NumPy array for the key points. Concatenating left-hand keypoints with right-hand key points will be then the final key points array, then apply each 30 frames of the feed to the LSTM model in order to predict the intended gestures out of the main five gestures that the system is trained with.

The detected gesture will be then sent to the car module to perform the intended action. We used Firebase real-time in order to send the key points array from the laptop where we performed the key points extractor function that uses Mediapipe, to Raspberry Pi 4 where the LSTM trained model will do the gesture detection, the detected gesture will then be sent to Arduino in order to be interpreted in terms of motor movement.

Training the LSTM Neural Network Model

We gathered nearly 1150 video clips, 230 clips for each gesture, we were able to gather such clips with the help of our colleagues from the college. The data set may seem relatively small, since our model only contains 203,525 parameters as seen in Figure 4.4.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            48896
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 203,525
Trainable params: 203,525
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.4 LSTM Neural Network Model Summary

Our training process was limited to 35 epochs given the small number of parameters we have in the model, and we noticed that beyond 35 epochs the model will over train providing a higher loss. Figure 4.5 presents the accuracy metric data throughout the training process, also Figure 4.6 shows the loss metric data throughout the training process.

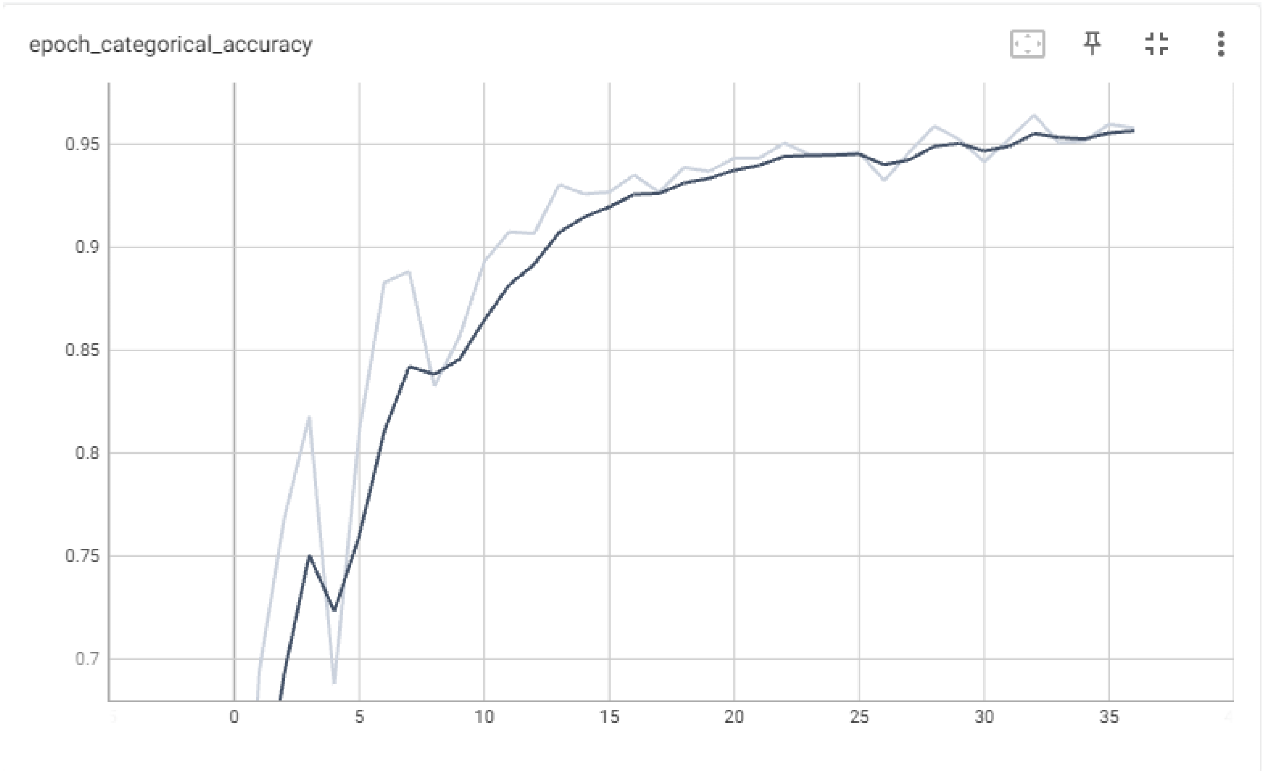


Figure 4.5 Epoch Categorical Accuracy

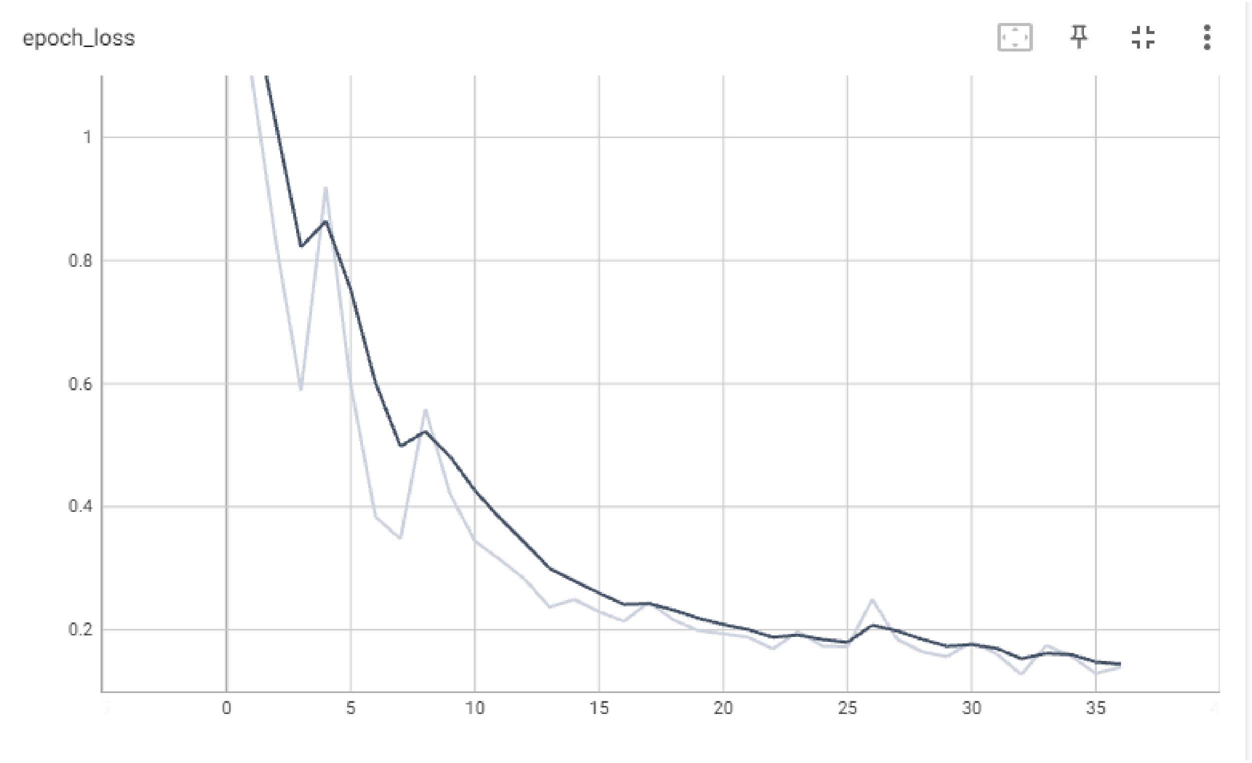


Figure 4.6 Epoch Loss

4.3.3 Arduino Implementation

We have used the Arduino to control the low-level hardware components, which are responsible for moving or stopping the car depending on the hand gestures or the ultrasonic sensors.

The following steps explain how we have implemented the robot components:

1. Every movement is referred to as a single character, as follows:
    a. Stop referred to as s.
    b. Forward referred to as f.
    c. Backward referred to as b.
    d. Right referred to as r.
    e. Left referred to as l.
2. The robot will be standing when the program is running.
3. The Arduino will read the distance between the robot and the objects in front and back of it. There are two scenarios for that as follows:

   a. If the movement is one of forwarding or right or left and the distance between the robot and any object in front of it is less than 30cm, the robot will stop immediately.
   b. If the movement is backward and the distance between the robot and any object behind it is less than 30cm, the robot will stop immediately.
4. As we mentioned in section 4.2, the Arduino connects with the Raspberry Pi with a serial communication cable to allow us to transfer the data from the Raspberry Pi to Arduino. The data is a single character that refers to the movement. The robot will move depending on this data as follows:
   a. If the character is f and there is no object in front of the robot, the robot moves to forward.
   b. If the character is r and there is no object in front of the robot, the robot moves to the right.
   c. If the character is l and there is no object in front of the robot, the robot moves to the left.
   d. If the character is b and there is no object behind the robot, the robot moves backward.
   e. If the character is s, the robot stops.

Chapter 5: Validation and Results

5.1 Overview

This chapter illustrates the testing process of the overall system, its components, and software. We test all the parts to ensure that all the functions work as expected and without errors.

5.2 Hardware Testing

This section discusses the testing process for the hardware components.

5.2.1 DC motors and L293D driver

After we've connected the car's components, we tested it by connecting it to the laptop. We've used the Arduino software and Serial Monitor to send data to the robot. The process was successful, and the robot moved into all sides without problems.

5.2.2 Ultrasonic sensors

The car will stop when we put an object in the front of it when it's moving forward or right or left, and the same thing when it moves backward.

5.2.3 Test the Camera with Raspberry Pi

We captured some videos through the camera while the car was moving, as it was clear, fast, and suitable for outdoor photography.

5.2.4 Test the communication between Raspberry Pi and Arduino.

We tried to send data from Raspberry Pi to Arduino using the Python library called serial. The process was fast and successful.

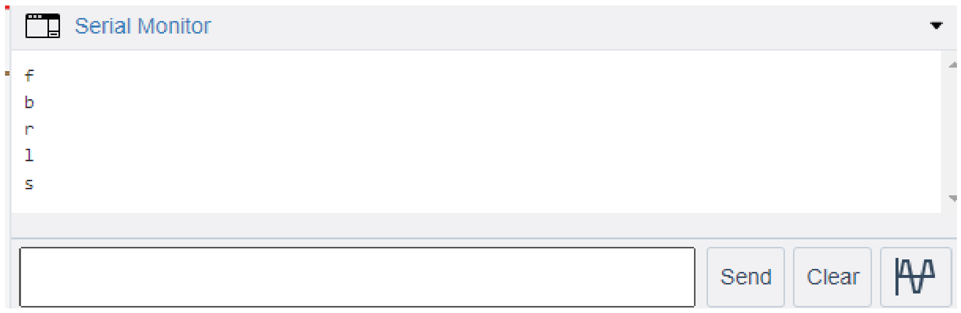
5.3 Software Testing

This section discusses the testing for our functions in the software

5.3.1 Test Arduino Software

We've used the serial monitor to test the software on the Arduino. We've entered a character inside the monitor, and we got the following results:

1. If the character is f, then the car moves forward.
2. If the character is b, then the car moves backward.
3. If the character is r, then the car moves to the right.
4. If the character is l, then the car moves to the left.
5. If the character is s, then the car stops.



```
f
b
r
l
s
```

Figure 5.1: Serial Monitor

5.3.2 Test Gesture detection Model

Our initial plan was to implement the entire Module on the Raspberry pi 4. As we capture the live video stream of the user hand gesture we will send this stream to the Module that is deployed in the Raspberry pi 4 and process it frame by frame in real-time.

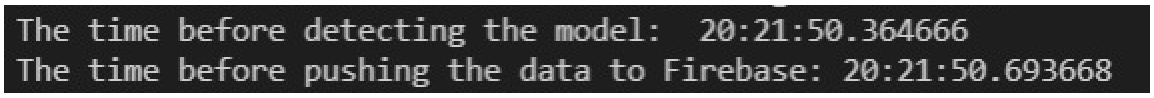
The detection process includes extracting the 42 key point values of both hands from the Mediapipe hand library, then passing those key points into our LSTM Neural Network, and finally, sending the predicted gesture to the Arduino.

Both Mediapipe and Tensorflow were installed successfully without any major problems to the Raspberry pi. Considering performance, this approach did not work very well, since we faced a huge noticeable delay to capture the Mediapipe key points from the user's hands.

That delay led us to try another approach to deal with the main cause of the delay, and run the Mediapipe key points extractor on a laptop, while the LSTM neural network will run on the Raspberry pi 4 using Tensorflow. We used firebase real-time to connect both devices together and send the key points vector from the laptop to the Raspberry pi for the gesture to be detected.

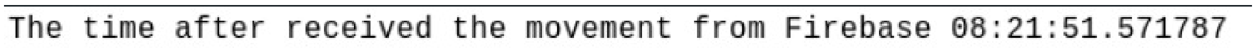
We ended up using the last approach since it was more reliable when it comes to the prediction time, and the user real-time experience. We calculated the time it takes for the user to see the changes of the provided gestures be implemented to the robot.

We tested out two approaches regarding running the gesture detection module. The first is to extract the key points from the user gesture and to run the LSTM neural network model in the laptop and send the detected gesture to the Raspberry pi via firebase real-time, Figure 5.2 (a) shows that the time needed to detect the gesture when the model is running on the laptop will be 3 milliseconds, and the time to send the detected gesture to the Raspberry pi will be nearly 8 milliseconds as shown on Figure 5.2 (b)



```
The time before detecting the model:  20:21:50.364666
The time before pushing the data to Firebase: 20:21:50.693668
```

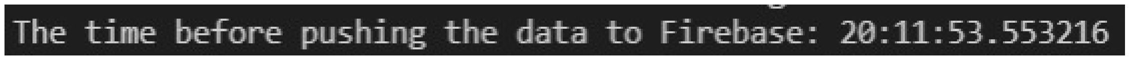
Figure 5.2 (a)



```
The time after received the movement from Firebase 08:21:51.571787
```

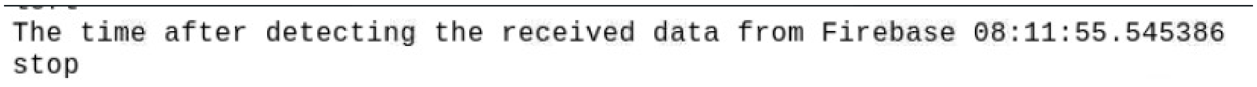
Figure 5.2 (b)

For the second approach we did only extract the key points in the laptop and send the key points array via firebase real-time, and as shown in Figure 5.3 (a) and (b) it took nearly 2 seconds



```
The time before pushing the data to Firebase: 20:11:53.553216
```

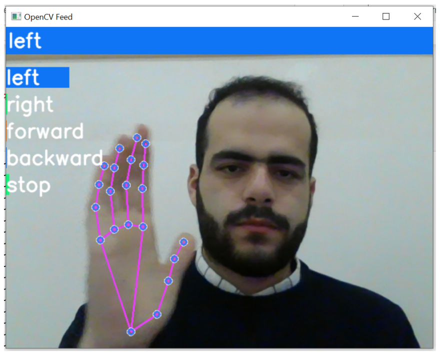
Figure 5.3 (a)



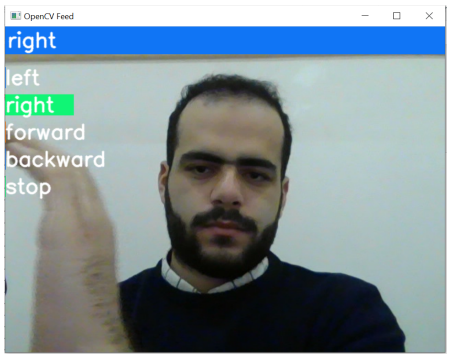
```
The time after detecting the received data from Firebase 08:11:55.545386
stop
```

Figure 5.3 (b)

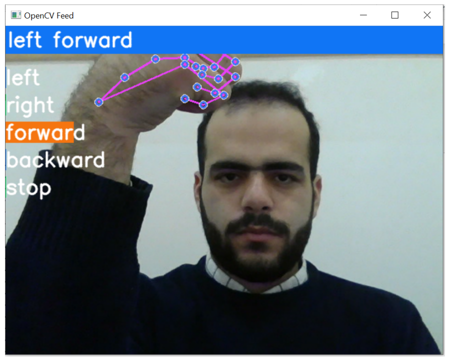
Gesture detection Model in-action



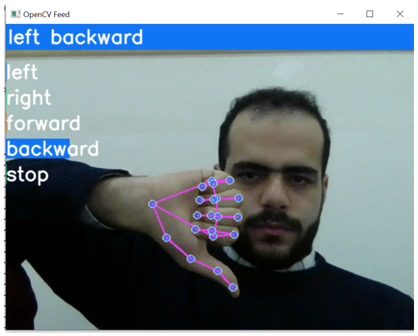
(a)



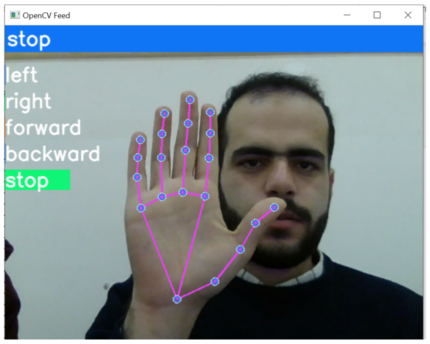
(b)



(c)



(d)



(e)

Figure 5.4 Dynamic Gesture Detection Module results

When it comes to the LSTM Neural Network model overall accuracy for a 30% testing set size out of 1150 gesture clips as our main data set, we were able to achieve 91.30%, but when it comes to the categorical accuracy our model achieved 96.14%. Figure 5.4 shows our results for all five gestures we are using in this project.

Chapter 6: Conclusions and Future Work

6.1 Conclusion

In this project we created an embedded system using deep learning and computer vision algorithms for real-time hand gestures detection and control of a robotic system with those detected gestures

A camera was installed to capture human gestures. Then the real-time captured stream of frames sent to our detection module in order to detect the intended gestures, the intended signal then be sent to the robot in order to control its movement.

The most fundamental part of the software system, the gesture detection module, was implemented utilizing deep learning with an LSTM Neural Networks.

6.2 Future Work

The general purpose of this project is to minimize the distance between the digital world and the physical world by building an application that controls a robot system using the hand gesture from a remote place. We can enhancement the current model in different aspects:

1. Use an illumination invariant hands key points extractor other than Mediapipe even if that required us to install some hardware on the user's hands if that was needed depending on the application.
2. We can use DC motors with an encoder to prevent impact on the speed when one of the motors is weaker than the other.
3. Extend the set of gestures for further controlling commands.
4. Optimize the workflow that involves pushing key points to Firebase to decrease the response time.

References:

[1] BAY AREA NEWS GROUP, Map: 33 people killed in California wildfires, 2020 season.” *The mercury news.*, 2 October 2020, https://www.mercurynews.com/2020/10/02/map-31-people-killed-in-california-wildfires-2020-season/#:~:text=Thirty%2Dthree%20deaths%20have%20been,Berry%20Creek%20and%20Feather%20Falls.

[2] Calma, Justine. “What you need to know about the Australia bushfires.” *The Verge,* 13 Feb 2020, https://www.theverge.com/2020/1/3/21048891/australia-wildfires-koalas-climate-change-bushfires-deaths-animals-damage.

[3] IBM Cloud Education. “Machine Learning.” *IBM*, 15 July 2020, https://www.ibm.com/cloud/learn/machine-learning.

[4] priyadharshini R. “Simple overview of machine learning.” *slideshare*, 5 aug 2017, https://www.slideshare.net/priyadharshiniR8/simple-overview-of-machine-learning

[5] IBM Cloud Education. “Neural Networks.” *IBM*, 17 August 2020, https://www.ibm.com/cloud/learn/neural-networks?mhsrc=ibmsearch_a&mhq=Neural%20Networks.

[6] IBM Cloud Education. “Deep Learning.” *IBM*, 1 may 2020, https://www.ibm.com/cloud/learn/deep-learning.

[7] Bhavsar, Dushyant. “Dispelling Myths: Deep Learning vs. Machine Learning.” *MERKLE*, 6 may 2020, https://www.merkleinc.com/blog/dispelling-myths-deep-learning-vs-machine-learning.

[8] Hochreiter & Schmidhuber. “Long Short-Term Memory.” *Neural Computation, 9*, 1997, http://www.bioinf.jku.at/publications/older/2604.pdf

[9] Alam, Abu Yousuf. “Designing and Implementation of a Wireless Gesture Controlled Robot for Disabled and Elderly People” *IEEE*, 4 April 2019, https://ieeexplore.ieee.org/abstract/document/8679290

[10] J. L. Raheja, R. Shyam, U. Kumar and P. B. Prasad, "Real-Time Robotic Hand Control Using Hand Gestures," *2010 Second International Conference on Machine Learning and Computing*, Bangalore, India, 2010, https://ieeexplore.ieee.org/abstract/document/5460699

[11] O. Köpüklü, A. Gunduz, N. Kose and G. Rigoll, "Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks," *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, Lille, France, 2019, https://ieeexplore.ieee.org/abstract/document/8756576

[12] Park, Kim, “HMM-Based Gesture Recognition for Robot Control” *2005 Iberian Conference on Pattern Recognition and Image Analysis*, Estoril, Portugal, https://www.researchgate.net/publication/221258820_HMM-Based_Gesture_Recognition_for_Robot_Control

[13] Singha, Laskar “Self co-articulation detection and trajectory guided recognition for dynamic hand gestures” *IET Computer Vision*, March 2016, https://digital-library.theiet.org/content/journals/10.1049/iet-cvi.2014.0432

[14] Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang, Yu-Bo Yuan, "Real-Time Hand Gesture Recognition Using Finger Segmentation", *The Scientific World Journal, vol. 2014*, https://doi.org/10.1155/2014/267872

[15] *Logitech*, https://download01.logitech.com/support/25173.1.0.pdf.

[16] “Raspberry Pi Camera Rev 1.3.” *Raspberry Pi Camera Rev 1.3*, https://circuit.rocks/raspberry-pi-camera.html.

[17]: Raspberry Pi 4. “Raspberry Pi 4 Tech Specs.” *raspberrypi*, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[18] “NVIDIA® Jetson Nano™ Developer Kit.”
https://docs.rs-online.com/5d0f/A700000006773861.pdf

[19] Joseph, Tom. “Arduino Uno.” *Arduino Uno Specification*,
https://www.tomsonelectronics.com/blogs/news/arduino-uno-specification. Accessed 02 Aug 2018.

[20] “Ultrasonic Distance Sensor - HC-SR04.” *Ultrasonic Distance Sensor - HC-SR04*,
https://www.sparkfun.com/products/15569.

[21] Mediapipe documentation,
https://google.github.io/mediapipe/

[22] Zhang, Fan, et al. “MediaPipe Hands: On-device Real-time Hand Tracking.”,
https://arxiv.org/pdf/2006.10214.pdf.

[23] *PyTorch tutorials,*
https://pytorch.org/

[24] “*tensorflow”,tensorflow.org/, 2020,*
https://www.tensorflow.org/

[25] OpenCV online documentation,
https://docs.opencv.org/4.1.0/

[26] NumPy online documentation,
https://docs.scipy.org/doc/numpy/user/whatisnumpy.html

[27] Graves, Alex. “Understanding LSTM Networks -- colah's blog.” *Colah's blog*, August 2015,
https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed 25 December 2021.

[28] Davis J., Shah M. (1994) Recognizing hand gestures. In: Eklundh JO. (eds) Computer Vision — ECCV '94. ECCV 1994. *Lecture Notes in Computer Science, vol 800*. Springer, Berlin, Heidelberg.
https://doi.org/10.1007/3-540-57956-7_37

[29] *WebRTC.org*,
https://webrtc.org/.

[30] Akbari, Mohammadreza. "Raspberry Pi Serial Communication (UART) w/ Arduino & PC." *Raspberry Pi Serial Communication (UART) w/ Arduino & PC*,
https://electropeak.com/learn/raspberry-pi-serial-communication-uart-w-arduino-pc/.