Swinburne University of Technology

Sarawak Campus

Faculty of Engineering, Computing and Science

RME 40006: Final Year Research Project 2

RME01: Roaming "Nagger"

Report

Bachelor of Engineering (Hons)

(Robotics & Mechatronics)

Chuah Ding Ken (101218820)

Semester 1, 2021

Declaration

I hereby declare that this report entitled **"Roaming Nagger"** is the result of my own project work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Swinburne University of Technology Sarawak Campus.

Name: Chuah Ding Ken

ID: 101218820

Date: 12th June 2021

Table of Contents

Lists of Tables and Figures

List of Abbreviations

Abbreviations	**Full Form**
SOP	Standard Operating Procedures
COVID-19	Corona Virus Disease-2019
AGVs	Automated Guided Vehicles
LIDAR	Light Detection and Ranging
SLAM	Simultaneous Localization and Mapping
IR Sensor	Infrared Sensor
AI	Artificial Intelligence

Acknowledgement

First and foremost, I, Chuah Ding Ken hereby would like to express my sincere gratitude to my project supervisor, Dr. Almon Chai for providing assistance and guiding me in the development of this entire research project.

Furthermore, I would like to thank the unit convenor of RME 40006 Final Year Research Project 2 (R&M), Dr. Lai Chean Hung for guiding me through the real engineering research, such as guiding me in report writing, referencing, and also workbook preparation, in order to complete the RME research project.

Lastly, I would also like to thank my family members and friends for supporting me in every circumstances.

Abstract

COVID-19 is an infectious disease which can spread through respiratory droplets and occasionally airborne method. With the implementation of Recovery Movement Control Order (RMCO), places like restaurants and shopping malls are able to open with strict Standard Operating Procedure (SOP) guidelines, such as setting the maximum number of people who are inside a premise, practicing social distancing and mask wearing measures. However, as the situation gets better, we as citizens tend to forget about the importance of social distancing and mask wearing practice. Thus, this project aims to design and develop a roaming "nagger" which produces nagging message when it detects someone who is not practicing social distancing and/or mask wearing measures. By doing so, the spread of the COVID-19 in community can be curbed and controlled, thus breaking the chain of COVID-19 infection. The roaming "nagger" is capable of navigates itself by detecting lines on the floor, wall sensing and also manual navigation which user can control the robot movement remotely. It is also capable of capture real-time video where it is sent from the vision systems to the user's PC. In this project, a conceptual design is developed using SolidWorks, where the height of the design reaches slightly above knee height. Software implementation, flowcharts, and source codes are presented as well.

1. Introduction

1.1 Problem Statement

As of June 2021, COVID-19 has been affecting 213 countries, including some territories. Some of the countries with more than 10 million cases are India, USA, and Brazil, with a handful of countries with 7-digit cases being listed behind. As for Malaysia, the number of cases rose to a 7-digit figure in this year. This shows that 3rd wave is approaching us.

With the continuous ongoing occurrence of COVID-19 cases in Malaysia, most of the states experienced a surge in cases, particularly in most of the Sabah districts and in West Malaysia. As the number of active cases increase exponentially in Malaysia, hygienic practice and social distancing has become the number 1 safety precaution, in order to reduce the spread of the virus. With the current condition, we as Malaysian citizens might need to adapt to a new norm in the next 1-2 years or possibly more. As for other countries which are severely hit by COVID-19, the time taken to return to a normal life is even longer.

This research project aims to increase the people's awareness of abiding SOP by following social distancing measures and face mask wearing practice by reminding them using nagging messages/phrases.

1.2 Research Questions

- How does the roaming robot monitor whether people abide by social distancing and proper mask wearing measures in an indoor area using vision-based sensing?
- How does the roaming robot navigate itself around an area?

1.3 Hypothesis

If people abide by SOP by following proper social distancing and mask wearing measures, then the risk of them getting infected will be reduced to very low. People who do not wear mask can spread the virus to others more at risk than themselves, if they are carrying the virus. This is because COVID-19 spreads via respiratory droplets and via airborne method.

People who follow proper social distancing and mask wearing will have extremely low risk of getting infected with the virus as wearing mask protects them from inhaling the virus while social distancing reduces the spread of the virus in the community.

1.4 Aims and Objectives

The aim of this project is to design and develop a user-friendly and cost-effective roaming robot which roams around an indoor area and nags citizens who do not abide by Standard Operating Procedures(SOP) such as mask wearing or social distancing.

The objectives of this project are:

- To implement the image processing algorithms to detect the absence of mask wearing or social distancing from citizens.

- To produce nagging message from the speaker to the citizens who do not abide by the Standard Operating Procedures (SOP).
- To increase the security of the roaming "nagger" by implementing an anti-theft system to prevent the robot from being carried away or stolen by any unauthorized person.
- To implement sensors for the robot to detect obstacles from the front and also the side when the robot navigates.

1.5 Project Scopes and Constraints

The project roaming "nagger" involves computer vision for detecting presence of social distancing and mask wearing using camera which sends real-time video to the user's PC via Wi-Fi transmission. By doing that, user can able to monitor the surrounding area remotely, which functions similarly as a CCTV feature. It consists of path navigation feature using sensors which guide the robot to roam an area. Additionally, security implementation is one of the scope for this project, as it aims to prevent the robot from being stolen by implementing digital failsafe features.

For the constraints, budget is one of the main identified constraints as some sensors are expensive which will exceed the actual budget. Hence, alternative sensor devices will be used for features such as path navigation. The second constraint is the weight of the prototype. Heavier prototype requires more powerful DC motors to move the robot around, which in turn also cause inconvenience when lifting up the robot. Hence, appropriate materials should be chosen, in order to reduce any extra weight on the robot.

1.6 Project Expenses

1.6.1 Mechanical Parts and Components

Components	**Unit Price (RM)**	**Quantity**	**Total Price (RM)**
MDF (30cm x 30cm)	2.20	2	4.40
Aluminium Composite Panel (30 cm x 30 cm)	11.00	1	11.00
Aluminium Composite Panel (45 cm x 30 cm)	13.50	4	54.00
Hollow Aluminium Square Tube (1 inch x 1 inch)	RM 22 per 20 ft	8 feet	8.80
L-shaped tube bracket connector	0.8	4	0.32
L-shaped metal bracket (50 mm x 50 mm)	1.70/4 pcs	4 pcs	1.70
L-shaped metal bracket (40 mm x 40 mm)	1.50/4 pcs	12	4.50
Shaft (40 cm, 8mm diameter)	17.50	1	17.50
Shaft (10 cm, 8mm diameter)	5.00	1	5.00
Motor Bracket	14.00	2	28.00
Motor Shaft Coupling (6 mm to 8 mm)	4.14	2	8.28
Pillow Bearing (KFL08, 8 mm bore)	5.80	4	23.20
Rivet (3.2 mm)	1.79/24 pcs	8	0.60

Wheels (65 mm diameter)	10.00	4	40.00
Wheel Hex Coupling	5.00	4	20.00
Machine Screw (M4*50, crosshead, flat screw)	0.37	8	2.96
Machine Screw (M4*40, crosshead, flat screw)	0.4	32	12.80
Machine Screw (M4*20, crosshead, flat screw)	0.2	16	3.20
Wood screw (10 mm)	0.05	8	0.40
Machine Screw (M3*35, cross, flathead screw)	0.46	1	0.46
Machine Screw (M3*25, cross, panhead screw)	0.33	20	6.60
Machine Screw (M3*20, cross, panhead screw)	0.30	6	1.80
Machine Screw (M3*10, cross, panhead screw)	0.21	8	1.68
Machine Nut (M3)	0.05	36	1.80
Machine Nut (M4)	0.06	56	3.36
3D-printed filament	RM 50/kg	300 g	15.00
Subtotal			277.36

1.6.2 Power Supply Components

Components	**Unit Price (RM)**	**Quantity**	**Total Price (RM)**
12 V battery pack	69.00	1	69.00
2-way terminal block – DC jack	3.80	1	3.80
Buck converter (12V/5V, 3A) (Adjustable current and voltage)	30.00	1	30.00
USB A to Type C Cable	8.00	1	8.00
Subtotal			**110.80**

1.6.3 Raspberry Pi 4B and its Peripherals

Components	**Unit Price (RM)**	**Quantity**	**Total Price (RM)**
Raspberry Pi Model 4B	239.00	1	239.00
Pi Camera Module (8MP)	110.00	1	110.00
Mini Speaker	20.00	1	20.00
40-pin GPIO cable extension (40-pins)	4.50	1	4.50
GPIO Extension Board	15.00	1	15.00
Subtotal			**388.50**

1.6.4 Electronics Sensors and Components

Components	**Unit Price (RM)**	**Quantity**	**Total Price (RM)**
Line Detection IR Sensor	2.50	3	7.50
PIR Sensor	2.90	1	2.90
Ultrasonic Sensor	2.90	2	5.80
Buzzer	3.60	1	3.60
L293D Brushed DC Motor Driver	2.70	1	2.70
MCP 3008	20.00	1	20.00
DC Motor (12 V)	55.00	2	110.00
Subtotal			**152.50**

1.6.5 Connectors and Circuit Boards

Components	**Unit Price (RM)**	**Quantity**	**Total Price (RM)**
2-pin headers	0.16	25	4.00
3-pin headers	0.24	5	1.20
4-pin headers	0.32	4	1.28
2-pin housings	0.06	25	1.50
3-pin housings	0.09	5	0.45
4-pin housings	0.12	4	0.48
Metal crimp pieces	0.10	81	8.10
Vero stripped board	2.00	2	4.00
Subtotal			21.01

Total Budget = $277.36 + 110.80 + 388.50 + 152.50 + 21.01 = RM\ 950.17$

1.7 FYP2 Gantt Chart

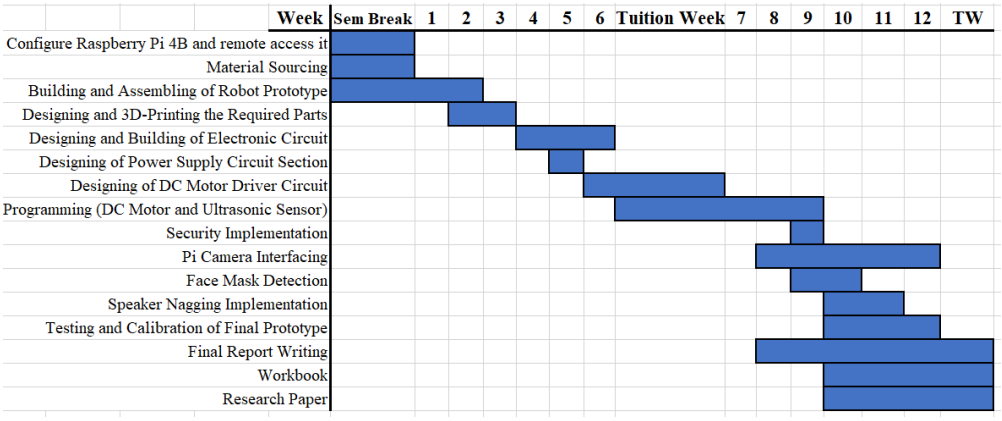


Figure 1: Gantt Chart for FYP2

2. Literature Review

2.1 Path Finding and Navigation Methods of Robot

In nowadays, most robots deployed in a public area have being installed with sensing technology devices, particularly laser or camera-based technologies. Below are a few path finding techniques for a roaming robot.

In many automated guided vehicles (AGVs), mobile robots are programmed to roam around an area or transport materials in a facility. According to Loadamaster, there are 3 main ways of how a mobile robot can navigate itself, with the 1st method being Laser Navigation.

Figure 2 below shows the 6 different types of path navigation implemented by mobile robots. [1] Method a (laser triangulation) and b implement laser navigation as they use laser device to detect ceiling mounted bar codes and reflectors mounted on the wall, these methods are commonly used in warehouse area. Laser navigation uses laser light as a emitter to strike at the reflectors mounted on the wall, hence the position, distance, and angle can be detected. Whilst Natural Navigation is more like a flexible but more expensive implementation compared to the other methods. One of the most common techniques of Natural Navigation is SLAM (Simultaneous Localization and Mapping). Another method of navigation is using Tape Follower Method. In this method, tapes are attached on the floor as the path, it can be either open-loop or closed-loop path. For closed-loop path, the robot moves around an area with no turning back of route. For open-loop path, when the robot reaches the end of the path, it simply turns a 180 degree back and go back for the path it follows previously.

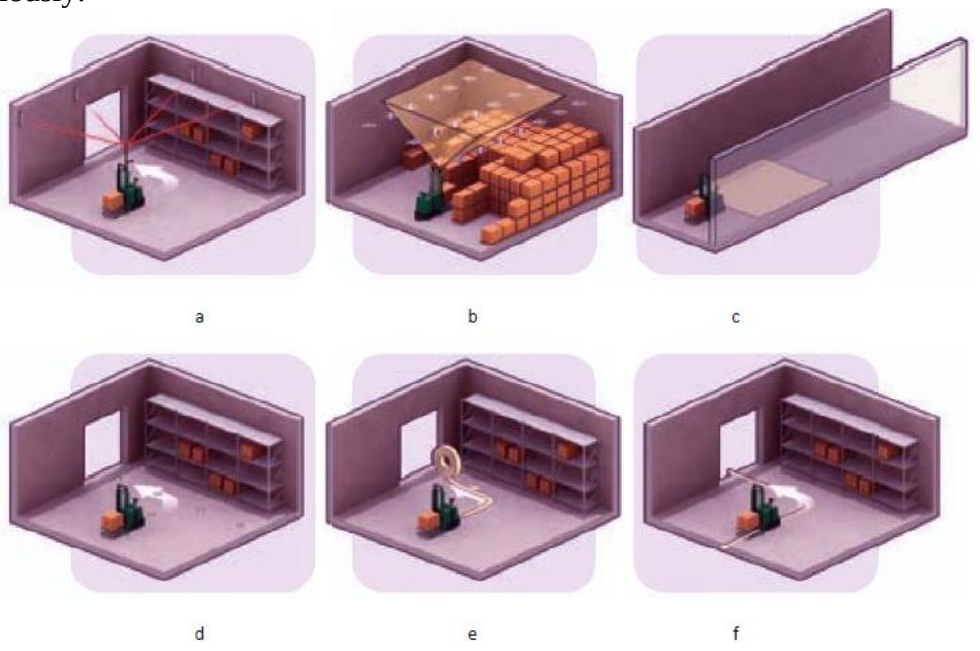


Figure 2: Illustration of 6 navigating methods for an automated guided vehicles (Shneier, 2015) [1]

2.1.1 Path Finding and Navigating using LIDAR Mapping/SLAM Algorithm

LIDAR is the acronym for Light Detection and Ranging. It is a remote sensing method which uses light as the emitter to measure distance in the form of laser, and it can generate 3-D information about its surroundings. For example, in a public indoor area, when the light pulse emitted by the LIDAR device hits a target (people, wall, or obstacles), it will bounce back and return to the LIDAR sensor. The distance can be approximated accurately by considering the time taken for the light to be emitted out and return back. Typically, a LIDAR sensor is mounted on a rotating platform stand which can be able to capture many readings around a 360-degree sweep. The maximum sampling rate of the device is around 1000 samples per second, hence it is able to capture sufficient data.

In other terms, LIDAR implements the algorithm of SLAM (Simultaneous Localization and Mapping). For SLAM method, a mobile robot will perform an estimation of map in an environment while simultaneously localizing itself with respect to the map estimated [2].

[5] Unlike vision-based SLAM algorithm, laser-based SLAM algorithm is more capable of performing indoor mapping in a more accurate way. It implements LIDAR which acts as a range finders to do laser scanning in 2D, it provides high resolution real-time images with high sampling rates, which contributes to a smoother image captured.

The LIDAR sensor model TFMini is shown in Figure 3 below:

Figure 3: LIDAR Sensor device (TFMini - Micro LIDAR Module), which can detect distance up to 12m

In Figure 4 below, the LIDAR performs a 2D Mapping which updates its surroundings when it is moving. The longer the robot moves around, more data will be collected. From the 2D Map created by the LIDAR, black dots represent data that has been obtained from LIDAR and is being updated into the map. Greenish-grey region represents the unmapped area, whereas light grey dots are free spaces which robot can move around, and white dots are the data scanned from the LIDAR.

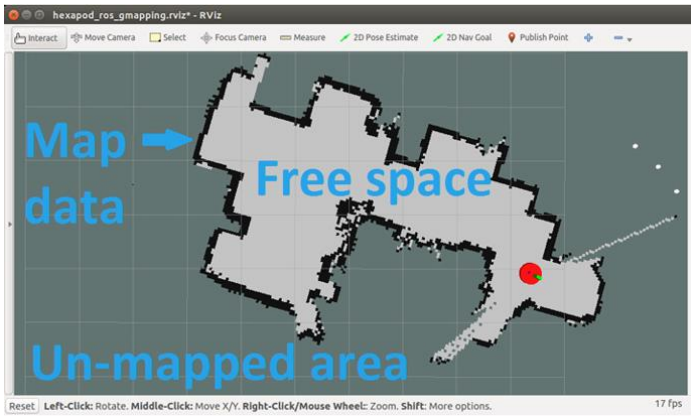


Figure 4: 2D Mapping Interface using LIDAR device

After completing the mapping process, next thing is to perform both navigating and obstacle avoiding task. This can only be done when the map is built. During these 2 processes, it consists of:

- Planning of path.
- Detection of newly added obstacles such as people moving around or something is placed in the middle.
- Updating path map every time a new obstacle has been detected.

The illustrations of how navigation and obstacle avoidance are explained below:

Step 1: After completing the mapping process, the user clicks the final destination for the robot to move to there. (Illustrations shown in Figure 5 below)

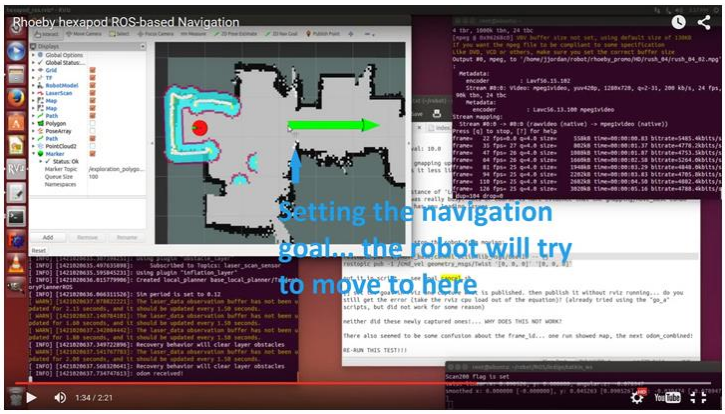


Figure 5: Setting of the final destination of the robot after it has done with the mapping task [3]

The red dot represents the location of where the robot is, the purple and light blue dots are referring to the object inflation, which is where obstacles are detected immediately upon finishing the mapping task. For the object avoidance, instead of nearly bump into the obstacle and dodge it, the object inflation will make the purple/light blue region appears larger in area for safety margin purpose. This is due to the fact that some obstacles are moving when the robot is moving as well, such as people moving around it. Hence, the purple edged dots indicate the danger zone where collision is likely to occur.

Step 2: Beginning of the Robot Movement (Illustrations shown in Figure 6 below)

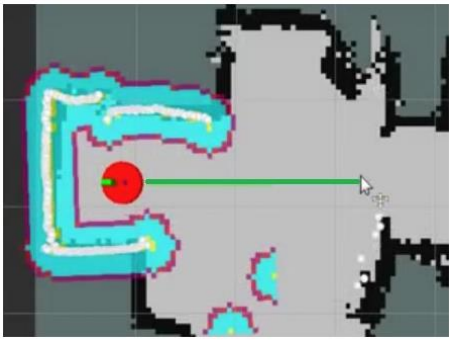


Figure 6: Route Indicator of the Robot (Robotics Tomorrow, 2015) [3]

Once the destination has been set, the robot will start moving from the initial point to the area clicked by the user. The green line indicates the path that the robot is supposed to follow.

Step 3: Introducing New Obstacle in the middle of the route (Illustrations shown in Figure 7 below)

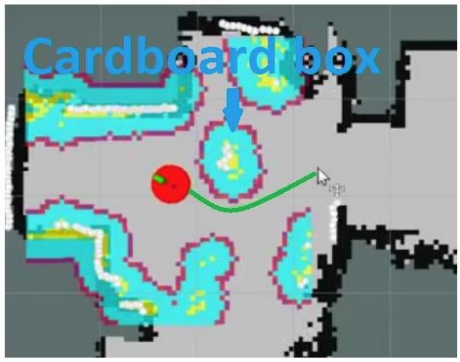


Figure 7: New Obstacle being added into the map (a card box)

In this scenario, a new obstacle has been placed in the middle which blocks the path of the robot. Hence, the new obstacle is represented by a light blue blob with purple edge, and it will be updated to the map for the robot to avoid it. As a result, the robot will move away from the obstacle and continue its route towards the final destination set by the user.

For SLAM Algorithm, it consists of 2 different types of mapping systems. The first one is a Terrestrial Laser Scanner, it is fixed in a location and captures data from surroundings when the laser sensor is rotated 360-degree. This mapping system has a higher accuracy as the alignment is already being calibrated with respect to the laser scanner. The second mapping system is using a mobile mapping system. This system possesses the same mechanism as previous mapping system, however it is not fixed at a point. Instead, the device is moving around while performing data capturing, causing some errors in the alignment process and thus decreases the accuracy of the final point. For this project, since the roaming "nagger" will be moving around an area, hence it will have some accuracy error in capturing data surroundings.

However, there is a solution which can be able to greatly reduce the accuracy error of SLAM algorithm. The method is called loop closure [4], loop closure is the method where the robot will return and overlap with the previous point where it had been scanned. By implementing this, the accuracy errors such as tracking error can be corrected, as shown in Figure 8 below:

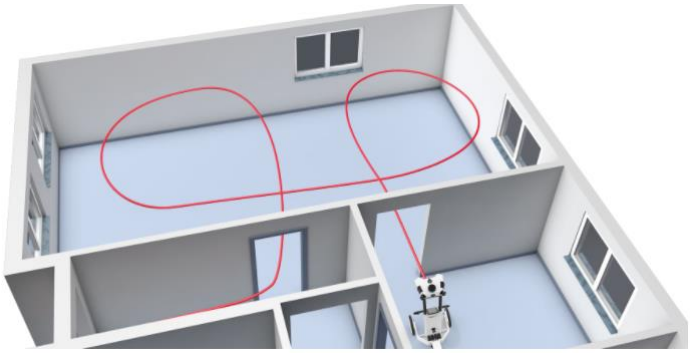


Figure 8: Loop Closure being carried out, as shown in the red line (Higgins, S, 2020) [4]

2.1.2 Path Finding and Navigating using Line Following Method

Line Following Robot is one of the most important yet common navigation for robot to roam around an area. On the floor, either black or white lines are drawn due to the behaviour of the sensors that it can only detect black or white colour. For the sensors, IR Sensor are used to detect the line on the floor. When IR Transmitter emits ray onto a surface, the reflected ray will be bounced back and return to the IR Receiver. The amount of rays reflected back depends on the contrast of the line. Figure 9 below shows how the IR sensor works in an indirect manner.

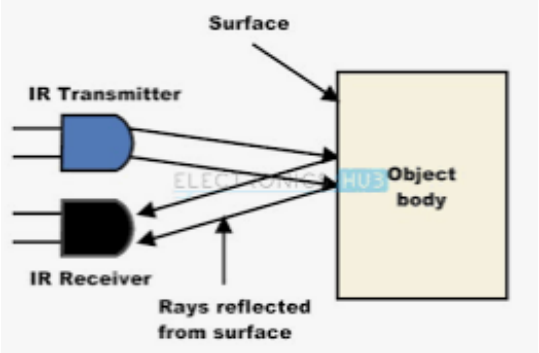


Figure 9: Illustration of how IR Sensor works

According to Chowdhury [6], a line follower robot can be implemented using 4 pairs of sensors attached adjacent to each other with a suitable distance between it. By implementing this amount of sensors, the robot can able to detect T-junctions and 90 degrees bends. The robot will then roam around the area with the guided line located between the sensors. There are a few scenarios where the robot will encounter. Figure 10 below shows the normal condition when the robot moves in a straight direction. Hence, all the four IR sensors will detect the black region, and both motors will run at full speed.

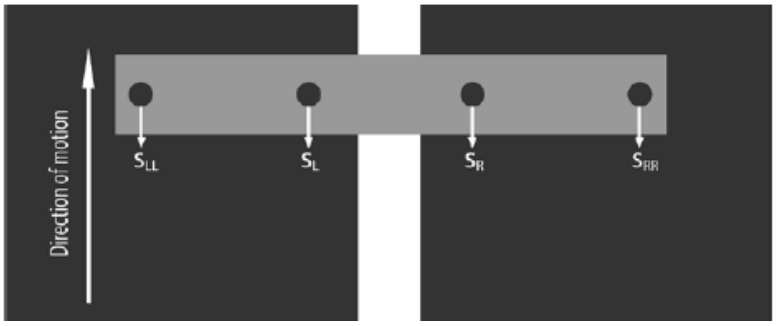


Figure 10: Top View of how the setup of Line Follower Robot (Chowdhury, N, 2017) [6]

When the robot starts moving, both sensors will detect the outer path which is not the black line. When there is a change in direction of the line (e.g. turning to left/right), one of the sensors will detect the lines on the floor because the robot moves in a straight line while the lines are changing direction. In the case where either the middle left or right sensor detects the line path, the robot needs to turn to the opposite direction until all the sensors are not detecting the black line. Figure 11 and 12 on the next page shows the rotation of the robot when either left/right sensor detects the line on the floor.

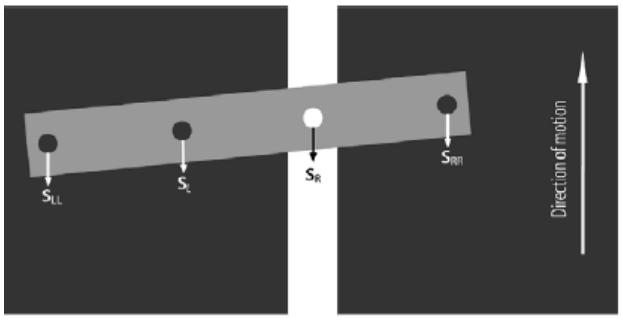


Figure 11: Scenario when the robot needs to rotate to the right. (Chowdhury, N, 2017) [9]

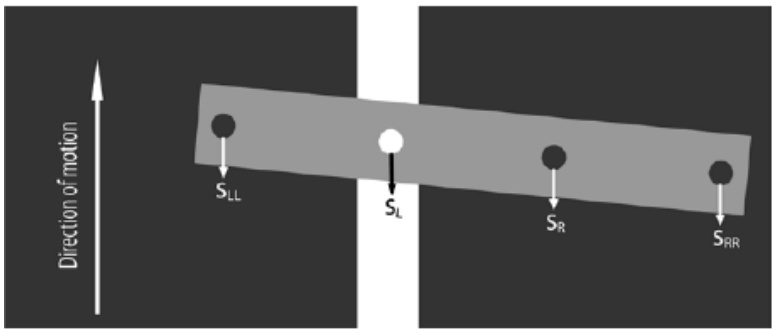


Figure 12: Scenario when the robot needs to rotate to the left. (Chowdhury, N, 2017) [6]

Also according to Chowdhury[6], the distance between placement of sensors affects the robot stability and the tendency of the robot to roam accurately. If the placement of the robot is closer to each other, then the roaming robot will have to correct its path more frequently as the marginal distance is smaller. However, if the sensors are placed further from each other, it will tend to move at more offset distance before the IR sensor detects and correct its path. Due to larger margin, the robot sometimes may skip the current path and follow other black lines if the lines are being placed at close to each other. Hence, in order for sensors to be further placed with each other, the lines must not be built close to each other. In shorter words, the longer the distance between sensors, the less stable of the robot and less rigid of the robot movement will be.

For the Line Following method proposed by Chowdhury [6], it uses 4 pairs of IR Sensor Module. The implementation made by Chowdhury is able to overcome some challenges such as 90 degrees bends and even T-junctions, this is because if the number of IR sensors used is decreased, the robot will not be able to detect critical paths in an effective way [9]. The placement of sensors will affect the path detection and navigation in different places. For example in a long hallway where the space is rather narrow, the distance between the sensors needs to be narrower due to limited spaces on the side of robot. However in an area with larger space, the sensors can be placed at a further distance, provided that the lines are not too close to each other. For the type of path, open-loop path is more suitable in long and narrow hallways, and for larger areas, both open-loop and closed-loop paths are flexible in this case.

2.1.3 Path Finding and Navigation using Wall-Sensing Method (Additional Method besides Line Following)

In some areas of airport where tapes are not allowed to be attached on the floor, the roaming "nagger" must rely on other obstacles as reference point. One of the most suitable reference point will be the wall, as the robot can rely on the wall and navigate itself along the wall with a certain distance between. The main objective of wall following navigation method is to maintain a fixed distance from the side walls**,** and 2 DC motors are used to adjust the orientation of the robot if the robot is too close or too further away from the wall. Ultrasonic sensor is used to detect the distance between the robot side and the wall.

Figure 13 below proposed flowchart of wall sensing robot navigation by Wei [7]:

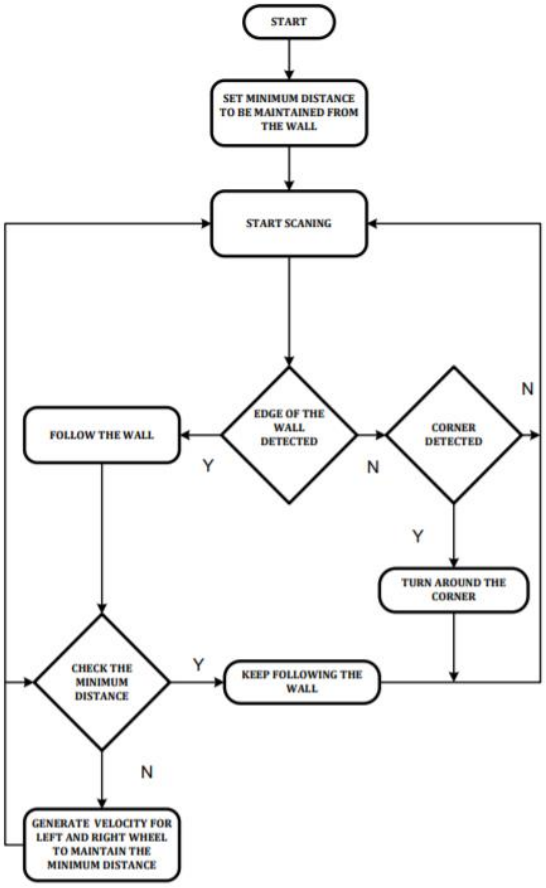


Figure 13: Software Algorithm Proposed by Wei [7]

For normal linear movement, both DC motors move in the same direction and speed. However, when an obvious change in the distance between the wall and robot is detected, the 2 DC motors will move in different direction, causing the robot to rotate along its midpoint. For constant movement of the robot, a range of distance is set for the robot to move along, so that the robot will not always adjust its orientation and thus decrease its velocity.

According to Wei [7], the implemented solution is able to detect wall and corner, most of the time it will follow the wall, unless a corner is detected, the robot will only turn. It also comes with a minimum threshold point where it only rotates to the left/right when the distance detected between the wall and the sensor is below the minimum value set.

2.1.4 Review of SLAM Algorithm and Line Following Method for Path Finding and Navigating

Table 1 below compares the advantages and drawbacks/limitations of using SLAM algorithm:

Table 1: Comparison between the Pros and Cons of 2 different Navigating Method

Navigation Methods	**Advantages**	**Disadvantages**
SLAM Method (Natural Navigation)	• Able to capture data at a fast rate when the LIDAR device is rotated 360 degrees. • Does not need a large memory size. • Suitable for large and wide spaces such as airports and shopping malls due to the fact that it can perform loop closure more easily.	• Requires more calibration and longer time taken for the mapping process to finish before the robot can navigate itself. • Loop closures are difficult to be implemented in long and narrow hallway. • More expensive due to the cost of LIDAR device.
Line Following Method (Tape Following Navigation)	• The line following algorithm is easier to be implemented. • The cost of implementation is cheaper as IR sensors are cheaper than LIDAR device. • The path can be set in such a way that it can cover most of the area with minimal blind spots.	• Distance between sensors need to be adjusted in different types of place. • It requires an external object avoidance sensor(e.g. PIR Sensor) as line follower algorithm does not have obstacle detection. • Requires regular replacement of black/white tapes due to constant wear and tear of the tapes. • Cannot be used in carpet terrains in indoor areas.
Wall Following Method	• Easier to be implemented • Cost of implementation is low	• Not suitable for places which are very big, where only limited path can be covered.

From the comparison, Line Following Navigation is slightly more suitable than SLAM Navigation due to its less complexity and more direct of algorithm. This algorithm uses cheap cost because the price of an IR Sensor is far less than a LIDAR device. Despite the SLAM Algorithm can be able to detect obstacles while navigating itself to another point, the LIDAR device is more expensive than IR Sensors. On the other hand, the roaming "nagger" is set to be used in indoor areas of even concrete terrain. Hence, line following method is more suitable for the roaming "nagger" to be implemented. Whereas for wall sensing method, it can be used together with line robot method in such a way that wall sensing method is more suitable for narrower place such as in a corridor. Hence, both of these implementations should be used together for the robot.

2.2 Types of Vision Systems

Long before machine and cameras are invented, human used the traditional method in inspecting an object or a part by using their own eye. Despite the fact that human eye is able to execute a task better than machine in some cases, they tend to get exhausted after a series of non-stop routine-based task, such as carry out object inspection. This applies the same in detecting face mask and social distancing in public areas, where authorities are able to spot if anyone did not wear a mask or not practicing social distancing immediately. However, due to current COVID-19 pandemic, monitoring of such conditions can be risky, especially if the citizen who is not wearing mask/following social distancing is exposed to the virus but showing no symptoms. Therefore, monitoring of such scenario by a roaming robot is a better and much safer option. Figure 14 below shows how a typical industrial vision system works.

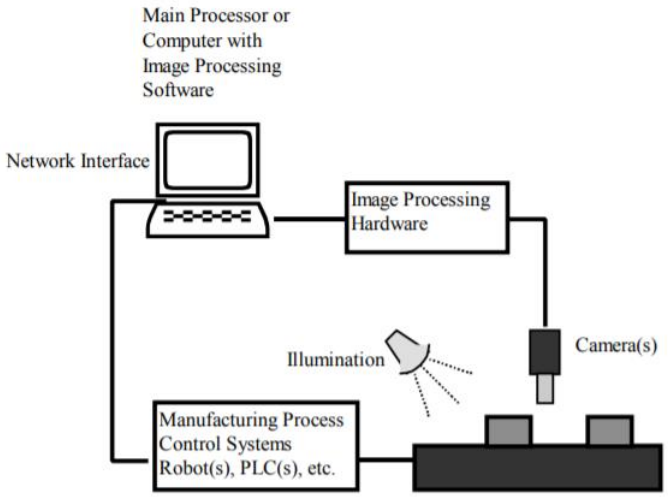


Figure 14: Inspection System used in Industrial Robot (Malamas, 2003) [8]

In order to interface a vision system, a computer is needed to process the input images by integrating image processing and software into it.

Machine vision is a subset of artificial intelligence where it integrates image processing and automation together to perform object detection in an automated manner. Machine vision can be either 1-D, 2D or 3D vision systems. In this section, 2D vision systems and 3D vision systems will be discussed, as well as their uses in industry, advantages and drawbacks.

2.2.1 2D Vision Systems

2D vision systems are vision system which analysis an image directly perpendicular from the image base. In other words, when performing 2D image analysing, the horizontal and vertical coordinates, as well as the orientation of the object will be known.

2D vision systems are the most commonly used inspection cameras that carry out area scans. In this vision system, a camera will be installed above the position or its capturing view line is perpendicular to the plane of the object. As shown in Figure 15 below, the object is positioned in 3 coordinate axis, which are x, y and z-axis.

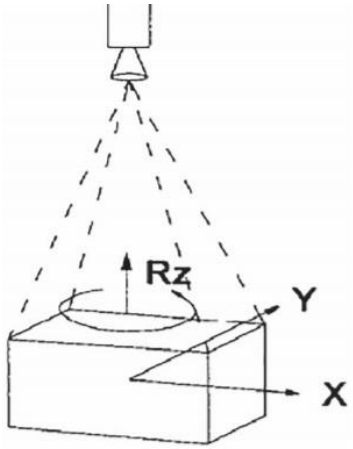


Figure 15: Schematic of how 2D Vision System Works (Cui, 2019) [9]

2D vision system provides area scans that work well for discrete parts [10], it is able to analyze images with various resolutions. However, this type of vision system does have limitations. The accuracy of the image detection can be affected if changes in ambient conditions or artificial lighting occurs. Therefore, 2D vision system is somehow light-sensitive. Another issue is that 2D machine cannot handle additional height difference if the image is placed flat on the ground. Therefore, 2D vision system is not suitable to be used in a pick and place application or detection of any image involving more than 2 axis.

2.2.2 3D Vision Systems

Unlike 2D vision system, 3D vision system provides 6 degree of freedoms instead of just 3. 3D vision system can be explained using analogy of our human eyes (2 eyes). When the object is being positioned, the 3D vision system utilizes 2 cameras with different orientation angle to capture the image simultaneously [11]. Due to its 6 degree of freedom positioning system, it is able to provide x, y, and z-axis parameter with the rotational information in these 3 respective axis, as shown in Figure 16 below:

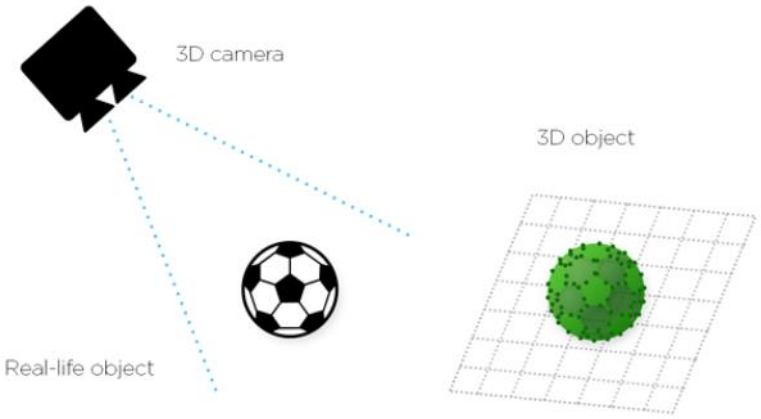


Figure 16: Illustration of how the Image is projected on the plane using 3D-vision system (Schumann-Olsen, 2021) [11]

3D vision system possesses several advantages compared to 2D vision system. First and foremost, it has a wider range of usage due to the additional orientation that can be detected. By utilising 3D vision system, more than one object can be analysed at the same time by using reference coordinate system. Hence, it can be used in a more complex and uncontrolled environments.

On the other hand, 3D vision system does have some limitations as well. Due to its nature of image processing method, 3D vision system takes more time to execute the task, and it also consume more power as it is very software intensive. However, since 3D [11] vision system has wider range of parameter detection due to the 6 degree of freedom positioning, 3D vision system is more recommended in executing object detection, particularly when the image is a live stream video that consists of objects moving around.

3. Methodology

3.1 Mechanical Design and Implementation

3.1.1 DC Motor Sizing Calculations

The estimated mass of the robot is around 6 kg, including the components which are to be installed inside and the mechanical chassis parts. For the speed of the roaming "nagger", it is set to be moving for 0.5m for 1 second.

Applying the kinematic equation:

$$s = v_o t + \frac{1}{2}at^2, where\ s = 0.8\ m, t = 1\ s, v_o = 0\ ms^{-1}$$
$$0.8 = \frac{1}{2}a(1)$$
$$\therefore a = 1.6\ ms^{-2}$$

Applying Newton's second law of motion:

$$F = ma = 5.5(1.6) = 8.8\ N$$

Assume an efficiency of 0.7, and the diameter of the wheel is 65 mm or 0.065 m. The torque is calculated using the equation:

$$\tau = \frac{Force \times Wheel\ Radius}{Efficiency} = \frac{8.8 \times 0.0325}{0.7} = 0.41\ Nm$$

For safety factor of 2,

$$Optimal\ Torque = 2(0.41) = 0.82\ Nm$$

In most DC motors sold in Cytron Technologies online store, the torque is given in kgf cm (kilogram force centimetres).

Therefore 0.82 Nm of torque is equivalent to 8.36 kgf cm. Therefore, the DC Motor model SPG50-100K which has a rated torque of 9 kgf cm is sufficient for the robot moving speed.

3.1.2 Finalized Design of the Prototype (SolidWorks)

Figure 17 below shows the finalized prototype drawn using SolidWorks with labelling of each parts.

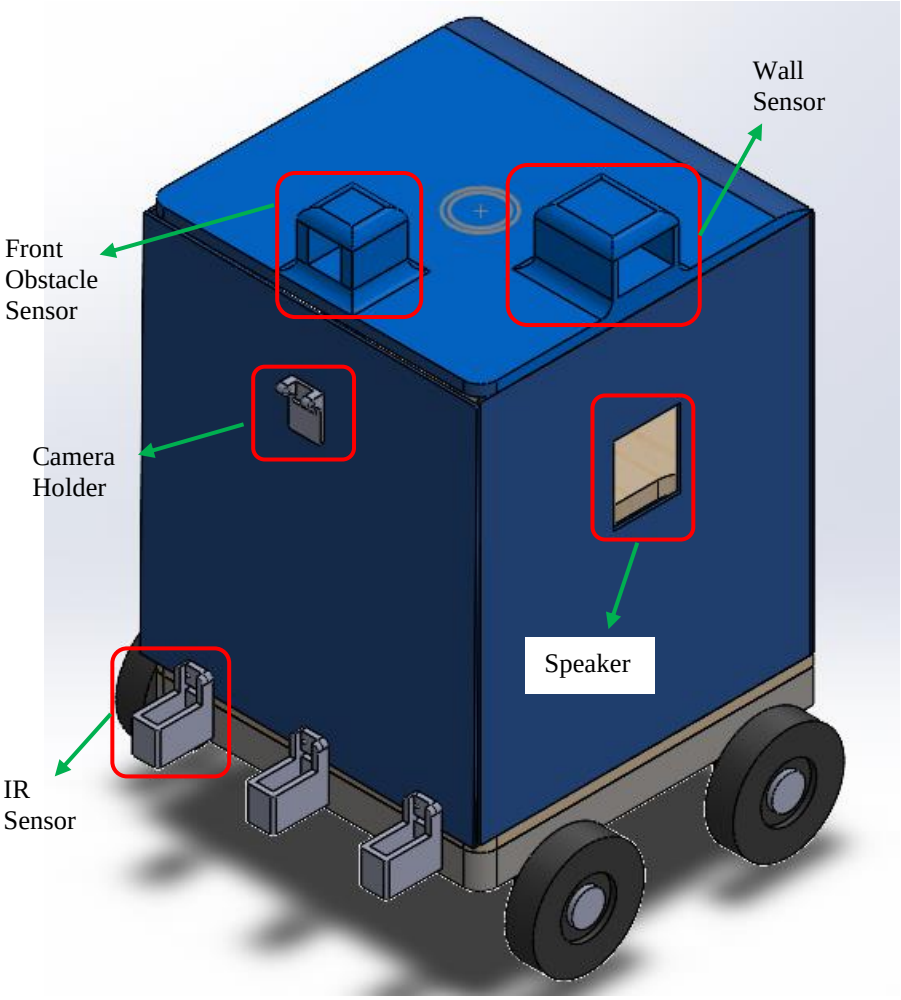


Figure 17: Exploded View of the Prototype.

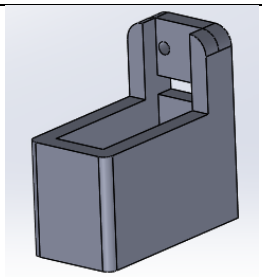
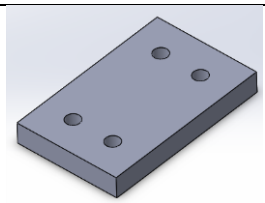
In Figure 17 above, the prototype consists of 2 level of components which are supported by aluminium square hollow tubes of 1 in. by 1 in. For the wheels, they are made of rubber with metallic part as the rim, and have diameter of 65 mm. In figure 1 above, the blue colour parts are made of aluminium of thickness 3 mm. For side covers, square holes are but so that the output sound from the speaker can be heard clearly.

3.1.3 Design Considerations of 3-D Printed Parts

IR Sensor Holder

Table 2 below shows the functions of each 3D-printed parts used for the IR sensor holder, with the 3D model for the parts shown in Figure 18 and 19 inside the table.

Table 2: Parts used for IR Sensor Holder

Components	Diagram	Functions
IR Sensor Slot Holder	 *Figure 18: IR Sensor Slot Holder using 3D Modelling*	For IR sensor placement
IR Sensor Extend	 *Figure 19: IR Sensor Extension*	Extend the sensor holder from the robot chassis so that the sensor is placed at distance as close as to the ground (2 cm to 3cm).

For the IR sensor holder, two rectangular holes are extruded cut so that the IR sensor and the vero stripped boards can be inserted properly, which can be illustrated in Figure 20 below:

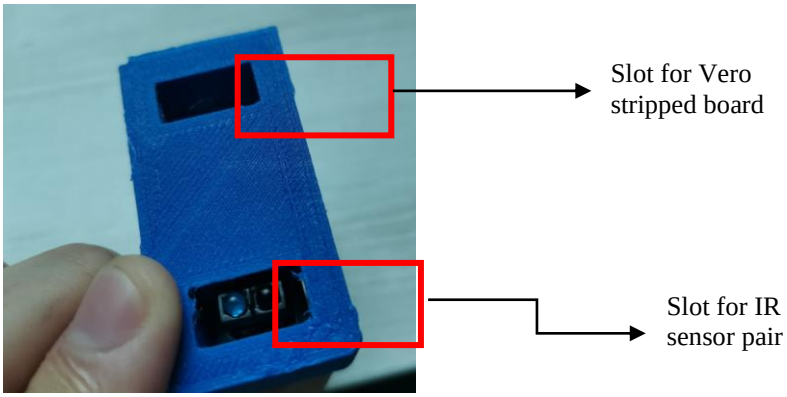


Figure 20: Position of Each Extruded Hole for IR sensor pair and vero stripped board

The width of the slot hole for IR sensor pair and vero stripped board are 8 mm and 7.5 mm respectively, so that the IR sensor pair can be fitted properly into the hole with some clearance left behind. The dimension of the IR sensor (from front to the tip of 4-pin header) is measured at 37 mm x 14 mm. Since 3-D printed parts will have some minor discrepancy when printing, hence the top hole is extruded at a dimension of 40 mm x 15 mm for proper insert and taking out of IR sensor whenever necessary.

DC Motor Platform

When the DC motor is connected to a DC motor bracket, there is still some spaces before the motor bracket can be installed on the robot base. Hence, a motor platform is required to be connected with the bracket and the robot base, so that the shaft can be inserted through the bearing, as well as connecting with the DC motor properly. Figure 21 below shows the 3D-model of the DC motor platform drawn in SolidWorks.

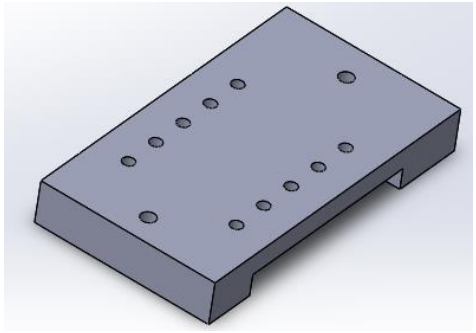


Figure 21: 3D Modelling of the DC Motor Platform

From the 3-D modelling, the platform consists of 10 holes of approximately 3 mm and 2 larger holes of slightly more than 4 mm. The smaller holes are used to connect the motor bracket and the platform whilst the larger holes are used to connect the platform to the base of robot. The motor platform is of maximum thickness of 12 mm, which are located at both ends of the platform, and a thickness of 6 mm so that a screw of length 10 mm is able to secure the bracket to the platform. Figure 22 shows the side view of DC motor platform with the direction of DC motor applied to it

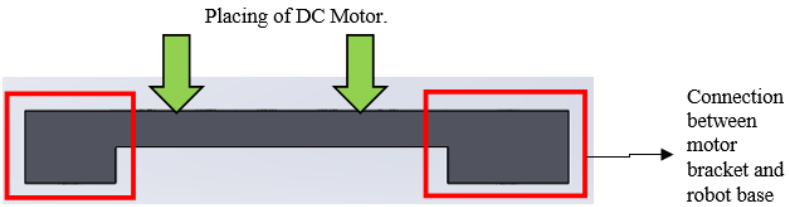


Figure 22: Side View of DC Motor Platform

The reason of implementing 10 smaller holes instead of the typical 4 holes is to allow adjustment of positions of DC motor, thus a more flexible design. Figure 23 below shows the location of where the platform is being placed beneath the robot base.

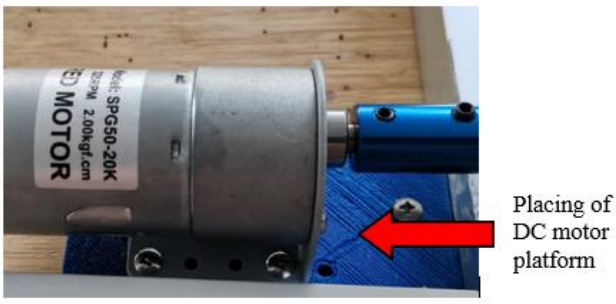


Figure 23: Placing of DC Motor Platform at the robot base

Ultrasonic Sensor Holder

This ultrasonic sensor holder is specifically designed to secure the ultrasonic sensor at the base of the robot. It consists of 4 extruded cut screw hole to allow the screw to be fit through the robot base and secured it with nut. The half bridge at the top surface of the sensor holder serves as a purpose to hold the sensors and make sure that it is held upright with minimal bending angle.

For the other ultrasonic sensor holder, it is designed such that the sensor is held upright and perpendicular to the ground. Minimal clearance is set in order to allow the ultrasonic sensor to be tightly fitted into the slot without slipping out from the holder. Table 3 below shows the ultrasonic sensor holders used in different locations, with the 3D model shown in Figure 24 and 25 below:

Table 3: Ultrasonic Sensor Holders used in different areas of robot

Base of Robot	Top of Robot

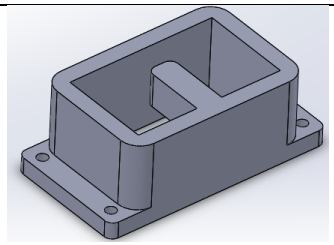
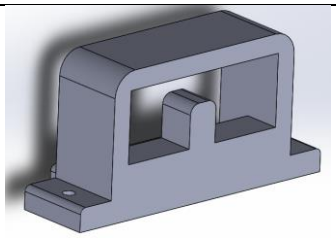

	

Figure 24: Ultrasonic Sensor Holder at the robot base	*Figure 25: Ultrasonic Sensor Holder on top of robot*

Figure 26 and 27 show the placement of ultrasonic sensor beneath the robot base and top of robot respectively.

Figure 26: Placement of Ultrasonic Sensor Holder (Robot Base)

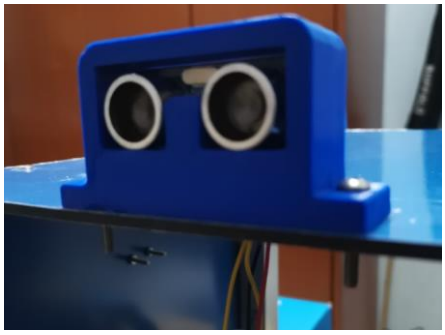


Figure 27: Placement of Ultrasonic Sensor Holder (Top of the Robot)

3.1.4 Assembly and Building of Prototype

Stage 1: Building of Base

During the 1st stage of assembly, the base is built by connecting the aluminium square hollow tubes with the L-shaped bracket, followed by inserting rivets between the bracket and the aluminium bar to hold on its position, as shown in Figure 28 below:

Figure 28: Connecting Aluminium Square Hollow Bar with backet using rivet

Upon securing the aluminium square frame, the next step that must be done is to drill 8 holes of diameter approximately 4 mm at the 4 sides of the aluminium frame. The purpose of doing so is to connect bearing with the frame using 8 screws (M4*50) and 8 M4 nuts. Figure 29 below shows how the bearings are connected with the frame.

Figure 29: Securing the bearing and the frame using screws and nuts

The resultant assembly needs to be connected with the wooden board of thickness around 10 mm. Therefore, M4 screws of length 50 mm is sufficient for it to connect through the bearing, frame, and the wooden board with some more length left for nut securing.

Step 2: Building of 1st level with 4 aluminium hollow bars.

After the wooden board is assembled together with the frame, the 4 aluminium bars are cut and are required to be placed on the 4 corners of the board. The aluminium bars are connected with the board using L-shaped bracket of length 50 mm each. 2 holes are drilled on each aluminium bars, so that screws can be fitted through the hollow bars. Before drilling, markings are made so that the L-shaped bracket can be connected properly with the aluminium hollow bar without any misalignment, as shown in Figure 30 below:

Figure 30: Pencil marks on the bar before drilling.

In Figure 31 below, despite only 2 holes are required for each aluminium hollow bar, the 3rd drilled hole will be used later, which is for securing the outer cover with the aluminium bar using screws. The screws used to connect the aluminium bar and the L-shaped bracket are of dimension M4*40, since the width of the aluminium is about 2.5 cm (or 1 in.).

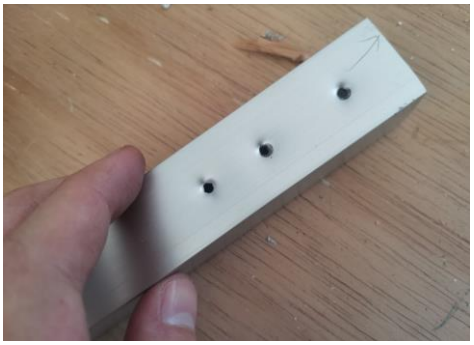


Figure 31: Drilling on the Aluminium Hollow Bar

In Figure 29, the 4 aluminium bars connected with the brackets will be secured on the wooden board using 8 wooden screws of length 10 mm. These assembly will act as the 4 pillars which connect between the ground level of compartment and the 2nd level of compartment.

Figure 32: Connecting L-shaped brackets and aluminium bar

In Figure 33 below, after placing the 4 pillars on the 1st level wooden block, a 2nd wooden block is placed on top of the 4 pillars as the 2nd level compartment. Here, 8 screws of dimension M4*25 are used to secure the L-shaped bracket and the 2nd level wooden base.

Figure 33: Securing 2nd level wooden block with brackets

For the other 4 pillars, same process will be carried out as the 1st level pillars.

Table 4 below lists out the size of the screws used in the robot chassis skeleton (excluding top and side covers):

Table 4: Size and quantity of the screws used in the robot chassis skeleton (excluding top cover)

Size of Screws (Dimension)	**Quantity**	**Place of Use**
M4*50 *(flathead machine screw)	8	Connecti bearings and the ground wooden base
M4*40 *(flathead machine screw)	16	Connect L-shaped bracket and the aluminium pillars
M4*25 *(flathead machine screw)	8	Connect L-shaped bracket with 2nd level wooden base
Wood Screw (M4*10)	8	Connect L-shaped bracket with ground wooden base

Step 3: Placing of DC Motors and sensors at the robot base

During this stage, most of the assembly will be done at the robot base. This stage focuses mainly on the installation of DC motors with the required mechanical parts, which are shaft coupling and cylindrical shaft, together with the rubber wheels. Before any drilling is started, the markings are done on the base of the board which is for placement of DC motor platform. For the connection between the platform and the robot base, 2 screws of dimension M4*40 are used, whereas 4 screws of dimension M3*10 are used to connect the DC Motor bracket and the motor platform. Table 5 lists the components used for assembling the wheel part, with the diagram of the components shown from Figure 34 to 39 below:

Table 5: Lists of Components required for the assembly of the wheel mechanism

Required Components	
DC Motor	 *Figure 34: DC Geared Motor (SPG50-100K)*
DC Motor Bracket	 *Figure 35: DC Motor Bracket*
6mm to 8mm shaft coupling	 *Figure 36: Shaft Coupling (6mm to 8mm)*
8mm shaft	 *Figure 37: Shaft (8mm diameter)*

Hex coupling for wheel	 *Figure 38: Hex Coupling for wheel*
Wheel	 *Figure 39: Wheel (65mm diameter)*

After assembling the wheel mechanism, Figure 40 shows the connection of the shaft couplings and the DC motors.

Figure 40: Placement of DC Motor and the required mechanical components

Figure 41 below shows how the ultrasonic sensor is being placed at the bottom of the robot base. The sensor holder is secured to the robot base using 4 pan head screws of dimension M3*25.

Figure 41: Placement of Ultrasonic Sensor at the base

Step 4: Placing of top cover and the 4 covers

The final stage is mostly about the covering of robot skeleton chassis using the aluminium composite panel. For the top base, aluminium panel of dimension 30cm by 30cm is cut and is placed to secure the top using 8 screws of dimension M4*25. Figure 42 shows how the ultrasonic sensor is being fixed on top of the prototype.

Figure 42: Placing of Top Cover with the addition of the ultrasonic sensor holder on top

For the front cover, two separate panels are cut and covered the chassis. This is because the bottom compartment consists of IR sensor holders connected to the cover, and the upper compartment consists of the Raspberry Pi camera attached at the wall of the cover. For the back cover, 1 single panel of dimension 30cm by 35cm will be connected with the aluminium hollow bar using 4 screws of dimension M4*40. The front covers consisting of 2 separately cut panels are shown in Figure 43 below:

Figure 43: Placing 2 separate covers in front of the robot

The reason why 2 separate panels are used to cover the front part of robot is to reduce the error of drilling when the measurements are made. When marking one drilling point, an alignment error of about 1 mm is made, hence the more the number of screws connected for each panel, the higher the alignment error, which causes the aluminium panel to unable to fit in properly.

After placing the front cover, the IR sensor holders are attached on the wall cover. Each IR sensor holders are connected with an extension so that the distance between the IR sensor and the ground is between 2 cm and 4 cm. For the connection between each sensor holder and the extension, 2 screws of dimension M3*15 are used, whereas screws of dimension M3*20 are used to connect the extension to the front cover of robot. Figure 44 on the next page shows how the IR sensors are attached to the bottom front cover of the robot with the extension holder.

Figure 44: Placing of 3 IR sensors in front of robot

In Figure 45 below, during the testing and calibration of robot, both sides are left opened first in case any removing of the connectors or adjustment of the position of Raspberry Pi 4B is carried out.

Figure 45: Isometric view of the prototype before covering the sides.

In Figure 46 below, after the testing and calibration has been done, the sides of the prototype are covered by the aluminium panels with a hole cut in the side to allow the speaker to distribute out the sound.

Figure 46: Final Prototype with cover (Isometric View)

3.2 Circuit Design and Hardware Implementation

3.2.1 Finalized Schematic Diagram

The finalized schematic diagram is drawn using Fritzing, as shown in Figure 47 below:

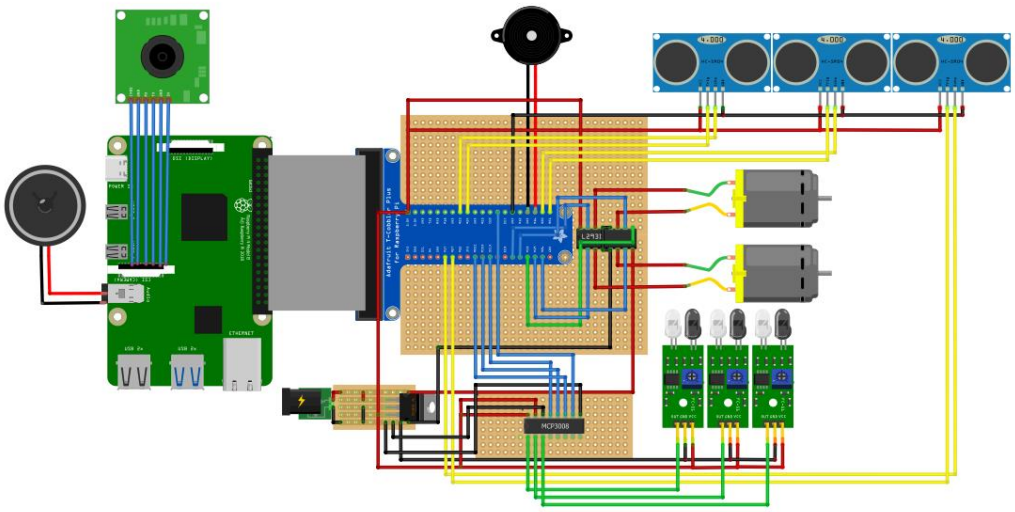


Figure 47: Finalised Schematic Diagram drawn using Fritzing

The block diagram of the robot is shown in Figure 48 below:

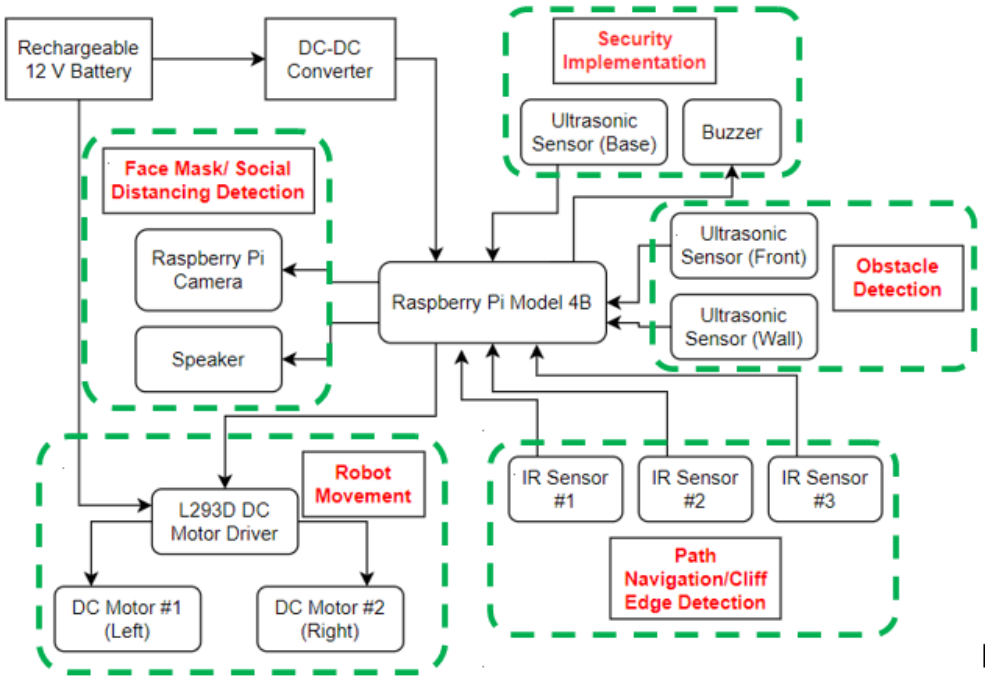


Figure 48: Finalized Block Diagram

Based on the schematic diagram and the block diagram shown previously, several modifications have been made compared to the previous schematic from FYRP1:

- 4x3 keypad matrix will not be used in the security implementation, due to the overusing of the GPIO pins. Instead, it will be implemented in software.
- Additional ultrasonic sensor will be used to detect obstacle in front.
- The ESP 8266 WiFi module will not be used, as Raspberry Pi 4 already has built in Wi-Fi.
- The ADC which converts the analogue signal from IR sensor to the digital signal input will be used instead of connecting the IR sensor directly to the GPIO pins. This is because Raspberry Pi 4 is not capable of handling analogue inputs without any comparators or ADC converters.

Table 6 below lists the GPIOs used for each sensor and component

Table 6: List of Sensors and Components corresponding to their GPIO pin number

Sensors or Components Used	**GPIO pin number**
2 inputs for DC Motor #1	GPIO 5 and 6
2 inputs for DC Motor #2	GPIO 26 and 19
PWM enable pins for L293D	GPIO 13
Inputs from ADC	SCLK, MISO, MOSI, SPI_CE1
Ultrasonic Sensor #1 (robot base)	GPIO 23 – Echo Pin GPIO 24 – Trigger Pin
Ultrasonic Sensor #2 (front)	GPIO 20 – Echo Pin GPIO 21 – Trigger Pin
Ultrasonic Sensor #3 (side)	GPIO 17 - Echo Pin GPIO 27 - Trigger Pin
Buzzer	GPIO 16

3.2.2 Sensors and Components Used

For the sensors used, there are only 2 types of sensors being used in this project, which are ultrasonic sensors and IR sensors. Figure 49 below shows the IR sensor used for line following navigation. In this project, 3 line-tracking sensors are used to detect the line on the floor any to also detect whether the centre of the robot has totally offset from the line position. Additionally, it is also used to detect any cliff edges so that robot won't fall off from a slightly elevated surface.

Figure 49: Ultrasonic Sensor, HC-SR04 and IR Line Tracking Sensor

In Figure 50 below, the Raspberry Pi Camera Module (8MP) will be connected to the camera port on the Raspberry Pi. The speaker is connected to the audio port of Raspberry Pi by an audio cable. Whenever a person is not following social distancing measures or wearing mask, Raspberry Pi will send a line of commands to the speaker with the mp3 file containing the nagging message.

Figure 50: Components Required for Video Capturing and Nagging

Figure 51 below shows the required components for rh robot movement. A L293D motor driver is used to drive 2 DC motors, for the Enable Pin at the L293D, PWM output from the same pin is used to drive two DC motors simultaneously, in order to ensure similar PWM pulsing.

Figure 51: Components for Robot Movement

Figure 52 below shows the IC of the ADC used in this project. In MCP 3008, the converter consists of 8 channels labelled CH0 to CH7 which can allow up to 8 analogue inputs to be connected at the same time. Multiplexing will be used to measure all the channels connected at an extremely fast speed.

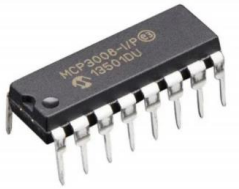


Figure 52: ADC MCP3008

3.2.3 Design of Different Types of Circuits

For the design of the ADC circuit, the schematic diagram is drawn as shown in Figure 53 below:

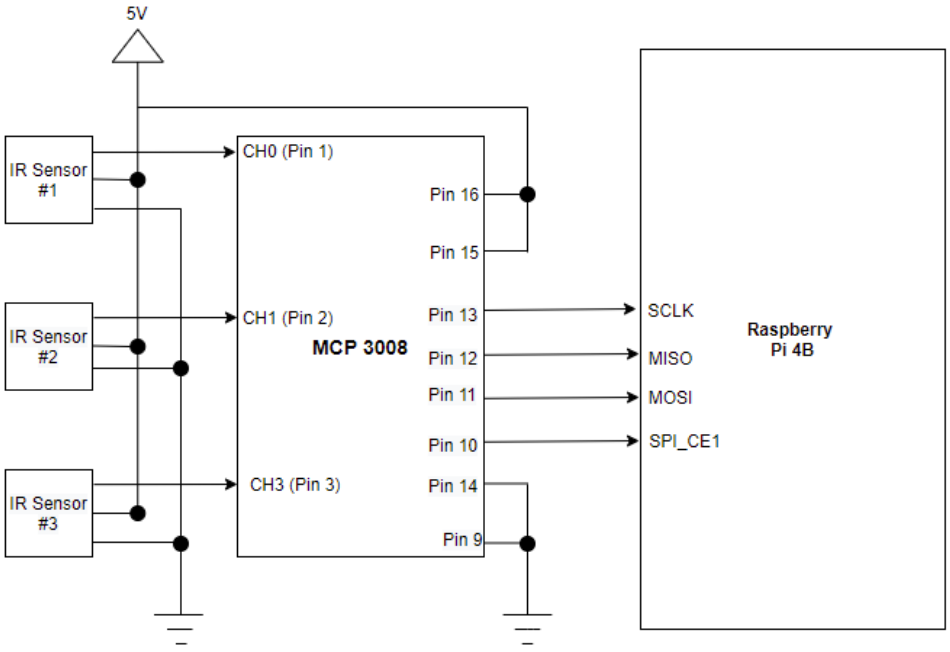


Figure 53: Schematic Diagram of ADC Circuit

For the ADC circuit, MCP 3008 consists of 2 sides of pins, where the left side consists of all the 8 channel pins which can receive 8 inputs at the same time. In this case, only 3 IR analogue inputs are used, therefore CH0-CH2 will be used. On the right side of MCP 3008, Pin 15 is the reference voltage which is 5 V as it is connected to the 5V supply. And pin 16 is the default power supply pin of the chip. Pin 9 and 14 are the digital and analogue ground pin for the ADC respectively.

In Figure 54 below, one 2, 3, and 4-way connectors will be used for the 5V/Ground, IR inputs and the 4 ADC outputs which are connected to the Raspberry Pi 4B respectively.

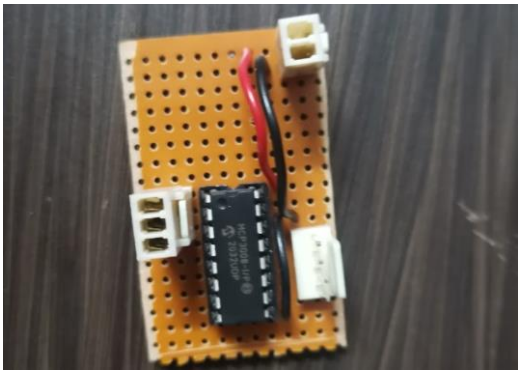


Figure 54: ADC circuit soldered on a very stripped board

Apart from the ADC circuit, there will be a DC motor drive circuit which is used to drive the 2 DC motors instead of using GPIO pins to drive directly. Figure 55 below shows the schematic circuit for the DC motor driver circuit:

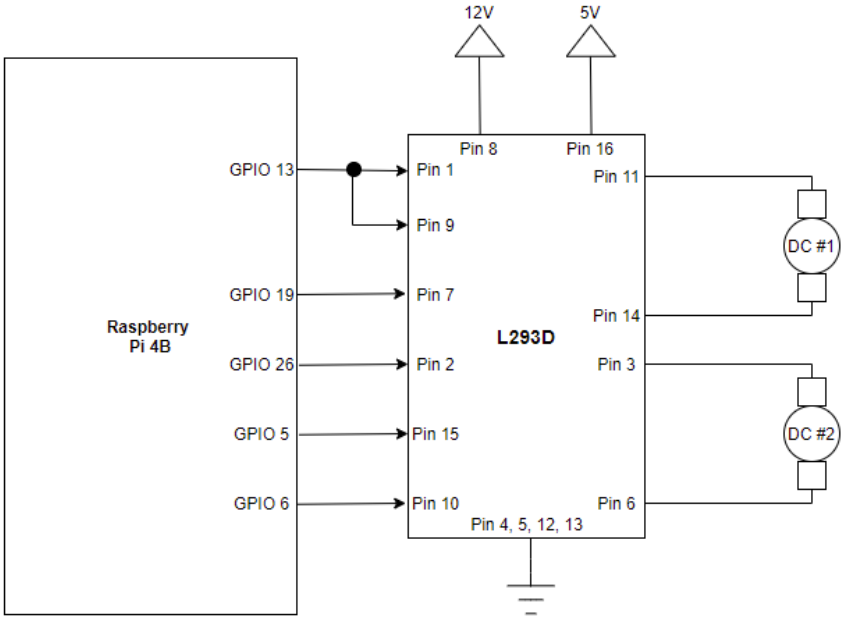


Figure 55: Schematic Diagram of DC Motor Driver Circuit

From the schematic diagram, the L293D Motor driver is used to drive the 2 DC motors, the 2 DC motors require 12 V supply, therefore the 12V voltage input is fed from the 12 V battery pack to the pin 8 of the chip. From the Raspberry Pi 4B side, GPIO 5 and 6 are used to control the direction of rotation of DC motor #1, and GPIO 19 and 26 are used to control the direction of rotation of DC motor #2. GPIO 13 serves as the PWM output from Raspberry Pi 4B to enable to motors at pin 1 and 9. Therefore, the speed of the DC motor can be synchronised in order for the robot to move in a straight line properly.

3.2.4 Camera Placement and Position Calibration

The specifications for the Pi Camera Module V2 are listed in Table 7 below:

Table 7: View angle specifications of Pi Camera 8MP

Resolution	8 Megapixels
Horizontal field of view	62.2 degrees
Vertical Field of view	48.8 degrees

Source: *https://www.raspberrypi.org/documentation/hardware/camera/*

In terms of horizontal field of view, the camera needs to be placed as close to the centre of the robot as possible. The most important parameter is the vertical field of view, as the robot not just detects the face of each person, but also the entire person for social distancing detection. Additionally, the robot is built at approximately 45 - 50 cm in height. Hence, the camera needs to be tilted up at certain degree so that proper and accurate detection of mask wearing and social distancing can be achieved simultaneously. The value of the tilted angle will be determined by considering the average height of an adult.

According to study by NCD Risk Factor Collaboration, the average height of a Malaysian male adult and female adult whose age is above 19 are 169.2 cm and 157.1 cm respectively. And the average adult height for Malaysian is 163.1 cm. Hence, approximately 160 cm will be taken as the reference height of detection, as the upper section allows greater height of detection.

According to the robot design, the camera is to be placed at 40 cm from the ground. And it is assumed that the maximum horizontal distance for the person's face to be detected is fixed at 150 cm. And for social distancing, the maximum distance is fixed at 450 cm. Consider the illustration shown in Figure 56 below (social distancing):

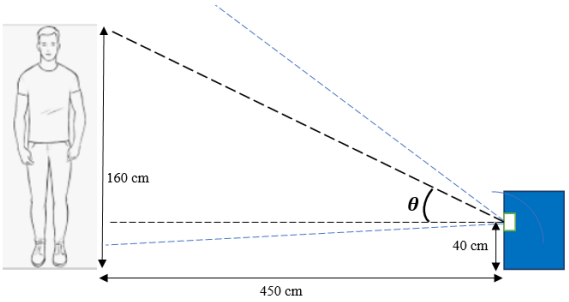


Figure 56: Illustration of the range of social distancing detection

Based on the figure above, the angle can be determined using Trigonometry Ratio formula:

$$\tan\theta = \frac{160 - 40}{450}$$

$$\theta_{\min\,(social\ distancing)} = 14.93°$$

If the distance is shortened to 250 cm:

$$\tan\theta = \frac{160 - 40}{250}$$

$$\theta_{\max\,(social\ distancing)} = 25.64°$$

Based on the calculation, for social distancing detection, the minimum distance between the citizens and the robot is 250 cm.

For face mask detection, the maximum distance which the mask can be detected is 170 cm. Consider the illustration shown in Figure 57 below (face mask detection):

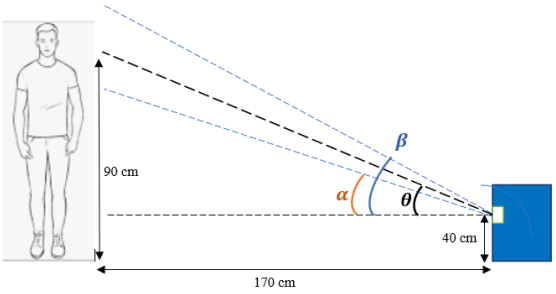


Figure 57: Illustration of the range of face mask detection

$$\tan\theta = \frac{90-40}{170}$$

$$\theta_{\min\,(mask)} = 16.39°, \beta = 16.39 + 24.4 = 40.79°$$

$$Let\ the\ maximum\ height\ of\ the\ person\ be\ h$$

$$\tan 40.79 = \frac{h-40}{170}$$

$$h = 186.7\ cm\ for\ the\ case\ when\ distance = 170cm$$

When the distance is reduced to 120 cm:

$$\tan\theta = \frac{90-40}{120}$$

$$\theta_{\max\,(mask)} = 22.62°, \beta = 22.62 + 24.4 = 47.02°$$

$$Let\ the\ maximum\ height\ of\ the\ person\ be\ h$$

$$\tan 47.02 = \frac{h-40}{120}$$

$$h = 168.8cm\ for\ the\ case\ when\ distance = 120cm$$

Based on the calculations, the suitable tilting angle that can be implemented is 20° as at this angle the camera is able to detect mask wearing and social distancing at a suitable distance. However, there will be a limitation, which is the maximum height of the detectable person is 187 cm when the robot is 170 cm away from the person, and 168.8 cm when the robot is 120 cm away from the person.

For the camera holder, the STL file for the 3D modelling of the pi camera can be downloaded from the Cytron official website. In Figure 58 below, the camera holder can be tilted from 0° to 180 °. The 3D printed parts are printed out separately, whereby one of the part is responsible for holding the camera, and the bottom part is involved in the adjustment of the tilting of the camera. To connect these 2 parts together, a M4*40 flat head screw is used with a nut. To secure the camera holder on the front part of robot, 2 pan head screws of dimension M3*25 are used.

Figure 58: 3D-Printed camera holder

3.2.4 Wiring and Assembly of Circuit Boards with Raspberry Pi 4B

Since the project involves a robot which moves around an area, therefore connecting the Raspberry Pi 4B to a plug adapter is inappropriate. On the other hand, a power supply circuit needs to be designed in order for the robot to be powered by a portable DC battery pack. Figure 59 below shows the block diagram for the power supply circuit:

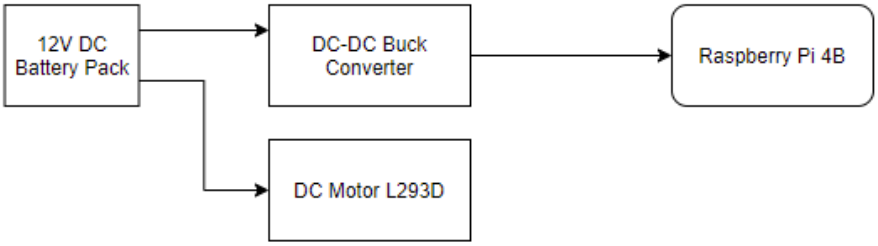


Figure 59: Block Diagram of the Power Supply Circuit

The 12V battery output is branched out in order to power up the L293D chip and to be converted into 5V to power up the Raspberry Pi 4B. At the DC-DC Converter, the output voltage and current are tuned to 5V and 3A respectively as Raspberry Pi 4B can only operate at a minimum input current of 3A. The battery pack is connected to the converter via a female DC jack socket terminal, and the output from the converter is connected to the Raspberry Pi 4B via a USB Type A-to-Type C cable, as there is a built in USB port at the converter module.

In Figure 60 below, the voltage can be adjusted directly from the potentiometer knob without any external wire configuration. However for the adjustment of current (shown in Figure 61 below), the outputs are shorted together with a wire before adjusting the potentiometer know for the current.

Figure 60: Adjustment of Output Voltage

Figure 61: Adjustment of Output Current

The 40 GPIO pins from Raspberry Pi 4B are connected to a T-Cobbler GPIO extension board which is for Raspberry Pi usage via a rainbow extension cable, inside the robot chassis, a rectangular is cut so that the extension cable can be extended from the 2nd compartment to the 1st compartment level. Figure 62 below shows how the 40 female-extension cable is connected from the Raspberry Pi 4B down to the GPIO extension board.

Figure 62: Connecting Raspberry Pi 4B to extension board using rainbow extension cable

In Figure 63 below, several connectors of 2/3/4 way are soldered around the extension board for connecting to other sensors and IC chip purpose. For some outputs, they are grouped together according to the GPIO extension board labelling. This is to ensure that the connectors can be soldered on the board and on the other hand to allow easy connecting and removing of the connectors which are catered for each specific sensors and components.

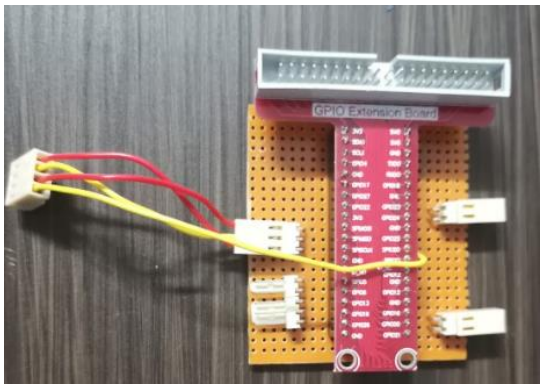


Figure 63: Wiring and Soldering of GPIO Extension board

Figure 64 below shows the completed soldered and wired circuit board for L293D motor driver circuit,

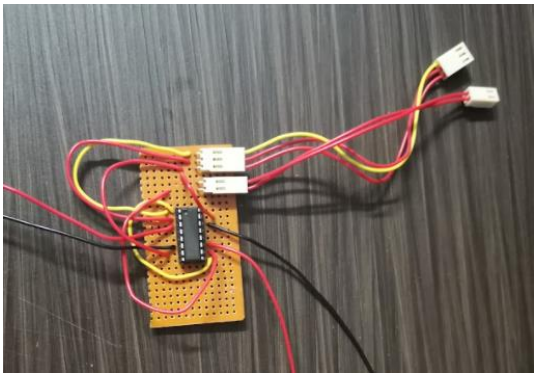


Figure 64: Wiring and Soldering of the L293 Motor Driver Circuit

In Figure 65 below, different colours of the wires are assigned to the wiring of L293 motor driver circuit. For example, yellow wires are indicated as PWM input to the Enable pins of both motors, and red wires are used in 5V or 12V input, as well as input and output pins of the motor, and black wires are distinguished as ground wires.

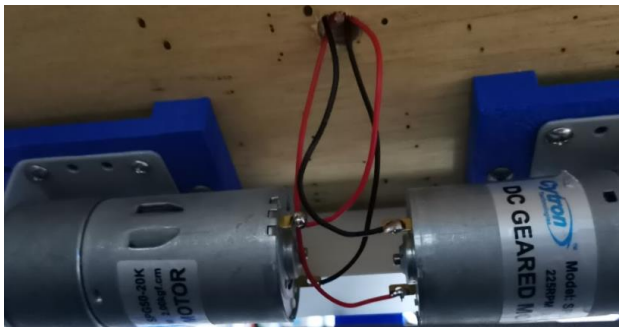


Figure 65: Soldering of wires from the L293 motor output pins to the DC motors

In Figure 66 below, the 3 analogue inputs of the IR sensors are indicated as yellow-coloured wires, and these 3 wires will be crimped and connected to the 3-way connector header, so that the connector can be plugged into the CH0-CH2 connector pin of the MCP 3008 ADC circuit board.

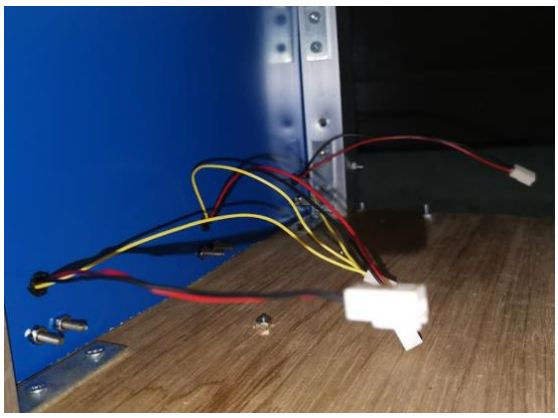


Figure 66: Internal Wiring Diagram from the IR sensors

3.3 Software Algorithm Implementation

3.3.1 Flowcharts

For the path navigation, there will be 2 types of mode in which user can select in different location. The first type of navigation is free navigation in which robot does not follow any indicators on the floor or on the wall, therefore the ultrasonic sensors at the front and side will be responsible for detecting any obstacles surrounding it. The free path navigation is usually used on slightly smaller spaces such as corridor and places where there are more non-moving obstacles surrounding. For the second path navigation, it will be line following navigation, the robot will follow the line on the floor and move in an open/closed-loop path. This type of path navigation is used in very large places such as the large empty space at the airport, where many people will queue up in front of a counter.

Before the robot starts moving, user can choose either one of the 2 path navigation, then the robot will follow the chosen path navigation. However due to the limited size of the testing area, usually the free navigation method will be used for testing calibration.

Figure 67 below shows the flowchart for selection of path navigation type by the user. The robot is able to run in either free path navigation or line tracking path navigation (dual navigation), it also depends on the initial selection of the user before the robot starts roaming. Normally, these 2 types of path navigation will be used in different conditions in an indoor area which is quite easy to be distinguished. Hence, when user wishes to move the robot to other places to roam, the program needs to be stopped before allowed to start so that user can able to choose the other path navigation before the robot starts roaming.

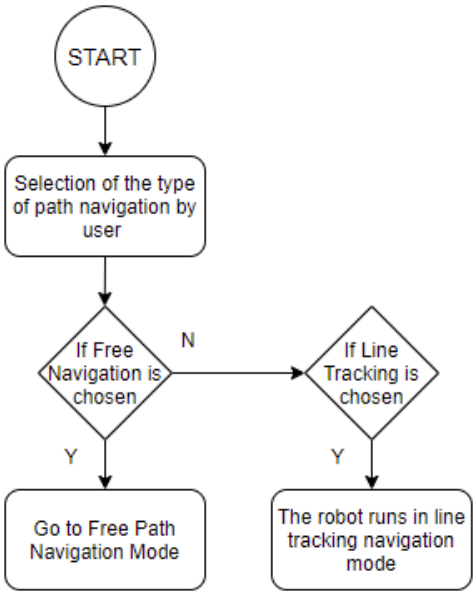


Figure 67: Flowchart for Selection of Path Navigation by the user

Figure 68 below shows the flowchart for free navigation of robot. For free navigation, the ultrasonic sensors at the front and side of the robot will be used to detect obstacle. The robot will turn 90 degree to the left/right side when an obstacle is detected in front.

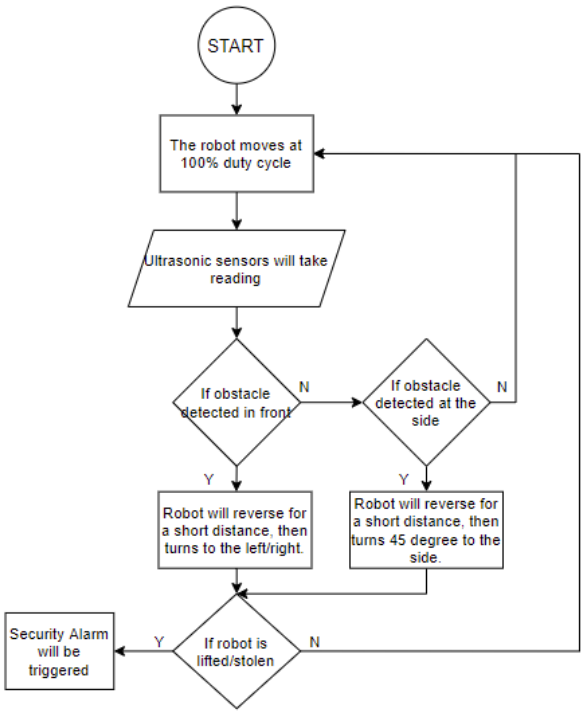


Figure 68: Flowchart for Free Navigation of Robot

Figure 69 below shows the line tracking path navigation of robot. For the line tracking path navigation, the three IR sensors will be used to detect the line on the floor, whereas only the ultrasonic sensor at the front side of the robot will be used to detect any obstacle along its fixed path. For this case, the robot will only stop and wait until the obstacle in front has been removed or the person in front has moved.

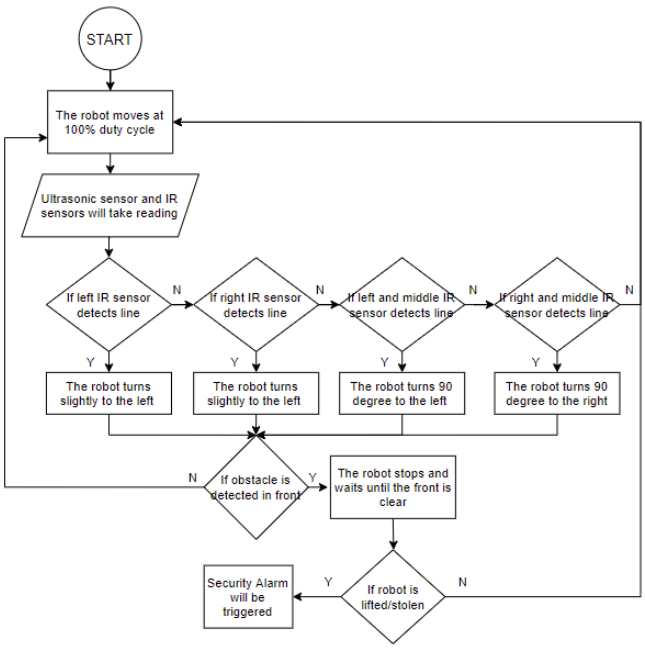


Figure 69: Flowchart for Line Tracking Navigation of Robot

Figure 70 below shows the flowchart for the security implementation, when the robot is stolen or lifted up until a certain height, the buzzer will be triggered. The program will then ask user to input the correct password, so that the buzzer will be disabled. Before the robot starts moving, it will prompt user whether the robot should resume its task, so that no buzzer will be triggered again when the robot is being moved from one place to another by the person in charge.

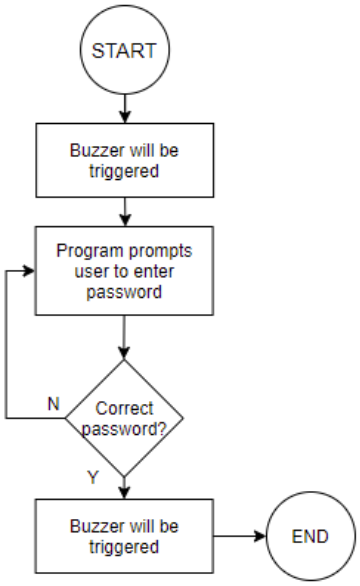


Figure 70: Flowchart for Security Implementation

Figure 71 below shows the flowchart for mask wearing detection, the Pi camera and the speaker will be run in a different program in Raspberry Pi using VNC viewer, as there are conflicts occur when the main program is integrated with the image processing program together. When the camera detects the absence of either social distancing or mask wearing, the speaker will produce the nagging message until the nagging message has been finished, then the camera can only resume its video capturing process.

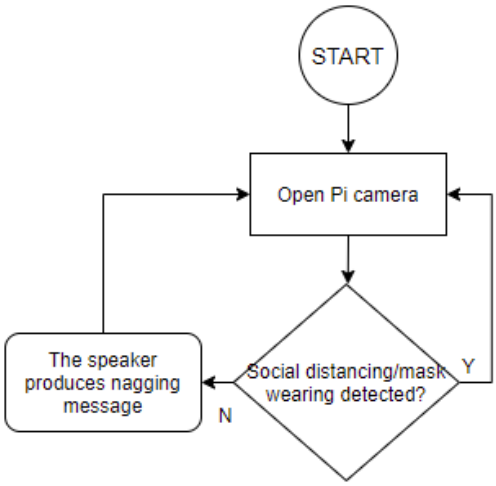


Figure 71: Flowchart for Mask Wearing detection

3.3.2 Explanation of Codes for each Function

First of all, to run the main program, some of the libraries which are required to run are *spidev* which is for configuring ADC converter, and also *time* and *RPi.GPIO*. Figure 72 below shows the libraries necessary for the GPIO interface.

```
#Import the required libraries
import spidev
import time
import RPi.GPIO as GPIO
```

Figure 72: Importing the Required Libraries

Secondly, the GPIO pins for each sensors or components are defined as well, as shown in Figure 73 below:

```
# Define sensor channels for each IR sensor
ir1_channel = 0 #left sensor
ir2_channel = 1 #middle sensor
ir3_channel = 2 #right sensor

#GPIO pin number for the DC motor inputs and PWM input
Left_DC_1 = 5
Left_DC_2 = 6
Right_DC_1 = 26
Right_DC_2 = 19
DC_PWM = 13

BUZZER = 16 #GPIO pin number for the buzzer

# setup gpio for echo & trig
echopin = [20, 23, 17]
trigpin = [21, 24, 27]
```

Figure 73: Code Snippet for defining all the GPIO pins required

Next, all the GPIO pins are defined as whether they are inputs or outputs in Figure 74 below:

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(Left_DC_1,GPIO.OUT)  # All motor input pins as Outputs
GPIO.setup(Left_DC_2,GPIO.OUT)
GPIO.setup(Right_DC_1,GPIO.OUT)
GPIO.setup(Right_DC_2,GPIO.OUT)
GPIO.setup(DC_PWM,GPIO.OUT)

GPIO.setup(BUZZER,GPIO.OUT)

#Echo pin as input, and Trigger pin as output for the Ultrasonic Sensor Configuration
for j in range(2):
    GPIO.setup(trigpin[j], GPIO.OUT)
    GPIO.setup(echopin[j], GPIO.IN)
```

Figure 74: Code Snippet for defining inputs and outputs of the GPIO pins used

For calculation of the distance between the ultrasonic sensor and the ground, Figure 75 below shows the following function that has been written to do the calculation mentioned above:

```
48  def ping(echo, trig):
49
50      GPIO.output(trig, False)
51      # Allow module to settle
52      time.sleep(0.5)
53
54      # Send 10us pulse to trigger
55      GPIO.output(trig, True)
56      time.sleep(0.00001)
57      GPIO.output(trig, False)
58      pulse_start = time.time()
59
60      # save StartTime
61      while GPIO.input(echo) == 0:
62          pulse_start = time.time()
63
64      # save time of arrival
65      while GPIO.input(echo) == 1:
66          pulse_end = time.time()
67
68      # time difference between start and arrival
69      pulse_duration = pulse_end - pulse_start
70
71      # mutiply with the sonic speed (34300 cm/s) and divide by 2, because there and back
72      distance = round(pulse_duration * 17150, 2)
73
74      return distance
```

Figure 75: Code Snippet for Calculating the distance between the ultrasonic sensor and the ground

From the code snippet above, in line 55 and 56, the ultrasonic sensor sends a short pulse of 10 μs and records the time when the pulse has been sent out (Line 62), after the pulse signal bounces back when an obstacle has been detected (Line 65), the Echo pin will receive the bounced-back signal thus recording the time when the pulse signal has been received by the Echo pin (Line 66). The difference between the time when Echo pin receives the pulse signal and the time when pulse signal is sent out from the Trigger pin is computed, which will be used to multiply with the sonic speed (34300 cm/s) and then divided by 2 (Line 69-72), thus calculating out the distance in cm.

At the main program, the program will print out the distances between the obstacle and all the 3 ultrasonic sensors in a very fast period of time by using assigning vector to each trigger and echo pins of the ultrasonic sensors. Figure 76 below shows the code for displaying the distances at the console.

```
135             # get distances and assemble data line for writing
136             results = ","
137             for j in range(2):
138
139                 distance = ping(echopin[j], trigpin[j])
140                 print("sensor", j+1, ": ", distance, "cm")
141                 results = results + str(distance) + ","
142
143             results = results + "\n"
```

Figure 76: Code Snippet for printing out the distances at the console

For the configuration of ADC MCP 3008, several configurations are required. Firstly, the *spidev* library is needed so that the maximum SPI clock frequency can be set at 1 MHz. Additionally, the channel pins used at the ADC are also assigned to each IR sensors. Figure 77 below shows the declaration of the analogue input channels and the definition of SPI clock frequency.

```
6   # Open SPI bus
7   spi = spidev.SpiDev()
8   spi.open(0, 0)
9   spi.max_speed_hz = 1000000
10
11  # Define sensor channels for each IR sensor
12  ir1_channel = 0 #left sensor
13  ir2_channel = 1 #middle sensor
14  ir3_channel = 2 #right sensor
```

Figure 77: Code Snippet for Defining SPI Clock frequency (Line 7-9) and channel pins assigning (Line 12-14)

The following code snippet shown in Figure 78 below lists out the functions used while converting the analogue input data from the IR sensors:

```
78  def ReadChannel(channel):
79      adc = spi.xfer2([1,(8+channel)<<4,0])
80      data = ((adc[1]&3) << 8) + adc[2]
81      return data
82
83  # Function to convert data to voltage level,
84  # rounded to specified number of decimal places.
85  def ConvertVolts(data,places):
86      volts = (data * 3.3) / float(1023)
87      volts = round(volts,places)
88      return volts
```

Figure 78: Code Snippet of the Functions used in ADC interface

From the code snippet above, the function *ReadChannel()* calculates the required values to be sent to the ADC so that the correct data from the desired channels can be obtained, then bit masking and shifting will be carried out in order to obtain the actual result, which is shown in Line 81. In line 79, the syntax *spi.xfer([1, (8+channel)<<4, 0])* implies that if channel 1 data is to be concerned, then 8+channel will get 9, then this value 9 will be shifted 4 bits to the left in binary form, so:

00001001 → 10010000 (which is 144 in decimal equivalent) [Channel 1]
00001010 → 10100000 (which is 160 in decimal equivalent) [Channel 2]
00001011 → 10110000 (which is 176 in decimal equivalent) [Channel 3]

The configuration of each bit of the ADC channel can be found in the Appendix section later on.

The code snippet for the security implementation can be explained by the following lines of code shown in Figure 79 below.

```
190         if (ping(echopin[0], trigpin[0])>20): #If robot is lifted up by more than 20cm.
191             GPIO.output(Left_DC_1,GPIO.LOW)
192             GPIO.output(Left_DC_2,GPIO.LOW)
193             GPIO.output(Right_DC_1,GPIO.LOW)
194             GPIO.output(Right_DC_2,GPIO.LOW) #The robot stops moving
195             GPIO.output(BUZZER,GPIO.HIGH) #Buzzer will be triggered
196             while True:
197 
198                 print('Please enter the correct password to disable the alarm:')
199                 password = input() #Prompts user to enter password
200 
201                 if (password != '12345'): #Default password is not matched
202                     print('Please enter the correct password to disable the alarm:')
203                     password = input()
204                 else:
205                     print('Alarm Disabled')
206                     GPIO.output(BUZZER,GPIO.LOW) #Alarm will be disabled.
207                     time.sleep(1)
208 
209                     while True: #Ask user whether should continue using the robot
210                         print("Continue moving the robot? (Y/N)")
211                         answer = input()
212 
213                         if (answer != 'Y' or answer!='Yes' or answer != 'YES'):
214                             print("Continue moving the robot? (Y/N)")
215                             answer = input()
216                         else:
217                             break
218                     break
219             break
```

Figure 79: Code Snippet for the Security Implementation

From the code snippet, right after the distance above 20cm has been detected by the ultrasonic sensor, the DC motors will stop and the buzzer will be triggered (Line 190-195). The program will prompt user to enter the default password which is fixed by the program. If the password entered matches the default password, then the alarm will be disabled (Line 201-207). However, the robot will not start moving until user reactivate the robot by entering either ‘Y’, ‘Yes’, or ‘YES’ so that the robot can resume its operation (Line 209-215). This is because when the robot is being moved from one place to another, the distance between the ground and the sensor will still be above 20 cm. Hence, only after the robot has been moved to another location for roaming purpose, user will then reactivate it to allow it to continue its roaming task.

3.4 Image Processing and Artificial Intelligence Implementation

3.4.1 Face Mask Detection

In order for the Pi camera to distinguish between human face with mask and of without mask, the robot needs to undergo some sort of image classification training before the Pi camera is being run by the program. To do so, there are a few libraries which are required for the face mask detection. A few of the most important library packages that are used are *tensorflow*, *keras*, *imutils*, and *sklearn*. *Tensorflow* is one of the most important library package platform specialised for AI and machine learning, as well as *sklearn*, whereas *keras* specialises in neural network learning. And usually *imutils* is used with OpenCV for image processing functions.

In order to train neural networks, gradient descent will be used. Gradient descent is a very commonly used algorithm used in training a machine learning model.

To begin the AI model training, first of all images from the two categories (without and without mask) need to be stored inside the specific folder, and will be analysed by extracting the images from the folder when the program is being run. Figure below shows the code used to import image datasets from 2 folders of different categories.

```
39  #Import the images from both folder (with and without mask)
40  for category in CATEGORIES:
41      path = os.path.join(DIRECTORY, category)
42      for img in os.listdir(path):
43          img_path = os.path.join(path, img)
44          image = load_img(img_path, target_size=(224, 224))
45          image = img_to_array(image)
46          image = preprocess_input(image)
47
48          #Store the image into the data array
49          data.append(image)
50          labels.append(category)
```

Figure 80: Code Snippet for extracting images from the 2 folders

In terms of AI model training wise, these data are classified as positive and negative dataset. Positive dataset contains images which consisting of the object to be detected present inside it, and negative dataset contains images which should not have the detected object inside. Therefore in this case positive dataset consists of people wearing mask of different colours, and negative dataset consists of people not wearing mask or people who pull down the mask straps to their neck.

In Figure 81 below, *LabelBinarizer()* is a function from the *sklearn* library that is responsible for receiving categorical data (data from the 2 categories) and convert it into numpy array.

```
52  # perform one-hot encoding on the labels
53  lb = LabelBinarizer()
54  labels = lb.fit_transform(labels)
55  labels = to_categorical(labels)
```

Figure 81: Code Snippet for

In AI model training, there are a few variable parameters which affects the results of the trained model, which are:

- Batch size
- Epoch
- Initial Learning Rate
- Iterations

Initial learning rate refers to the parameter that controls the rate at which the model learns from the training data. Batch size is the number of times the image dataset is loaded into the neural network. And epoch refers to the number of iterations completed for the entire dataset. For example, in this face mask detection algorithm, approximately 4000 sample images from 2 categories are used, and the batch size is assumed to be 32, therefore it will take almost 125 iterations to complete 1 epoch.

For these 3 parameters, any increase/decrease in these 3 values might require the other 2 values to be changed as well. For example, initial learning rate should be less than 1 but need to be more than 10^{-6}. Initial learning rate which too high took less time for the model to be trained, but may cause the loss value to be oscillating along the epoch line, on in other words divergences occur instead of converging of value [13]. Initial learning rate which is too slow does not only take longer time to complete the training, but also won't decreasing the training error. [12] Therefore, generally choosing the best initial learning rate is not possible [15].

For batch size, it is recommended to start from 32, 64, 128, 258 and so on. However, the batch size cannot be more than the number of sample images in the dataset. Since 4000 samples of images will be used for face mask detection, therefore the batch size can only reach up to 2048.

In terms of epochs, more epochs does not indicate higher accuracy. If the number of epochs exceed the optimum value, the model will be overtrained, causing the AI to unable to learn new data but memorise the data instead.

The suitability of the model for face mask detection can be explained using the learning curve. Learning curve allows the user to see whether the model has been overtrained, undertrained, or optimum for the dataset [14].

Figure 82 below shows the AI model being trained in progress, as viewed from the console:

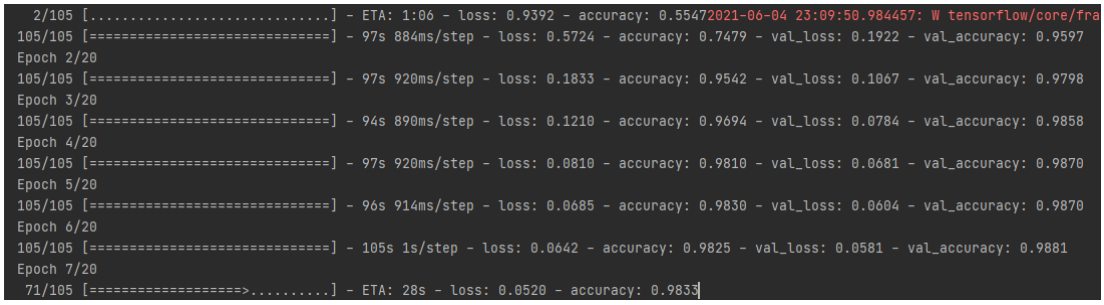


Figure 82: Process of model training in console window

As shown in the console, each epoch has a iterations of 105, which is indicated by 105/105 below the epoch number.

After training the model, a model.h file will be created, which will be fed into the mask detection program, and the program only can detect the face mask.

Figure 83 below shows the confidence level for the detection of presence/absence of face mask.

```
31      for i in range(0, detections.shape[2]):
32          # extract the confidence (i.e., probability) associated with
33          # the detection
34          confidence = detections[0, 0, i, 2]
35
36          # filter out weak detections by ensuring the confidence is
37          # greater than the minimum confidence
38          if confidence > 0.5:
39              # compute the (x, y)-coordinates of the bounding box for
40              # the object
41              box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
42              (startX, startY, endX, endY) = box.astype("int")
43
44              # ensure the bounding boxes fall within the dimensions of
45              # the frame
46              (startX, startY) = (max(0, startX), max(0, startY))
47              (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
48
49              # extract the face ROI, convert it from BGR to RGB channel
50              # ordering, resize it to 224x224, and preprocess it
51              face = frame[startY:endY, startX:endX]
52              face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
53              face = cv2.resize(face, (224, 224))
54              face = img_to_array(face)
55              face = preprocess_input(face)
56
57              # add the face and bounding boxes to their respective
58              # lists
59              faces.append(face)
60              locs.append((startX, startY, endX, endY))
```

Figure 83: Code Snippet of how the detection is considered valid

In line 38-55, if the confidence is less than 0.5, the algorithm will skip the weak detections and only detect the face which is either with/without mask. For example, if a person who is not wearing mask is standing too far away from the camera (e.g. more than 2 m away) from the camera, the algorithm will produce a confidence level less than 0.5 as the face of the person is too small for the confidence level to be at least 0.5. Right until the person stands slightly closer, which makes the detection easier, then the algorithm will probably have a higher confidence value, thus making it able to detect the presence/absence of mask. As shown in line 51-55, a frame will be created to surround the person's face, in which the parameters of the frame created which are *startX*, *startY*, *endX*, and *endY* refer to the x and y coordinates of where the frame will be placed on the screen with reference to the face detected (line 41-47). If they are more than 1 face detected, the faces detected and the bounding box frames will be appended into their respective lists.

Figure 84 below shows the checking of presence/absence of face mask wearing of a person/the persons:

```
97      # loop over the detected face locations and their corresponding
98      # locations
99      for (box, pred) in zip(locs, preds):
100         # unpack the bounding box and predictions
101         (startX, startY, endX, endY) = box
102         (mask, withoutMask) = pred
103
104         # determine the class label and color we'll use to draw
105         # the bounding box and text
106         label = "Mask" if mask > withoutMask else "No Mask"
107         color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
108
109         # include the probability in the label
110         label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
111
112         # display the label and bounding box rectangle on the output
113         # frame
114         cv2.putText(frame, label, (startX, startY - 10),
115                     cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
116         cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

Figure 84: Code Snippet for placing different bounding boxes to people with/without mask

From the faces detected and the bounding boxes appended into the lists previously, the parameter *pred* refers to the predictions made from the initial detection. In line 102, the *mask* and *withoutMask* parameter will have the prediction value assigned into it. For example, the prediction value in the variable *mask* is defined as how possibly the face detected consists of face mask. And in line 106-107, if the *mask* value is greater than *withoutMask*, then the algorithm will identify the person as not wearing mask. In contrary, if the value of *withoutMask* is higher, then no mask will be detected. For better indication and detection, green bounding box will be used for the presence of mask which is written as (0, 255, 0) in BGR representation, and red bounding box will be used for the absence of mask, which is written as (0, 0, 255) in BGR representation.

4. Results and Discussions

4.1 Testing of IR Sensors from different heights and surface

IR sensor consists of both analogue and digital type. For line following IR sensor, analogue type IR sensor is a better option as the differences between the input voltage read by the sensor when it detects black line and the input voltage when it detects region other than black colour must be large enough for the robot to follow the line properly.

Some testing is required such that the voltage read from the sensor in different coloured surface detected needs to be obtained. Additionally, different values of the distance between the sensor and the ground are used, ranging from 2 cm to 5 cm with 1 cm increment. Figure 85 below shows the setup of the IR sensors connected to the breadboard from different height:

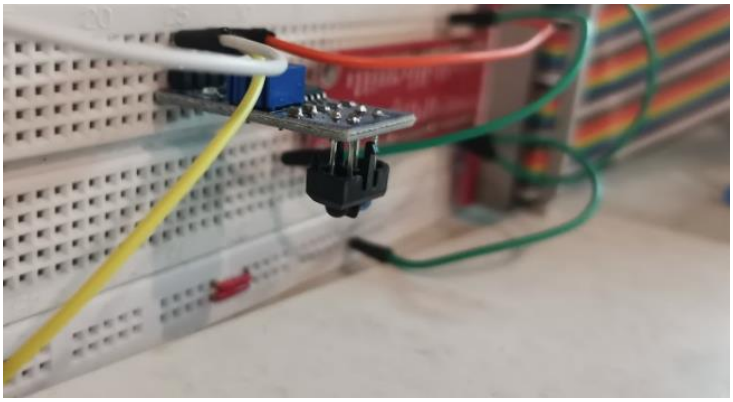


Figure 85: Setup of IR Sensor connection with breadboard

From the setup of the circuit in Figure 85 above, the IR sensor is connected to the breadboard which is held vertically in order for the IR sensor to detect the ground surface. Each time when the IR sensor detects the surface at different height, the voltage reading will be observed and recorded in the tabulation of data. Table 8 tabulates all the sensor input data vs the height between the IR sensor and the ground.

Table 8: IR sensor voltage from different height

	Input IR Sensor Voltage (V)		
	Black/Dark Grey surface	**White surface**	**Yellowish surface**
Height, h (cm)			
2	2.21	0.447	0.363
3	2.11	0.385	0.341
4	1.31	2.21	2.06
5	1.62	2.29	2.86

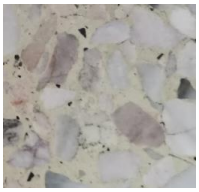
From the tabulation of data above, the IR sensor showed normal range of voltage readings when the height is between 2 cm and 3 cm. As the height is increased to 4 cm, odd and fluctuating readings are produced, thus resulting in inaccurate detection of line on the floor. Therefore, the maximum height the sensor can detect is 3 cm. For safety margin purpose, the most suitable height for the IR sensor placement is 2 cm above the ground, since IR sensor has a limited range of distance of detection.

Also from the Table 8 above, it is shown that black or dark grey region has a similar range of voltage reading (more than 2 V), and white or lighter colour surface has very small voltage reading (less than 0.5 V). Hence, the threshold can be set such that when the voltage reading is less than 1.2 V, the IR sensor will detect the region as white colour. Whereas for voltage reading exceeding 1.8 V, the IR sensor will detect the region as the line.

For the next IR sensor testing, a more thorough and detailed testing will be carried out, which is to determine the threshold point for different surfaces at a fixed height of 2.5 cm. An additional condition is added such that the robot is moving at the same time the IR sensor is taking the measurement. There are 4 types of surfaces to be tested (including the white surface for line detection). The robot will be move for 10 seconds in a straight line, and the measurements are taken by the sensors.

Table 9 below tabulates the input sensor value for each of the 3 IR sensors of different colour of surfaces (labelled as Figure 86-89 respectively) vs time.

Table 9: input sensor value for each of the 3 IR sensors of different colour of surfaces vs time

	Analogue Input IR Sensor Value (Legend: Left/Middle/Right)			
Time (s)	**Black/Dark Grey surface**  *Figure 86: Black surface*	**Brownish surface**  *Figure 87: Brown wooden surface*	**Coloured Limestone Surface**  *Figure 88: Coloured Limestone Surface*	**White surface**  *Figure 89: White surface*
1	720/582/584	383/416/114	231/268/39	88/181/35
2	714/576/614	416/416/179	212/260/37	92/184/28
3	708/573/587	397/409/137	222/261/40	102/190/30
4	707/573/585	417/403/125	251/237/37	105/191/31
5	709/570/570	380/410/107	222/266/39	102/189/32
6	668/560/571	396/416/147	220/253/42	102/192/32
7	703/567/585	409/414/184	203/253/40	92/182/44
8	701/570/608	368/406/123	210/234/41	88/181/44
9	716/571/647	406/420/132	165/240/40	97/186/45
10	702/567/609	455/445/260	237/256/42	108/198/45

Based on the Table 9 above, 3 graphs of the sensor value vs the surfaces used are plotted, with each graph belongs to each sensor.

Figure 90-92 on the next 2 pages show the graph plotted for the analogue sensor input values of left, middle, and right sensors vs time respectively.

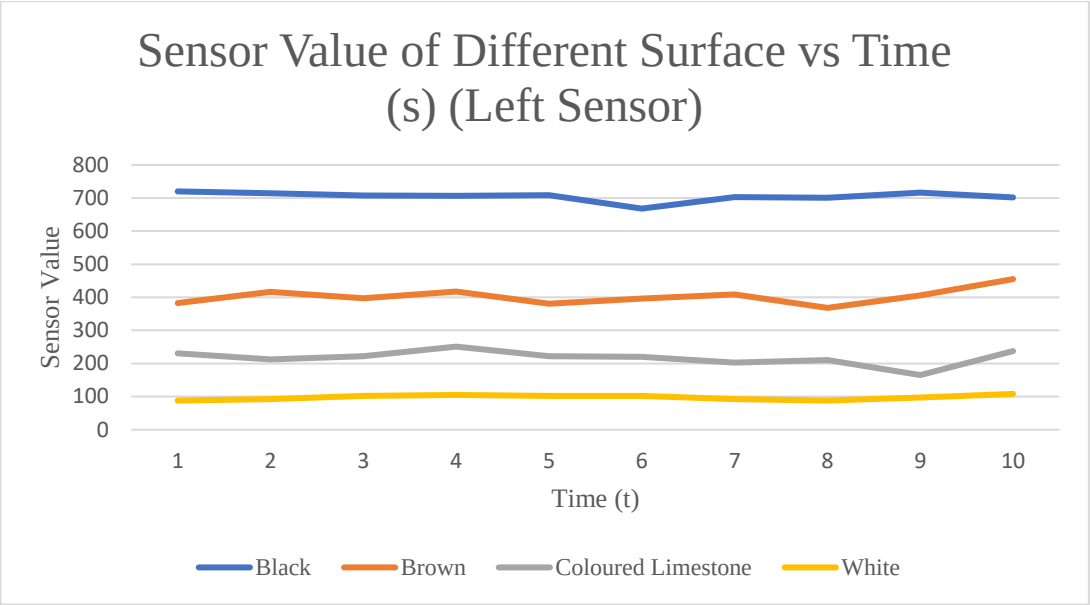


Figure 90: Graph of Sensor Value of different surfaces vs time(s) for left IR sensor

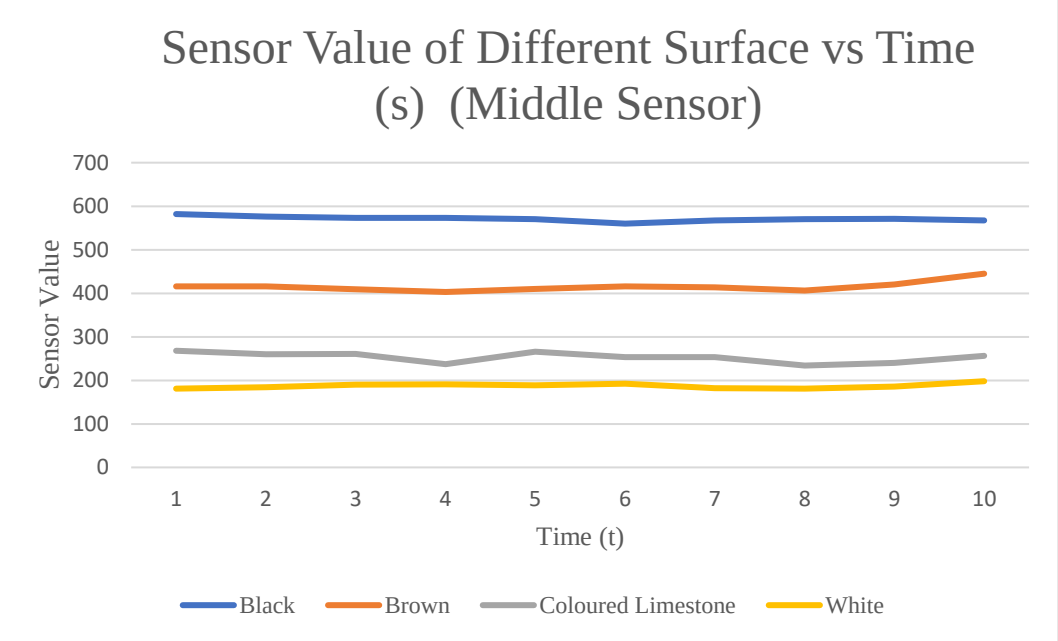


Figure 91: Graph of Sensor Value of different surfaces vs time(s) for middle IR sensor

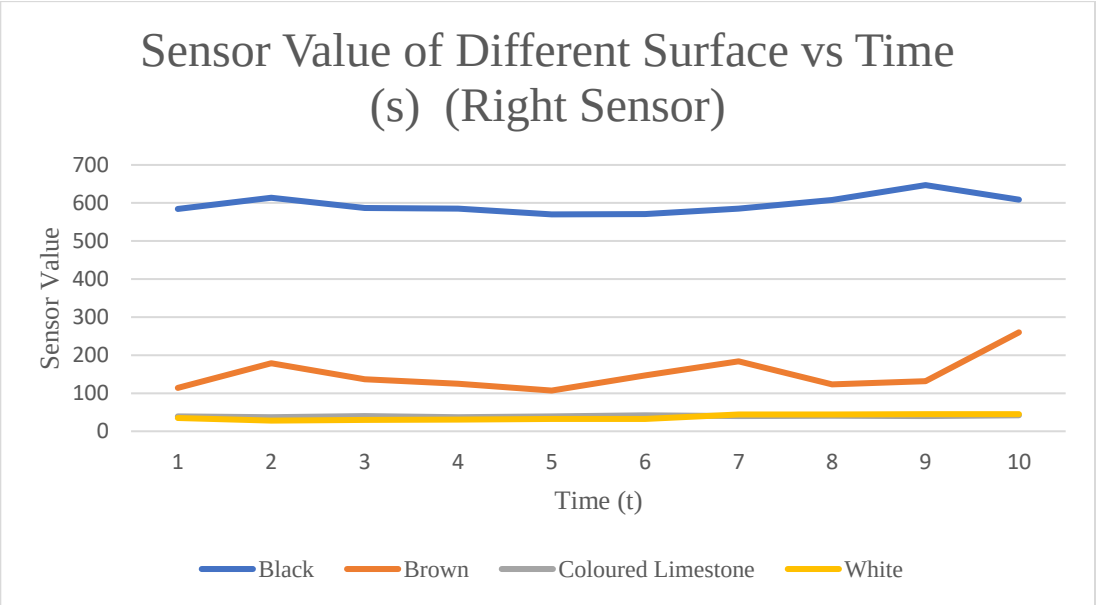


Figure 92: Graph of Sensor Value of different surfaces vs time(s) for right IR sensor

From the 3 graphs plotted, the difference between sensor values of different surface is very obvious for left sensor and middle sensor. However for the right sensor, white and coloured limestone surface cause the sensor value to be in the same range of value. As a result, choosing different coloured tapes is crucial for accurate path navigation. For example, if the floor surface is a coloured limestone or white/bright in colour, then the tape chosen for the line path should be a black tape. And if the floor surface has sensor value almost the same as the brown surface, then a bright/white-coloured tape is recommended. By doing so, the sensor is able to tell the difference between the floor and the line. In any situations, white tape should not be paired together with the floor which has similar colour as the coloured limestone surface.

4.2 Safety Features of the Internal Compartment

Inside the lower compartment of the robot, many screws and nuts are protruding out due to the connection between the wooden board and the L-shaped brackets, bearings, and even DC motor platforms. Therefore, there will be a risk of short circuit occurred at the base of the circuit board if the circuit board were to contact with the screws or nuts during the testing and calibration. Therefore, some plasticines are used to cover the screws and nuts, so that the metallic components can be isolated from the metal strips at the base of the vero boards.

Figure 93 and 94 below show the covering of metallic part such as screws and nuts with plasticine so that the bottom of the vero stripped board won't be in contact with the parts, thus preventing any potential short circuit hazard.

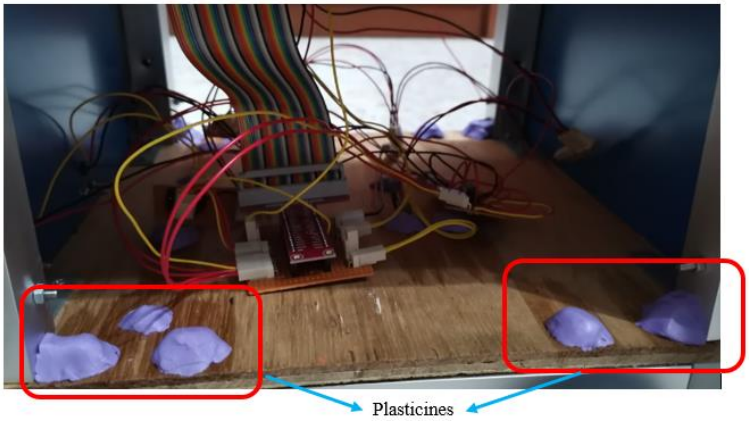


Figure 93: Covering of screws and nuts from the bearings at the robot base

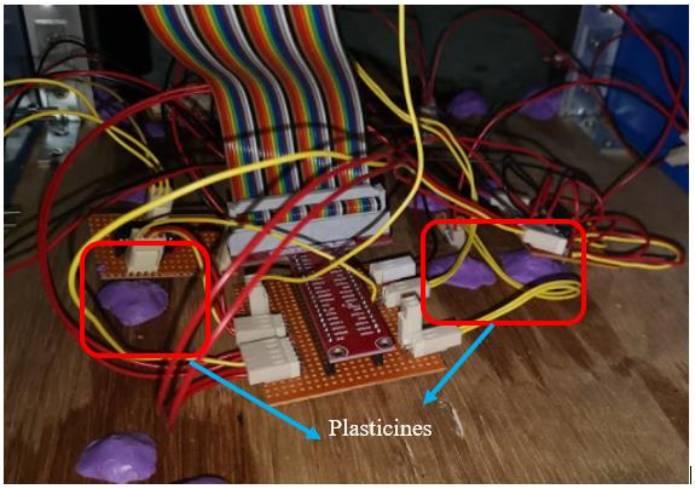


Figure 94: Covering of screws and nuts from the ultrasonic sensor holder and DC motor platforms

4.3 Troubleshooting Circuit Issue and Testing Connectivity of the Circuit

After soldering and wiring of the circuit, troubleshooting of the circuit is necessary to be done. As we know that, we may accidentally "over-solder" the components until a little portion of the molten solder accidentally slips to the 2nd metal strip below the circuit board, causing short circuit which the feature won't work properly. Therefore, a multimeter with a connectivity function is used to check the presence of short circuit occurred. During the testing, if a beeping sound is heard continuously, it indicates that there's a short circuit occurs on the point probed by the multimeter. Therefore, in order to solve this short circuit issue, a de-soldering pump will be used together with the soldering tool, in which the soldering metal will be melted by the soldering tool before the pump is able to suck it out.

Figures 95 and 96 show how the testing of short circuit is being done using the multimeter probes.

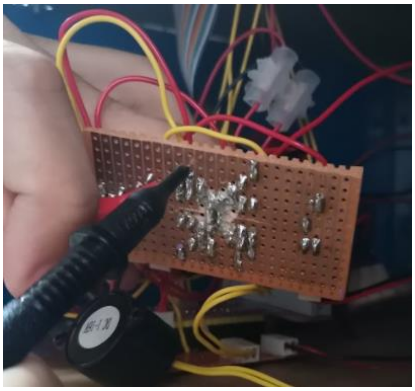


Figure 95: Multimeter probing on the point of interest below the vero stripped board

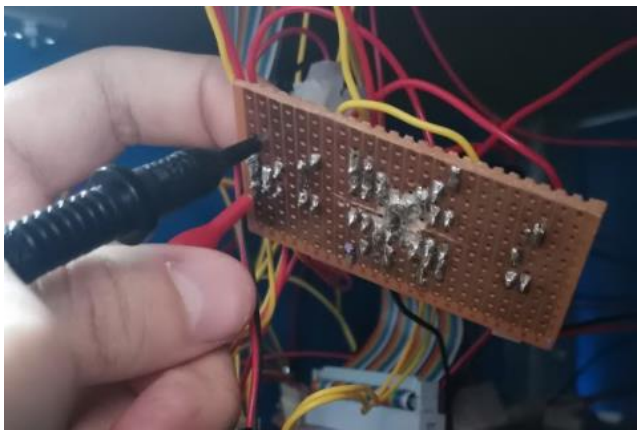


Figure 96: Multimeter probing on the point of interest below the vero stripped board (2nd image)

On top of checking whether there is any short circuit occurs within the circuit boards, there is also a need of testing whether the wiring connections made from the GPIO board/circuit boards to the sensors/DC motors have continuous connectivity. This is because when the wire is not crimped tightly with the crimp pins, loose connection which breaks the connection may occur, if the loosely crimped wire is inserted into the 2/3/4-pin header.

Therefore, in order to test the wiring connectivity, again the connectivity function of the multimeter probe will be used again. This time, one of the probe will be pointed on the circuit board and the other probe will be pointed at the sensor pins/DC motors. If a beeping sound is heard, then it indicates that the connection is present, thus the sensors/DC motors are able to work properly.

Figures 97 and 98 show how the connection test is being carried out for a DC motor connection.

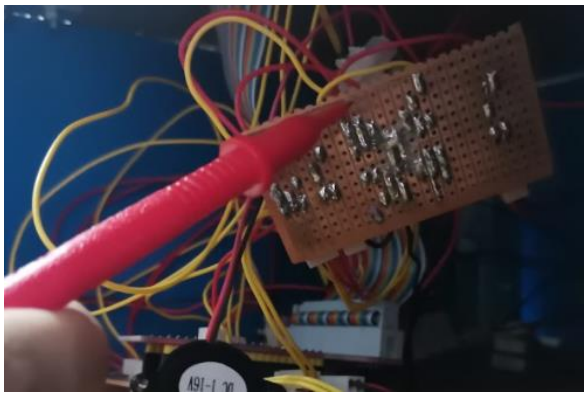


Figure 97: Probing of the output of L293 DC motor driver which is connected to the DC motor directly

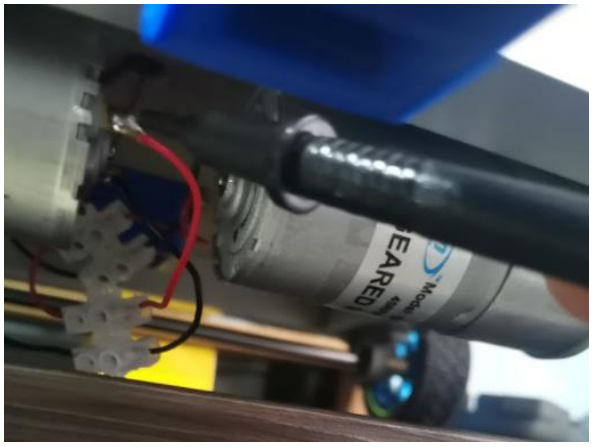


Figure 98: Probing of the one of the polarity pins of DC motor

4.4 Results from Mask Wearing and Machine Learning Model Training

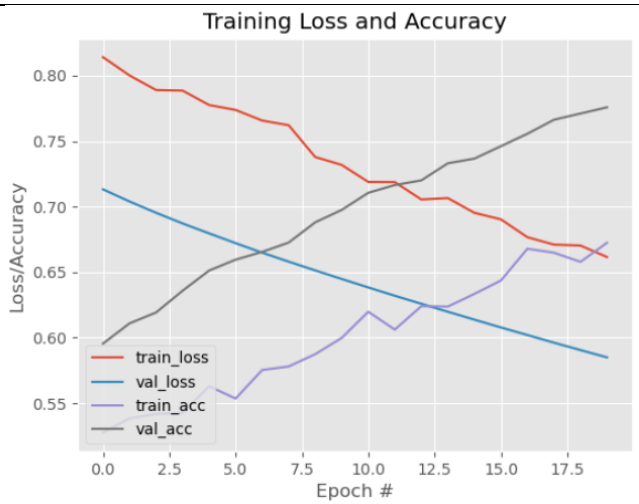
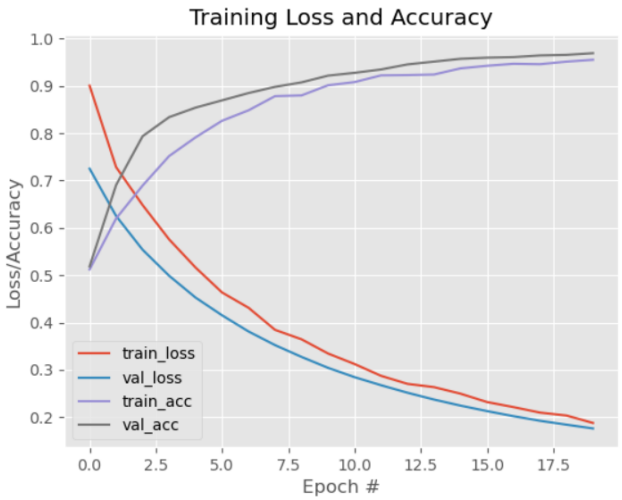
4.4.1 Changing Machine Learning Training Model Parameters

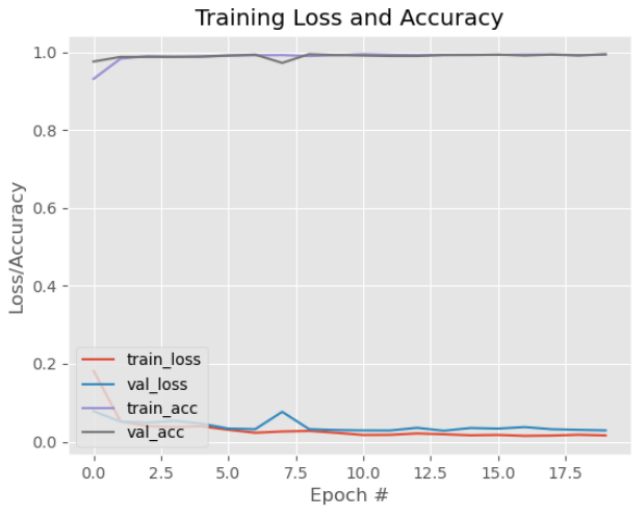
Experiment 1: Changing initial learning Rate while fixing the value for batch size and the epoch

In this section, the three parameters, initial learning rate, batch size and the epoch will be manipulated for the model training task. For this experiment, the epoch and the batch size will be fixed, while changing the initial learning rate. The epoch and the batch size will be fixed at 20 and 64 respectively. While the initial learning rate will be set at $10^{-6}, 10^{-5}, 10^{-4}, and\ 10^{-3}$.

Table 10 below tabulated the learning curve plotted by the AI model training algorithm for different initial learning rate used, with Figures 99-102 below showing the learning curves plotted for each of the initial learning rates:

Table 10:Tabulation of Learning Curve Plotted for different initial learning rates

Initial Learning Rate	Learning Curve
10^{-6}	 *Figure 99: Learning Curve (Initial Learning Rate=10^{-6})*
10^{-5}	 *Figure 100: Learning Curve (Initial Learning Rate=10^{-5})*

10^{-4}	 *Figure 101: Learning Curve (Initial Learning Rate=10^{-4})*
10^{-3}	 *Figure 102: Learning Curve (Initial Learning Rate=10^{-3})*

From the results obtained, at an initial learning rate of 10^{-6} shown in Figure 99 above, the training and validation losses do not converge at the end, and it decreases and also continue to decrease at the end of the graph. This shows that the model is a underfit model as it still has further room for learning improvements. For initial learning rate of 10^{-5} shown in Figure 100 above, the curve decreases until the end of the plot. However, it does not show any steady state towards the end of the plot. This shows that the learning curve is somehow between underfit and good fit. For the maximum initial learning rate set which is 10^{-3} shown in Figure 101 above, both the training and validation loss almost converge at the end of the plot, but there is a slight spike of the validation loss in the middle. Therefore, initial learning rate of 10^{-4} shown in Figure 101 above is the optimum rate value as both losses converge at the end of the plot, and they decreases to a point of stability. Furthermore, the training loss is lower than the validation loss towards the end of plot. Therefore, an initial learning rate of 10^{-4} produces an almost good fit model.

Experiment 2: Changing batch size while fixing the value for initial learning rate and the epoch

For this experiment, the epoch and the initial learning rate will be fixed, while changing the initial learning rate. The epoch and the initial learning rate will be fixed at 20 and 10^{-4}, respectively. While the initial learning rate will be set at 32, 64, 128, and 256.

Table 11 below tabulated the learning curve plotted by the AI model training algorithm for different batch sizes used, with Figures 103-106 below showing the learning curves plotted for each of the initial learning rates:

Table 11: Tabulation of Learning Curve Plotted for different batch sizes

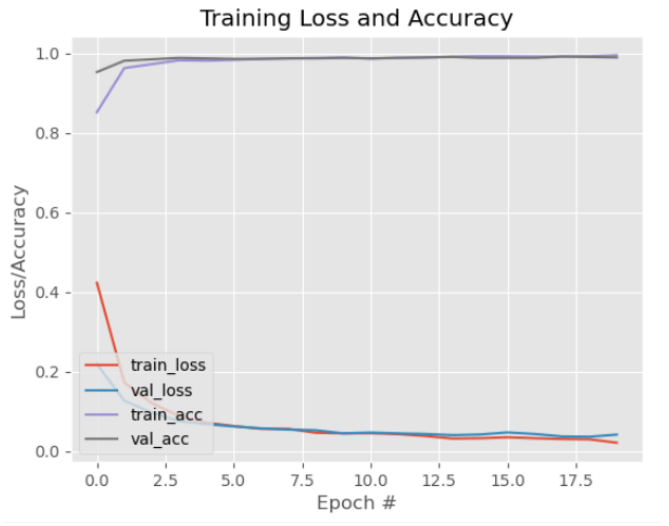
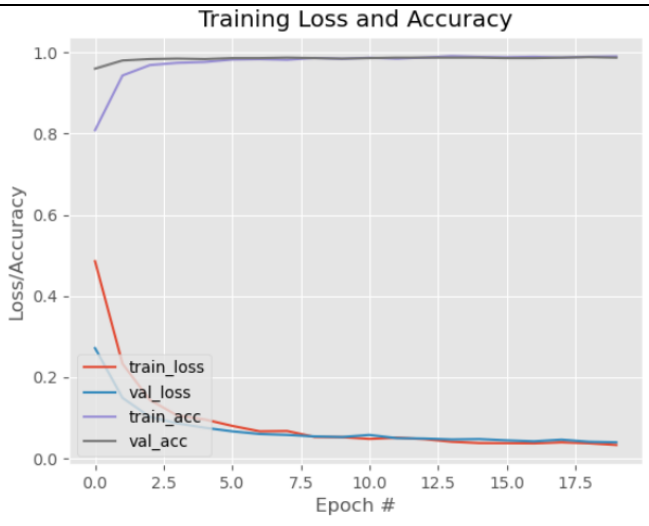
Batch Size	Learning Curve
32	
64	

Figure 103: Learning Curve (Batch Size = 32)

Figure 104: Learning Curve (Batch Size = 64)

128	

Figure 105: Learning Curve (Batch Size = 128)

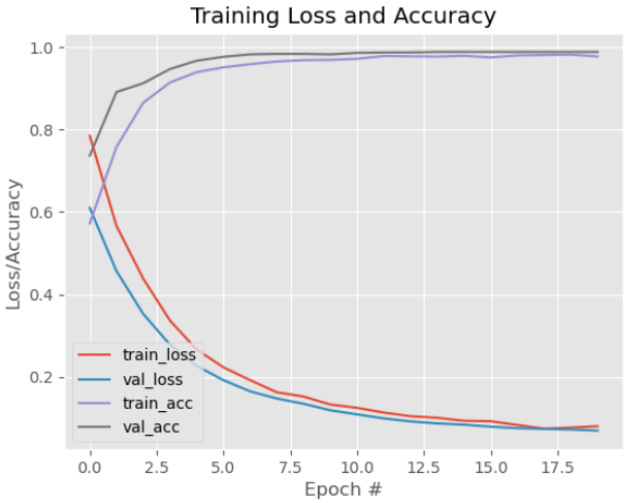
256	

Figure 106: Learning Curve (Batch Size = 256)

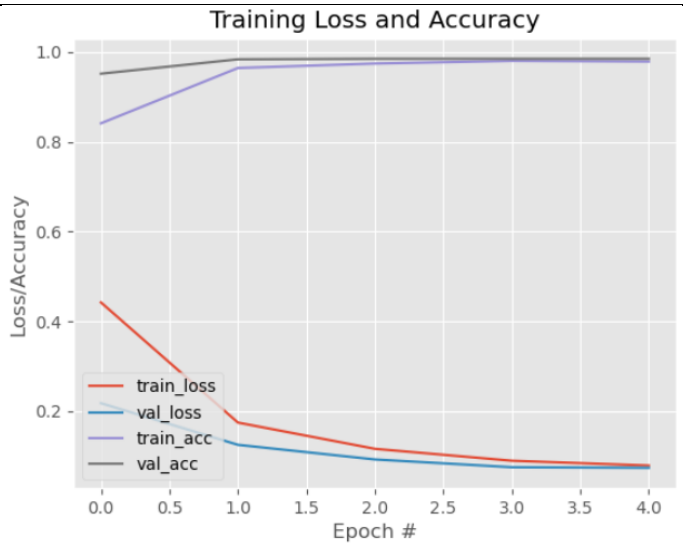
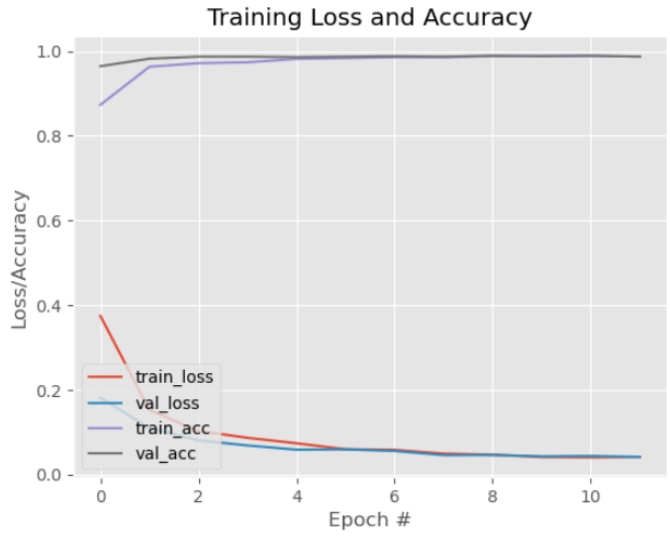
From the results obtained, for batch size equal to 32, 64 or 128 shown in Figure 103 - 105, the learning curves show a near good fit model, as the training and validation loss come to a steady point after decreasing from the beginning. Both losses converge towards each other at the end of the plot. For batch size equals to 256 shown in Figure 106, the training and validation loss also decrease without any increase again, which is also considered acceptable. However, these 2 curves do not show much constant point of steady at the end of plot. Therefore, batch size of 32 and 64 are the most suitable.

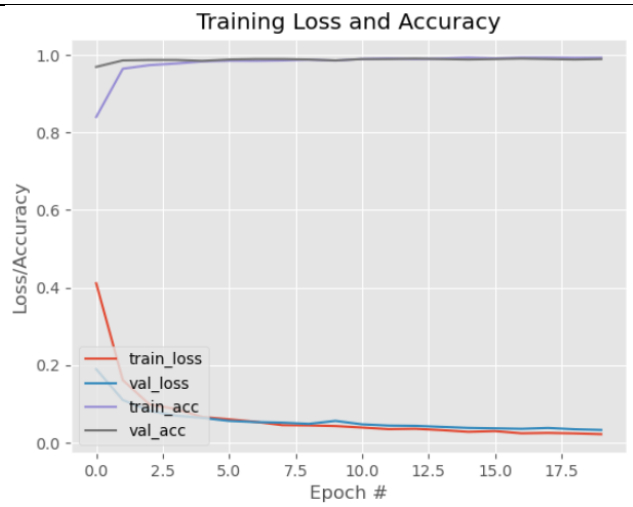
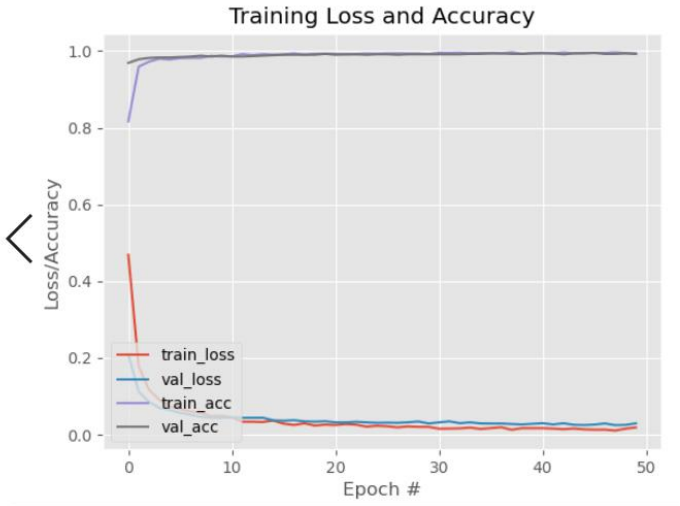
Experiment 3: Changing epoch while fixing the value for initial learning rate and the batch size

For this experiment, the batch size and the initial learning rate will be fixed, while changing the epoch. The batch size and the initial learning rate will be fixed at 20 and 10^{-4}, respectively. While the epoch will be set at 5, 12, 20, and 50.

Table 12 below tabulated the learning curve plotted by the AI model training algorithm for different epochs used, with Figures 107-110 below showing the learning curves plotted for each of the initial learning rates:

Table 12: Tabulation of Learning Curve Plotted for different epochs

Epochs	Learning Curve
5	 *Figure 107: Learning Curve (Epoch = 5)*
12	 *Figure 108: Learning Curve (Epoch = 12)*

20	 *Figure 109: Learning Curve (Epoch = 20)*
50	 *Figure 110: Learning Curve (Epoch = 50)*

Based on the results obtained, learning curve with an epoch of 5 is not enough for the model to learn further as the length of the point of steady is not sufficient. For epoch of 12 or 20, it can be seen that the training and validation loss reaches the point of steady before the epoch reaches the end of the plot. Therefore, an epoch of between 12 and 20 is suitable for the model to be trained. For learning curve of epoch = 50, the training and validation loss have already reached the point of steady for certain amount of epochs, therefore additional training until the epoch reaches 50 is not necessary and not recommended, as if the number of epochs is set too high, it can cause the model to be overfit.

4.4.2 Results from Mask Wearing

Figures 111 and 112 below show the results obtained from the face mask detection.

Figure 111: Results from Camera (With Mask)

Figure 112: Results from Camera (Without Mask)

Based on the results, it is shown that the algorithm is able to distinguish a person wearing a mask and also not wearing a mask. From Figure 111 above, the number value beside the word "Mask" shows the prediction score for the detection of presence/absence of mask. Here in Figure 111, '99.97%' indicates that the algorithm is 99.97% sure that the person is wearing a mask. Even if the person covers partially his face with hand, the algorithm is still able to detect the presence of mask, as most of the mask still can be seen from the camera. In Figure 112 above, the algorithm will not able to detect the mask on a person's face if he/she is not wearing any mask or pulls down the mask to the chin.

5. Summary of the Objectives and Tasks Achieved

Table 13 below summarizes the objectives and the tasks achieved throughout this final year project:

Table 13: List of Tasks Accomplished

Objectives/Tasks	Remarks
Path Navigation and Obstacle Detection	• The robot is able to detect the obstacles from side and in front. • The turning of the robot is almost 90 degree for a smooth surface • The robot is also able to detect the wall at the side of the robot when it moves too close along the wall.
Cliff Detection	• The robot is able to detect the edge of the cliff before the robot runs down the cliff and cause damage to itself. • For cliff of height around 2-3 cm, the robot will just run down to the lower side of the floor without damaging itself due to the smaller distance.
Security Implementation	• The robot is able to detect whether someone attempts to steal the robot. The buzzer works properly with the alarm disabling feature as well.
Detection of whether citizens abide by SOP	• The camera is able to detect the absence/presence of mask wearing and is able to produce the nagging message. • The algorithm for detection of social distancing is currently under development and testing

Table 14 below shows 3 limitations for the roaming "nagger", with the first 2 limitations are mainly due to the budget constraint.

Table 14: List of Limitations present in this project and their suggested solutions if no budget constraint

Limitations	**Solutions if there is no budget constraint**
The robot is unable to perform a perfect 90 degree turn when placed in a rougher surface due to the frictional force acted on the wheels.	Using of omnidirectional wheel will help greatly in this limitation. As the robot does not even need to turn around as the omnidirectional wheels are able to take care the motion in all 4 directions.
The robot is unable to perform indoor mapping and automated path navigation. Instead, it relies mostly on the analogue sensors such as ultrasonic sensors.	LIDAR sensor is able to perform an indoor mapping using SLAM algorithm.
2 programs are required to run the robot every time. The 1st program is written on desktop and will control and program the GPIOs of Raspberry Pi 4B through SSH method. The 2nd program is run from the VNC viewer so that Raspberry Pi desktop can be accessed remotely. The 2nd program in charge of only Pi camera interface and speaker.	This limitation is still being looked through. Therefore, there is no better approach for this limitation at the moment.

6. Conclusion

In overall, most of the features are able to work properly as expected. Student managed to integrate the mechanical, electronic, software, and computer vision field altogether so that the functions of the roaming "nagger" can be achieved. On top of gaining research skills, student also learn numerous tasks involving hands on, such as building of the mechanical prototype and soldering and wiring of the circuit boards. However, further testing and calibrations are still required so that the results will be more accurate, particularly in the turning of the robot and the face mask detection. Student also learn that controlling the turning of the robot using timing function in the program will not provide the best result. Therefore, additional sensors are required to detect the angle of rotation of the robot. Due to the budget constraints, some of the features are not working in their best performance. For example, due to budget constraint LIDAR sensor which is capable of performing SLAM algorithm and indoor mapping is not purchased.

7. Future Work and Recommendation

For the future work, a web application can be created so that the video captured by the camera can be streamed online. For the security implementation, instead of displaying the password input at the console, a Graphic User Interface (GUI) can be created to prompt user to enter the correct default set password so that the buzzer can be deactivated. From these 2 recommendations, it will enhance the user-friendliness of the robot. Additionally, the detection for social distancing will be implanted as well, other than just the face mask detection. There are limited range of distance in which the AI algorithm can detect the face mask, which is capped maximum at 1.8 m. In order to increase the maximum distance it can detect, more images of the person wearing/not wearing mask far from the camera must be taken and loaded into the dataset, so that the algorithm can learn how to detect face mask in longer distance away from the camera. For better angle detection for face mask detection, the dimension of the prototype can be scaled up from 30cm x 30cm x 50cm to around 45cm x 45cm x 75cm, so that the detection of face mask can be improved.

8. References

[1] Shneier, M & Bostelman, R 2015, *Literature Review of Mobile Robots for Manufacturing*.

[2] Clemens, J, Reineking, T & Kluth, T 2016, "An evidential approach to SLAM, path planning, and active exploration", *International Journal of Approximate Reasoning*, vol. 73, pp. 1-26.

[3] Robotics Tomorrow, 2015. 'Low-Cost, LIDAR-based Navigation for Mobile Robotics', Robotics Tomorrow, viewed 7 November 2020, < https://www.roboticstomorrow.com/article/2015/11/low-cost-lidar-based-navigation-for-mobile-robotics/7270 > (accessed 5th November 2020)

[4] Higgins, S, 2020. 'How SLAM affects the accuracy of your scan (and how to improve it)', Navvis, viewed 11 November 2020, < https://www.navvis.com/blog/how-slam-affects-the-accuracy-of-your-scan-and-how-to-improve-it > (accessed 5th November 2020)

[5] Kumar, G, Patil, A, Patil, R, Park, S & Chai, Y 2017, "A LiDAR and IMU Integrated Indoor Navigation System for UAVs and Its Application in Real-Time Pipeline Classification", *Sensors*, vol. 17, no. 6, p. 1268.

[6] Chowdhury, N, Khushi, D & Rashid, M 2017, "Algorithm for Line Follower Robots to Follow Critical Paths with Minimum Number of Sensors", *International Journal of Computer (IJC)*, vol. 24, no. 1, pp. 13-22

[7] Wei, X, Dong, E, Liu, C, Han, G & Yang, J 2017, "A wall-following algorithm based on dynamic virtual walls for mobile robots navigation", *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 46-51.

[8] Malamas, E, Petrakis, E, Zervakis, M, Petit, L & Legat, J 2003, "A survey on industrial vision systems, applications and tools", *Image and Vision Computing*, vol. 21, no. 2, pp. 171-188.

[9] Cui, L 2019, "Complex industrial automation data stream mining algorithm based on random Internet of robotic things", *Automatika*, vol. 60, no. 5, pp. 570-579.

[10] Anon 2021, *Types of machine vision systems*, [Online] <https://www.vision-systems.com/knowledge-zone/article/14040180/types-of-machine-vision-systems#:~:text=2D%20vision%20systems%20provide%20area,for%20most%20machine%20vision%20applications.&text=Camera%20sensors%20in%2010%2C%2020,standard%202D%20product%20lineups%2C%20however.>.

[11] Schumann-Olsen, H 2021, *Why 3D machine vision? What's wrong with 2D machine vision?*, viewed 3 June, 2021, <https://blog.zivid.com/why-3d-machine-vision-whats-wrong-with-2d-machine-vision>.

[12] Goodfellow, I, Bengio, Y & Courville, A 2016, *Deep learning*, The MIT Press; Illustrated edition (November 18, 2016), p. 429.

[13] Bishop, C 1995, *Neural Networks for Pattern Recognition*, 1st ed, Clarendon Press; 1st edition (January 18, 1996), p. 95.

[14] Anzanello, M, Fogliatto, F 2011, "Learning curve models and applications: Literature review and research directions", International Journal of Industrial Ergonomics, vol. 41, pp. 573-583

[15] James, G, Witten, D, Hastie, T & Tibshirani, R 1999, *An introduction to statistical learning*, Bradford Books; Illustrated edition (February 17, 1999), pp. 22-24

9. Appendix

Appendix A: CAD Drawings and Dimensions

CAD Drawing for the Final Prototype Assembly

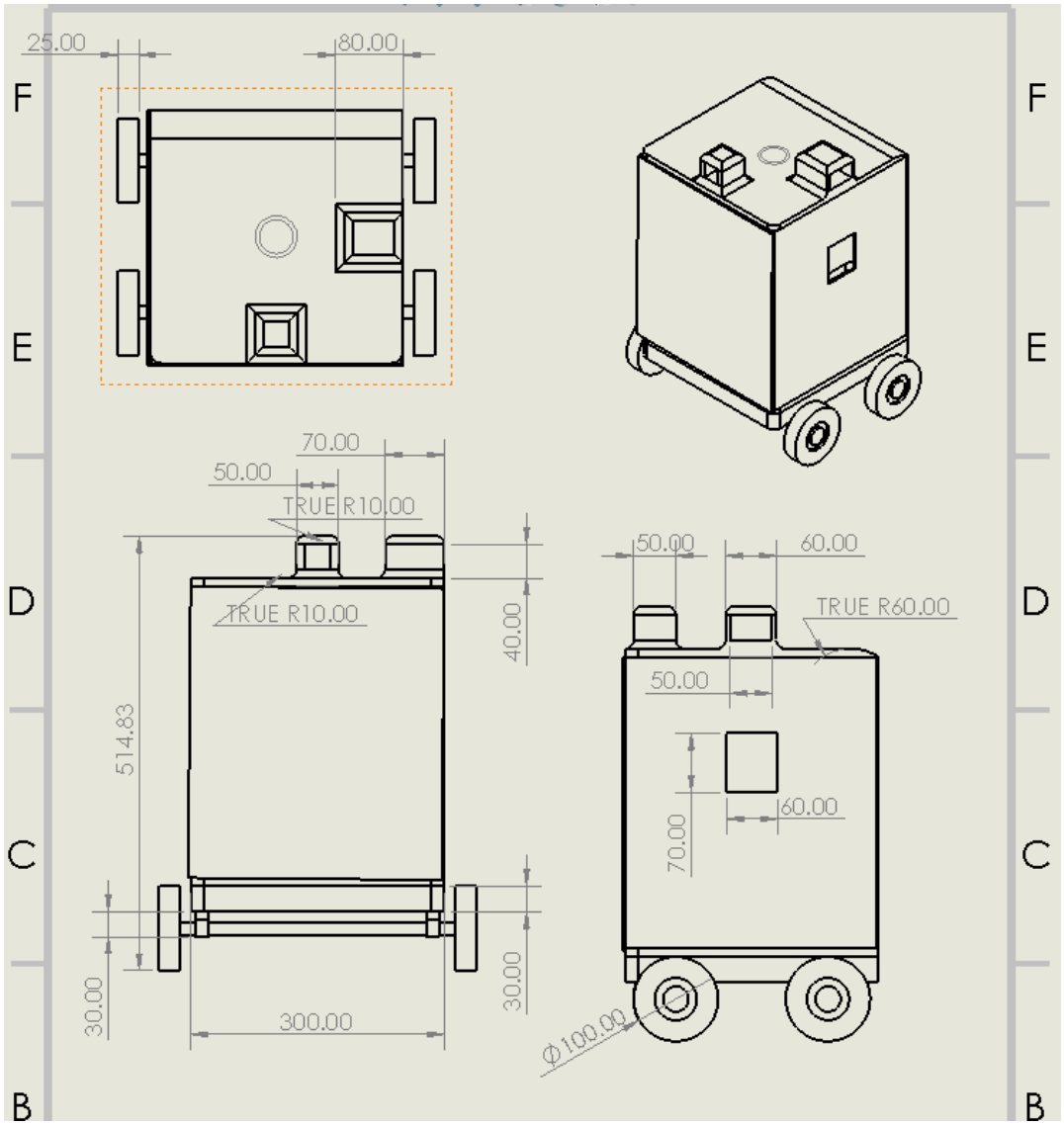


Figure 113: CAD Drawing Diagram for the Prototype

CAD Drawing for the DC Motor Platform

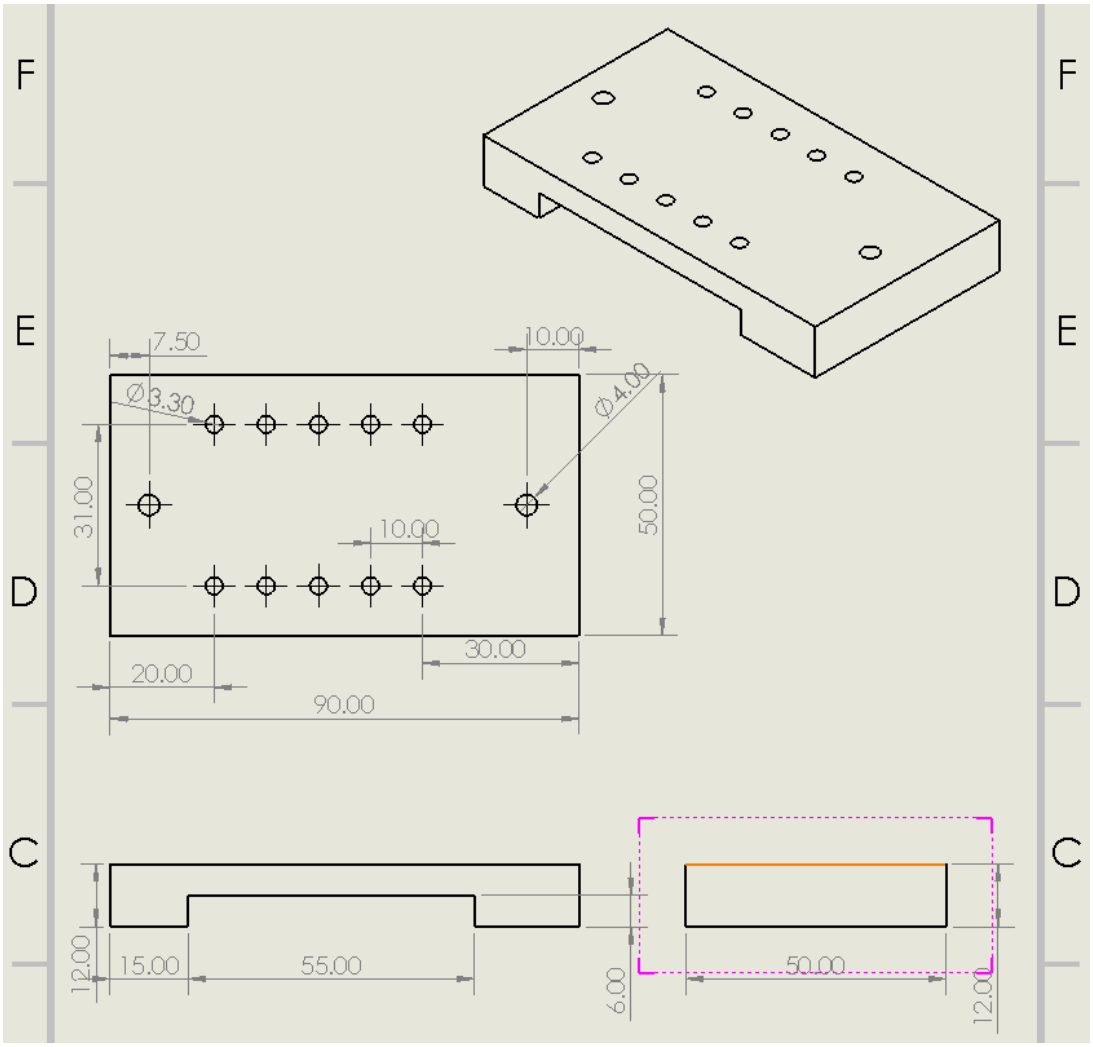


Figure 114: CAD Drawing Diagram for the DC Motor Platform

CAD Drawing for IR Sensor Holder

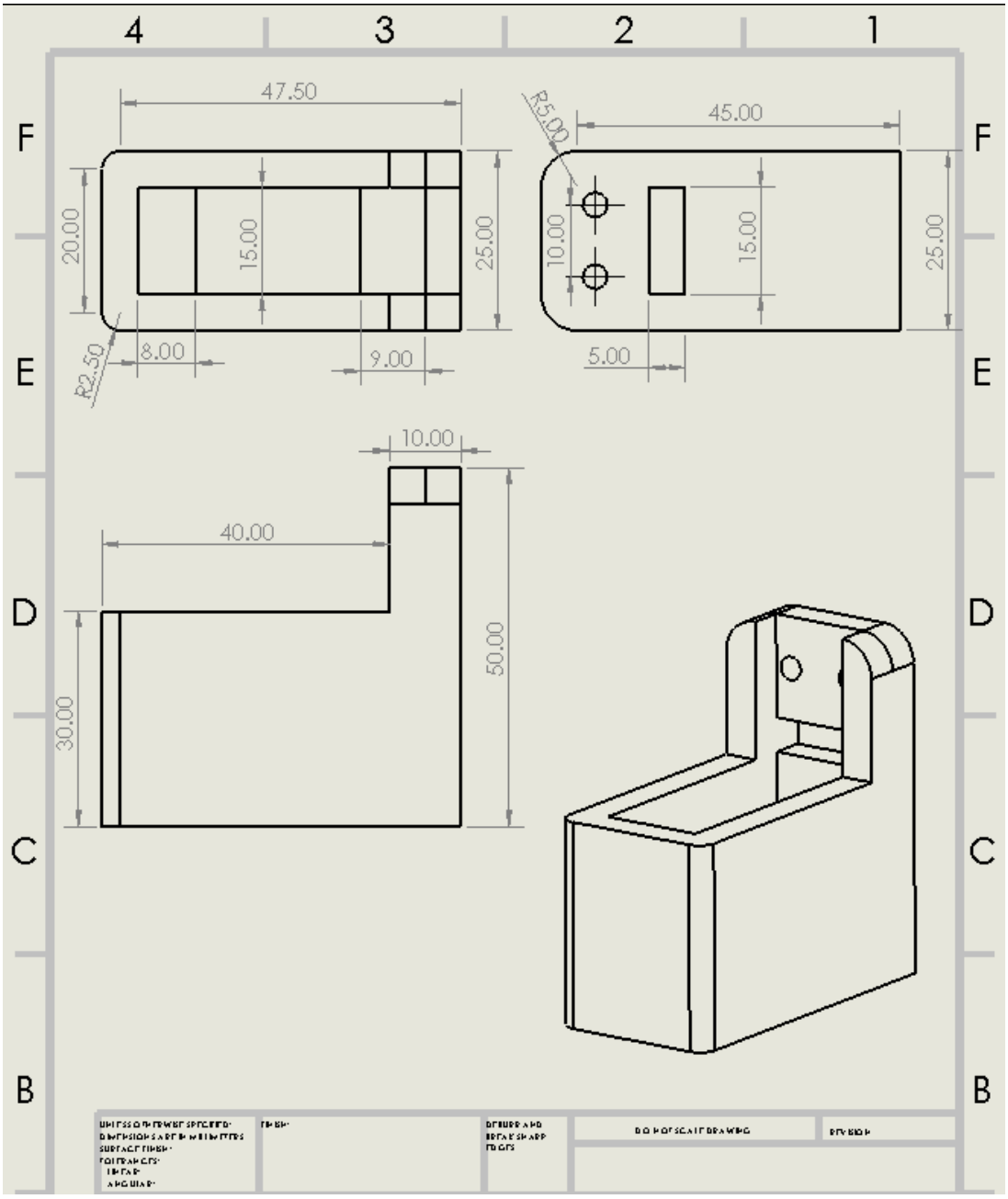


Figure 115: CAD Drawing Diagram for the IR Sensor Holder

CAD Drawing for the Ultrasonic Sensor Holder (Robot Base)

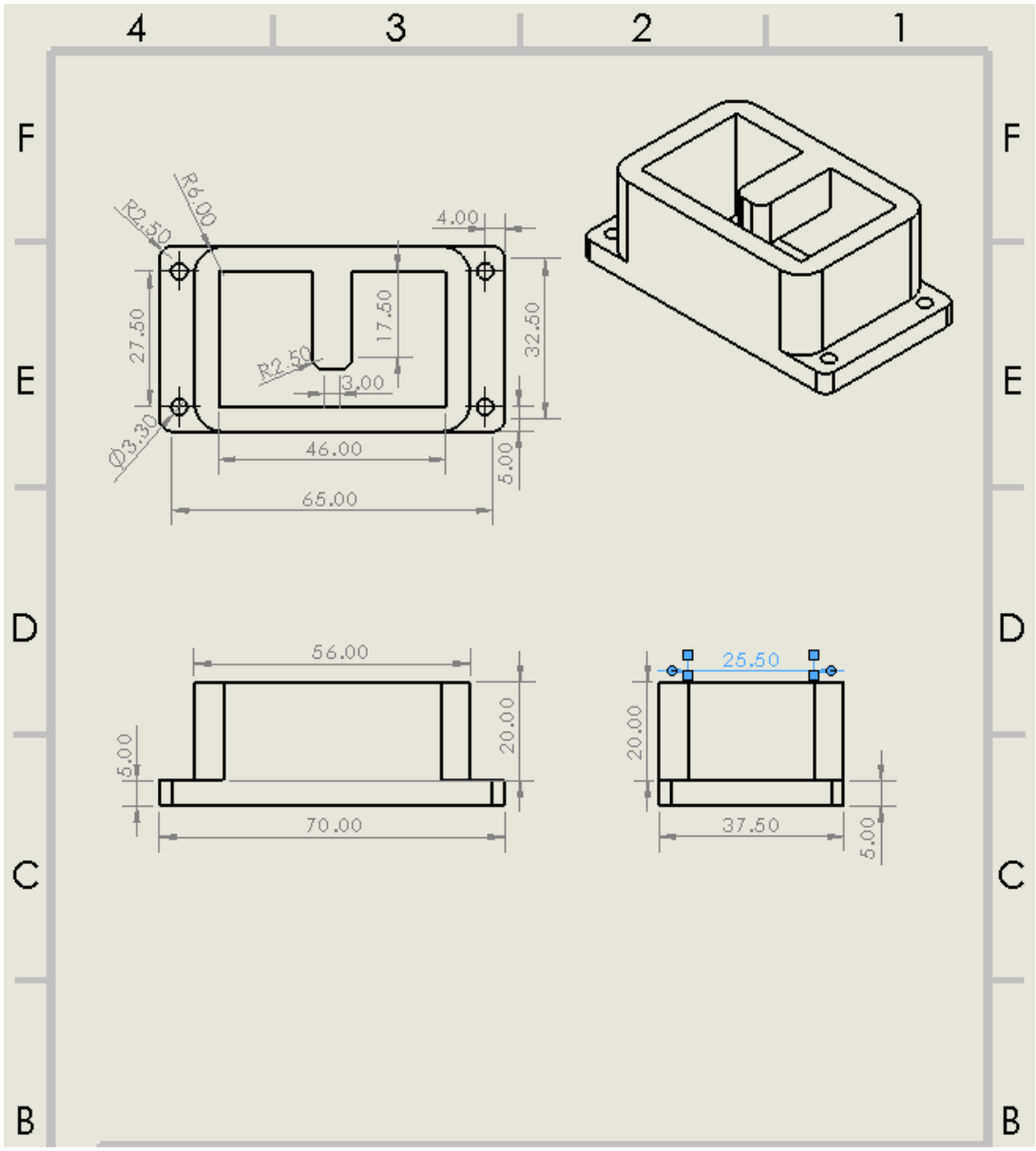


Figure 116: CAD Drawing Diagram for the Ultrasonic Sensor Holder (Robot Base)

CAD Drawing for Ultrasonic Sensor Holder (Wall)

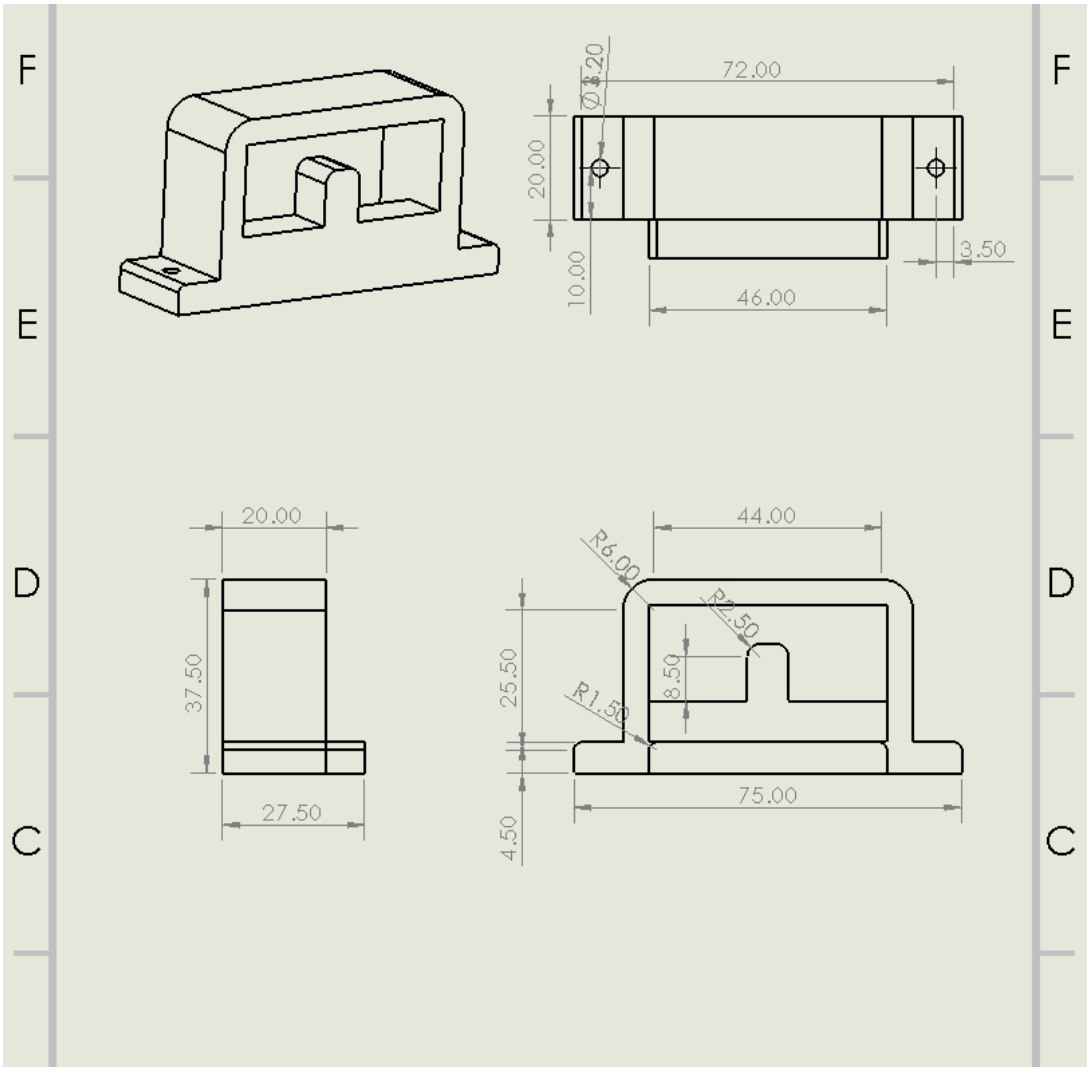


Figure 117: CAD Drawing Diagram for the Ultrasonic Sensor Holder (Top of Robot)

CAD Drawing for IR Sensor Connector/Extend

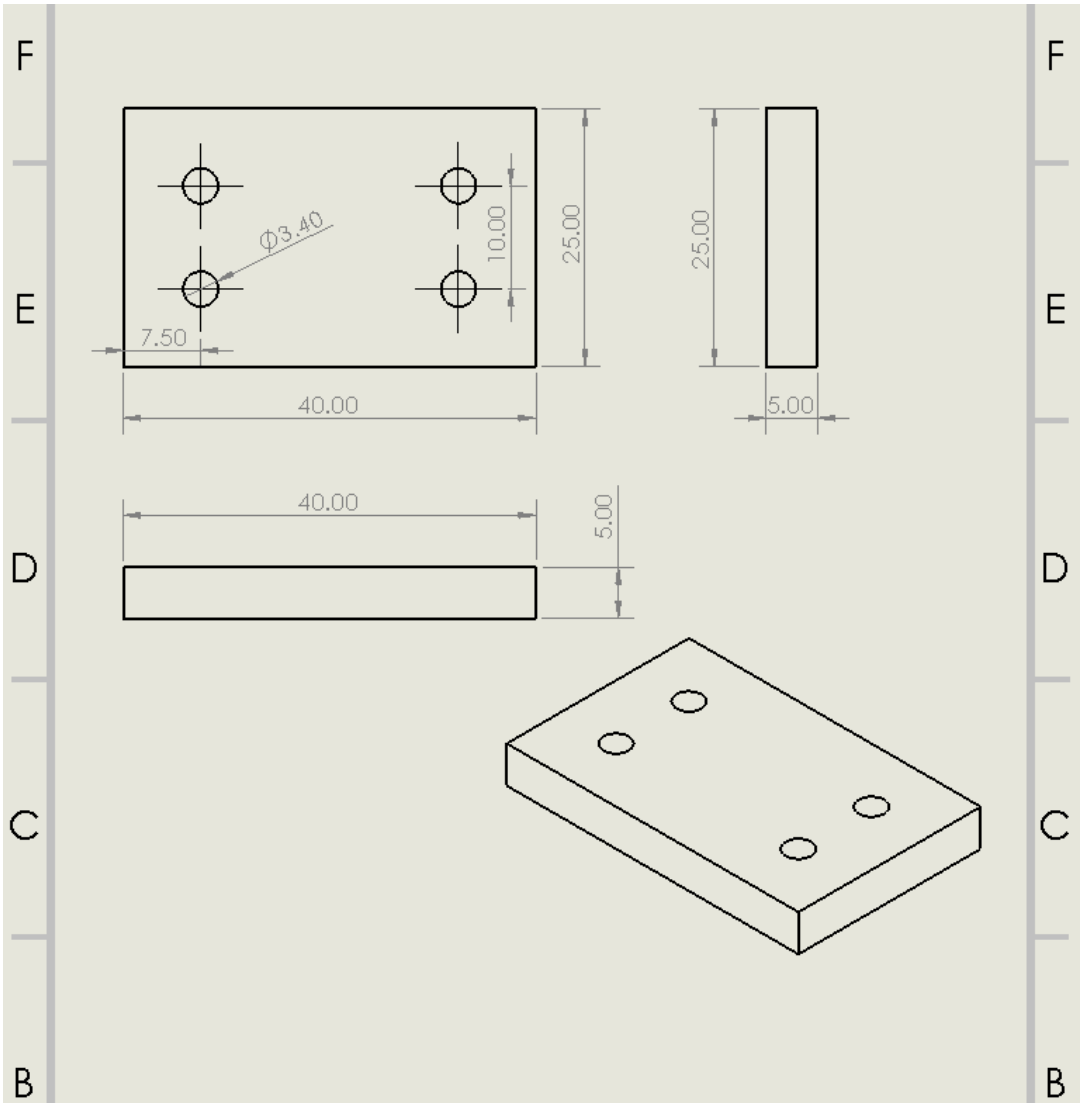


Figure 118: CAD Drawing Diagram for the IR Sensor Extend

CAD Drawing for Lower Camera Holder

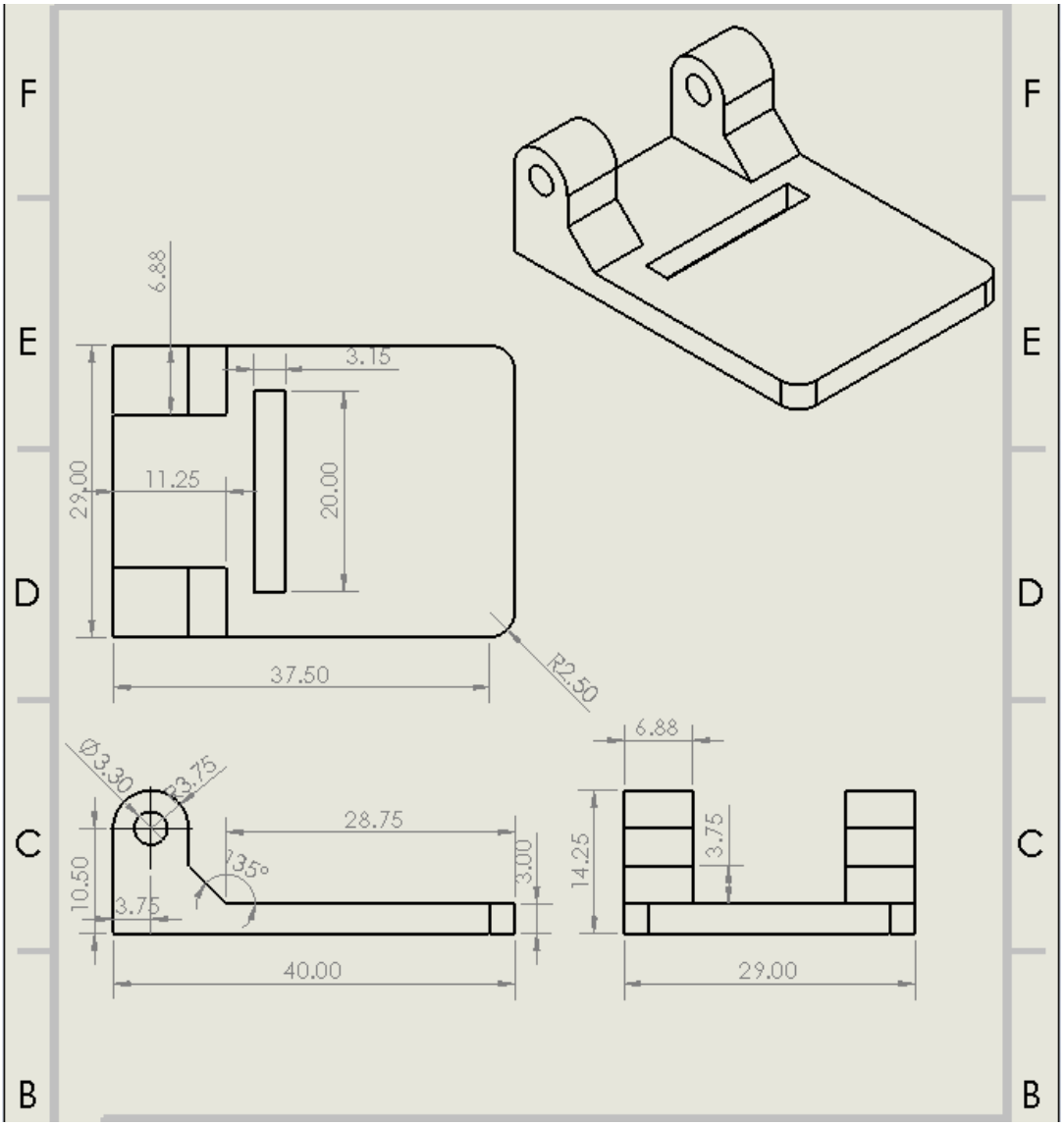


Figure 119: CAD Drawing Diagram for the Camera Holder (Lower Part)

Appendix B: Source Code (Main Program)

```
#Import the required libraries

import spidev
import time
import RPi.GPIO as GPIO
import random

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 1000000

# Define sensor channels for each IR sensor
ir1_channel = 0 #left sensor
ir2_channel = 1 #middle sensor
ir3_channel = 2 #right sensor

#GPIO pin number for the DC motor inputs and PWM input
Left_DC_1 = 5
Left_DC_2 = 6
Right_DC_1 = 26
Right_DC_2 = 19
DC_PWM = 13

BUZZER = 16 #GPIO pin number for the buzzer

# setup gpio for echo & trig
echopin = [20, 23, 17]
trigpin = [21, 24, 27]

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(Left_DC_1,GPIO.OUT)  # All motor input pins as
Outputs
GPIO.setup(Left_DC_2,GPIO.OUT)
GPIO.setup(Right_DC_1,GPIO.OUT)
GPIO.setup(Right_DC_2,GPIO.OUT)
GPIO.setup(DC_PWM,GPIO.OUT)

GPIO.setup(BUZZER,GPIO.OUT)

#Echo pin as input, and Trigger pin as output for the
Ultrasonic Sensor Configuration
for j in range(3):
    GPIO.setup(trigpin[j], GPIO.OUT)
    GPIO.setup(echopin[j], GPIO.IN)

# Define delay between readings
delay = 1
```

```
def ping(echo, trig):

    GPIO.output(trig, False)
    # Allow module to settle
    time.sleep(0.5)

    # Send 10us pulse to trigger
    GPIO.output(trig, True)
    time.sleep(0.00001)
    GPIO.output(trig, False)
    pulse_start = time.time()

    # save StartTime
    while GPIO.input(echo) == 0:
        pulse_start = time.time()

    # save time of arrival
    while GPIO.input(echo) == 1:
        pulse_end = time.time()

    # time difference between start and arrival
    pulse_duration = pulse_end - pulse_start

    # mutiply with the sonic speed (34300 cm/s) and
divide by 2, because there and back
    distance = round(pulse_duration * 17150, 2)

    return distance

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts

# Function to calculate temperature from
# TMP36 data, rounded to specified
# number of decimal places.
def ConvertTemp(data,places):
    temp = ((data * 330)/float(1023))-50
    temp = round(temp,places)
    return temp
```

```
def RobotForward():
    GPIO.output(Left_DC_1,GPIO.LOW)
    GPIO.output(Left_DC_2,GPIO.HIGH)
    GPIO.output(Right_DC_1,GPIO.HIGH)
    GPIO.output(Right_DC_2,GPIO.LOW)

def RobotReverse():
    GPIO.output(Left_DC_1,GPIO.HIGH)
    GPIO.output(Left_DC_2,GPIO.LOW)
    GPIO.output(Right_DC_1,GPIO.LOW)
    GPIO.output(Right_DC_2,GPIO.HIGH)

def Turn_Right():
    GPIO.output(Left_DC_1,GPIO.HIGH)
    GPIO.output(Left_DC_2,GPIO.LOW)
    GPIO.output(Right_DC_1,GPIO.HIGH)
    GPIO.output(Right_DC_2,GPIO.LOW)

def Turn_Left():
    GPIO.output(Left_DC_1,GPIO.LOW)
    GPIO.output(Left_DC_2,GPIO.HIGH)
    GPIO.output(Right_DC_1,GPIO.LOW)
    GPIO.output(Right_DC_2,GPIO.HIGH)

def RobotStop():
    GPIO.output(Left_DC_1,GPIO.LOW)
    GPIO.output(Left_DC_2,GPIO.LOW)
    GPIO.output(Right_DC_1,GPIO.LOW)
    GPIO.output(Right_DC_2,GPIO.LOW)

pwm=GPIO.PWM(13,50)

while True:

    RobotForward()
    GPIO.output(BUZZER,GPIO.LOW)

    try:
        while True:
            pwm.start(100) #The motor will run at full
speed due to the high torque

            #pwm.start(10)
            # Read the light sensor data
            ir1_level = ReadChannel(ir1_channel)
            ir1_volts = ConvertVolts(ir1_level,2)

            # Read the temperature sensor data
            ir2_level = ReadChannel(ir2_channel)
            ir2_volts = ConvertVolts(ir2_level,2)
```

```
            ir3_level = ReadChannel(ir3_channel)
            ir3_volts = ConvertVolts(ir3_level,2)

            # get distances and assemble data line for
writing
            results = ","
            for j in range(3):

                distance = ping(echopin[j], trigpin[j])
                print ("sensor", j+1,": ",distance,"cm")
                results = results + str(distance) + ","

            results = results + "\n"

            if (ir1_level > 400 or ir2_level > 700 or
ir3_level > 700):
                RobotStop()
                time.sleep(0.25)

                RobotReverse()
                time.sleep(2)

                RobotStop()
                time.sleep(1)

                break

            if (ping(echopin[2], trigpin[2])<10):#If side
is close to wall/obstacles
                RobotStop()
                time.sleep(0.5)

                RobotReverse()
                time.sleep(1.5)

                Turn_Right()
                time.sleep(0.5)

                RobotForward()
                time.sleep(3)

                RobotStop()
                time.sleep(0.5)

                Turn_Left()
                time.sleep(3)

                RobotStop()
                time.sleep(0.5)
```

```
                Turn_Left()
                time.sleep(3)

                RobotStop()
                time.sleep(0.5)

                break


            if (ping(echopin[1], trigpin[1])<40): #If
obstacle is 55cm or less from the robot
                choice = random.randint(0, 1)#The robot
will turn left/right based on the random number generated

                RobotStop()
                time.sleep(1)

                RobotReverse()
                time.sleep(3.5)

                RobotStop()
                time.sleep(1)

                if (choice == 0): #If random number 0 is
generated by the program
                    #Robot will turn right
                    for a in range(4):
                        Turn_Right()
                        time.sleep(2.5)

                        RobotStop()
                        time.sleep(0.5)

                    break

                elif (choice == 1): #If random number 1
is generated by the program
                    #Robot will turn left
                    for a in range(4):
                        Turn_Left()
                        time.sleep(2.5)

                        RobotStop()
                        time.sleep(0.5)

                    break
                break

            if (ping(echopin[0], trigpin[0])>10): #If
robot is lifted up by more than 20cm.
                RobotStop()
```

```
                GPIO.output(BUZZER,GPIO.HIGH) #Buzzer
will be triggered
                while True:

                    print('Please enter the correct
password to disable the alarm:')
                    password = input() #Prompts user to
enter password

                    if (password != '12345'): #Default
password is not matched
                        print('Please enter the correct
password to disable the alarm:')
                        password = input()
                    else:
                        print('Alarm Disabled')
                        GPIO.output(BUZZER,GPIO.LOW)
#Alarm will be disabled.
                        time.sleep(1)

                        while True: #Ask user whether
should continue using the robot
                            print("Continue moving the
robot? (Y/N)")
                            answer = input()

                            if (answer != 'Y' or
answer!='Yes' or answer != 'YES'):
                                print("Continue moving
the robot? (Y/N)")
                                answer = input()
                            else:
                                break
                        break
                break

            # Print out results
            print("----------------------------------------
------")
            print("IR Sensor (Left)) : {}
({}V)".format(ir1_level,ir1_volts))
            print("IR Sensor (Middle) : {}
({}V)".format(ir2_level,ir2_volts))
            print("IR Sensor (Right) : {}
({}V)".format(ir3_level,ir3_volts))

            # Wait before repeating loop
            time.sleep(delay/50)

    except KeyboardInterrupt:
```

```
        pwm.stop()
        GPIO.cleanup()
```

Appendix C: Source Code (Machine Vision and Artificial Intelligence)

Source Code for AI Training Model

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input
from tensorflow.keras.preprocessing.image import
img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

# initialize the initial learning rate, number of epochs
to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"D:\Desktop\Degree Uni\Semester 6\RME 40006
Final Year Research Project 2 (RME)\train_model2\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then
initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

#Import the images from both folder (with and without
mask)
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
```

```
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

        #Store the image into the data array
        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data,
labels,

test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data
augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC
layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet",
include_top=False,
                        input_tensor=Input(shape=(224,
224, 3)))

# construct the head of the model that will be placed on
top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

```
# place the head FC model on top of the base model (this
will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them
so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the
index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1),
predIdxs,
                            target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"],
label="train_loss")
```

```
plt.plot(np.arange(0, N), H.history["val_loss"],
label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"],
label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"],
label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

Source Code for Face Mask Detection

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input
from tensorflow.keras.preprocessing.image import
img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct
a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                 (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the
face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    # initialize our list of faces, their corresponding
locations,
    # and the list of predictions from our face mask
network
    faces = []
    locs = []
    preds = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability)
associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the
confidence is
        # greater than the minimum confidence
        if confidence > 0.5:
            # compute the (x, y)-coordinates of the
bounding box for
            # the object
```

```
            box = detections[0, 0, i, 3:7] * np.array([w,
h, w, h])
            (startX, startY, endX, endY) =
box.astype("int")

            # ensure the bounding boxes fall within the
dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0,
startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1,
endY))

            # extract the face ROI, convert it from BGR
to RGB channel
            # ordering, resize it to 224x224, and
preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)

            # add the face and bounding boxes to their
respective
            # lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))

    # only make a predictions if at least one face was
detected
    if len(faces) > 0:
        # for faster inference we'll make batch
predictions on *all*
        # faces at the same time rather than one-by-one
predictions
        # in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return a 2-tuple of the face locations and their
corresponding
    # locations
    return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath =
r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

```
# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and
resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # detect faces in the frame and determine if they are
wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame,
faceNet, maskNet)

    # loop over the detected face locations and their
corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        # determine the class label and color we'll use
to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No
Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0,
255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask,
withoutMask) * 100)

        # display the label and bounding box rectangle on
the output
        # frame
        cv2.putText(frame, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color,
2)
        cv2.rectangle(frame, (startX, startY), (endX,
endY), color, 2)

    # show the output frame
```

```
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Appendix D: Datasheets

Datasheet for Raspberry Pi 4B

1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with siginficantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is avaiable with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:

- Up to 6x UART
- Up to 6x I2C
- Up to 5x SPI
- 1x SDIO interface
- 1x DPI (Parallel RGB Display)
- 1x PCM
- Up to 2x PWM channels
- Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

3 Mechanical Specification

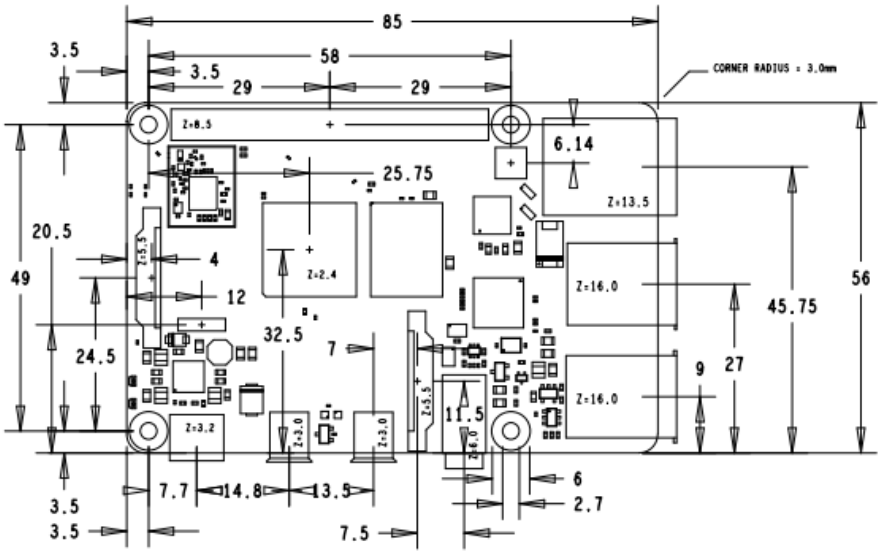


Figure 1: Mechanical Dimensions

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

Figure 2: Digital IO Characteristics

4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

5 Peripherals

5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

5.1.1 GPIO Pin Assignments

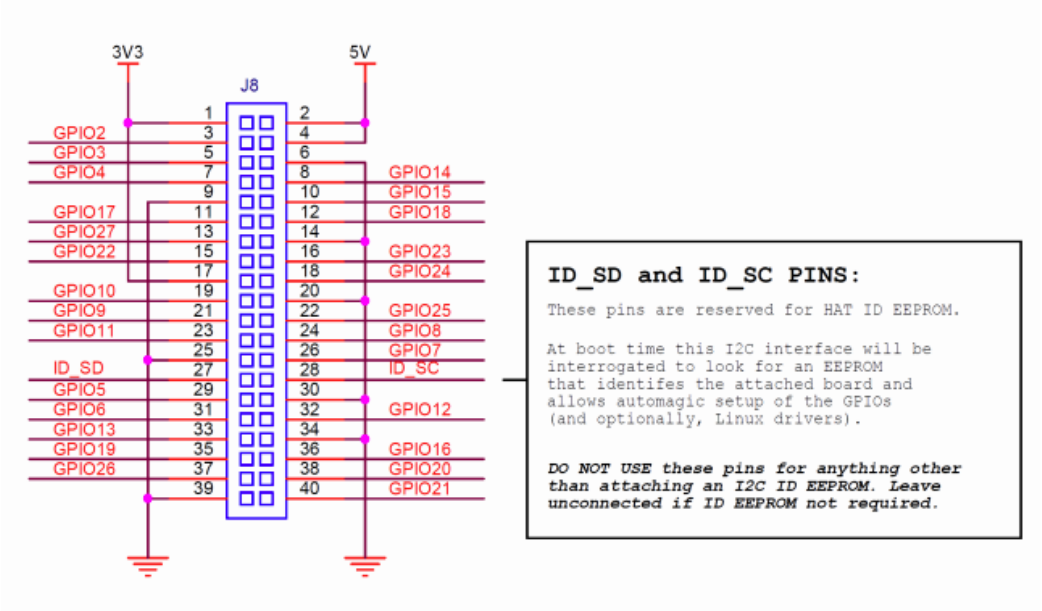


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the hardware documentation section of the website.

5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature at full performance, further cooling may be needed.

Datasheet for L293D DC Motor Driver

TEXAS INSTRUMENTS

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

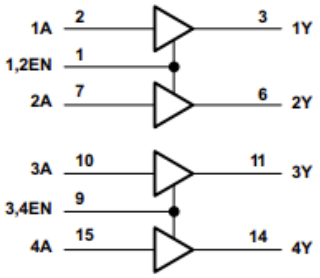
The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

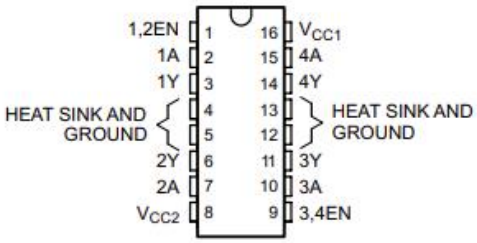
(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram



5 Pin Configuration and Functions

NE Package
16-Pin PDIP
Top View



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

	MIN	MAX	UNIT
Supply voltage, V_{CC1}(2)		36	V
Output supply voltage, V_{CC2}		36	V
Input voltage, V_I		7	V
Output voltage, V_O	–3	V_{CC2} + 3	V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	–2	2	A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	–1.2	1.2	A
Continuous output current, I_O: L293	–1	1	A
Continuous output current, I_O: L293D	–600	600	mA
Maximum junction temperature, T_J		150	°C
Storage temperature, T_{stg}	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to the network ground terminal.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±1000	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
	Supply voltage	V_{CC1}	4.5		7	V
		V_{CC2}	V_{CC1}		36	
V_{IH}	High-level input voltage	$V_{CC1} \le 7$ V	2.3		V_{CC1}	V
		$V_{CC1} \ge 7$ V	2.3		7	V
V_{IL}	Low-level output voltage		–0.3(1)		1.5	V
T_A	Operating free-air temperature		0		70	°C

(1) The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

6.4 Thermal Information

THERMAL METRIC(1)		L293, L293D NE (PDIP) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance (2)	36.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	22.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.5	°C/W
ψ_{JT}	Junction-to-top characterization parameter	7.1	°C/W
ψ_{JB}	Junction-to-board characterization parameter	16.3	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.
(2) The package thermal impedance is calculated in accordance with JESD 51-7.

6.5 Electrical Characteristics

over operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = –1 A		V_{CC2} – 1.8	V_{CC2} – 1.4		V
			L293D: I_{OH} = – 0.6 A					
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A			1.2	1.8	V
			L293D: I_{OL} = 0.6 A					
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = –0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	µA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			–3	–10	µA
		EN				–2	–100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

6.6 Switching Characteristics

over operating free-air temperature range (unless otherwise noted) V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output from A input	L293NE, L293DNE	C_L = 30 pF, See Figure 2		800		ns
		L293DWP, L293N L293DN			750		
t_{PHL}	Propagation delay time, high-to-low-level output from A input	L293NE, L293DNE			400		ns
		L293DWP, L293N L293DN			200		
t_{TLH}	Transition time, low-to-high-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			100		
t_{THL}	Transition time, high-to-low-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			350		

6.7 Typical Characteristics

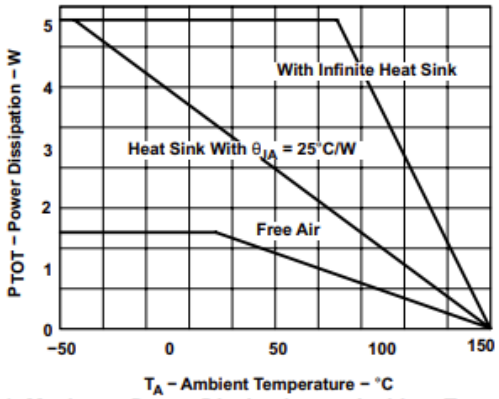
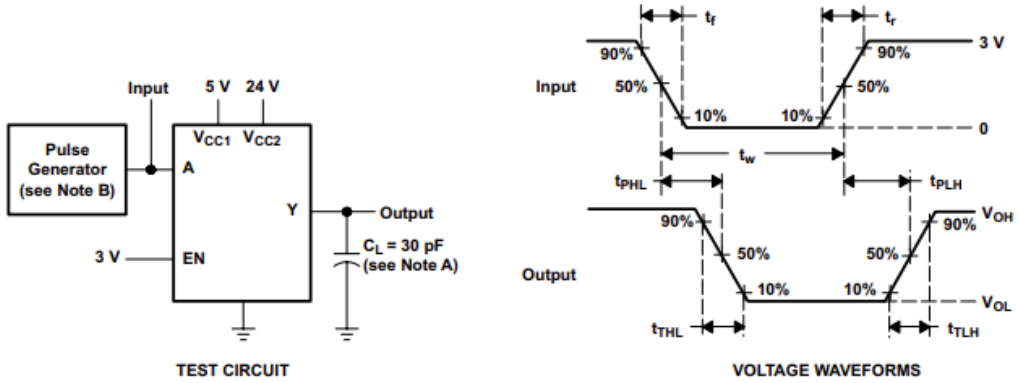


Figure 1. Maximum Power Dissipation vs Ambient Temperature

7 Parameter Measurement Information



NOTES: A. C_L includes probe and jig capacitance.
B. The pulse generator has the following characteristics: $t_r \leq 10$ ns, $t_f \leq 10$ ns, $t_w = 10$ µs, PRR = 5 kHz, $Z_O = 50$ Ω.

Figure 2. Test Circuit and Voltage Waveforms

8 Detailed Description

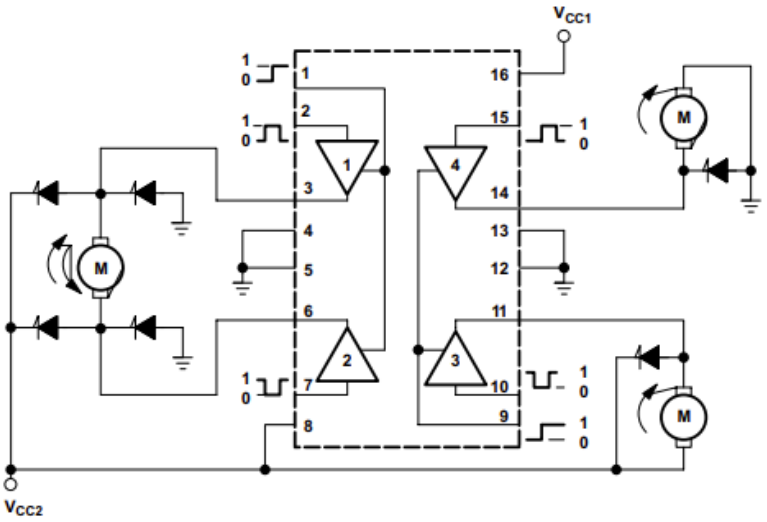
8.1 Overview

The L293 and L293D are quadruple high-current half-H drivers. These devices are designed to drive a wide array of inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current and high-voltage loads. All inputs are TTL compatible and tolerant up to 7 V.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression. On the L293D, these diodes are integrated to reduce system complexity and overall system size. A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation. The L293 and L293D are characterized for operation from 0°C to 70°C.

8.2 Functional Block Diagram



Output diodes are internal in L293D.

8.3 Feature Description

The L293x has TTL-compatible inputs and high voltage outputs for inductive load driving. Current outputs can get up to 2 A using the L293.

8.4 Device Functional Modes

Table 1 lists the fuctional modes of the L293x.

Table 1. Function Table (Each Driver)(1)

INPUTS (2)		OUTPUT (Y)
A	EN	
H	H	H
L	H	L
X	L	Z

(1) H = high level, L = low level, X = irrelevant, Z = high impedance (off)
(2) In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

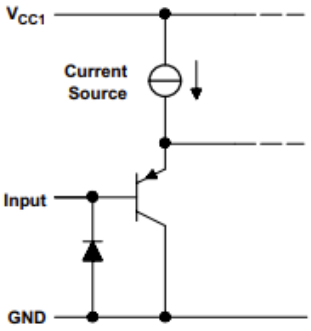


Figure 3. Schematic of Inputs for the L293x

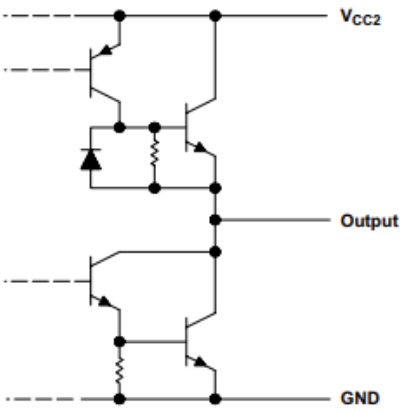


Figure 4. Schematic of Outputs for the L293

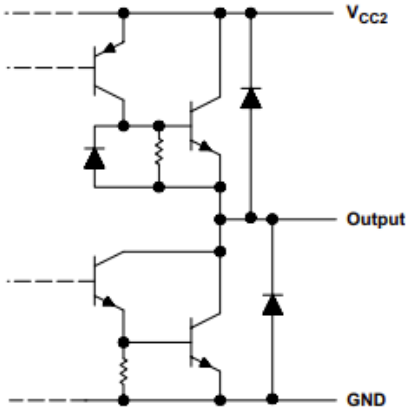


Figure 5. Schematic of Outputs for the L293D

9 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

9.1 Application Information

A typical application for the L293 device is driving a two-phase motor. Below is an example schematic displaying how to properly connect a two-phase motor to the L293 device.

Provide a 5-V supply to V_{CC1} and valid logic input levels to data and enable inputs. V_{CC2} must be connected to a power supply capable of supplying the needed current and voltage demand for the loads connected to the outputs.

9.2 Typical Application

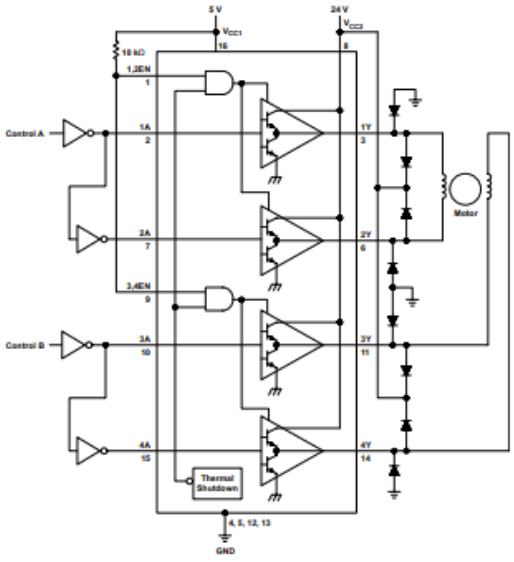


Figure 6. Two-Phase Motor Driver (L293)

9.2.1 Design Requirements

The design techniques in the application above as well as the applications below should fall within the following design requirements.

1. V_{CC1} should fall within the limits described in the *Recommended Operating Conditions*.
2. V_{CC2} should fall within the limits described in the *Recommended Operating Conditions*.
3. The current per channel should not exceed 1 A for the L293 (600mA for the L293D).

9.2.2 Detailed Design Procedure

When designing with the L293 or L293D, careful consideration should be made to ensure the device does not exceed the operating temperature of the device. Proper heatsinking will allow for operation over a larger range of current per channel. Refer to the *Power Supply Recommendations* as well as the *Layout Example*.

Typical Application (continued)

9.2.3 Application Curve

Refer to *Power Supply Recommendations* for additional information with regards to appropriate power dissipation. Figure 7 describes thermal dissipation based on Figure 14.

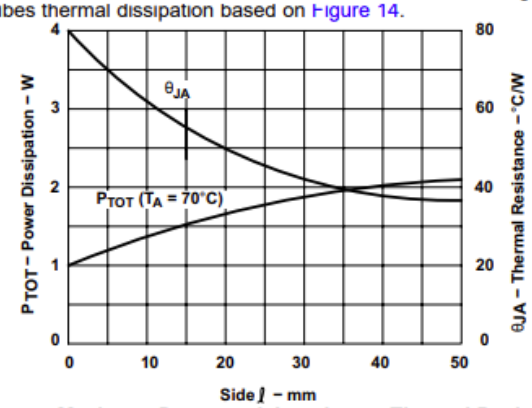


Figure 7. Maximum Power and Junction vs Thermal Resistance

9.3 System Examples

9.3.1 L293D as a Two-Phase Motor Driver

Figure 8 below depicts a typical setup for using the L293D as a two-phase motor driver. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.

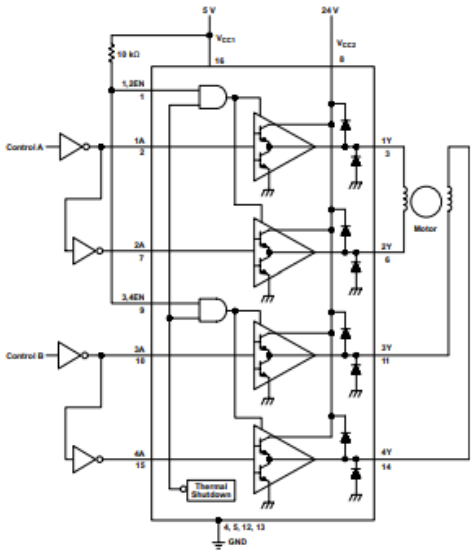
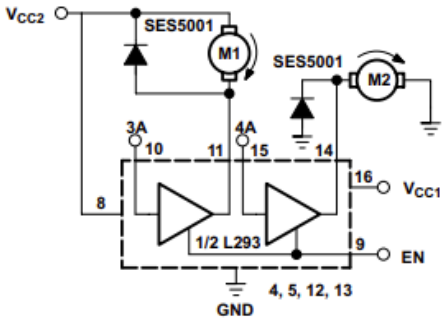


Figure 8. Two-Phase Motor Driver (L293D)

System Examples (continued)

9.3.2 DC Motor Controls

Figure 9 and Figure 10 below depict a typical setup for using the L293 device as a controller for DC motors. Note that the L293 device can be used as a simple driver for a motor to turn on and off in one direction, and can also be used to drive a motor in both directions. Refer to the function tables below to understand unidirectional vs bidirectional motor control. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.



Connections to ground and to supply voltage

Figure 9. DC Motor Controls

Table 2. Unidirectional DC Motor Control

EN	3A	M1 (1)	4A	M2
H	H	Fast motor stop	H	Run
H	L	run	L	Fast motor stop
L	X	Free-running motor stop	X	Free-running motor stop

(1) L = low, H = high, X = don't care

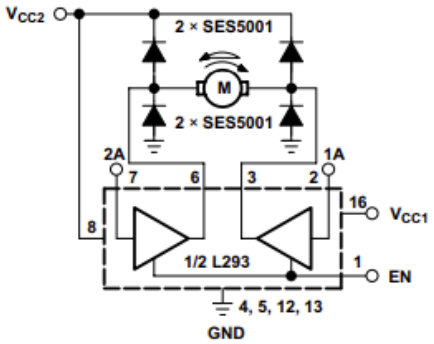


Figure 10. Bidirectional DC Motor Control

Table 3. Bidirectional DC Motor Control

EN	1A	2A	FUNCTION (1)
H	L	H	Turn right
H	H	L	Turn left

Table 3. Bidirectional DC Motor Control (continued)

EN	1A	2A	FUNCTION(1)
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Free-running motor stop

9.3.3 Bipolar Stepping-Motor Control

Figure 11 below depicts a typical setup for using the L293D as a two-phase motor driver. Refer to the *Recommended Operating Conditions* when considering the appropriate input high and input low voltage levels to enable each channel of the device.

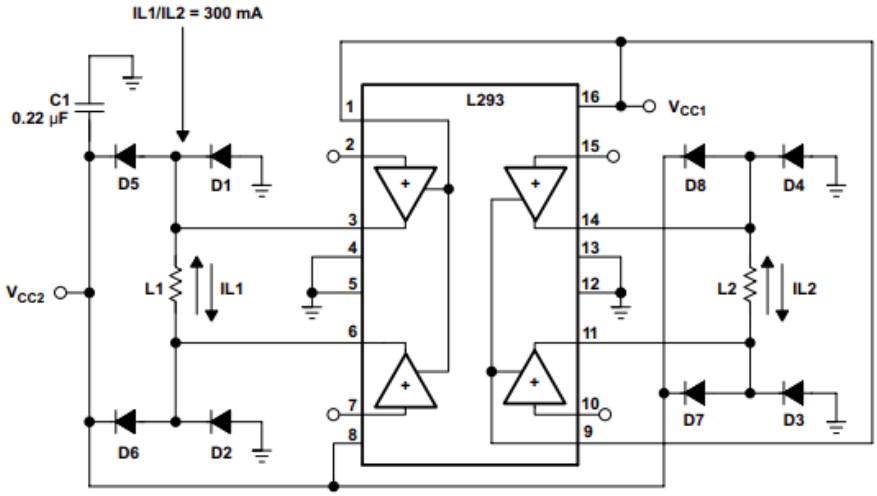


Figure 11. Bipolar Stepping-Motor Control

Datasheet for MCP 3007

MICROCHIP

MCP3004/3008

2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface

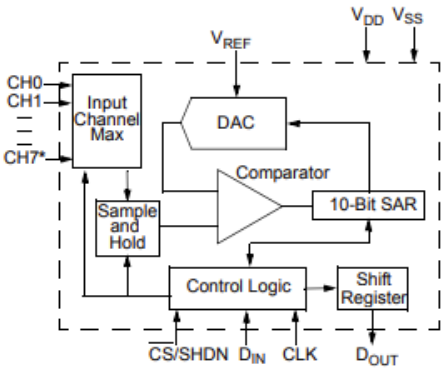
Features

- 10-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL
- 4 (MCP3004) or 8 (MCP3008) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 200 ksps max. sampling rate at V_{DD} = 5V
- 75 ksps max. sampling rate at V_{DD} = 2.7V
- Low power CMOS technology
- 5 nA typical standby current, 2 µA max.
- 500 µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Functional Block Diagram



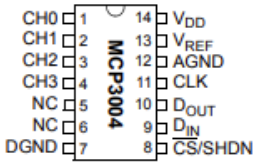
* **Note:** Channels 4-7 are available on MCP3008 Only

Description

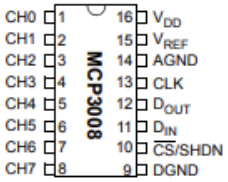
The Microchip Technology Inc. MCP3004/3008 devices are successive approximation 10-bit Analog-to-Digital (A/D) converters with on-board sample and hold circuitry. The MCP3004 is programmable to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3008 is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. Differential Nonlinearity (DNL) and Integral Nonlinearity (INL) are specified at ±1 LSB. Communication with the devices is accomplished using a simple serial interface compatible with the SPI protocol. The devices are capable of conversion rates of up to 200 ksps. The MCP3004/3008 devices operate over a broad voltage range (2.7V - 5.5V). Low-current design permits operation with typical standby currents of only 5 nA and typical active currents of 320 µA. The MCP3004 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages, while the MCP3008 is offered in 16-pin PDIP and SOIC packages.

Package Types

PDIP, SOIC, TSSOP



PDIP, SOIC



1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD} 7.0V
All Inputs and Outputs w.r.t. V_{SS} .....– 0.6V to V_{DD} + 0.6V
Storage Temperature .....–65°C to +150°C
Ambient temperature with power applied.....–65°C to +150°C
Soldering temperature of leads (10 seconds) .....+300°C
ESD Protection On All Pins (HBM).....≥ 4 kV

† Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 200 ksps and f_{CLK} = 18*f_{SAMPLE}. Unless otherwise noted, typical values apply for V_{DD} = 5V, T_A = +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Conversion Rate						
Conversion Time	t_{CONV}	—	—	10	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}	—	—	200 75	ksps ksps	V_{DD} = V_{REF} = 5V V_{DD} = V_{REF} = 2.7V
DC Accuracy						
Resolution			10		bits	
Integral Nonlinearity	INL	—	±0.5	±1	LSB	
Differential Nonlinearity	DNL	—	±0.25	±1	LSB	No missing codes over temperature
Offset Error		—	—	±1.5	LSB	
Gain Error		—	—	±1.0	LSB	
Dynamic Performance						
Total Harmonic Distortion		—	-76		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Signal-to-Noise and Distortion (SINAD)		—	61		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Spurious Free Dynamic Range		—	78		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Reference Input						
Voltage Range		0.25	—	V_{DD}	V	Note 2
Current Drain		—	100 0.001	150 3	µA µA	 $\overline{CS}$ = V_{DD} = 5V
Analog Inputs						
Input Voltage Range for CH0 or CH1 in Single-Ended Mode		V_{SS}	—	V_{REF}	V	
Input Voltage Range for IN+ in pseudo-differential mode		IN-	—	V_{REF}+IN-		
Input Voltage Range for IN- in pseudo-differential mode		V_{SS}-100	—	V_{SS}+100	mV	

Note 1: This parameter is established by characterization and not 100% tested.
2: See graphs that relate linearity performance to V_{REF} levels.
3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for more information.

ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 200 ksps and f_{CLK} = 18*f_{SAMPLE}. Unless otherwise noted, typical values apply for V_{DD} = 5V, T_A = +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Leakage Current		—	0.001	±1	µA	
Switch Resistance		—	1000	—	Ω	See Figure 4-1
Sample Capacitor		—	20	—	pF	See Figure 4-1
Digital Input/Output						
Data Coding Format		Straight Binary				
High Level Input Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low Level Input Voltage	V_{IL}		—	0.3 V_{DD}	V	
High Level Output Voltage	V_{OH}	4.1	—	—	V	I_{OH} = -1 mA, V_{DD} = 4.5V
Low Level Output Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 1 mA, V_{DD} = 4.5V
Input Leakage Current	I_{LI}	-10	—	10	µA	V_{IN} = V_{SS} or V_{DD}
Output Leakage Current	I_{LO}	-10	—	10	µA	V_{OUT} = V_{SS} or V_{DD}
Pin Capacitance (All Inputs/Outputs)	C_{IN}, C_{OUT}	—	—	10	pF	V_{DD} = 5.0V (**Note 1**) T_A = 25°C, f = 1 MHz
Timing Parameters						
Clock Frequency	f_{CLK}	—	—	3.6 1.35	MHz MHz	V_{DD} = 5V (**Note 3**) V_{DD} = 2.7V (**Note 3**)
Clock High Time	t_{HI}	125	—	—	ns	
Clock Low Time	t_{LO}	125	—	—	ns	
$\overline{CS}$ Fall To First Rising CLK Edge	t_{SUCS}	100	—	—	ns	
$\overline{CS}$ Fall To Falling CLK Edge	t_{CSD}	—	—	0	ns	
Data Input Setup Time	t_{SU}	50	—	—	ns	
Data Input Hold Time	t_{HD}	50	—	—	ns	
CLK Fall To Output Data Valid	t_{DO}	—	—	125 200	ns ns	V_{DD} = 5V, See Figure 1-2 V_{DD} = 2.7V, See Figure 1-2
CLK Fall To Output Enable	t_{EN}	—	—	125 200	ns ns	V_{DD} = 5V, See Figure 1-2 V_{DD} = 2.7V, See Figure 1-2
$\overline{CS}$ Rise To Output Disable	t_{DIS}	—	—	100	ns	See Test Circuits, Figure 1-2
$\overline{CS}$ Disable Time	t_{CSH}	270	—	—	ns	
D_{OUT} Rise Time	t_R	—	—	100	ns	See Test Circuits, Figure 1-2 (**Note 1**)
D_{OUT} Fall Time	t_F	—	—	100	ns	See Test Circuits, Figure 1-2 (**Note 1**)

Note 1: This parameter is established by characterization and not 100% tested.
2: See graphs that relate linearity performance to V_{REF} levels.
3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for more information.

ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 200 ksps and f_{CLK} = 18*f_{SAMPLE}. Unless otherwise noted, typical values apply for V_{DD} = 5V, T_A = +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	425 225	550	µA	V_{DD} = V_{REF} = 5V, D_{OUT} unloaded V_{DD} = V_{REF} = 2.7V, D_{OUT} unloaded
Standby Current	I_{DDS}	—	0.005	2	µA	$\overline{CS}$ = V_{DD} = 5.0V

Note 1: This parameter is established by characterization and not 100% tested.
2: See graphs that relate linearity performance to V_{REF} levels.
3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for more information.

3.0 PIN DESCRIPTIONS

The descriptions of the pins are listed in Table 3-1. Additional descriptions of the device pins follows.

TABLE 3-1: PIN FUNCTION TABLE

MCP3004	MCP3008	Symbol	Description
PDIP, SOIC, TSSOP	PDIP, SOIC		
1	1	CH0	Analog Input
2	2	CH1	Analog Input
3	3	CH2	Analog Input
4	4	CH3	Analog Input
–	5	CH4	Analog Input
–	6	CH5	Analog Input
–	7	CH6	Analog Input
–	8	CH7	Analog Input
7	9	DGND	Digital Ground
8	10	$\overline{CS}$/SHDN	Chip Select/Shutdown Input
9	11	D_{IN}	Serial Data In
10	12	D_{OUT}	Serial Data Out
11	13	CLK	Serial Clock
12	14	AGND	Analog Ground
13	15	V_{REF}	Reference Voltage Input
14	16	V_{DD}	+2.7V to 5.5V Power Supply
5,6	–	NC	No Connection

3.1 Digital Ground (DGND)

Digital ground connection to internal digital circuitry.

3.2 Analog Ground (AGND)

Analog ground connection to internal analog circuitry.

3.3 Analog inputs (CH0 - CH7)

Analog inputs for channels 0 - 7, respectively, for the multiplexed inputs. Each pair of channels can be programmed to be used as two independent channels in single-ended mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN. See **Section 4.1 "Analog Inputs"**, "Analog Inputs", and **Section 5.0 "Serial Communication"**, "Serial Communication", for information on programming the channel configuration.

3.4 Serial Clock (CLK)

The SPI clock pin is used to initiate a conversion and clock out each bit of the conversion as it takes place. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for constraints on clock speed.

3.5 Serial Data Input (D_{IN})

The SPI port serial data input pin is used to load channel configuration data into the device.

3.6 Serial Data Output (D_{OUT})

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

3.7 Chip Select/Shutdown ($\overline{CS}$/SHDN)

The $\overline{CS}$/SHDN pin is used to initiate communication with the device when pulled low. When pulled high, it will end a conversion and put the device in low-power standby. The $\overline{CS}$/SHDN pin must be pulled high between conversions.

4.0 DEVICE OPERATION

The MCP3004/3008 A/D converters employ a conventional SAR architecture. With this architecture, a sample is acquired on an internal sample/hold capacitor for 1.5 clock cycles starting on the first rising edge of the serial clock once $\overline{CS}$ has been pulled low. Following this sample time, the device uses the collected charge on the internal sample and hold capacitor to produce a serial 10-bit digital output code. Conversion rates of 100 ksps are possible on the MCP3004/3008. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for information on minimum clock rates. Communication with the device is accomplished using a 4-wire SPI-compatible interface.

4.1 Analog Inputs

The MCP3004/3008 devices offer the choice of using the analog input channels configured as single-ended inputs or pseudo-differential pairs. The MCP3004 can be configured to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3008 can be configured to provide four pseudo-differential input pairs or eight single-ended inputs. Configuration is done as part of the serial command before each conversion begins. When used in the pseudo-differential mode, each channel pair (i.e., CH0 and CH1, CH2 and CH3 etc.) are programmed as the IN+ and IN- inputs as part of the command string transmitted to the device. The IN+ input can range from IN- to (V_{REF} + IN-). The IN- input is limited to ±100 mV from the V_{SS} rail. The IN- input can be used to cancel small signal common-mode noise, which is present on both the IN+ and IN- inputs.

When operating in the pseudo-differential mode, if the voltage level of IN+ is equal to or less than IN-, the resultant code will be `000h`. If the voltage at IN+ is equal to or greater than {[V_{REF} + (IN-)] - 1 LSB}, then the output code will be `3FFh`. If the voltage level at IN- is more than 1 LSB below V_{SS}, the voltage level at the IN+ input will have to go below V_{SS} to see the `000h` output code. Conversely, if IN- is more than 1 LSB above V_{SS}, the `3FFh` code will not be seen unless the IN+ input level goes above V_{REF} level.

For the A/D converter to meet specification, the charge holding capacitor (C_{SAMPLE}) must be given enough time to acquire a 10-bit accurate voltage level during the 1.5 clock cycle sampling period. The analog input model is shown in Figure 4-1.

This diagram illustrates that the source impedance (R_S) adds to the internal sampling switch (R_{SS}) impedance, directly affecting the time that is required to charge the capacitor (C_{SAMPLE}). Consequently, larger source impedances increase the offset, gain and integral linearity errors of the conversion (see Figure 4-2).

4.2 Reference Input

For each device in the family, the reference input (V_{REF}) determines the analog input voltage range. As the reference input is reduced, the LSB size is reduced accordingly.

EQUATION 4-1: LSB SIZE CALCULATION

$$LSB\ Size = \frac{V_{REF}}{1024}$$

The theoretical digital output code produced by the A/D converter is a function of the analog input signal and the reference input, as shown below.

EQUATION 4-2: DIGITAL OUTPUT CODE CALCULATION

$$Digital\ Output\ Code = \frac{1024 \times V_{IN}}{V_{REF}}$$

Where:

V_{IN} = analog input voltage

V_{REF} = analog input voltage

When using an external voltage reference device, the system designer should always refer to the manufacturer's recommendations for circuit layout. Any instability in the operation of the reference device will have a direct effect on the operation of the A/D converter.

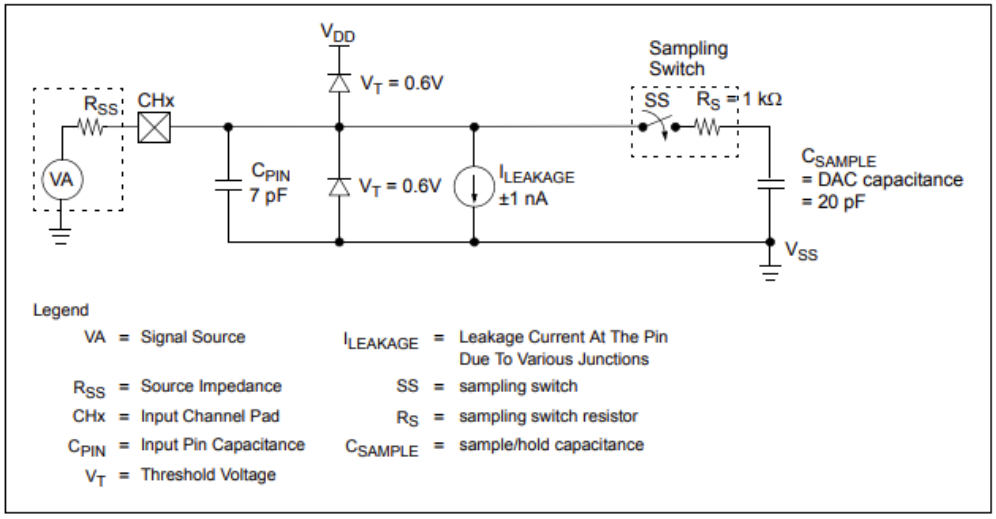


FIGURE 4-1: *Analog Input Model.*

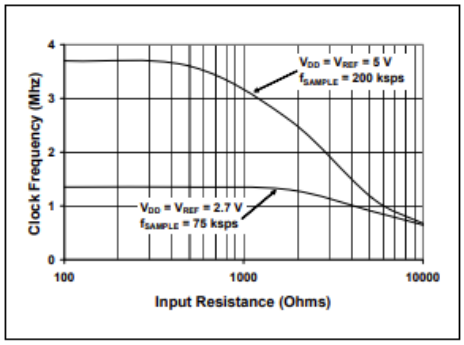


FIGURE 4-2: *Maximum Clock Frequency vs. Input resistance (R_S) to maintain less than a 0.1 LSB deviation in INL from nominal conditions.*

5.0 SERIAL COMMUNICATION

Communication with the MCP3004/3008 devices is accomplished using a standard SPI-compatible serial interface. Initiating communication with either device is done by bringing the $\overline{CS}$ line low (see Figure 5-1). If the device was powered up with the $\overline{CS}$ pin low, it must be brought high and back low to initiate communication. The first clock received with $\overline{CS}$ low and D_{IN} high will constitute a start bit. The SGL/DIFF bit follows the start bit and will determine if the conversion will be done using single-ended or differential input mode. The next three bits (D0, D1 and D2) are used to select the input channel configuration. Table 5-1 and Table 5-2 show the configuration bits for the MCP3004 and MCP3008, respectively. The device will begin to sample the analog input on the fourth rising edge of the clock after the start bit has been received. The sample period will end on the falling edge of the fifth clock following the start bit.

Once the D0 bit is input, one more clock is required to complete the sample and hold period (D_{IN} is a "don't care" for this clock). On the falling edge of the next clock, the device will output a low null bit. The next 10 clocks will output the result of the conversion with MSB first, as shown in Figure 5-1. Data is always output from the device on the falling edge of the clock. If all 10 data bits have been transmitted and the device continues to receive clocks while the $\overline{CS}$ is held low, the device will output the conversion result LSB first, as is shown in Figure 5-2. If more clocks are provided to the device while $\overline{CS}$ is still low (after the LSB first data has been transmitted), the device will clock out zeros indefinitely.

If necessary, it is possible to bring $\overline{CS}$ low and clock in leading zeros on the D_{IN} line before the start bit. This is often done when dealing with microcontroller-based SPI ports that must send 8 bits at a time. Refer to **Section 6.1 "Using the MCP3004/3008 with Micro-controller (MCU) SPI Ports"**, "Using the MCP3004/3008 with Microcontroller (MCU) SPI Ports", for more details on using the MCP3004/3008 devices with hardware SPI ports.

TABLE 5-1: CONFIGURE BITS FOR THE MCP3004

Control Bit Selections				Input Configuration	Channel Selection
Single/ Diff	D2*	D1	D0		
1	X	0	0	single-ended	CH0
1	X	0	1	single-ended	CH1
1	X	1	0	single-ended	CH2
1	X	1	1	single-ended	CH3
0	X	0	0	differential	CH0 = IN+ CH1 = IN-
0	X	0	1	differential	CH0 = IN- CH1 = IN+
0	X	1	0	differential	CH2 = IN+ CH3 = IN-
0	X	1	1	differential	CH2 = IN- CH3 = IN+

* D2 is "don't care" for MCP3004

TABLE 5-2: CONFIGURE BITS FOR THE MCP3008

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+