

**RMIT UNIVERSITY VIETNAM  
SCHOOL OF SCIENCE AND TECHNOLOGY**



**EEET2473: ENGINEERING DESIGN 3A  
ASSIGNMENT 3  
FINAL DESIGN PROJECT REPORT AND DEMONSTRATION**

**A Semi-Autonomous Cargo Robot**

**Instructed by:** Dr. Minh Tran Quang

**Date of submission:** 13.01.2020

**Team No:** C

Student name	ID number
Nguyen Huu Tho	s3712942
Tran Do Chi Hai	s3653997
Weeramundage Leshan Fernando	s3765778
Alexander Nuwan Amila	s3766009
Vang Huynh Minh Tri	s3726101
Mohamed Ibrahim Atheef Mohammed	s3789831

## 1 ABSTRACT – EXECUTIVE SUMMARY

With this final report for the project 3A, team C provides the current update of the project work procedures including hardware and software developments regarding task 1, task 2, task 3 and task 4 which were required to be completed by the course delivery period and also the results obtained after the completion of the respective tasks, the final design, overall strengths and weaknesses of the performance, list of results for each task, group evaluations and future works of the project regarding task 3 and task 4 including object detection and colour detection of the program, Engineering design 3A. task 1 and task 2 mainly corresponds with the manual handling and the path following operation of the robot and task 3 and 4 corresponds to the obstacle detection and color detection respectively.

In accordance with the requirements of the project descriptions and materials, teams' schedules, methods, acknowledgements and recommendations for the next level of the project procedures, have been attached at the second portion of this report as well.

Mainly, this documentation includes the comparisons between the previous models of the autonomous cargo robot which our team has presented in the initial project proposal and the problems, midterm evaluation stages, strengths and weaknesses and what were the reason behind in switching into a new version of the robot other than the proposed version of the prototype respectively for each task.

In an agreement among the team, the goal was to understand the system for task 1 and task 2 and let the problems in hardware and software which was to occur, avoid imagining about the solutions and start working upon the solutions instead without doing any harm to the overall system or to any personnel in the team.

In the latter sections underneath the main contents of the report, all the supporting documents and meeting minutes documents of the team have been attached along with the appendices. All the changes which were made throughout the completion of task 1 and task 2 compared to the project proposal of this project have been recorded in this report and compared with each other as mentioned before. Additionally, supporting videos from the demonstrations at the midterm evaluation have also been attached in the attachment section as well.

Final thoughts for the completion of the required tasks and the future developments which could be performed to this robot prototype that we have presented in the final demonstrations have been added by the team members at the very last part of the report and the comments must be validated by the instructor referring to our meeting minutes and all the reference documents which our team has share with the instructor via a shared folder., Gantt chart and the overall progress of the project design throughout the course until the final demonstration of the project to was referred to assess the work of the team in accordance with the assignment marking criteria and rubrics.

To achieve the targets explained above, we, group C, have cooperated with our team members to apply knowledge from wide range of resource persons from electrical, robotic and software engineering sectors. The short-term targets are using components to build the prototype, creating a mobile controller application and applying Arduino and Raspberry Pi to program to achieve all 4 required functions. The long-term objectives are upgrading the robot to work more effective, learn how to cooperate with others and gain knowledge in different engineering areas to get ready to our future career.

## 2 LITERATURE REVIEW

At this stage of the project, all the tasks have been completed which are required to be completed by the end of the course program, Engineering project 3A. as the midterm report was completed three weeks back, up to this day, task 1 and task 2 was completed as a milestone at the end of the course delivery period. Task 1 and task 2 specifically were based on RC control of the prototype and the path following operations. to implement, test and finalize those tasks initial research on certain approaches which are already available in the market and industry was performed, and those researches were attached in the literature review section of the Midterm report of team C. With reference to that, task 1 and task 2 were performed. Those approaches and their functionalities along with the results obtained from them could be analyzed and observed through the content of the midterm report and the project proposal which have been attached in a linked form to the containing folder on Google drive in Attachment A. Also, the folder containing all the documentation and support files for task 1, task 2 completion and video demonstration files with hardware implementation solid work files and codes have been attached in the shareable folder on Google drive which is also included in Attachment B.

Additional research has been carried out to investigate extra knowledge and approaches to implement the required functions on the robot prototype which are regarding task 3 and task 4 which differs from the initial literature review and research which was attached in the midterm report as well.

The task 3 requires the robot detects an obstacle in front of it and stops in a safe distance. There are many devices from basic to advance that can connect to Arduino or Raspberry Pi to detect an object. One of the basic common devices is HC-SR04 Ultrasonic Sensor, suggested by Abhiemanyu Pandit in Circuit Digest website [1]. The principle of this device is transmitting the ultrasonic beams and capturing it back if it hits the surface of the object in front of it. The distance result between the sensor and the object is calculated based on the time it takes to reflect the ultrasonic waves. This result can be sent to the Arduino to change robot direction or stop when it reaches the safe distance. This HS-SR04 is also suggested by Solanke Jyoti Tanaji but it will be connected to

Raspberry PI and coded in Python [2]. The distance is calculated by the formula:  $\text{distance} = ((\text{Stop time} - \text{Start time}) * 34300)/2$ . It is divided by 2 because it takes time for both sending and receiving routes.

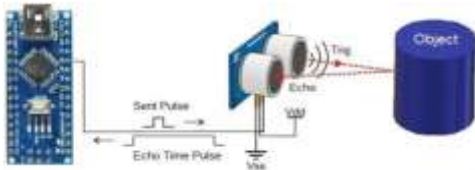


Table 1: Ultrasonic sensor [1]

Another device that can detect the obstacle is Infrared sensors. Aswirth Raj in Circuit Digest website used this device to detect objects by sending and receiving IR ray [3]. However, based on the research of Liz Miller, the robotic engineer and a creator of Learn Robotics website, said that the Ultrasonic sensors provide more reliable and accurate data. The IR sensor is often used to detect the presence of the objects rather than the distance. [4]



Table 2: Infrared Sensor [3]



Table 3: Deep Learning object detection [5]

The advance method to detect the obstacle is using (DL technique) Deep Learning and OpenCV. Ph.D Adrian Rosebrock applied this method by using Camera and Raspberry Pi to look for objects in each frame. This method can detect the kind of object with a confidence level, draw a bounding box around it and associate it with a label [5]. After discussions in our group, we decide to use the Camera and Raspberry Pi to detect color of the cubic and garage position, so we will use the Ultrasonic sensor and Arduino to detect the obstacle.

An approach to detect the colors using Raspberry Pie was carried out by Anahgha B. Kulkarni, Pranjali S.Jaisingpure, Dr. Lenina SVB from Department of Electronics and Telecommunication Engineering, Osmania

University Campus, India in 15<sup>th</sup> October 2017 to implement a automated sorting system using raspberry pi 2(Raspbian operating system) and a USB camera for color detection of objects. OpenCV (Open Source Computer Vision) to implement color detection algorithm. They have analyzed the color detection outputs using RGB and HSV color models [6]. There are three major types of color models in the use, namely,

- RGB color model (Red, Green, Blue)
- HSV color model (Hue – Dominant wavelength, Saturation – purity/shades of colors and Value – Intensity of colors)
- CMYK color model

Specifically, in OpenCV, color ranges from [0,0,0] to [179,255,255] and with trial and error methods, they have been able to calibrate the lower level lower level and the upper ranges of the hue for maximum possible colors and have created a mask for the image to extract the desired hue ranges.

Below are two resultant hue ranges that they have obtained after applying the mask to detect the desired hue range, respectively for colors, red and orange.

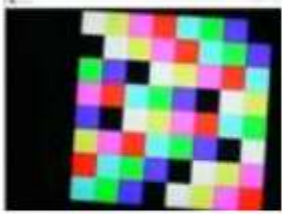



Hue Ranges	Original image	Resultant image
Red 0 -10		
Orange 10-20		

Table 4: Hue ranges obtained after applying the mask to obtain the desired hue range

Above mentioned research has been carried out addition to the prior mentioned research in the mid term report as extra knowledge about the methods and functions to implement color detection and obstacle detection procedures in to the robot prototype. Additionally, a research have been attached finally after the research from Dr. Ph.D Adrian Rosebrock, that our team has referred to greatly with many of the approaches which are discussed by the author of the documentation, files and videos, have also attached from the initial research as well, as the approaches for the color detection procedures to be implemented in the robot have been extracted from the above mentioned resource, mainly the color range and calibration of the default program functions of raspberry pi. with reference to this, and the initial research which are included in the midterm report and for the project proposal which we attached together in the midterm report, several key approaches were figured out and those approaches regarding the set forth topics have been applied to our software solutions and hardware solutions as well. Following sections will reflect the entire work of the team starting from the scratch until the final demonstrations and discussing about the major differences specially between the initial mechanical design and the final mechanical design and how the software approach aligns with the changed functions and implementations of the robot. At the end of this document, several attachments which are required to support the content of this document are attached as attachments and appendix as well. And the list of references is also called in the references section of this document for the corresponding citations as well.

### **3 KEYWORDS**

List of all keywords and acronyms that are used in this document.

- RGB – Red, Green and Blue
- HSV – Hue, Saturation and Value
- CMYK – Cyan Magenta Yellow and Key (Black)
- Mm – Millimetres
- IR – Infrared
- US – Ultrasonic
- Kg – Kilogram
- Cm – Centimetres
- OpenCV – Open Source Computer Vision
- V - Volts
- A – Amperes
- VDC – Direct current Volts
- RHSV – Right Hand Side View
- LHSV – Left Hand side view
- TV – Top view
- FV – Front view
- Mm -Millimeters
- DL – Deep Learning
- IMU – Internal Measurement Unit
- Gyro - Gyroscope

## TABLE OF CONTENTS

<b>1</b>	<b>ABSTRACT – EXECUTIVE SUMMARY</b>	<b>2</b>
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>3</b>
<b>3</b>	<b>KEYWORDS</b>	<b>5</b>
<b>4</b>	<b>INTRODUCTION</b>	<b>10</b>
<b>5</b>	<b>PROJECT DESCRIPTION AND DESIG SPECIFICATIONS</b>	<b>11</b>
<b>6</b>	<b>FINAL HARDWARE DESIGN OF THE ROBOT PROTOTYPE</b>	<b>12</b>
6.1	Final design model of the robot prototype	12
6.2	Final design of the robot prototype using real physical components and modules	14
<b>7</b>	<b>VERSION BY VERSION COMPARISON OF THE MECHANICAL DESIGN STRUCTURE</b>	<b>14</b>
7.1	Version 1: initial basement design and the final basement design made to place the components	15
7.2	Version 2: Setup for the IR sensor array initially and switching to a new approach	15
7.3	Version 3: Overall placing and of the components of the first stage on the final basement design	16
7.4	Version 4: Comparison between the initial second stage and the final design second stage	16
7.5	Version 5: design of a camera holder to place the raspberry pi camera on top of the second stage	17
7.6	Version 6: initial design of the robot without a cover and the final design with a cover.	17
<b>8</b>	<b>BACKGROUND INFORMATION</b>	<b>18</b>
8.1	Task 3 - Obstacle Detecting	18
8.2	Color Detection	19
8.3	Visual Properties of Images in Computational Workspaces	20
8.4	Understanding how Colors are Represented in Computation	20
8.5	Different Types of Color Spaces	20
<b>9</b>	<b>PROCEDURES AND METHODS APPLIED TO DATE</b>	<b>21</b>
9.1	Task 3	21
9.2	Introduction to Computer Vision	26
9.3	Approach to Task 4	29
9.4	Major Code Implementations	31
<b>10</b>	<b>FUTURE WORKS</b>	<b>43</b>
<b>11</b>	<b>RISK ANALYSIS</b>	<b>45</b>
<b>12</b>	<b>COMMUNICATIONS</b>	<b>46</b>
<b>13</b>	<b>POLICIES AND EXPECTATIONS OF THE TEAM</b>	<b>46</b>
<b>14</b>	<b>GANTT CHART FOR THE OVERALL PROJECT PROCEDURES</b>	<b>47</b>
<b>15</b>	<b>DISCUSSION AND OBTAINED RESULTS</b>	<b>47</b>
<b>16</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>47</b>
<b>17</b>	<b>MEMBER CONTRIBUTION</b>	<b>48</b>
<b>18</b>	<b>ACKNOWLEDGEMENTS</b>	<b>50</b>
<b>19</b>	<b>REFERENCES</b>	<b>51</b>

<b>20</b>	<b>APPENDICES</b>	<b>52</b>
20.1	Appendix A	52
20.2	Appendix B	52
20.3	Appendix C	52
20.4	Appendix D	52
<b>21</b>	<b>ATTACHEMNTS</b>	<b>52</b>
21.1	Attachment A	52
21.2	Attachment B	52
21.3	Attachment C	52
21.4	Attachment D	52



## LIST OF FIGURES

Figure 1:Ultrasonic sensor [1] .....	3
Figure 2:Infared Sensor [3] .....	3
Figure 3:Deep Learning object detection [5].....	3
Figure 4: Hue ranges obtained after applying the mask to obtain the desired hue range .....	4
Figure 5: Final Robot Assembly Outside the Factory Floor .....	12
Figure 6: Final robot prototype design using real physical components and modules .....	14
Figure 7: Transmission of the signals.....	18
Figure 11: Color detection frequency range .....	19
Figure 8:Initial concept.....	21
Figure 9:Picture of actual HC-SR04 on robot .....	22
Figure 12:Arduino Code to implement the above function.....	24
Figure 13:how the complete system runs with the second sub method .....	25
Figure 14: Pie camera used for image processing .....	27
Figure 15: Camera connect to RASP-pie .....	28
Figure 16: Camera mount to the robot prototype .....	28
Figure 17: Preliminary prototype design of mount .....	29
Figure 18:Preveiew of the text file .....	32
Figure 19:COde snippet for the set forth function.....	33
Figure 20:File preview .....	33
Figure 21: Data retrieval code .....	34
Figure 22:Fucntion of the taskbars (Calibration) .....	34
Figure 23: Code to implement the set forth function .....	35



## LIST OF TABLES

Table 1: Design specifications for the robot.....	11
Table 2: Final design structure of the robot in solid works.....	13
Table 3: Basement design comparison .....	15
Table 4: IR sensor array placement between initial and final designs .....	15
Table 5: Overall placing of components on the basement in initial and final designs.....	16
Table 6: Comparison between the initial second stage and the final second stage .....	16
Table 7: Comparison between initial and final camera mount.....	17
Table 8: Comparison between initial and final wiring design.....	17
Table 10: Color perceive to the human brain .....	19
Table 9: Specifications of the task.....	22
Table 11: Ultrasonic sensor used to detect obstacles .....	22
Table 12: Reading the pulse by Arduino via its pins and its functionality.....	23
Table 13: Difficulty variation of the tasks.....	29
Table 14: Color range for three color required.....	35
Table 15: Code implementation .....	36

## 4 INTRODUCTION

Engineering design 3A is a project-based course which is delivered in the third semester of academic year 2019. During this course, every one of our project team is having the chance to work in a group of students from different programs (Electrical and electronics engineering, robotics and mechatronics engineering and software engineering) to find a solution to a practical engineering problem. The focal point of our project is to implement a robot design to transport goods cargo from the production line to specific warehouse depending on the type(colour) of the cargo. This goal should be achieved by completing several milestones along the way of the program outline and it is expected by then our project team would complete the design prototype and ready to showcase it in week 13 of this respective semester.

Im completion of this project, several learning outcomes could be achieved such as, Comprehensive, theory-based understanding of the underpinning natural and physical sciences and the engineering fundamentals applicable to the engineering discipline., Understanding of the scope, principles, norms, accountabilities and bounds of contemporary engineering practice in the specific discipline., Application of systematic engineering synthesis and design processes., Professional use and management of information. These are some of the key learning outcome which could be achieved by completing this course successfully. Majorly, working with a team bring s us a huge experience in many aspects of our lives and to our career as to be a professional engineer. And, on completion of this course, and project works, all the members of our team would be able to Utilise fundamental knowledge and skills in engineering and apply it effectively to a project., Plan and manage your time effectively as an individual and as a team. Design and develop a functional product prototype while working in a team. Manage any disputes and conflicts within and outside of your team., Present your work to peers, academics, general and industry community. With regard to the requirements of the project outline, our team has sharply adjusted the system behaviour of our team to achieve all theses targets and accomplish our goals which are set forth at the proposal stage of our project.

And also, this course and project works will prepare with knowledge and skills in working in a moderate complex project topic with other team members as explained previously as well. Other than that, good study practices have been carried out up to the second milestone of our project, which could be enlisted namely, pay attention in class and participate in class activities., Do exercises, tutorials, laboratories and assignments., No plagiarism at all time. When plagiarism occurs, it will result in ZERO for ALL parties involved and the case will be recorded for further process., Ask questions and engage in class (Specially in group meeting sessions with the instructor at the end of every week)

Dr. Minh Quang Tran who is a lecture in the department of engineering, school of science and technology, RMIT Vietnam obtained his Bachelor of Engineering (Honours) in Aerospace engineering from the University of Technology Vietnam National University and PhD in Maritime Engineering and Hydrodynamics from the Australian Maritime College, College of science and Engineering, University of Tasmania(Australia), holds the position of the lecturer and the course coordinator of this project-based course, and he provides our team, all the resources, knowledge and aids and guides our team to reach the required milestones during the given period and gives us a bid support in managing the time, add professional ethics and qualities to our careers and teaches us how to manage a team successfully and how to troubleshoot the problems arise while running a project and shaping us to face the new world of technology. Our project group consists of six members from thee different sectors of studies such as, Electrical and electronics engineering, Robotics and mechatronics engineering and software engineering. A short description about our team and the background of our team members could be placed as follows,

### **i.Nguyen Huu Tho**

A Mechatronic Engineering student who has experience in hardware design (Solid work, Autodesk) and a bit of software knowledge (Arduino). Passionate about designing robots and making new inventions.

### **ii. Tran Do Chi Hai**

An Electronic Engineering student who has experience in hardware design (build the robot) and a bit of software. Develops software to enable hardware applications to run smoothly. Helps debug programming to ensure the expected outcome is received.

### iii. Weeramundage Leshan Tiranga Fernando

A Mechatronics Engineering student who has experience in designing line following robots, grid solving robots, both hydraulic and autonomous pick and place robots, maze solving robots in my past academic career having the ability to design, and model design prototypes of projects using software such as Solid works, Autodesk inventor, AutoCAD, PTC Creo etc. additionally, consists with a moderate knowledge about Arduino programming and Java and document formatting/writing/editing skills in Microsoft Word.

### iv. Alexander Nuwan Amila

Mechatronics Engineering student who has former experience in designing line sensing robots and manipulating robotic arms. Has additional experience in coding and programming languages such as Python, Java and C. Also presents ability in working with proteus to prototype custom circuits. Proficient in SOLIDWORKS, AutoCAD.

### v. Vang Huynh Minh Tri

Has a bachelor's in finance and Banking and is currently undertaking a Engineering degree. Worked as a banker in 2 years. Passion for creating stuffs. In his opinion, computers are powerful and can help us do lots of things, but we need to create codes to communicate with them. That is why he choose to take a second degree.

### vi. Mohamed Ibrahim Atheef Mohamed

Mechatronics and robotics engineering student .I am good at designing and implementing a robot because I have an adequate experience in making stuffs like maze detecting and line following robot, electrical scooter, robotic arm which are done by me as a part of my degree in previous semesters .In addition to that I am good at AUTOCAD,INVENTOR,CREO like designing packages and simulation and designing soft wares like fritzing and proteus professional

This project, which is basically based on the topic of the smart transportation, focuses on four main operations which acts as prototypical approaches to implement these systems in real life factory environments to adapt to industry 4.0 methodologies and achieve desired goals in an efficient manner. Those specific operations are defined as four main task which are to be completed by the end of the course delivery time which are namely,

1. Task 1: Manually remote control
2. Task 2: Path following
3. Task 3: Obstacle avoidance
4. Task 4: Object recognition

This is the final report that describe all our main versions and the final design for both hardware and software design after we completed all 4 tasks. Based on our working process, the advantage and disadvantage of each options and updates will be documented. The object of this report is updating the process from the mid-term report. It will describe all design for all tasks. The final performance of the product will also describe based on our work on the demonstration day. This report also describes technical and non-technical skills that we learn throughout the project. The schedule and Gantt chart will be mentioned to show how we manage time and tasks for each member. Finally, the work contribution shows how each member contribute to this project

## 5 PROJECT DESCRIPTION AND DESIG SPECIFICATIONS

Major design specifications of the robot design could be tabulated as follows in the table below.

Table 5: Design specification s for the robot

Specification	Measurements
Size and weight	297x210x210 mm in size. Weight <10kg
Power	Self-contained power Must not exceed 24VDC
Materials	Any kind of material with any electronics devices/micro controllers or micro processors
Budget	< 7million VND

Safety	Robot must pose no danger Must have a switch which should be easily accessible No hazardous/explosive chemicals should be used Lasers: class 2 or lower
--------	--

And a description on the tasks which have to be performed on the mid semester demonstration and at the final demonstration as well (task 1, 2,3,4) are as already we have described in the project proposal and the midterm report. Please referee to that which is also attached the link to the folder in attachment A and Attachment D

## 6 FINAL HARDWARE DESIGN OF THE ROBOT PROTOTYPE

Final hardware design of the robot prototype could be presented in two approaches as it could be clariid in the modelling stage and in the designed stage. Modelling stage is referred to the teams' solution for the final design which is performed using solid works modelling and the designed stage is referred to the final physical appearance and the outcome of the robot which was presented in the final demonstrations as well. Obviously, this design is way more different compared to the previous design ideas and implemented robot structures for the robot prototypes specially for the midterm stage of the report. And as well, compared to design proposal pf the robot, the functions, components, and design approaches and requirements have been differed a lot in the final design of the robot prototype. These differences occur due to the requirements and need of performances which were needed along the way of the design until the final design of the prototype. Two images below depict the modeling design of the robot prototype in solid works and the real implementation of the modelled robot design in real physical life with real components and modules.

### 6.1 Final design model of the robot prototype

Final design of the robot prototype could be depicted in the outside factory floor as in the following figure. The design files from solid works for the robot prototype is attached in the Attachment C in the attachments section.

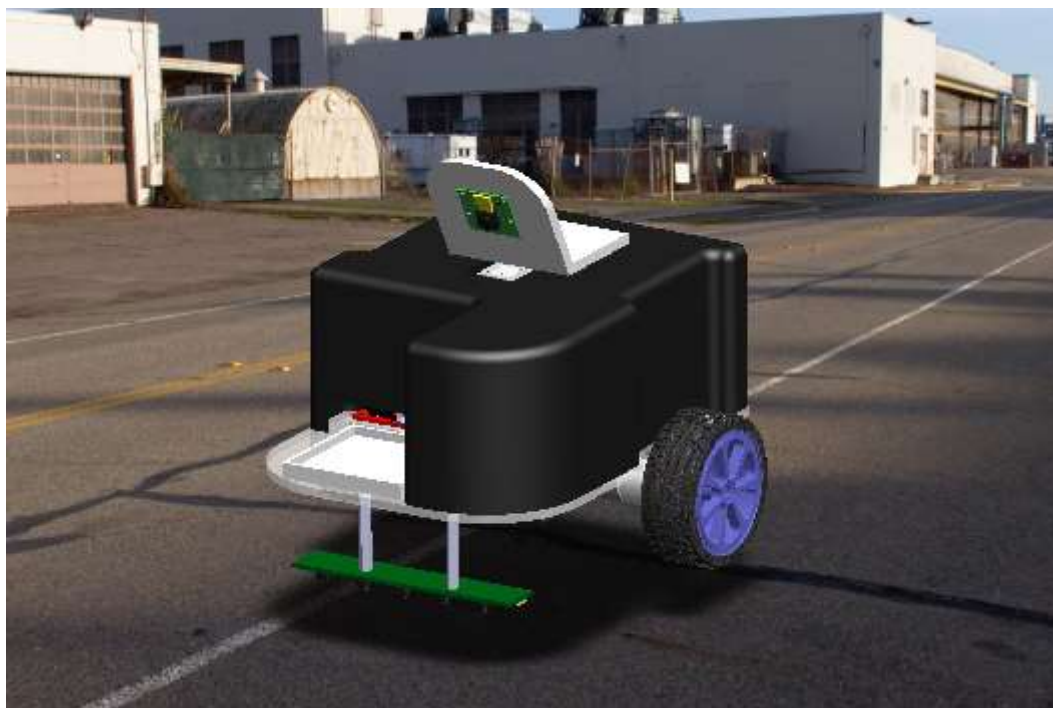
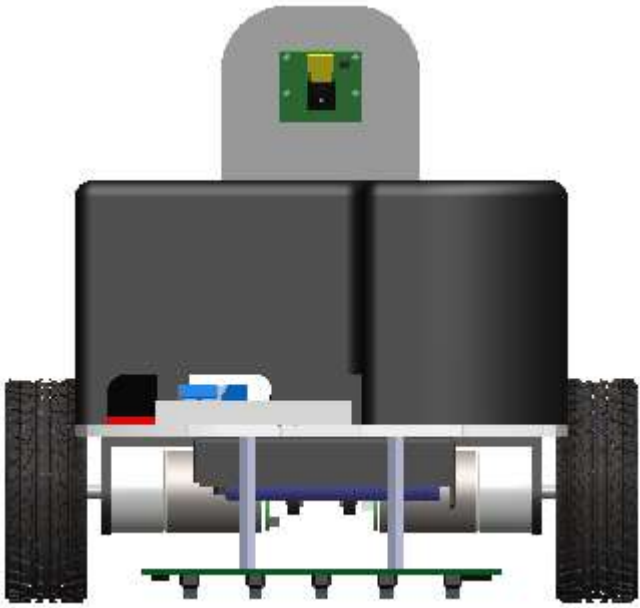
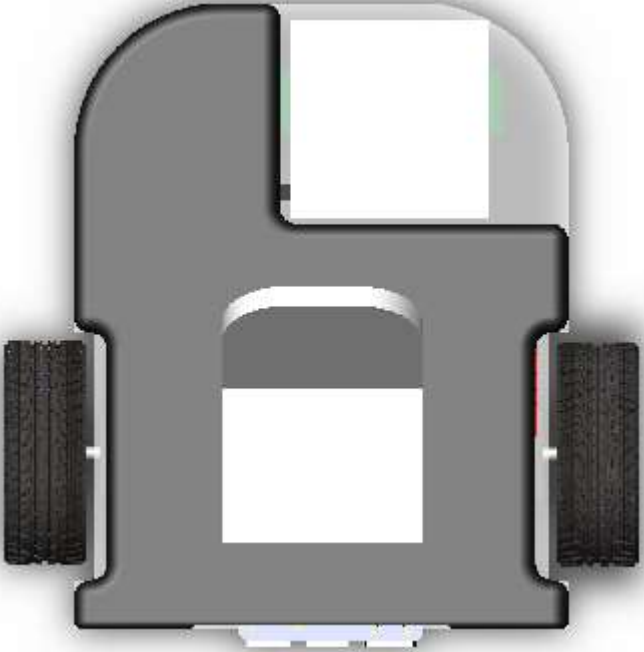

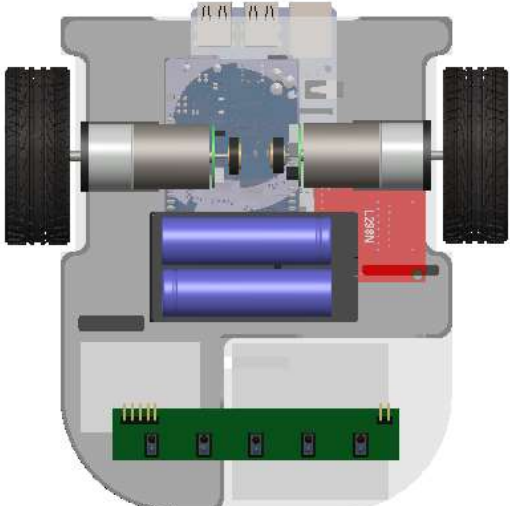


Table 6: Final Robot Assembly Outside the Factory Floor

Table 7: Final design structure of the robot in solid works

Front view (FV)	Top view (TV)
	
Left hand side view (LHSV)	Bottom view (BM)
	



## 6.2 Final design of the robot prototype using real physical components and modules



Table 8: Final robot prototype design using real physical components and modules

## 7 VERSION BY VERSION COMPARISON OF THE MECHANICAL DESIGN STRUCTURE

At the final design we have made changes along the way until the final stage for the robot prototype differing from the project proposal stage to the midterm evaluation stage and all the way to the final stage. These differences were made to make the functionalities of the robot structure more convenient, well-functioning, lower cost estimation, keep the design appearance at a good behavior and overall, aligning to the design specifications and the required guidelines of the project description set forth the proposing stage of the project. As there were many versions for the robot of our team, there occurs a difference in the mechanical design when we compare each stage with each other. Moreover, not only the outlook appearance's, but also the reason behind we omit the initial design idea and moved on to the final design idea and the gained results from the switching between those design stages could be carried out as follows according to each version of the design prototype of our team

## 7.1 Version 1: initial basement design and the final basement design made to place the components

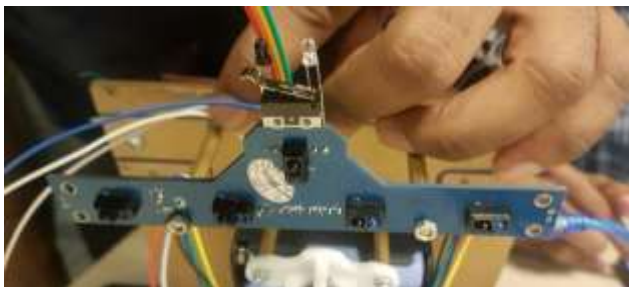

Table 9: Basement design comparison

Initial design	Final design
	

Initial design of the basement was not a modelled one, instead it was manually cut and placed the holes and attachments according to the need after the cutting, manually by using a hand cutter and drill etc. final stage design for the final prototype of the robot was modelled using solidworks by obtaining the measurements from the real life components and attachments. In this stage, our team has designed this basement design to place the required components and arrange them in an organized manner. For this objective to be achieved, several measurements from the components and the distances between them also had to be measured and apply them into the blueprint of the design to avoid collisions of the objects with each other.

## 7.2 Version 2: Setup for the IR sensor array initially and switching to a new approach

Table 10: IR sensor array placement between initial and final designs

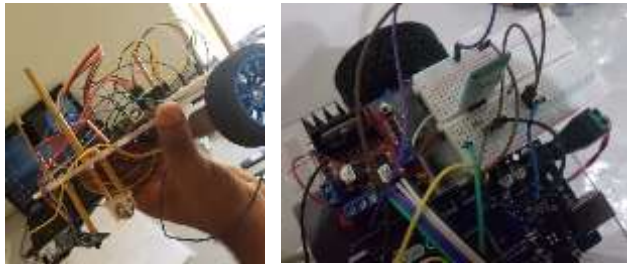
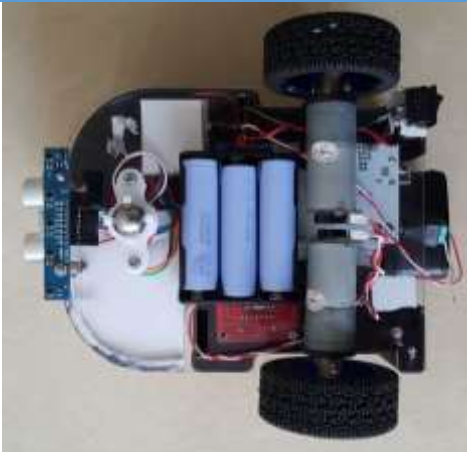
Initial design	Final design
	

Firstly, according to the proposed design of the line following operation of the robot prototype, a 6 channel IR sensor array was used to implement the line following operation. But for the final design of the prototype, it was used a 5 channel IR sensor for path detection and following operation. This was chosen, because of the inaccuracy of the readings that were obtained from the 6 channel IR sensor as the distance between the two middle sensors was quite larger than it was expected, and it lies outside the path which should be followed. So that, an IR sensor, (QTR-5RC) was substituted to the previous 6 channel IR sensor array which uses only 5 channels and comparatively accurate than the 6-channel sensor array.



### 7.3 Version 3: Overall placing and of the components of the first stage on the final basement design

Table 11: Overall placing of components on the basement in initial and final designs

Initial design	Final design
	

To omit the inaccessibility to the components on the basement for testing, and troubleshooting of the system with the aid of the software solution, and get rid of the untidy placing of components without any measurement/improper wiring structure between the components on the basement and of the tangled behavior of the wire connections, simple circuit components (ex: decreased size breadboard, resistors, power jacks etc.) were used to implement the same hardware setup on the basement stage in the final design.

### 7.4 Version 4: Comparison between the initial second stage and the final design second stage

Table 12: Comparison between the initial second stage and the final second stage




Initial design	Final design
	

For the initial design of the second stage, it was decided to use the same basement that we used in the first stage. This was decided after the completion of task 1 and task 2. Because, there was no any second stage for the prototype when the task 1 and task 2 were going on. Although the initial decision was to use the basement for the second stage as well, it was changed at the week before the completion of the robot prototype which was decided to use the cover design to place all the components which were decided to be placed on it. Firstly, it seemed as a quite challenging decision and an impossible one to achieve. But after some sensitive measurements of the components which were needed to be placed on the second stage and of the cover design, our team was able to place and organize all the components on the underlay of the cover design with a tight and precise arrangement. The components which were needed to be attached on to the second stage was the Raspberry Pi module, and the camera to detect the Rubik's cube and the color-coded unloading bays. Therefore, as we were able to omit a second stage for the final design prototype of the robot and ultimately when looking at this as a real-life approach to be

implement, we can save that cost allocation for a second stage and lower the total cost for the robot. Therefore, our team suggests that it was a good decision which has been taken to omit the second stage and surviving with the cover design of the robot and use it to function the process from a second stage.

### 7.5 Version 5: design of a camera holder to place the raspberry pi camera on top of the second stage

Table 13: Comparison between initial and final camera mount

Initial design	Final design
 	

An initial design approach to place the camera holder was to make a box on top of the second stage and to place the camera on one side of it. But, that idea was eliminated due to the need of optimizing the camera angle and its range of vision according to the preference and for the convenience of detecting the rubric cube and the surrounding at a clearer view. So that, we could state that, with the initial design approach for the camera holder, the required task could not be done successfully and instead, with the final design for a camera holder, the specified task and requirement to obtain a considerable good range for the camera view could be obtained.

### 7.6 Version 6: initial design of the robot without a cover and the final design with a cover.

Table 14: Comparison between initial and final wiring design

Initial design	Final design
	

First stage design was without a cover for the robot and the inside could be easily accessed by anyone outside. This was kept intentionally open for the ease of troubleshooting and fixing the issues that arise while testing the robot before the final demonstrations. And although we already omitted the second stage, there were no holding stage to hold the raspberry pi processor and raspberry pi camera. So that, we anyway had to have the cover design ready for the testing as well (Trials before the demonstrations). The cover design was designed using solid works after getting all the required measurements from the existing version of the robot at that stage. And we should say, this was pretty challenging as there were wire connections made from the Arduino MEGA, raspberry pi processor, motor driver, Bluetooth module, power supplies and the camera module which were implemented in a quite managed behavior and had to organize the placing of the cover design without crashing any connection between those modules and to function the system well without losing the outward appearance of the robot as well. So that, the final cover design was used to cover the robot without making a negative impact on the circuit setups and component builds in the system design of the robot as we can see in the final design structure in the above table.

## 8 BACKGROUND INFORMATION

### 8.1 Task 3 - Obstacle Detecting

Historically, obstacle detection can be carried out by a wide number of methods and systems. These systems range in complexity and encompassing range, the challenge is for the user to find the proper trade -off between the anticipated direction which an obstacle approaches and the nature of the vector it moves in. (for example if an obstacle approaches the moving body in a linear fashion) For advanced scenarios and complete identification of foreign objects in the immediate surrounding a system such as RADAR might be implemented via a spinning emitter/receiver. The emitted signals are received by receiver and then translated to distance information to determine if an object is present in vicinity. If no object is present the emitted signal does not bounce back from anything and hence shows no proximity readout.

Currently the project involves detecting a stationary object relatively to the ground plane. The robot perceives the object moving a velocity opposite to current velocity if the object does not move. The vector addition of velocity components of the object determines how fast the external object approaches the physical bounds of the robot. The current task, however, requires simply the distance information of nearby objects.

This can be accomplished in a wide variety of ways but is most commonly done with the commonly known principles of wave dynamics. The distance information of an object can be calculated by measuring the duration it takes for a signal to emit, reflect and finally arrive back at the origin transmitter/receiver.

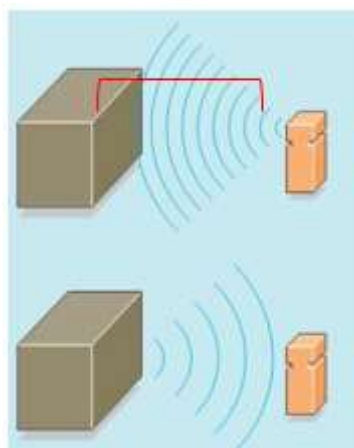


Table 15: Transmission of the signals

$$\text{distance} = \text{velocity of wave} \times (\text{time taken for wave to bounce back})$$

The above technique is the fundamental principle used during detecting obstacles. The following types of sensor rely on the above principle

1. RADAR
2. Ultrasonic Sensor
3. LIDAR

## 8.2 Color Detection

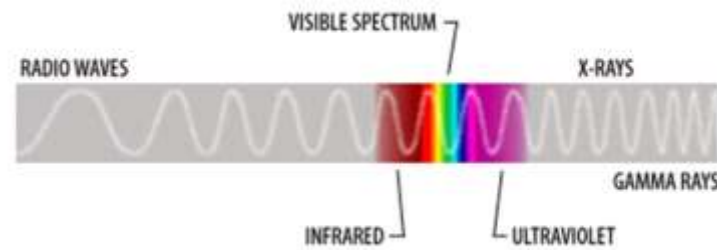












Table 16: Color detection frequency range

Detecting the intensity of light reflecting off a surface and measuring dominant wavelengths is known as color detection. Every object has a certain tendency to absorb more frequencies of light in the visible spectrum than others. If an object absorbs all incoming light it will be perceived as black. Whereas if it reflects all incoming light in the visible spectrum it will be as white.

The following chart summarizes how color is perceived to the human brain: This example assumes the three primary colors fall on an object and it has ideal reflecting capabilities in each of its color specific properties:

Table 17: Color perceive to the human brain

Wavelengths falling on material	Wavelengths Absorbed by Material	Perceived Color
	None	 White
		 Green
		 Blue
	All	 Black

Since the camera works on the same range of wavelengths the human eye is capable of seeing, it observes the world more or less to the way humans see (quality and focus aside). Hence the color information can be used to track the location or size of the object captured in the field of view of the camera.

### 8.3 Visual Properties of Images in Computational Workspaces

Unlike humans' computers rely on different types of cameras to perceive the outside world. Cameras come in a wide variety of sensitivities based on the type of wavelength needed to be primarily captured and other factors such as zoom, focus, frame rate and quality.

Cameras, like the human eye are designed to capture the intensity of the falling radiation onto a photo-sensitive surface. They also, like the human eye need to focus light onto a small region to capture meaningful results. For this they use lenses which refract light onto a photosensitive chip. This chip translates the captured photons based on their intensity values/wavelengths and converts them to a language a computer can understand.

As an example, if a greyscale image sensor is considered, it maps out the falling radiation into a matrix depending on intensity. The image is separated into discrete chunks of intensity having a range of values between a high and low point. The high point is pure white, and the low point is pure black. Depending on the sensitivity of the sensor it can have a wide range of values in between these two boundary points. The number of points determines how much fidelity is captured in the image. An 8MP image sensor (similar to the one used in this project) can capture up to 8 Million points of light in all three primary color bands of the spectrum (red, green, blue). Immediately it becomes clear the sheer amount of data captured in a single instant by the sensor.

This sheer amount of data is the primary source of input for Image processing.

### 8.4 Understanding how Colors are Represented in Computation

Previously, it was discussed how images are translated into a language a computer can read. However, it is simply not enough to adapt this language for humans as it is in the form of long binary strings and OP codes. Hence a middle ground is necessary. For this computer rely on color spaces. These are the standards used to represent color data in a way both humans and computers can understand. Some different color spaces are listed below

1. RGB Color
2. HSV Color
3. Grayscale
4. CMYK Scale

The most widely used color scale is the RGB scale due to its simplistic nature. (The certain downsides it has will be discussed further on). In this scale each color is divided into three different subcomponents, namely i) red, ii) green, iii) blue. These components are saved as 3 bytes (for 8-bit color) which as a whole is called a pixel. Each color ranges from (0,255), where 0 denotes the lowest possible intensity of the color and 255 represents the highest. These values are set based on the aforementioned way photons fall on a photosensitive chip. Each pixel is arranged in a matrix which has the same number of elements and the corresponding image size in MP (mega pixels). Once these matrices are sufficiently large enough an image can be generated from the different intensities observed on the screen.

This pixel data is the primary source of input the program uses to determine which objects are of interest and which are not. In addition to color data other types of data such as edges, corners, features, shapes can also be used to detect objects of interest.

Image thresholding based on certain ranges in then performed on the image to identify other aspects such as moments, perimeters, area and **perceived location relative to the center of the frame.**

### 8.5 Different Types of Color Spaces

A color space can is the way a computer encodes each pixel's color data. It can vary depending on the type of colorspace used. Normally when dealing with colored images RGB is used, however when dealing with ranges and when specifying a range of colors moving to a different color space such as HSV is advised due to the way color is understood. In RGB color is described by all three components as given below



R - Red

G - Green

B - Blue

Any alternations in the color values means all three of these values will change independently from one another. This shift can occur in the real world depending on the lighting conditions and the time of day. In addition shadows and highlights also contribute towards change in the “apparent color” of an object. Humans have the ability of automatically adapting to a shift in color because their brains compensate for color differences. (ie. Even in night a red apple will appear red, because the brain understands the color red and makes an informed “guess” based on past experiences) Computers on the other hand need to be told what a color is based on a color definition.

In HSV color is defined according to the following

H - Hue

S - Saturation

V - Brightness

This adaptation of color can range more easily as only the first variable determines the primary color of the object. Both the second and third descriptors specify the amount of color in an object and how bright it appears to be. Therefore, for image processing purposes switching to HSV was applied to extract colors.

## 9 PROCEDURES AND METHODS APPLIED TO DATE

### 9.1 Task 3

At first, our team intention was to design a frame to hold the Ultrasonic Sensor like in the picture



*Table 18:Initial concept*

However, due to time-limitation and exceeding external fee, we decide to omit this design. Instead, we use the frame holding the path following sensor to also hold the ultrasonic sensor. With this way, we have minimized the cost.

Below is the actual footage of the Ultrasonic Sensor attached to the metal rod by small metal wire.



Table 19: Picture of actual HC-SR04 on robot

Table 20: Specifications of the task

MCU	Raspberry Pi + Color Sensor	Arduino + Ultrasonic Sensor
Implementation	Hard – need to comply with Task 1 and 2	Easily assemble with Task 1 and 2 as they have the same MCU
Coding	Hard - need research throughout	Moderate since team member has experience with Ultrasonic Sensor in Arduino
Physical design	Moderate - We intend to put the Pi underneath the 2nd stage	Simple - the Arduino Mega is built in the robot already Sensor can be easily attached on existing metal rod
Conclusion	Final design use Arduino + Ultrasonic Sensor	



Table 21: Ultrasonic sensor used to detect obstacles

Obstacle detection was primarily performed with the aid of the hc-sr04 ultrasonic sensor. The sensor consists of 4 pins, vcc, gnd, trigger and echo. The trigger pin is responsible for transmitting a signal and the echo signal is responsible for obtaining a readout of the reflected ultrasound signal. The operation of this module is discussed below. This sensor provides distance data to object directly in front of it. The objects shape and surface type affect how good the data can be received (ie. If the object surface is not flat, reflected sound waves are distorted)

The hc-sr04 sensor consists of two main sub modules. Namely,

1. Transmitter
2. Receiver

The physical construction of the transmitter is an open-ended steel cylinder with a meshed covering on the top. Inside the cylinder a mini concave dish is encapsulated at the center. The receiver is similarly built. The reason they are built this way is to localize the signals emitted.



The transmitter is designed to send an ultrasonic pulse and the receiver is designed to interpret the reflected pulse signal off an object. If the signal is not present the device interprets the object being at infinity (or beyond the device's range).

The device works by measuring the time between the pulse sent and the received pulse. This time is then multiplied by the constant (speed of sound) to obtain the distance to an object.

$$\text{distance} = \frac{\text{speed of sound}}{2} \times (t_1 - t_2)$$

Where  $t_1$  is time pulse is sent  
and  $t_2$  is time pulse is received

The ultrasonic hardware itself is responsible for measuring the sent and received time signals. It then outputs this information to the Arduino.

The procedure to obtain a distance readout from the sensor is as follows.

1. To begin the process an outbound signal is required. To generate this signal a 10-microsecond pulse is required on the trigger pin. (set pin HIGH then LOW for  $t = 10\mu\text{s}$ )
2. The ultrasonic sensor then generates an 8-cycle burst via the transmitter
3. If an object is present in the near vicinity of the sensor it will reflect this generated pulse back to the sensor.
4. The sensor uses the receiver to measure how long it took for the pulse to return. It then creates a pulse on the echo PIN which has a pulse width equal to this time.
5. The final step is where the Arduino reads the pulse width on any digital pin via pulseIn() function.

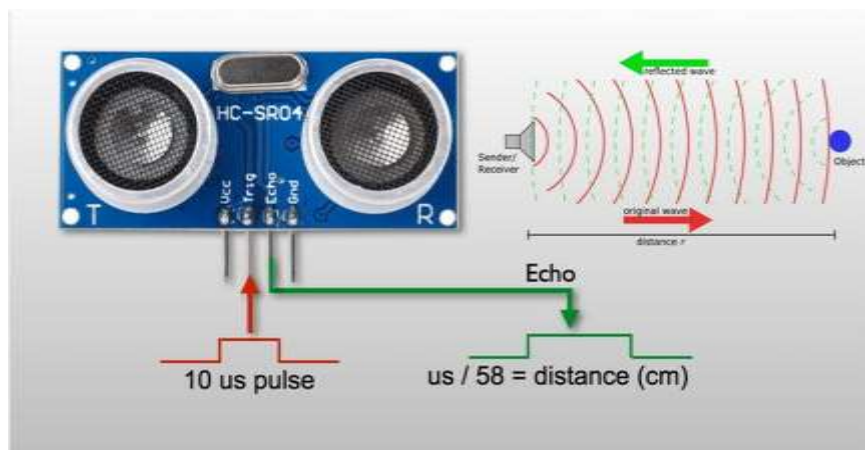


Table 22: Reading the pulse by Arduino via its pins and its functionality

For the project the 3<sup>rd</sup> task involves detecting an obstacle set on the path during the line following stage. To integrate this into the system the program checks for an obstacle every iteration of the main while loop in the Arduino. Since the program iterates through the loop multiple times a second it does not face any latency issues when detecting objects when travelling at low speeds.

The following lines of code briefly explain its function:

The full function is divided into two sub methods which retrieve the distance data.

```
// Team C
long getDistance() {
    digitalWrite(trig, LOW);
    delayMicroseconds(5);

    digitalWrite(trig, HIGH);           // positive pulse for 10us
    delayMicroseconds(10);
    digitalWrite(trig, LOW);

    duration = pulseIn(echo, HIGH);     //measure duration of returned pulse

    dist = duration * 0.032 / 2;         //speed of sound 320 ms-1

    return dist;
}
```

Table 23: Arduino Code to implement the above function

The above method returns the distance readout by the sensor every time it is called from the main function or another method. The next snippet will describe how the complete system runs with the second sub method.

```
// Team C
int distVerify = 0;

void checkCollision() {

    dist = getDistance();
    if (dist > 50) {
        return;
    }

    Serial.print("Distance ");
    Serial.println(dist);

    if (dist < DIST) {

        distVerify++;

        if (distVerify > 10) {

            while (dist < DIST) {
                dist = getDistance();
                analogWrite(ena, 0);
                analogWrite(enb, 0);
                Serial.println("under");
            }

            distVerify = 0;
        }
    }
}
```

Table 24: how the complete system runs with the second sub method

### Code Explanation

- The two sub methods which determine if an obstacle is present are explained in the snippets above. They are, respectively
  1. `checkDistance()`;
  2. `checkCollision()`;
- ❖ The first method returns a distance readout each time it is called. (ie. if an obstacle is 15cm/14cm..etc)
- ❖ The second method (`checkCollision`) is called in the main function. It runs the full function of reading, checking and also deciding what to do if the distance passes a certain threshold.
- ❖ **`checkDistance()` Function**
  - Function which instructs the sensor to begin a receive and transmit function
  - It returns the distance acquired from the sensor

### ❖ **checkCollision() Function**

- ❖ Calls the checkDistance() function and receives the readout of the distance
- ❖ If the distance is greater than 50cm the function returns without running the rest of the code
- ❖ If the function is less than 50cm it reads and prints the distance out

### Error Checking & Preventing False Positives

- If the distance is less than 6cm, it increments the sampling variable **distVerify** by 1.
- The variable **distVerify** is incremented by 1
- **Only once 10 samples are obtained does the sensor give a positive readout that an obstacle is present.**

Therefore, the sensor only signals an obstacle is **present** when a proper number of samples are received. Applying a single condition to detect whether an obstacle is present raises some challenges since sensor readings can present false values. These errors are mainly random and must be minimized as much as possible.

## Task 4 - Color Recognition

The final task is the most challenging task of the project. It requires the seamless communication of both the Raspberry Pi and Arduino in addition to all the other components on the robot. For instance, the camera needs to provide visual information to the pi, the pi needs to tell the Arduino what to perform. The Arduino needs to operate the mechanism needed to drive the wheels and steer/move straight and the ultrasonic sensor needs to give a readout in case a collision detection is met while parking in final “garage” destination. The 4<sup>th</sup> task can be considered as a culmination of most of the aspects of this project. It encapsulates some important concepts such as Pi  $\leftrightarrow$  Arduino UART communication, OpenCV color recognition and feature tracking, PID tuning based on object of interest location and a majority of python programming.

The task of color recognition was primarily done based on a few key principles which will be discussed in the sections below. The full task can be divided into two major sub-sections as follows

1. Raspberry Pi Operation
  2. Arduino Operation
- ❖ Raspberry Pi operation considers all the aspects of task 4 which was completed on the Raspberry Pi board. Operations such as OpenCV image processing fall under this category.
  - ❖ Arduino Operation describes the operational aspects of the Arduino board that are executed during Task 4. Serial communication and turning in the junction can be considered as tasks which fall under this category.

## 9.2 Introduction to Computer Vision

Computer vision is one of the aspects of modern computing which deals with visually processing images based on their pixel information. Not only are individual pixels processed during computation, pixels surrounding a certain pixel are also considered as a whole when applying certain algorithmic methods/functions. That is, a kernel (N x N matrix) is used when detecting certain aspects in an image which are significant. In other words, inter-pixel data (neighboring pixel information) is just as important as data contained within each individual pixel. The certain aspects put forth during implementation will be discussed in the sections below.

For a computer to share the same concept of vision as humans an architecture or framework is required to distinguish information from all the millions of bytes of data received every second. That is to see important aspects of an image such as lines, corners, edges, curves, objects etc the computer must be able to make sense of

all the 1's and 0's and turn it into something meaningful. The underlying framework which is used for this task is OpenCV v4. It is a set of libraries which contain image processing functions including much more advanced functions such as machine learning and neural network building as well. These libraries have been compiled into a single import class which can be called using cv2.<function/method Call> (assuming import cv2 is used during initialization) Once a function is called it will return information after applying certain algorithmic methods to the image frame specified. Since video is just a series of frames the image processing algorithms which are applied to image processing are more or less the same as the image processing algorithms applied to image processing. So, there is no separate algorithms for video processing and image processing. Instead videos are reduced into frames and each frame is analyzed/processed. By using this procedure data can be processed/captured/recompiled very efficiently while retaining the functionality of any given method/function. For instance, as an example if a user need identify the lines of a certain image they can apply "Hough Line Transform" and identify major horizontal or vertical lines within the image. They can apply it both on an image and a video due to the aforementioned process of reducing videos to separate frames for processing.

### Importance of Numpy

Images, as previously discussed are essentially massive matrices of data in which each element contains a pixels brightness/color intensity. Additionally, the location of each pixel signifies how the image will be finally displayed to the user. Both these types of data are crucial for image processing. For matrix processing a separate more efficient framework is required in addition to the default frame supplied by default in python. This framework is known as numpy and is used for plotting certain data points, defining certain ranges of colors and processing other aspects such as contour data more efficiently than python's normal default matrix compiling counterpart. Numpy is a library which facilitates multi-dimensional array logic in python. This enables better performance of the system and prevents unnecessary stalls which results in frame skipping and low frame rates. Therefore, matrix processing in numpy is used extensively side by side with OpenCV.

### Main Hardware Components Used in Image Processing

#### Pi Camera 2



Table 25: Pie camera used for image processing

For visual sensory input the 2<sup>nd</sup> revision of the pi camera was used for task 4. This camera has a resolution of 8MP and is capable of recording video with up to a max resolution of 1080p. (1920 x 1080). However, for the current requirements video of this high quality is not required. The quality of the video determines how large each singular frame is how many pixels need be processed in each frame. The processing time is exponentially proportionate to the resolution of the image being processed. For example, an image with a resolution of 200 x 200 has 4x as many pixels (40,000 pixels) as an image with 100 x 100 resolution (10,000 pixels). Therefore, proper resolutions and resizing should be set prior to processing each frame to prevent unwanted use of hardware resources and lag. Depending on the requirements of a task these values can be set accordingly. The frame rate of this camera can also be considered for slow motion videos, the quality of the video reduces as the frame rate increases because the total throughput of the camera is a set value.

The Pi camera 2 connects to the main Raspberry Pi board with a 15cm default provided ribbon cable which interfaces directly with the CSI port on the Rasp Pi Model B. This ribbon cable is quite fragile and should be

handled with care. The ribbon can also be replaced or upgraded easily via the easy to remove latching mechanism of the camera.

Below is an image of how the camera was connected to the Pi.



Table 26: Camera connect to RASP-pie

### Supporting Hardware for Camera

For the camera to properly read images it needs to be set up in the right location. Setting the camera at the optimum location improves image recognition quality and coverage of objects. In order to get the best possible design a custom bracket was created with a simple plastic board. The design was both simple and functional and was hence included.

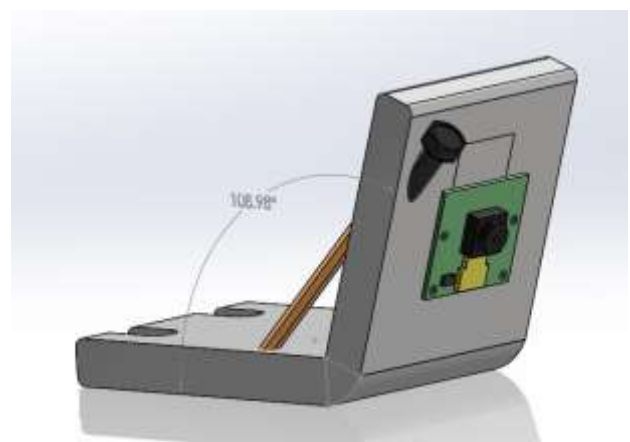
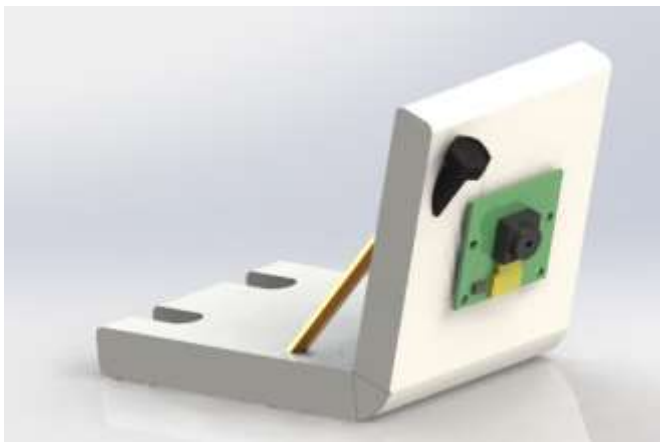


Table 27: Camera mount to the robot prototype

The mount base contains two base points for sliding the camera forward or backwards. It also contains a screw which when tightened decreases the angle shown below. This setup was designed with flexibility in mind to enable changing slight positional data during practical testing to obtain best possible results for camera image. **Flexibility** is an important design ideology to consider especially when the quality of information from the sensor depends on the sensor location.



*Table 28: Preliminary prototype design of mount*

### 9.3 Approach to Task 4

An important point worth highlighting is, when considering how to create a feasible method for task 4 two main options were considered. Of these options one relied heavily on a multiple variable factor staying perfectly constant while the other was a more flexible more reliable design

1. Hard Coding Method - Relying on delays to run the motors for specific amounts
2. Active Color Tracking - Relying on Raspberry Pi + Camera to process surroundings

*Table 29: Difficulty variation of the tasks*



Hard Coding vs. Active Tracking	
Method 1 - Hard Code	Method 2 - Active Color Track
<ul style="list-style-type: none"> <li>Slight changes in battery voltage will produce significantly undesirable results</li> </ul>	<ul style="list-style-type: none"> <li>Battery voltage has no significant effect</li> </ul>
<ul style="list-style-type: none"> <li>During turning/traversing slight variations of friction on the surface will produce overshoot or undershoot</li> </ul>	<ul style="list-style-type: none"> <li>Friction does not have effect</li> </ul>
<ul style="list-style-type: none"> <li>A minor adjustment of the garage location (even by a cm) will require the robot to be reprogrammed</li> </ul>	<ul style="list-style-type: none"> <li>Can tolerate slight adjustments of garage location</li> </ul>
<ul style="list-style-type: none"> <li>Good for one extremely inflexible use case scenario</li> </ul>	<ul style="list-style-type: none"> <li>Flexible for multiple use cases where garage could be near or far</li> </ul>
<ul style="list-style-type: none"> <li>Single run design; slight shift of any external components means complete program upload</li> </ul>	<ul style="list-style-type: none"> <li>Multiple run design</li> </ul>
<ul style="list-style-type: none"> <li>Relies on motor torque and accurate timing to travel a certain distance</li> </ul>	<ul style="list-style-type: none"> <li>Relies on location of most concentrated region of color</li> </ul>
<ul style="list-style-type: none"> <li>During turning a slight angle offset from desired location will cause extreme offset from the final stop location due to angle drift</li> </ul>	<ul style="list-style-type: none"> <li>Can compensate for angle drift by actively centering the desired location to travel to</li> </ul>
<div>❖ Difficulty:</div> <ul style="list-style-type: none"> <li><i>Programmatically</i> - Simple</li> <li><i>Practically</i> - Difficult</li> </ul>	<div>❖ Difficulty:</div> <ul style="list-style-type: none"> <li><i>Programmatically</i> - Hard</li> <li><i>Practically</i> - Relatively Simple if calibrated properly</li> </ul>

Of the two designs the Team's choice was to go with the second more complex design both as a way to mitigate errors but also as a way to create a robust system that works in multiple case scenarios despite Method 2 being more programmatically difficult.

### Software Design for Task 4

Once all the hardware requirements were satisfied the next major hurdle was tackling the momentous task of writing the code to combine everything together. To attempt this problem a step by step approach was taken to gradually develop the code to write the full solution. The initial stage testing and trial "blueprint" of the final code can be summarized according to the following list.

#### Raspberry Pi

1. Obtaining an Image/Video from the Webcam on PC in OpenCV-python
2. Obtaining an Image/Video on Raspberry Pi Cam in OpenCV-python

3. Detecting a single color and using masks to obtain select region where color exists
4. Understanding HSV color range finding more flexible ways to input it so color is detected seamlessly
5. Detecting multiple colors simultaneously and returning the one detected based on largest moment area
6. Obtaining image moments based on tracked color and shape of object of interest
7. Finding the center of object interest using Moments
8. Reading the value of the center of the image w.r.t. to the center of the frame
9. Constructing more flexible framework to tweak/calibrate color depending on lighting conditions and save data onto board (trackbars)
10. Implementing a system to write stop function to robot when no color is present
11. Investigating on how to send information from Raspberry Pi to Arduino
12. Determining Serial UART communication as the definitive solution due to its benefits over the other option. (will be discussed)
13. Setting up serial communication to send/receive a single byte to Arduino/Pi
14. Writing more complex data to Pi/Arduino (multiple chars)
15. Constructing methods that when called return the range values of predefined colors (blue, yellow, green)
16. Reading sub frame from main frame to get color of cube
17. Optimizing code to only run color detection on the cube for a certain time (sample counting)
18. Setting up stalling mechanism to prevent unnecessary hogging of CPU resources when running other stages (Task 1,2,3)
19. Error checking and Preventing false positives by implementing a system to react to the maximum area of concentrated color regions rather than any region
20. Testing and Debugging for proper function, adjusting sample count threshold variables, toggle image previews only when required

#### Arduino

1. Reading serial data transmitted from Raspberry Pi
2. Create system to read more than one character at a time
3. Create all new motor control function to process received error data from the Raspberry pi
4. Implement framework to process received data from pi and run certain commands (stop, turn right, left etc.)
5. Turning function
6. Add new steps and integrate to final FSM

## **9.4 Major Code Implementations**

### **Flexible Color Recognition**

For color tracking to work as intended colors need to be defined properly in given ranges. But since colors can appear to change depending on factors such as sunlight, day/night, lighting conditions or even shadows a more flexible system has to be implemented to quickly adjust and keep track of the required colors. The system thus implemented was to use a set of trackbars to initially calibrate the required color until it falls into the range and then save these lower and upper bounds as .txt files. The next time the program runs it will read these text files and load in these values. Hence data is persistent.

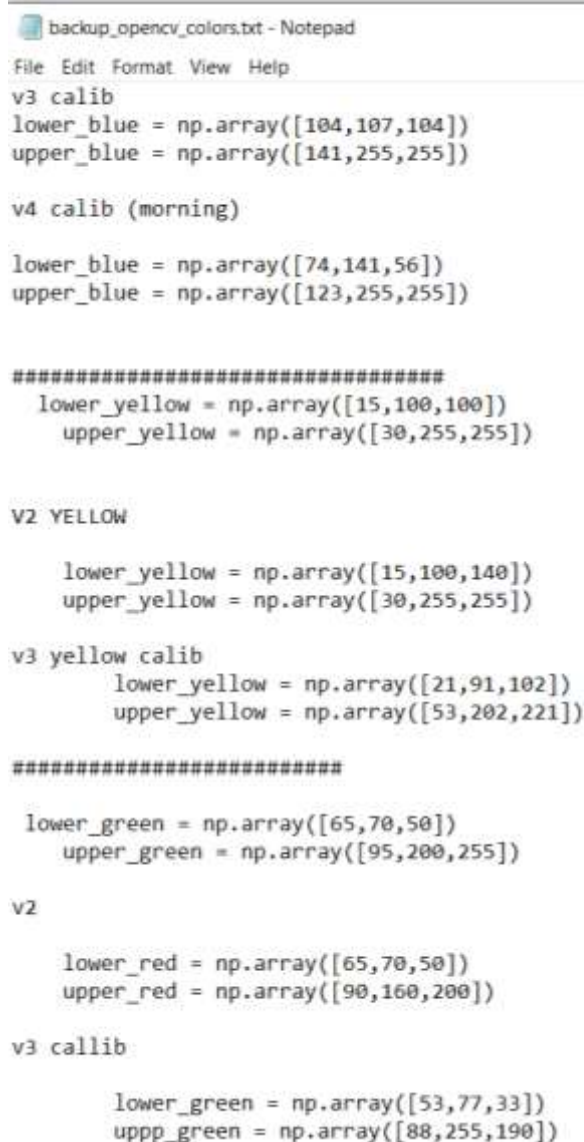
Additionally, data can even be set to written to separate text files or even within the same text file depending on the time of day etc. This way a user can load in the calib data from the files depending on the time of day or lighting conditions.

The preview below shows how such a system would be implemented. However, in the current assignment the use of separate text files was considered as the intended running time of the robot is only within the range of an hour in relatively fixed lighting conditions of the lab.

*Please note following preview is the manual implementation of the described method above, where users have to manually test a color, see if it works then copy paste the lines of code corresponding to the ranges into the text file.*

*The automated version is discussed next.*

Preview of the text file is as follows,



```

backup_opencv_colors.txt - Notepad
File Edit Format View Help
v3 calib
lower_blue = np.array([104,107,104])
upper_blue = np.array([141,255,255])

v4 calib (morning)

lower_blue = np.array([74,141,56])
upper_blue = np.array([123,255,255])

#####
lower_yellow = np.array([15,100,100])
upper_yellow = np.array([30,255,255])

V2 YELLOW

lower_yellow = np.array([15,100,140])
upper_yellow = np.array([30,255,255])

v3 yellow calib
lower_yellow = np.array([21,91,102])
upper_yellow = np.array([53,202,221])

#####

lower_green = np.array([65,70,50])
upper_green = np.array([95,200,255])

v2

lower_red = np.array([65,70,50])
upper_red = np.array([90,160,200])

v3 callib

lower_green = np.array([53,77,33])
uppp_green = np.array([88,255,190])
  
```

Table 30: Preview of the text file

### Semi-Automated System to Save and Store Calibrated Color Ranges

To facilitate storing and retrieving color data relatively quickly the following system was set up. The code snippet below explains how it functions.

Data Storing - press letter 'b' to store blue color data

```
//Team C

... //while loop

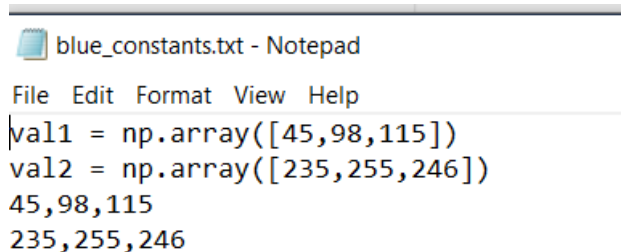
k = cv2.waitKey(5) & 0xFF //wait for key press every 5ms
if k ==27: //if keypress is Esc close loop
break

elif k == ord('b'): //else if key press is letter 'b'
with open('blue_constants.txt', 'w+') as the_file: //open file as write
    str1='val1 = np.array([{},{},{ }]).format(l_h,l_s,l_v) //write these lines of text
    str2='val2 = np.array([{},{},{ }]).format(u_h,u_s,u_v) //(for quick, easy copy pasting)
    str3 = '{ },{ },{ }'.format(l_h,l_s,l_v) //(for actual data loading to prog)
    str4 = '{ },{ },{ }'.format(u_h,u_s,u_v)
    the_file.write(str1+'\n'+str2+'\n'+str3+'\n'+str4) //write function

... //similar for other colors, y ,b
```

Table 31: Code snippet for the set forth function

## File Preview



```
blue_constants.txt - Notepad
File Edit Format View Help
val1 = np.array([45,98,115])
val2 = np.array([235,255,246])
45,98,115
235,255,246
```

Table 32: File preview

## Data Retrieval

//Team C

```

with open('blue_constants.txt','r') as fg:
    lines = fg.readlines()           //read all lines and store to array
    line3 = lines[2]                 //line 3 of array
    line4 = lines[3]                 //line 4 of array

    l_h_val = int(line3.split(',')[0]) //split wrt ',' and read val as int
    l_s_val = int(line3.split(',')[1])
    l_v_val = int(line3.split(',')[2])

    u_h_val = int(line4.split(',')[0])
    u_s_val = int(line4.split(',')[1])
    u_v_val = int(line4.split(',')[2])

print(l_v_val )

```

Table 33: Data retrieval code

The following methods above proved very useful and quick for storing color data quickly and efficiently. However, the problem with this method is that the values need be specified manually first. That is the user has to make an informed guess about what the HSV colors might be. **They cannot visually see what's happening when the HSV are changed.**

To combat this problem **trackbars are used to do calibration** on the go.

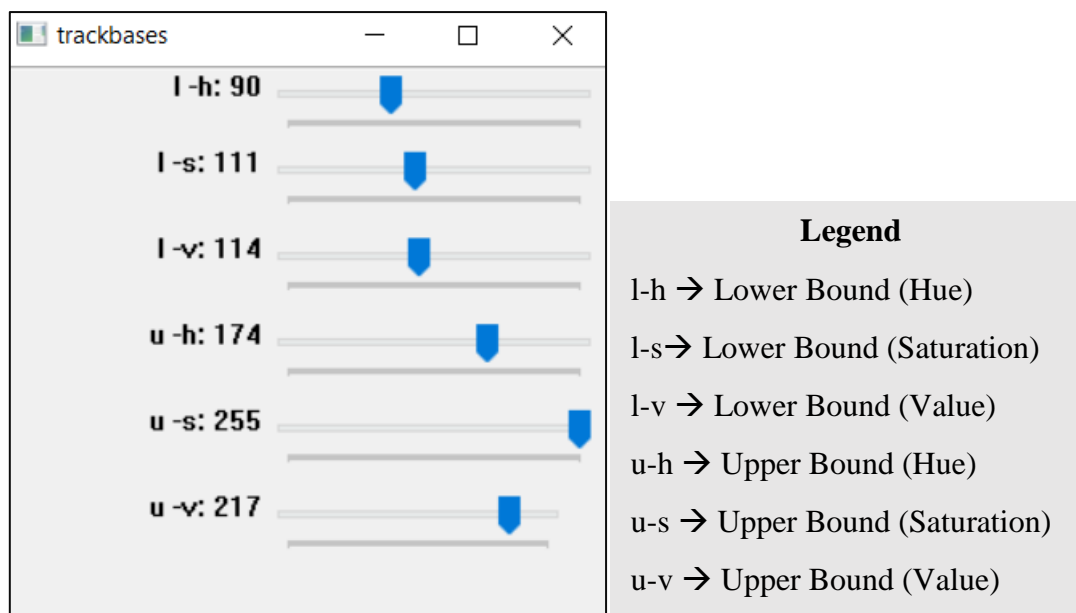


Table 34: Function of the trackbars (Calibration)

These values are then retrieved from the trackbars and used as the range values for determining color. For this system one color can, be tuned at a time.

```
//Team C
with open('blue_constants.txt','r') as fg:
    lines = fg.readlines()           //read all lines and store to array
    line3 = lines[2]
    line4 = lines[3]

    l_h_val = int(line3.split(',')[0])   //get values as integers
    l_s_val = int(line3.split(',')[1])
    l_v_val = int(line3.split(',')[2])

    u_h_val = int(line4.split(',')[0])
    u_s_val = int(line4.split(',')[1])
    u_v_val = int(line4.split(',')[2])

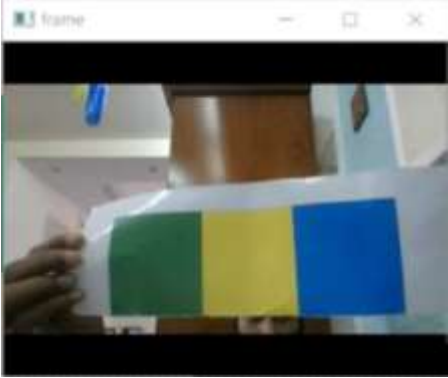


    val_blue_low = np.array([l_h, l_s, l_v])   //set range values
    val_blue_high = np.array([u_h, u_s, u_v])
```

Table 35: Code to implement the set forth function

### Color Ranging and Object of Interest Extraction

The reason color is extracted from the image is to find the object of interest. Since the objects are of distinct color and have uniform color throughout the surface, color is a quick and effective way to find the object of interest. The following preview window shows what object of interest looks like when applied properly

Table 36: Color range for three color required

		
Original Image	Bitwise Reduction Mask	Applied Mask to Original Image

The above image shows how color can be separated using OpenCV. The reference image is the colors that are required to be detected. From these colors color ranging is used to specify “green” in HSV. After specifying the color, the program then creates a mask where all the pixels in that range are set to 1 and all the pixels which are not are set to zero. This is also known as thresholding.

After thresholding has been completed the program takes the mask and applies it to the original image. The white pixels allow the original image to be rendered while the black pixels do not. The first and third images are for visualization of the process. The second image is the information that the program requires to determine key aspects like the moments and centers etc.

```
mask = cv2.inRange(hsv,lower_red,upper_red)           // 1 if in range 0 if not
if(cv2.countNonZero(mask)<1000):                       //count number of white px in mask
    port.write(bytes('s', 'utf-8'))                  //if no color detected write 'stop'

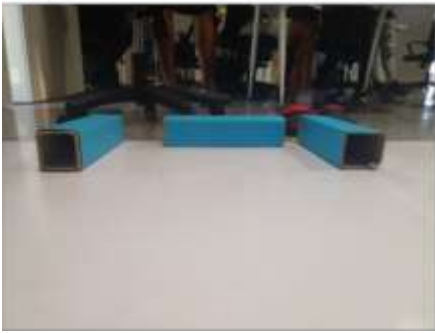
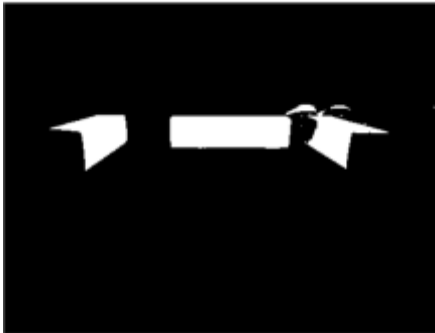
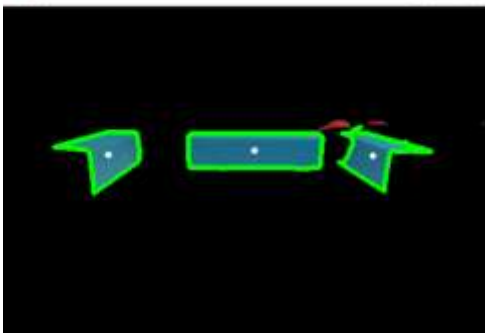
print(cv2.countNonZero(mask))                         //display value to user
res = cv2.bitwise_and(frame,frame,mask=mask)          // to show overlay of mask on original
```

Table 37: Code implementation

### PID + Image Recognition via Moments

The next part of the process is the key component of the system that enables task 4 to work elegantly. Instead of using hard coded methods the PID algorithmic method enables the use of image processing to properly detect where the parking garage is no matter where it is placed. (As long as the camera can see it)

Table 38:Image processing

		
Original Image	Bitwise Mask with Ranged HSV colors	Contours and Centre of mass using Moments

The above method is used to obtain the image moments in openCV. Once the data is properly returned the next step is to identify the biggest moment of the system. (which is most likely to be the middle box since it's directly facing the camera)



## // Team C

...

mask = cv2.inRange(hsv,lower\_blue,upper\_blue) //filter only 'blue' color

//returns the moments of the image according to an algorithm

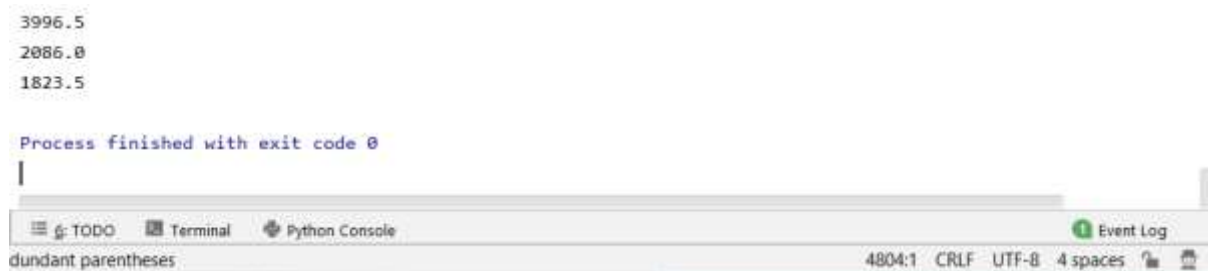
```
cnts= cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
```

```
for c in cnts:
    area = cv2.contourArea(c) //calculates area of moments
    if(area>1000):
        cv2.drawContours(res, [c], -1, (0, 255, 0), 3, -1) //draw contours in green line color
        M = cv2.moments(c) //calculate image moments
        cx = int(M["m10"] / M["m00"]) //calculate center of area
        cy = int(M["m01"] / M["m00"])
        cv2.circle(res, (cx, cy), 3, (255, 255, 255), -1) //draw white circle
        print(cv2.contourArea(c))
```

Size of the 3 areas underneath the moments are as follows

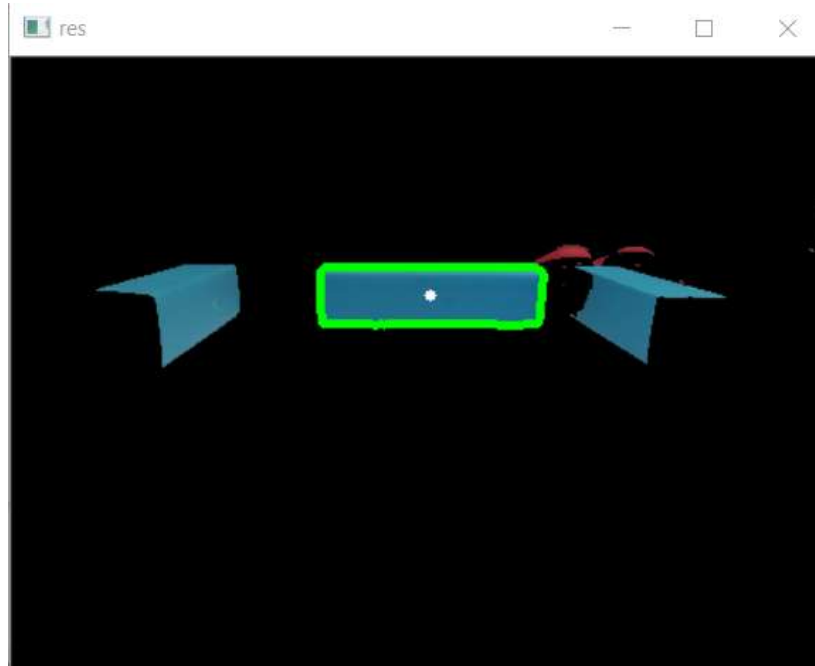
```
3996.5
2086.0
1823.5

Process finished with exit code 0
```



If we only require the biggest one, we can edit the code to detect moments above a certain threshold.

```
area = cv2.contourArea(c)
if(area>3500):
```



The next step is to gradually develop the required steps needed to implement a PID algorithm based on this information. The following lines help explain how this can be implemented to obtain the value of x.. The error thus calculated can see below.

```
width = frame.shape[1]
print('set point :{ } '.format(width/2))
print('location of x : { }'.format(cx))
error = (width/2 - cx)
print('error: { }'.format(error))
```

The above lines of code when implemented give the following output for the above image.

```
set point :230.5
location of x : 236
-5.5
```

```
Process finished with exit code 0
```

The value which is sent to the Arduino is the location of x. It is sent via UART communication to the board. The Arduino then calculates the error based on a known image frame size (which is constant).

*Please note actual values may differ from above demonstrative examples.*

```

void RaspiSerial() { //Arduino Code

    if (Serial3.available()) { //check for incoming Serial bytes from raspberry pi

        char str = Serial3.read();
        if (str == 's') { // stop if no color is detected
            stop1();

        }
        else if (isdigit(str)) { //navigate to "parking garage"
            recieved_pi_Char[0] = str;
            sscanf(recieved_pi_Char, "%d", &cvt_int);

            int object_frame_error = 5 - cvt_int ; //find error (5 is set point)

            int speedMotor_PI = Rkp * object_frame_error + Rkd * (error - prev_error); //positional error of
            object of interest

            MotorControl_PI(speedMotor_PI); //call method to run motors

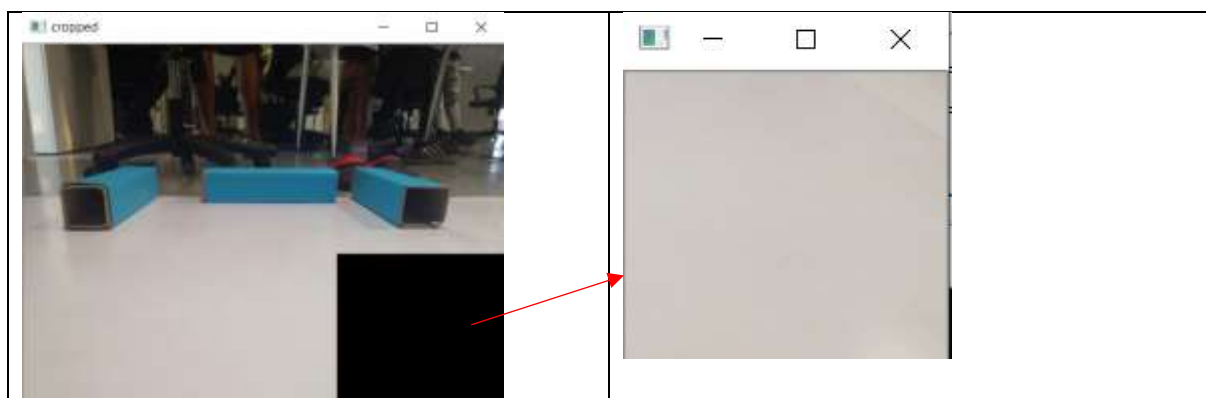
            prev_error_object = object_frame_error;
            Serial.print("char");
            Serial.println(recieved_pi_Char);
        }
    }
}

```

### Turning Function (to find Proper garage to park)

The turning function is implemented via a separate method which runs indefinitely until a color is properly detected. Once a color has been detected it increments a variable and breaks; out of the while loop. Then normal image recognition and PID tracking can begin.

Before moving on to the turning function however another critical aspect of the program must be discussed. To detect a cube and the surroundings at the same time without two cameras or moving the camera a system was developed to crop a certain region of the image out of the frame. This cropped region is purely used for recognizing the cube's color. After the cube has been set it does not interfere with the robot's normal color



Main camera feed is blacked out to prevent the camera from reading values from the cube	Image sent for processing color of cube
---	---

tracking functions.

```

while sampleCount<160:                                //when global variable sampleCount thresholds
    _, frame = cap.read()
    frame = cv2.resize(frame, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)

    y_crop = cv2.getTrackbarPos("y", "cropper")          //get crop dimensions
    x_crop = cv2.getTrackbarPos("x", "cropper")

    crop_segment = frame[y_crop:640, x_crop:300]         //get cropped cube image
    checkColor(crop_segment)                             //send to check color method
    cv2.imshow("frame", frame)

    cv2.imshow("cropped", crop_segment)
    print(sampleCount)

    k = cv2.waitKey(5) & 0xFF                            //override and escape if
req.
    if k ==27:
        break

cv2.destroyAllWindows()

```

This crop window also has trackbars and can be adjusted easily to properly cut off the right amount off the main camera feed.

## Method which check color and sends data to Arduino via UART

```

class Color:                                //Team C
    colname_lower=0                          //lower range of color
    colname_higher=0                        //upper range of color

color = Color()                             //separate class which stores color ranges

def checkColor(crop_segment):

    hsv1 = cv2.cvtColor(crop_segment,cv2.COLOR_BGR2HSV)    //convert received cube image to HSV

    mask_blue = cv2.inRange(hsv1,lower_blue,upper_blue)    //mask definitions
    mask_yellow = cv2.inRange(hsv1, lower_yellow, upper_yellow)
    mask_green = cv2.inRange(hsv1, lower_green, upper_green)

    blue_count = cv2.countNonZero(mask_blue)                //count how much blue is present
    yellow_count =cv2.countNonZero(mask_yellow)             //count how much yellow is present
    green_count =cv2.countNonZero(mask_green)               //count how much green is present

    # print("blue: { }".format(blue_count))
    arr = np.array([500,green_count,blue_count,yellow_count]) //find most prevalent color
    max_index = np.where(arr==np.amax(arr))[0][0]
    global sampleCount

    if(max_index==0):

        return "nocolor"
    elif(max_index==1):

        sampleCount+=1                                //increments sampleCount
        color.colname_lower = lower_green              //saves the mask for the color in a separate class
        color.colname_higher = upper_green
        port.write(bytes('g', 'utf-8'))
    elif(max_index==2):

        sampleCount+=1
        color.colname_lower = lower_blue
        color.colname_higher = upper_blue
        port.write(bytes('b', 'utf-8'))

    elif(max_index==3):
        sampleCount+=1
        color.colname_lower = lower_yellow
        color.colname_higher = upper_yellow
        port.write(bytes('y', 'utf-8'))

    else:

        return "nocolor"

```

## Arduino Side for Turning

Once the proper cube color has been detected the raspberry pi then transmits this data to the Arduino via Serial UART. The Arduino then has the task of turning right/left or moving straight ahead.

The Arduino follows the following code structure to determine which direction to turn in.

Table 39: Turing from Arduino

CHARACTER RECEIVED	METHOD/ FUNCTION CALLED	DESCRIPTION
'b'	rightSpecial();	Turn Right
'y'	goStraight();	Go Striaight
'g'	leftSpecial();	Turn Left

```
void directionSelect(char str) { //serial data received from Pi

  switch (str) {

    case 'b': rightSpecial(100); Serial.println("blue"); break; //100 → PWM strength
    case 'g': leftSpecial(100); Serial.println("green"); break;
    case 'y': goStraight(100); Serial.println("yellow"); break;
    default : Serial.println("no data"); break;
  }

  delay(900); //turn for 900ms (until garage comes to view of camera )
  forward(5); //default motor configuration (forward drive)
  stop1(); //stop for brief moment
  state = RASP_PI; //switch state to color tracking
}
```

## Problems and Challenges Faced

Table 40: Problems faced during the process

Category	Problem	Issue	Solution	Overcame?
Arduino	L298N refusing to respond to PWM signal	GND connected to motor driver but not common	Enable 1 single common GND for all components	✓
	Line following sensor 1 refuse to give proper output	Sensor was calibrated to work at extremely close levels to the ground	Get new sensor which could tolerate higher heights off the ground	✓



	Line following sensor 2 refuse to show proper readings	Library required to operate sensor properly	QTRSensors library included	✓
	IR sensor reporting outputs but incorrect	Calibration settings affecting proper reading	Remove calibration and set manual values	✓
	IR sensor worked fine previous day, but day of demo did not	During design change IR sensor was slightly more raised off the ground	Changed calibrated values on each sensor	✓
	Bluetooth module refused to work with Serial monitor	Incorrect model version listed as hc-06	Guides for HC-05 were used	✓
Raspberry Pi	Using Raspbian OS without keyboard	Keyboard required for configuration	VNC connection used	✓
	Serial connection not working	Raspberry pi has default serial set to Bluetooth	Disabled default and set to /dev/serial0	✓
	Installation of OpenCV	OpenCV not detected as installed	Python-opencv-contrib + other dependencies installed	✓
	Raspberry Pi reporting incorrect 2-digit numbers via serial	Serial sends one byte at a time	Single digit numbers/letters used	✓

## 10 FUTURE WORKS

Although this project requires only 4 tasks, there are some improvements that we can implement in the future. The first one is improving user experience with the robot and the controller application. San Joe suggest the idea that we can send the video from the Raspberry Pi to the Android phone via Wireless so that users can watch the real-time video streaming [7]. We can also build an iOS application based on instructions of Owen L Brown to connect to the Arduino and control the robot. Therefore, the usable devices are not limited in Android only. [8]

In order to improve the hardware design, Jeremy S Cook proposes the idea that is using Omni wheel which can helps our robot runs smoothly, forward, backward and slide sideways with almost no friction [9] Jorge Rance also show the ideas that we can put 3 HS-SR04 ultrasonic sensors around the robot so that it can detect obstacles not only in front but also on sides of the robot. In this way, we can protect our product from damages when it turns right or left. Moreover, the robot can also be functioned to avoid the obstacle by applying these sensors. It can be coded to keep the distance to the object on the sideways and go around it [10]





As our above research, deep learning can also be an advance upgrade for our future work. Sarthak Jain shows that this real-time object detection method helps to detect and predict the kind of objects with confidence levels. Then, the Raspberry Pi can send the results to user's mobile application to inform object type and also warn if they are too close and pose a threat. By tracking the object's kind and moving, we can predict its directions to avoid crashing. [11]

To improve the tracking functions of camera, Andrian Rosebrock suggest the idea that we use the Pimoroni pan tilt HAT full kit. This device has two servos for panning left or right, tilting up or down. In this way, we don't need to manually set time for the turning to seek for the position of garage. The camera can look around and detect the desired object. Furthermore, he also suggests to using an HDMI screen to display the video. This helps to visualize and debug. [12]

Finally, an advance upgrade could be installing a Robotic Arm connected to Arduino. The shaft of its servo can reach around 180 degrees. With this design, our robot can pick up the load and put on the container. It can also be able to remove the obstacle in front of it. [13]

Overall, with all these suggestions, we expect to improve our robots in software, hardware and user experience also.



## 11 RISK ANALYSIS

Table 41: Risk assessment

Risk Assessment Form	Task Name:	Final design	
	Assessor(s):	Group C	Date: 10 Jan 2019
	Those at risk:	Participants, Workers, Lectures	
	Risk:	Moderate	
Process/Job Description:	Design a robot to meet specific task in Project 3A course.		
Task	Hazards	Controls Already in Place	Action (Hierarchy of Control)
Nuts and bolts	Choking	Contain them in small plastic package	Personal Protective Equipment
	Eye damages	Don't tighten them too hard	(PPE)
	Finger scratches		Administration
Drilling objects	Dropping on foot, damage the frame	Proceed with caution Ask for help	Personal Protective Equipment (PPE) Administration
Grinding objects	Scratches Damage the frame	Carefully proceed grinding Seek for advice if possible	Personal Protective Equipment (PPE) Administration
Assembling the robot	Scratches Dropping on foot Break the robot Incorrect measurement	Proceed with care Carefully ask friend to help Proceed with the same manufacturer	Personal Protective Equipment (PPE) Administration
Robot storage and equipment	Damage the robot Sharp objects can cause injuries	Use big carton to store the robot Use separate box to store equipment	Administration
Robot wheel	Miss-alignment Heavy and bulky	Carefully calibrate the wheel's pin Design the robot with light-weight material	Personal Protective Equipment (PPE) Administration
Electrical circuit assembling (Raspberry Arduino, Battery, etc.)	Electrical shock Burning devices	Carefully check and double check if there short circuit Use multi-meter to check connectivity	Personal Protective Equipment (PPE) Administration
Using HC-SR04, Raspberry Pi and its camera	Low quality devices Failed tasks Burning other equipment	Apply basic coding to test the device Field test is mandatory Use multi-meter to check connectivity	Personal Protective Equipment (PPE) Administration
2 <sup>nd</sup> stage design	Low quality frame can easily damaged during transporting	Use at least 2.5mm acrylic board. For loading area, use hardened styro-foam	Personal Protective Equipment (PPE) Administration

Team management	Disagreement Miscommunication	Discuss in Microsoft Teams Leader takes role Seek for lecturer's advice	Administration
External fee	Exceeding the total allowance	Keep the bill record Share the total external bill	Administration
Final demonstration	Low mark Late testing	Minor individual task adjustment needed Team gathering at the lab	Administration

## 12 COMMUNICATIONS

When we do a group project the communication is very important to consider because it might lead to any chaos like misunderstandings, wrong implementations, etc.... We already knew about these issues. Therefore we wanted to make a proper system to communicate with each other to share the ideas, works, and needed a good platform to argue about the final outputs. So first we had an idea to use zalo app which is very popular in Vietnam. However, we needed more official than the formal chatting apps like WhatsApp, messenger, zalo etc. So finally, we switched to Microsoft teams as our communicational option to make the connection with the other members. We made some restrictions and rules to make the communication easily and effectively. Which are

- No one was allowed to share unwanted contents in the group like funny videos, sarcastic posts
- The standard communication language was set as English
- Every team member was asked to submit their final outputs in teams
- Team members were asked to reply quickly once they get the message in teams.

### The Four Communication Skills

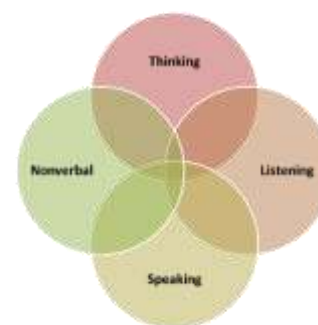


Table 42: Communication skills sets

So above methods were followed by teams' members strictly and they were warned that incase if they failed to follow these rules they would be complained to the lecture.

We arranged at least one meeting per week to develop our ideas and share our works and people were freely let to work according to their working styles. If it was not possible not meet our peers at the campus during the holidays we arranged the meeting at one team member's place.

## 13 POLICIES AND EXPECTATIONS OF THE TEAM

Our team need to tidy up the workspace when done, bring along as many objects as possible to school to get thing ready when needed. Each team member must finish their assigned task. When someone finishes, helps the other. By the end of 25 and 29 of January, we should be able to finish Task 3 and Task 4. In week 12, we must be done with final design. By the end of week 12 and week 13, we must already test all function and prepare to write report after week 13.

## 14 GANTT CHART FOR THE OVERALL PROJECT PROCEDURES



Table 43: Gantt chart for the overall procedures

## 15 DISCUSSION AND OBTAINED RESULTS

The overall project can be considered as a success. All aspects of the project were implemented, and a complete design was made by the end of the project. When considering each task individually the tasks can be summarized according to the following.

The first task was relatively easy to handle thanks to a well build android application and proper coding in the Arduino side. All of the signals received from the android application were interpreted as expected and no problems were faced whatsoever while traversing the grid. The addition of speed control was immensely helpful for the task because it allowed the operator to smoothly travel along the track without the need for brakes. If the user wishes to move faster, then can use the slider to a higher setting and allow the motors to use more power to go faster. The PWM code architecture including the methods and functions within the Arduino performed as expected. The robot did not under any circumstance go into a state where it had to be restarted all over again. The controller design enables quick switching to manual and line following states for easy and quick mode changing. Once the line following mode was enabled the robot would no longer require any operator contact, physical or virtual for the completion of the task. All the other tasks following line following were also carried out autonomously thanks to the automated state switching within the Arduino. To prevent unwanted false positives from triggering undesirable states error checking code was implemented throughout the software framework. The entire software does not rely on one single trigger for it's important state switching functions (such as when it reaches the junction) the software checks an input and then waits a while and checks the input again. If the input has not changed the software will know that it is indeed a true positive. Other techniques such as taking multiple samples before proceeding with the code have also been implemented in both the raspberry pi and the Arduino. This ensures that random errors which are caused by variable uncontrollable factors have less effect on the outcome of the project. One slight issue the team faced, however, during the demo was the function of obstacle detection.

## 16 CONCLUSIONS AND RECOMMENDATIONS

In conclusion, we already apply the algorithm into testing Task 3 while Task 2 is running and find out we where can put the sensor, so it can be work properly. Moreover, we finally develop the basic code to detect the camera. And we finalize the design for our robot.

When we were working on tasks, all tasks helped us to improve both technical knowledge and soft skills. We got to know about some modern different tiny components such as L298N motor driver, HC-05 Bluetooth module, ultrasonic sensors, etc. which made our tasks very much easier to accomplish the required goals successfully. Furthermore, we learned about leadership, time management, team coordination, communication skills, professional writing, etc. At the end of the tasks one and two, we acquired vast knowledge in line following, sensors, developing small apps to control a basic level robot and some real-world problems in vehicles.



The overall completion of the project was done a few days prior to the final trial and final testing was done on the day of the trial (due to lab availability). The project consisted of a large scope covering both hardware and software equally. The task at hand was to develop a fully functioning robot which had the ability of traversing across a field with different modes. Of these modes the first was manual control, the implementation of manual control was done via a custom-built app on Android and communication was done via Bluetooth. This task created challenges in developing code to figure out how send commands and receive commands of the same “language”. The next task was to develop the system such that it could autonomously follow a black line placed on the track. This task proved to be a challenging because upon the normal implementation of the manual control new code had to developed to accommodate for this as well. This is where FSM was considered for the full structure of the robot. By using FSM, it was easy to enable two different mode to run without having another mode conflict with the other. By using global variables and by setting their states accordingly the team was able to implement Finite state machine which not only accommodated for Task 1 & 2 but task 3 & 4 as well. The complete system could run in one single file while minimizing resource usage and computation accordingly. The 3<sup>rd</sup> Task was handled by using a non-contact sensor known as the HC-SR04 which a sensor that has the capability of sending and receiving ultrasonic pulses and then determining how long it took to receive the pulse. By using this information, it was possible to calculate the distance between the robot and any obstacles that lay ahead in the track. This not only proved useful for the autonomous line following but for parking in the relevant garage space as well. By using this sensor, it was possible to avoid obstacles all together no matter what situation the robot was placed even. Even if the robot was driven with the intention of being crashed it would cease to run as soon as the obstacle detection sensed an collision ahead. This mode is useful for safeguarding the surrounding in addition to safeguarding the robot. The final task was overall more challenging than the other tasks because it required knowledge on an entirely new board and a new software programming language (python). However, despite this challenge the team was successfully able to recognize the color of a cube placed and move to the garage. The most notable part about this section is the fact that hard cording was used only for turning (angling) the robot towards the garage. After the garage came in to the camera range the robot would then center itself so it can go inside the garage.

## **17 MEMBER CONTRIBUTION**

### **i. Nguyen Huu Tho**

Have Contributed the manufacturing process for the robot since I’m a local and I know many places that have needed materials. Contributed in the assembling process of the robot. Attempted preliminary testing of ultrasonic sensors of task 3 for developments.

### **ii. Weeramundage Leshan Tiranga Fernando**

Majorly contributed in Mechanical design implementation with Solid works modelling, designing the previous designs for the stages, present basement and cover and measuring each component and organizing the layout of the prototype components on each stage, drilling, managing the allocated spaces for prototype components (power transmitter, switch, Arduino, motor driver etc.), rubric holder, camera mount and other external devices connected for both testing and demonstration evaluations. And also attempted in preliminary testing for the two version of IR sensors (6 channel (custom made from the market) and QTR-5Rc sensors for task 2 for further developments.

### **iii. Alexander Nuwan Amila**

Contributed towards developing the framework necessary to run Raspberry Pi and Arduino in a single Robot including configuring Serial communication. Developed two different versions of PID to run for line following and active color recognition tracking via OpenCV. Additionally, contributed towards developing an Android app for manual control. Implemented multiple error checking mechanisms to prevent false positives while detecting the cube and detecting obstacles. Contributed towards developing a system which uses active processing for maneuvering towards the garage instead of relying on hard coding. Implemented two different versions of color detection using the same camera to observe both the surroundings and cube at the same time.

### **iv. Vang Huynh Minh Tri**



In task 1: I build an android application with bluetooth connection, send direction and change speed to the Aduino. I also write Arduino code suitable for my Android app for the Manual Controlling.

In task 2: I write an Arduino code that follow that path by receive data from 5 sensors. It follows the path by detecting these 5 values.

**v. Mohamed Ibrahim Atheef Mohamed**

I worked in chassis design (not SOLIDWORKS) hardware process the design of the vehicle including drilling cutting etc.. furthermore I did not work in any coding but however I have worked in Bluetooth module and it sensors which means the way to control, connections, limitations and contributed in attaching components with the chassis.

## **18 ACKNOWLEDGEMENTS**

As we have mentioned in the executive summary of this document as well, as a team, we would like to thank Dr. Minh Tran Quang for all the deliverables which were provided to our team and the knowledge shared throughout technical workshops step by step to guide the project procedures of the team in a professional manner with a quality. And for tracking our progress until mid-term to successfully complete all the tasks regarding this project 3A

Furthermore, Mr. Khoa Tran for inspecting us in the demonstrations and advising us to do the demonstrations and trial sessions successfully and updating us with the laboratory rules and procedures throughout the process.

## 19 REFERENCES

- [1] A. Pandit, "Circuit Digest," 12 March 2019. [Online]. Available: <https://circuitdigest.com/microcontroller-projects/arduino-obstacle-avoiding-robot>.
- [2] "Code Electron," 13 January 2020. [Online]. Available: <http://codeelectron.com/measure-distance-ultrasonic-sensor-pi-hc-sr04/>.
- [3] A. Raj, "Circuit Digest," 14 September 2017. [Online]. Available: <https://circuitdigest.com/microcontroller-projects/raspberry-pi-ir-sensor-tutorial>.
- [4] L. Miller, "Learn Robotics," 5 October 2019. [Online]. Available: <https://www.learnrobotics.org/blog/ir-sensor-vs-ultrasonic-sensor/>.
- [5] A. Rosebrock, "PyImageSearch," 16 October 2017. [Online]. Available: <https://www.pyimagesearch.com/2017/10/16/raspberry-pi-deep-learning-object-detection-with-opencv/>.
- [6] D. L. SVB, "Automated Object Sorting Based On Colour Detection," in *2nd International Conference on Science, Technology and Management ICSTM, At Hyderabad, India*, Hyderabad, India, 2017.
- [7] [Online]. Available: <https://sanjosetech.blogspot.com/2013/03/web-cam-streaming-from-raspberry-pi-to.html>.
- [8] [Online]. Available: <https://www.raywenderlich.com/2295-arduino-tutorial-integrating-bluetooth-le-and-ios>.
- [9] [Online]. Available: <https://create.arduino.cc/projecthub/JeremySCook/omni-wheel-robotics-platform-dd48f7>.
- [10] [Online]. Available: <https://www.vexrobotics.com/omni-wheels.html>.
- [11] [Online]. Available: <https://nanonets.com/blog/how-to-easily-detect-objects-with-deep-learning-on-raspberry-pi/>.
- [12] [Online]. Available: <https://www.pyimagesearch.com/2019/04/01/pan-tilt-face-tracking-with-a-raspberry-pi-and-opencv/>.
- [13] [Online]. Available: <https://www.instructables.com/id/ROBOTIC-ARM-Arduino-Controlled/>.

## 20 APPENDICES

### 20.1 Appendix A

Manual handling video link (Task 1 performance video)

<https://drive.google.com/drive/folders/1WmFFcKkvC4tAmDNseJfqLdgLoN-ykraQ>

### 20.2 Appendix B

Line following video link (Task 2 performance video)

<https://drive.google.com/drive/folders/1VZltJiiKlaYx9aoAhg-qzqgHWvKuU2u3>

### 20.3 Appendix C

Final demonstration video link (Full demonstration video for midterm)

<https://drive.google.com/drive/folders/1iZ7oP0JvmDAhe1c44PSMwA1s2WOrwVpp>

### 20.4 Appendix D

Final code for the software solution of the project:

<https://github.com/TeamC-3A/Arduino-Raspberry-Pi-SemiAuto-Robot>

## 21 ATTACHEMNTS

### 21.1 Attachment A

Shareable link for the midterm report and reference to the literature review specifically for task 1 and task 2:

[https://drive.google.com/drive/folders/1h5LzHoed-KWy9oy5HB8RsECeqZEU\\_VS2](https://drive.google.com/drive/folders/1h5LzHoed-KWy9oy5HB8RsECeqZEU_VS2)

### 21.2 Attachment B

Shareable link for the Team folder containing all documentation and support files:

<https://drive.google.com/drive/folders/1mq62L2ntTmrSmABPgAOapxalkQKbRII>

### 21.3 Attachment C

Solid work files for the final assembly:

[https://drive.google.com/drive/folders/1QrxmFnL0HYHOgpQggo2m\\_xx-6cHbko4o](https://drive.google.com/drive/folders/1QrxmFnL0HYHOgpQggo2m_xx-6cHbko4o)

### 21.4 Attachment D

Project Proposal of the team:

<https://drive.google.com/drive/folders/1s-TfhJd1FQCI3hp7F-icINEgM7rVVBYL>

----- The End -----