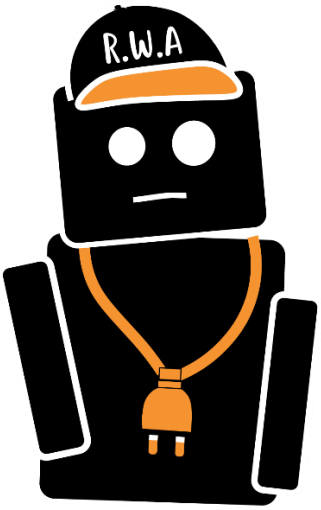
ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

ROBOTZ WITH ATTITUDE

EE494 FINAL REPORT



AFŞİN GÖKBERK PEKER | 2094357 | afsinn.peker@gmail.com
ALİ SEFKAN ULUDAĞ | 2031540 | alisefkan@gmail.com
BARKIN TUNCER | 2031490 | tuncer.barkin@gmail.com
DİLGE HÜMA AYDIN | 2030187 | dilgeaydin@gmail.com
EKİN ALP BİÇER | 2030385 | ekinalpbicer@gmail.com
10/05/2019

Table of Contents

1. EXECUTIVE SUMMARY

In this age of technology, there is little to be kept unautomated. With this in mind, as Robotz with Attitude, we started the project "Vehicles chasing each other around a closed course with varying properties" with request from our customer. This was an adequate challenge for the engineering team and also, we thought that it could be a stepping stone for future generation of automated cars.

Firstly, before going into the solution to the problem or the project deliverables, let's talk about the project itself. In this project, we are tasked with creating a robot that is both fast and agile. The robot will be sprinting around a closed contour, which is not necessarily a circle, but more of an oval shape. The track does not have a specific width, it expands and narrow down depending on the position. It is elevated from the ground by 2 centimeters and the standard committee has decided that it is going to be made of Styrofoam and its going have a green cardboard cover on it. The track also can have some small obstacles and the robot should be able to overcome them. The robot should autonomously track the road and decide on which direction to face in order not to fall off the course. Another very important part of the task is speed. There will be another robot on the other side of the road that we need to catch. In order to catch it, we need to be faster than them. Being faster than the opponent is not enough in our project. The opponent also needs to acknowledge that it's been caught (or, although it is very unlikely, our robot also needs to acknowledge that it has been caught). In order to do this, there will be a handshake protocol. This handshake has been standardized and it will take place over TCP/IP communication protocol. The robots will initiate a peer to peer handshake protocol when one robot reaches 5 cm behind the other robot, and both robots will stop.

Now that we have talked about the problem, let's talk about our solutions. Firstly, we are detecting the road using image processing via a camera. The camera is sending the images to raspberry pi, which is the brains of our operation. Raspberry pi detects the road and decides how much steering and speed the vehicle needs. After the decision, Raspberry sends the orders to Arduino. Arduino is in charge of the motor control. We decided on this to reduce the workload of raspberry pi to have more frames per second on the image processing part. Arduino is connected to a L298N motor driver and we are using micro motors. In order to figure out the distance between the robot in front or back of ours, we are using ultrasound distance sensors. Finally, the whole system is powered from Li-Po batteries.

When we look at our robot now, it is fully operational. All of the subparts have been integrated, and work together nicely. Our robot can reliably do clockwise or counterclockwise circuiting in less than 10 seconds. The robot does not need to be rebooted to switch from clockwise direction to counter-clockwise and the switching can be done on the go. Our robot also

can reliably create a TCP connection with other robots to communicate with them to simultaneously stop.

The Robots with Attitude team has the right engineers that gets the job done. Our team consists of a computer specialist, a power electronics expert and three control engineering professionals. This dream team has all the skills(image processing, mechanical design, programming languages, project management and many more) required to form a robot from ground and get it to sweep the competition.

The estimated cost for the project is 200 dollars, and project was completed in total of seven months.

2. INTRODUCTION

Automatization is becoming larger and larger in our daily lives. As the time passes, we see more and more automated objects around us. Next stop in the automatization train is vehicles, and we are here to make it happen.

As the Robotz with Attitude team, we decided to take the project “Vehicles chasing each other around a closed course with varying properties” since we got a request from a customer and we decided that we were up to the challenge.

The race will take place on a round track with changing width and some minor obstacles. The track will be lifted from ground about two centimeters and it will be made from Styrofoam. Our robot is going to keep track of the road and steer, accordingly, go as fast as possible and when it catches the opponent or gets caught, perform a handshake protocol with the opponent when in 5 cm proximity to simultaneously stop.

In order to be successful in the project there are some bare minimums, and there are some goals. The bare minimum is not to fall off the track and not hit the opponent. The goal is to be faster than the opponents and tag them.

Our final product can achieve the bare minimums stated above and we are confident in our design to be one of the top contenders in the races that will come.

3. DESIGN DESCRIPTION

3.1 TOP MODULE

In the system, top module is responsible for the management of the submodules, processing the data gathered by the submodules and communication with the Arduino. Block diagram of the overall system can be seen in Figure xx1.

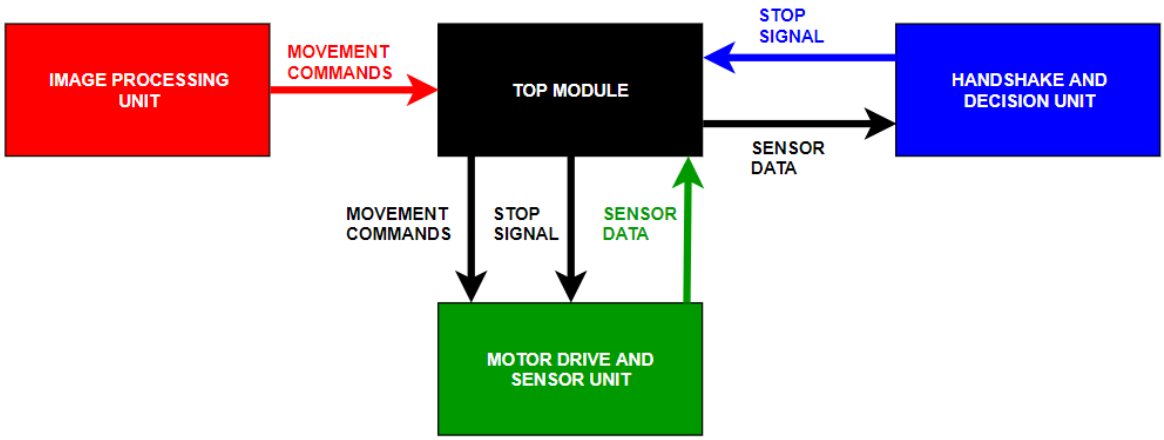


Figure 1: Block diagram of the overall system

As can be seen from Figure 1 above, top module mostly acts as a bridge between the submodules. It forwards the movement commands sent from the image processing unit to motor drive and sensor unit. Similarly, sensor data gathered by the ultrasonic distance sensors on Arduino is forwarded to the handshake and decision unit through the top module. If handshake is successful, stop signal from the handshake unit is also forwarded through the top module to the motor drive and sensor unit. A flowchart of the overall system can be found in Figure xx2.

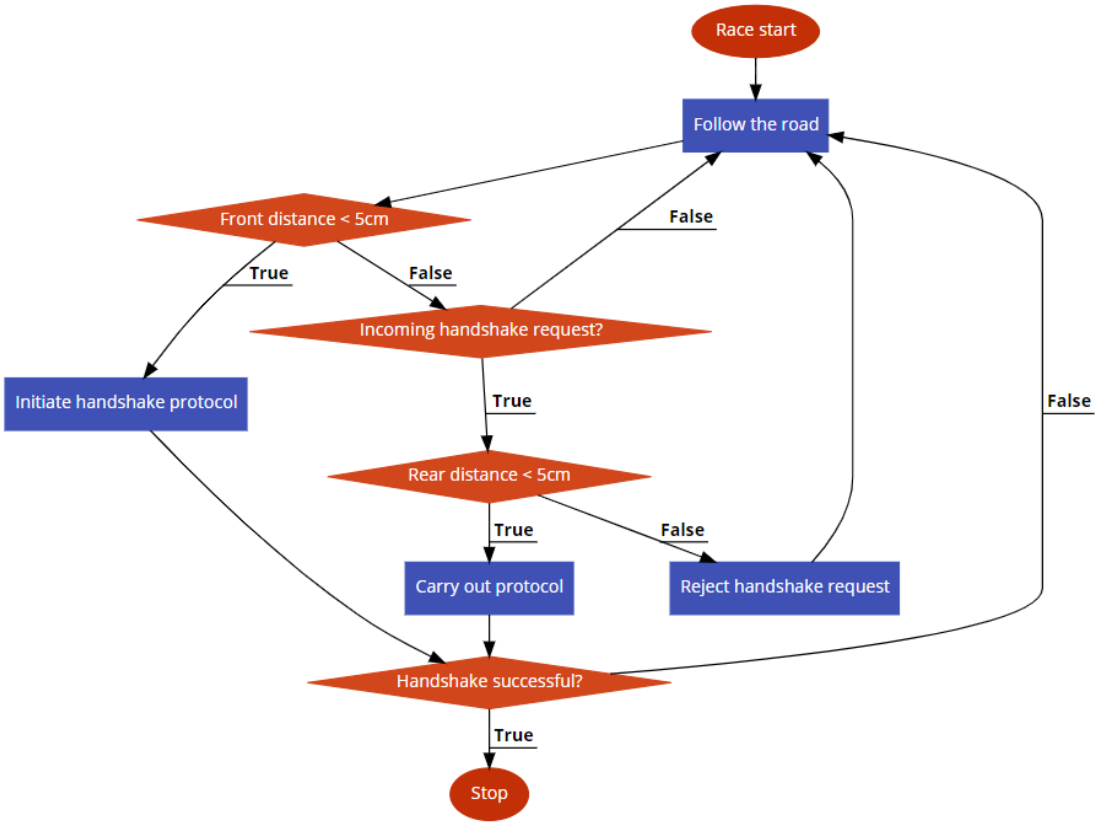


Figure 2: The flowchart of the overall system

The top module oversees the operation of 3 submodules in the system, namely, image processing unit, handshake and decision unit and motor drive and sensor unit. A summary of these submodules' functionality will be provided here. The details of these submodules will be elaborated in the following sections of the report.

Image processing unit is responsible for road detection and generation of the movement commands. It utilizes the camera of the Raspberry Pi3 to detect the road and generates 2 lines on the image, each of them defined by 3 points. Depending on the position of these points, movement commands are issued to the motor drive and sensor unit.

Handshake and decision unit is responsible for the handshake process when a vehicle catches up to the other one. It is continuously fed by the sensor data from the Arduino. If the front sensor reads a distance of less than 5 cm, this submodule initiates the handshake protocol that is defined in the standards. If the connection with the other robot is completed successfully, a stop signal to the motors will be sent from this submodule. This submodule also checks the incoming handshake requests. If a handshake is requested by the other robot, rear sensors will be checked to confirm that the robot is indeed in the 5 cm range, depending on the result of the check, either the handshake protocol is carried out or the request is simply denied, and the normal operation continues.

Motor drive and sensor unit is responsible for the movement of the robot and acquisition of the sensor data. This submodule resides on the Arduino. The sensor readings will be passed to the handshake and decision unit. Depending on the movement commands that are created by the image processing unit and handshake and decision unit, motors will be controlled within this submodule.

3.2 IMAGE PROCESSING SUBSYSTEM

In this product, image processing is utilized in order to detect the path and to orient the robot while co-operating with the motor driving Arduino. The goal is meant to be achieved with a Raspberry Pi camera and a Raspberry Pi. The product standards related with this subsystem are given below.

- The path will be of a cardboard with the color of dark green which has the code "Color 120/Yeşil/UCA-113".
- The cardboard will be placed on a 2 cm foam board which has no standards regarding its color.
- The background can be arbitrary.
- There is not any standard regarding the lightning conditions.
- The background color is not specified.
- The lightning conditions are not specified.
- The robot should be able to finish the parkour in 20 seconds and if possible, catch the other robot.

3.2.1 Solution of Image Processing Problem

Since the background color is not specified it is not straightforward to detect the path by subtracting two different colors, namely the path and the background, from the image captured by the camera. Hence, a color detection algorithm is needed. However, most of the color detection algorithms, which are done in RGB color space, lack of robustness when the lightning conditions are not specified. Therefore, a color detection algorithm which can differentiate the color from the given frame while being able to handle the abrupt changes in the illumination is highly demanded. A common approach for this problem would be using the HSV color space. HSV color space consists of three bands, as in RGB color space. The 'H' indicates the space of 'Hue'. Hue basically represents the color as a range from 0 to 180, in OpenCV. The range goes as red, yellow, green, cyan, blue, purple and red again. The 'S' in HSV represents the saturation value of the color. With the help of this feature it is feasible to have an answer to the lightning problem. The saturation value goes from 0 to 255. The 0 value indicates the brightest while 255 represents the darkest. Lastly, 'V' indicates the value of the color. Value is the maximum value among the

red, green and blue of a specific color. The HSV color space is calculated as in Figure 3, then scaled according to the data type of the image, i.e. 8-bit, 16-bit, etc.

$$V \leftarrow max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - min(R, G, B)) & \text{if } V = B \end{cases}$$

Figure 3: The calculation of HSV color space

Using HSV color space, masks can be generated, and these masks would be able to detect colors with varying illuminations. Also, this approach is fast which is needed since the robot is intended to finish the parkour in less than 20 seconds. Considering this, in our solution we started with masking the frame with using HSV color space.

After several trials including different light conditions, the region that is desired to be subtracted from the frame is decided to be between (48,120,40) and (92,255,255) in HSV color space. The image after applying the HSV mask will be referred as the mask image from now on. Some of the results for color detection can be seen in Figure 4, in which the meaning of the horizontal lines will be explained later in this chapter.

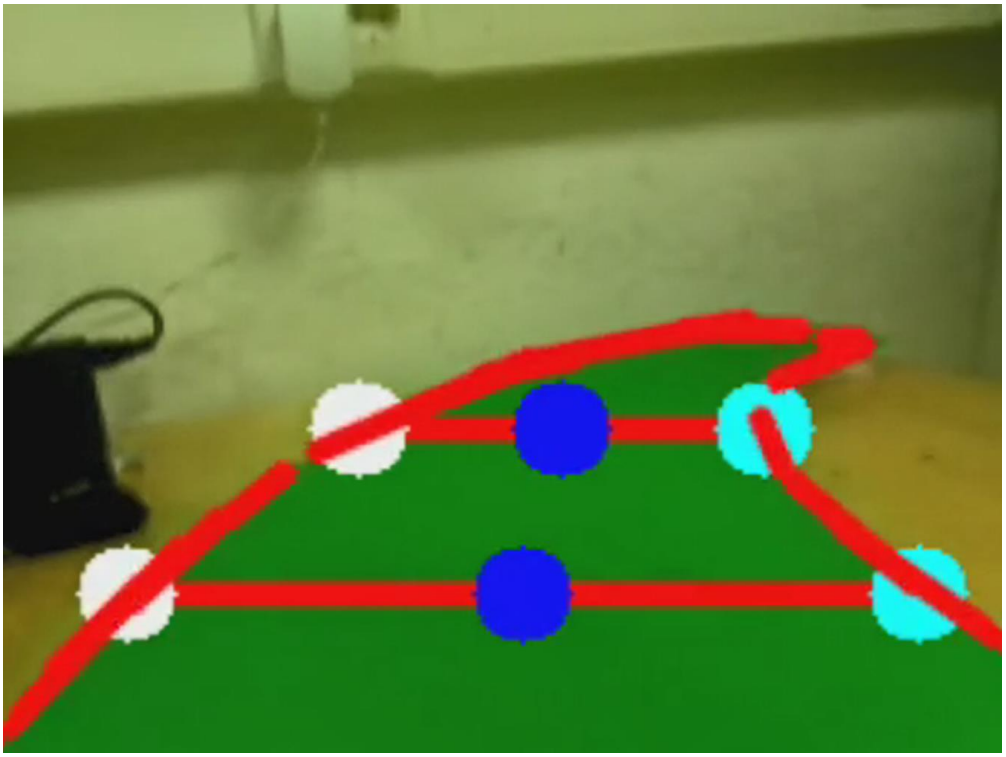


Figure 4: The result of the color detection algorithm.

In addition to color detection, to have a more robust performance a template matching scheme is added to the system. For the template choice, a part of the path is chosen, Figure 5.

Figure 5: The template

The working procedure of the template matching algorithm is that a predefined template is convolved with the image and the result is filtered with a threshold. The parts that are highly correlated with the template result in higher value and the remaining parts result in lower value. Hence by thresholding one can obtain the object he/she is looking for. This process is slower than color detection. However, we used the mask that is generated earlier on the template as well, and the new template is convolved with the mask image hence we decreased the complexity by x3 times since now there is only 1 channel for color. By the help of template matching now the disturbances are eliminated, however the overall complexity of our solution is increased. Therefore, to reduce the process time and increase the frame per second process we implemented template matching algorithm and color detection algorithm as a multiprocessing process. To be more specific, the frame is obtained by the camera and put in a queue for our multiprocessing algorithm. Whenever there is a frame in the queue color detection(mask) and template matching is applied to it and the resulting frame is put to another queue, while the

camera is capturing another frame. At this point, we have a nice representation of the path and it can be seen in Figure 6

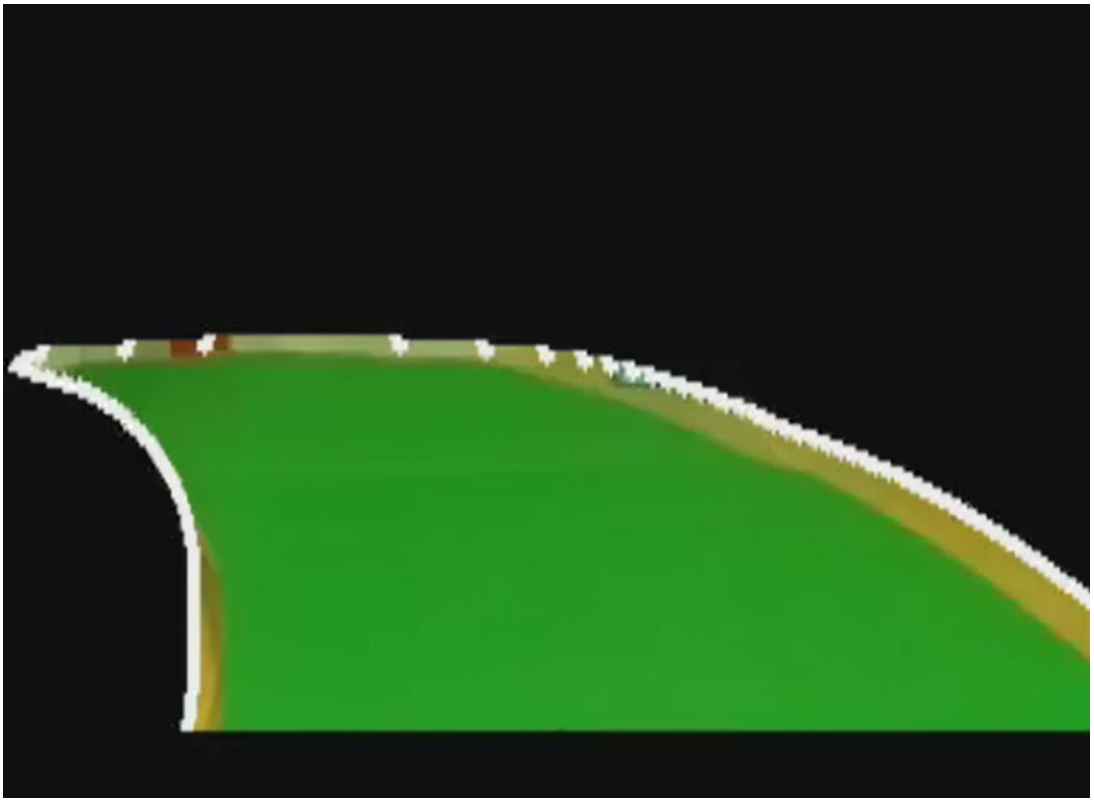


Figure 6: The result of template matching and color detection combined

After eliminating the surroundings completely, we have defined two-line locations. For test purposes, we defined the first line position to be the 125^{th} pixel and the second line as 90^{th} pixel. The pixels having these line locations as y value is considered as the eyes of the robot, one is looking further, and one is looking slightly closer to the robot. The points there the crossing, from background to path, is decided on these two lines and marked with white and cyan, while the mid-point is described with a blue circle. A resulting frame with the lines and circles generated after color detection and template matching and drawn on the camera view can be seen in Figure 7.

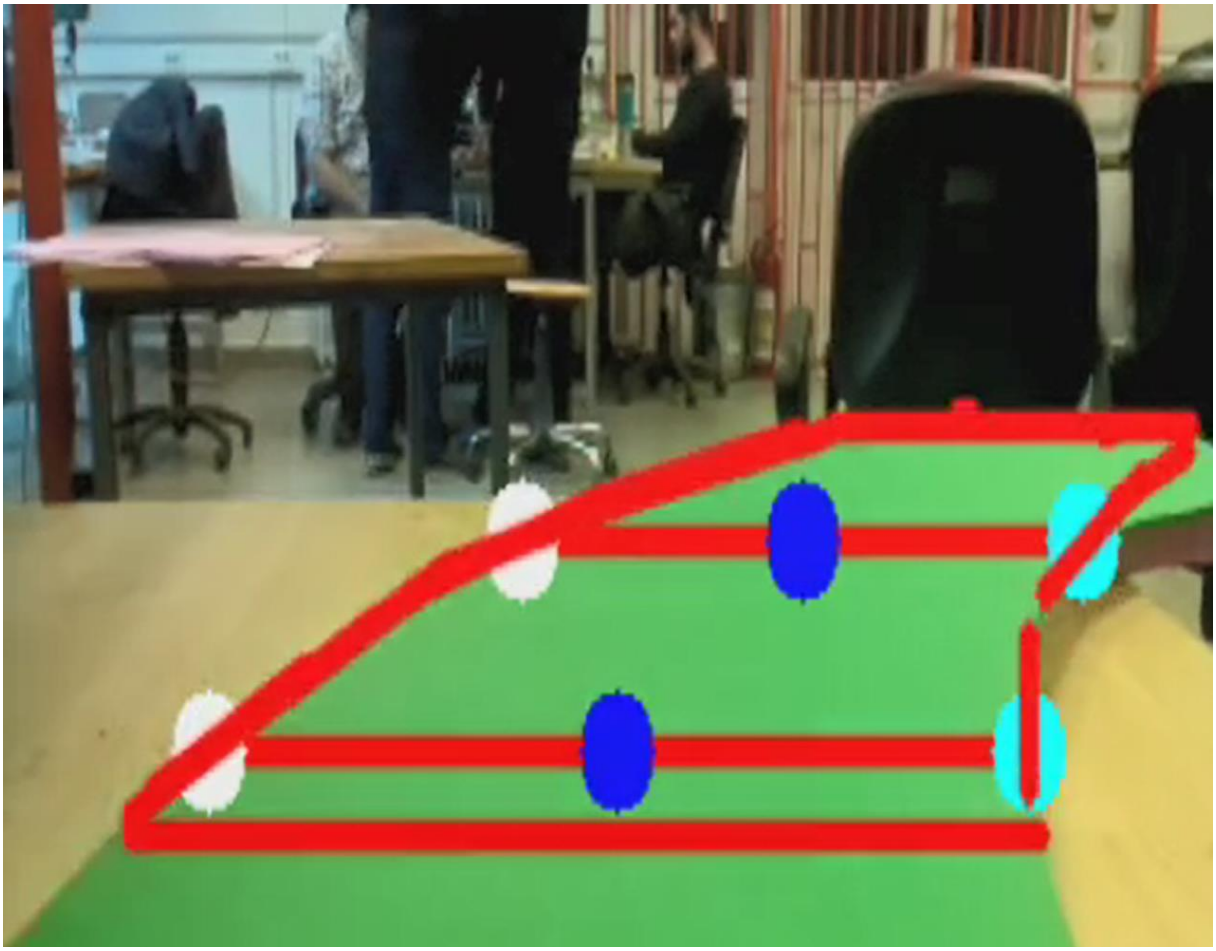


Figure 7: The resulting view of the camera

The mid-points are intended to be used in navigation and to regulate the speed of the robot. The left and right points determine if the robot has the path in its view as properly. If the points are at extremes, i.e. at first or last pixel points, the robot must rotate accordingly to take the path in its full view.

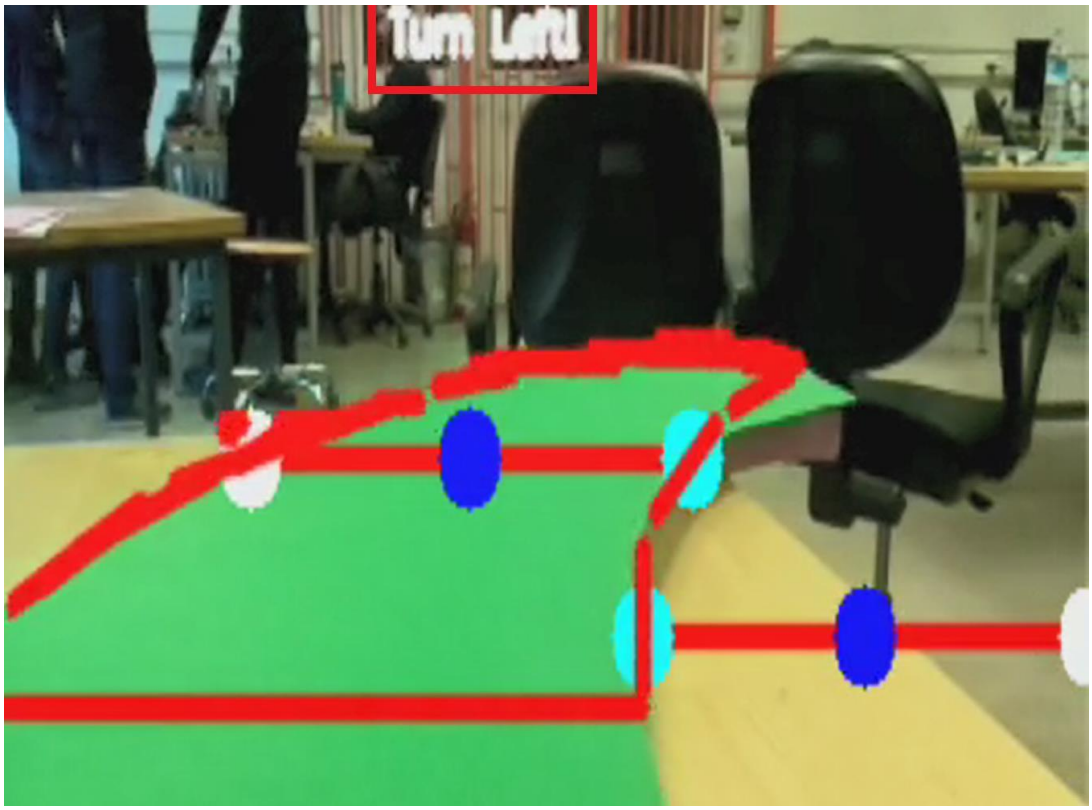


Figure 8: Robot has to turn left to have the path in its sight

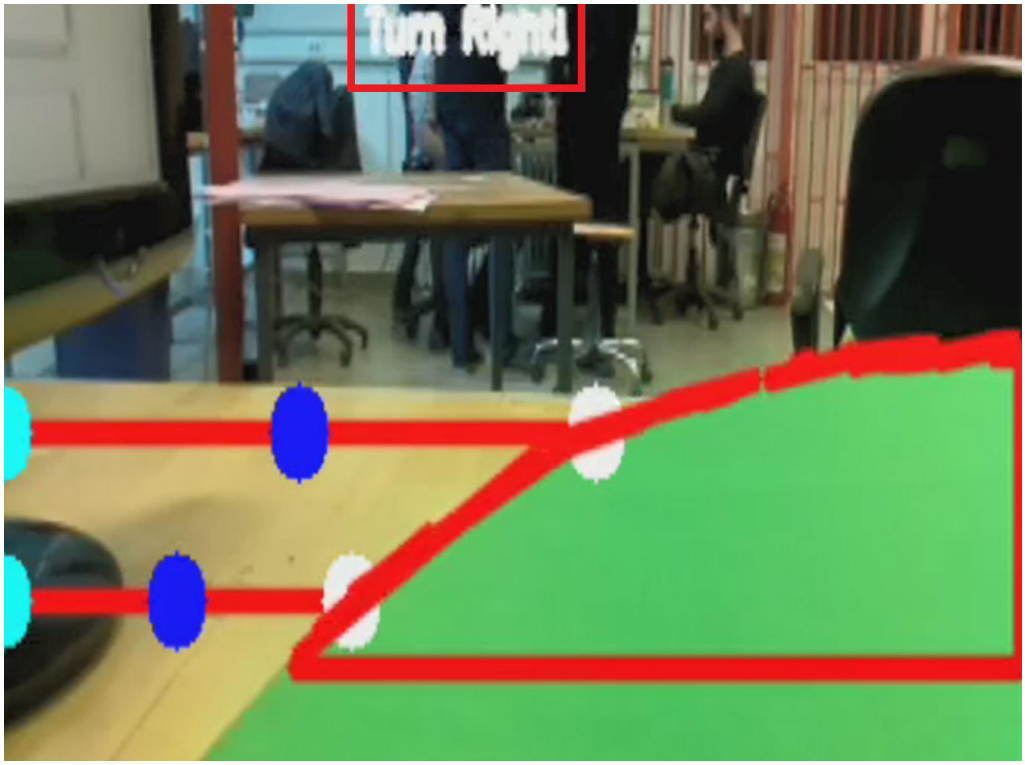


Figure 9: Robot must turn right to have the path in its sight

Although we do not expect the robot to be out of alignments so much that both measurements are taken at the border of the view, we cannot ignore such cases since if faced, they will result in robot falling out of the path. So, the movement algorithm first checks if any of the measurements are taken at the border, then proceeds to calculate the speed of the motor to move and navigate the robot.

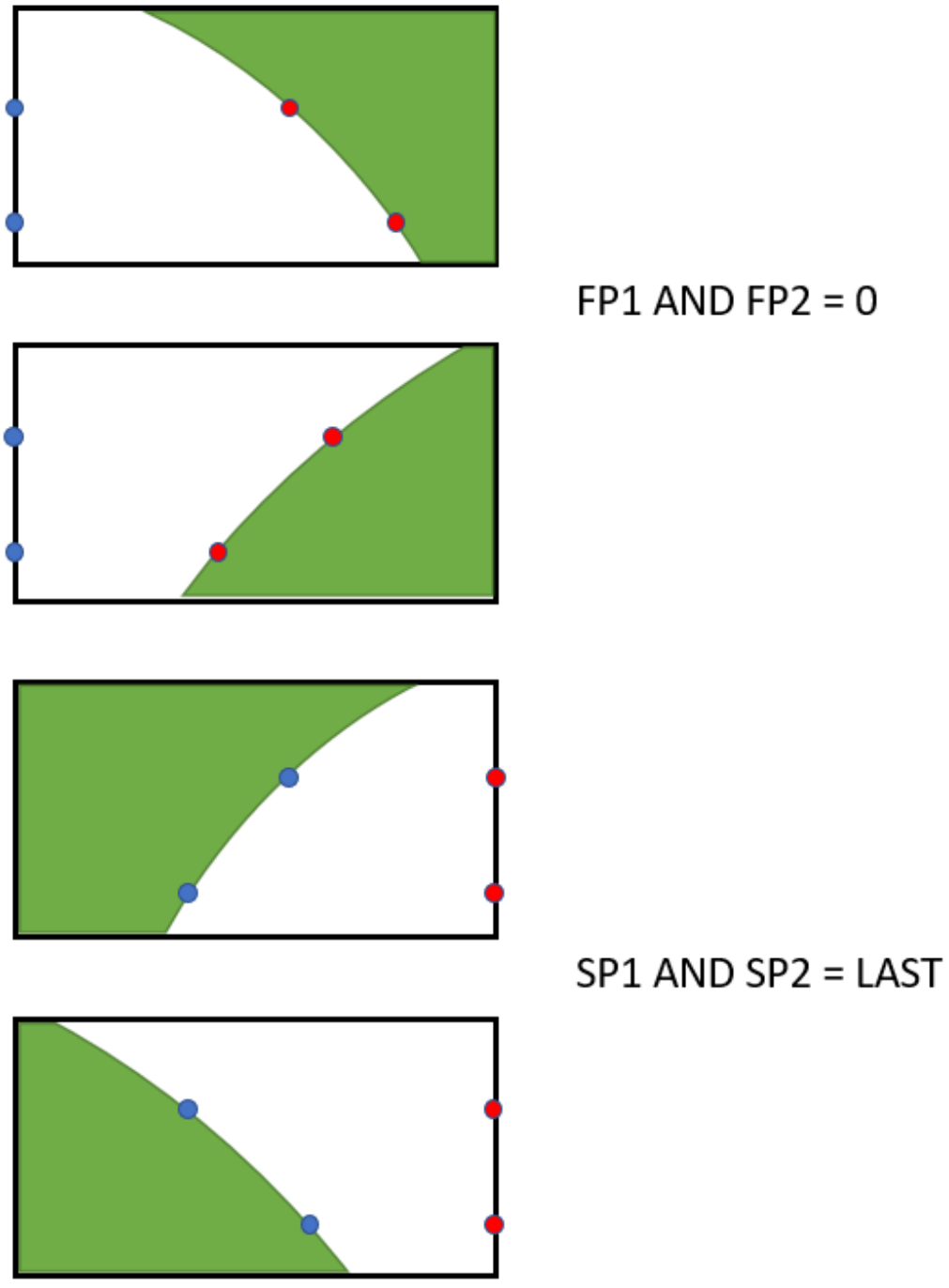


Figure 10: Illustrations of extreme misalignment

Figure 10 explains the view of the camera when both of the measurements are taken at either left or right border. Blue dots represent first points (FP1 and FP2) and red dots represents the second points (SP1 and SP2) and number 1 is given to the points closer to the robot. If measurements are taken at the right border, in other words second points are at the last pixels of the corresponding rows, then the robot stops and starts to rotate left until other edge of the road is in the frame. Similarly, if both measurements are taken at the left border and first points are at the first pixels of their rows, the robot stops and rotates right.

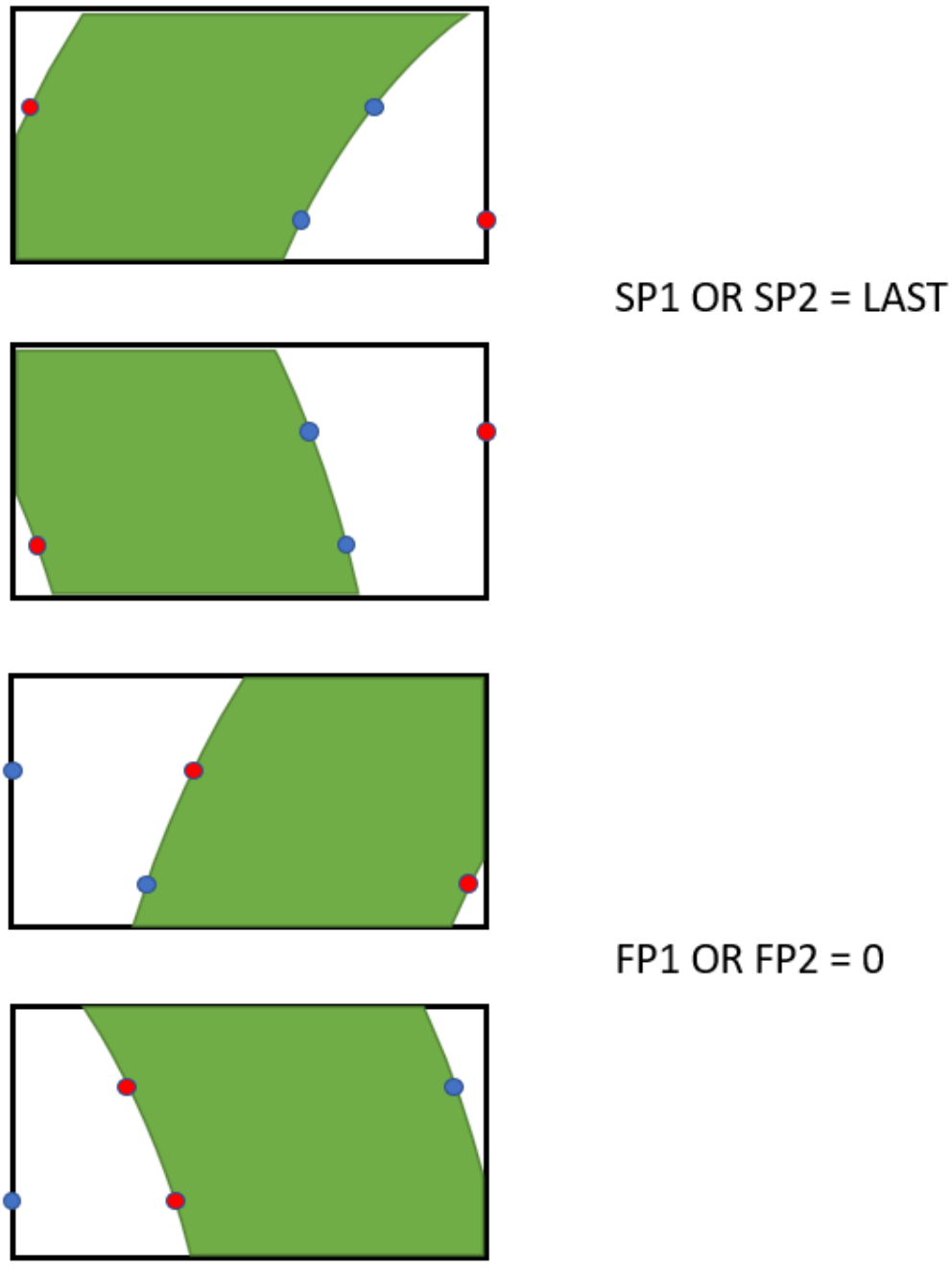


Figure 11: Illustration of slight misalignment

Since the road gets very tight at some points, we might expect only one of the measurements taken at the border occasionally. Figure 11 illustrates the cases when the robot is slightly misaligned and only one point is at the border of the view. When one of the second points is placed at the rightmost pixel, the robot slows down and turns left until the robot is aligned properly again. In addition, when one of the first points is at the left border, the robot slows down

and turns right. When proper alignment is reached, the robot continues its routine to calculate the individual speeds of the motors according to the mid points.

The difference of the mid-points will work as a factor when deciding the speed of the robot. If the difference between mid-points are less that means the robot is on the straight part of the path and hence it can go faster. On the other hand, if the difference is larger the robot will make a turn soon so, it must go slower. The overall image processing procedure works approximately with 20-25 frame per second. A diagram indicating the algorithm of the image processing subsystem is given in Figure 12. The test procedures, test results and the alternative solutions are given in the next chapter.

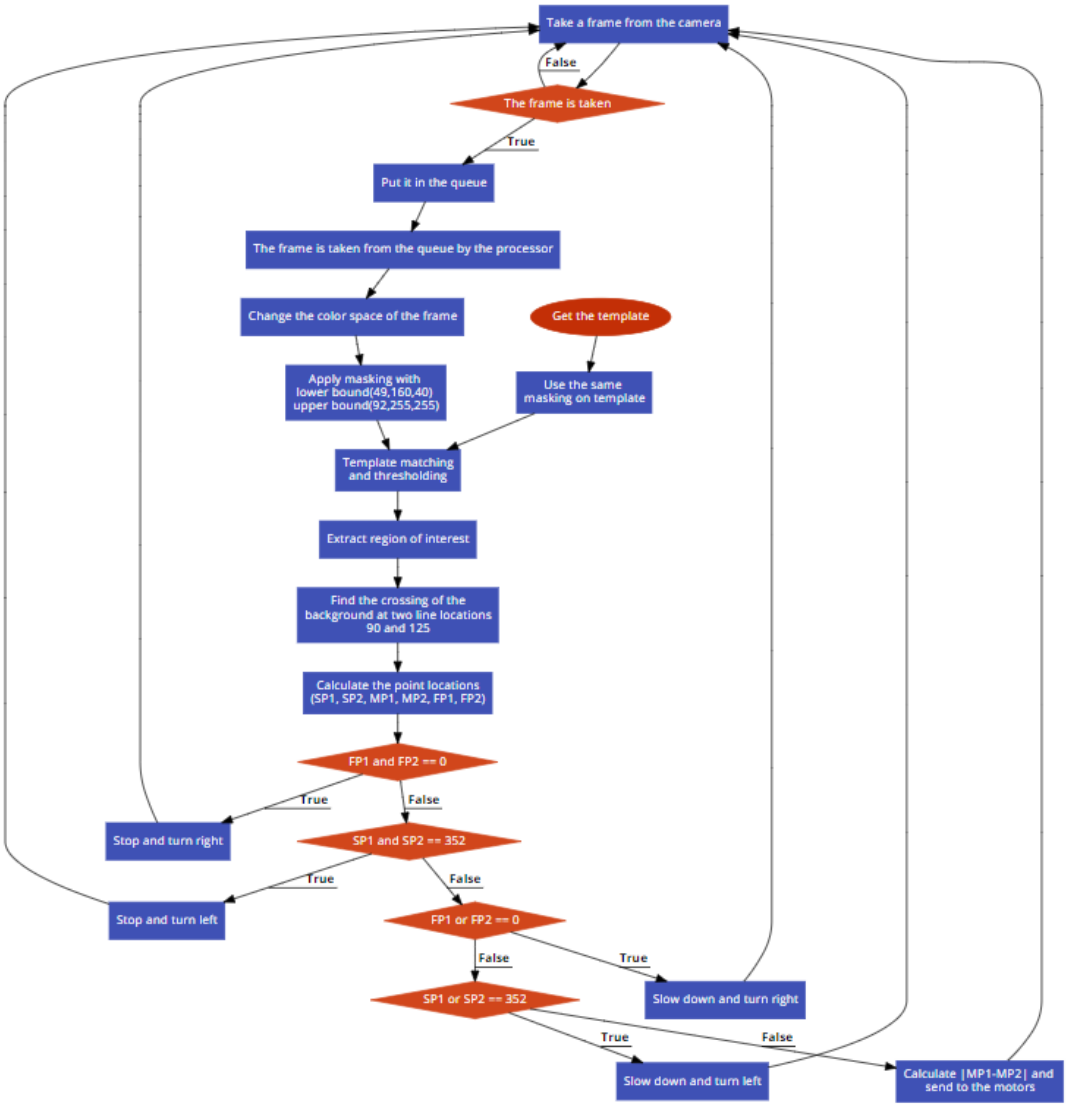


Figure 12: The flow of the image processing algorithm

3.3 Handshake Subpart

The race ends after both of the robots acknowledges the distance between the robots. The agreement is done after the participants send positive message to each other through a peer-to-peer network. Ad Hoc network is utilized to provide a peer-to-peer easy to maintain network. The connection is established with TCP/IP protocols. Raspberry Pi is used as a server or client during the race, which is decided beforehand. The socket pair is obtained just before the race begins and is maintained until the race finishes. While implementing the communication code, 'socket' package in Python3 is used.

First of all, a mutual Ad Hoc network is established between the racing robots to provide a communication channel. Ad Hoc network is chosen since the standard states that the communication should be done through a peer-to-peer wireless channel. The Wi-Fi interface of the Raspberry Pi is set to Ad-Hoc mode during boot. The on-board Wi-Fi adapter, which is shown in Figure 17 of Raspberry Pi can act as an access point for other devices as well. Using this feature, one of the robots 'hosts' the network while the other one connects to it. However, it should be noticed that the connecting robot can maintain the network even if the host robot disconnects provided that the device establishing the connection can serve a wireless network as well. The infrastructure-less configuration of Ad Hoc network enables each user on the network to act as host. This characteristic of Ad Hoc networks provides a peer-to-peer connection. Therefore wireless Ad Hoc network is chosen as the channel.

Figure 12: On-board Wi-Fi Adapter of Raspberry Pi 3

Secondly, a socket pair for receiving and sending data is obtained. The used communication protocol is TCP/IP. TCP/IP defines the way data is transferred through the network by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination. TCP/IP requires little central management, and it is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network. TCP (Transmission Control Protocol) specifies the procedure of creation of channel of communication across a channel. TCP connection is established with a 3 way handshake as illustrated in the Figure 18. This protocol also manages assembling the data to be sent into smaller packets before transmission over the network and reassembled in the right order at the destination address.

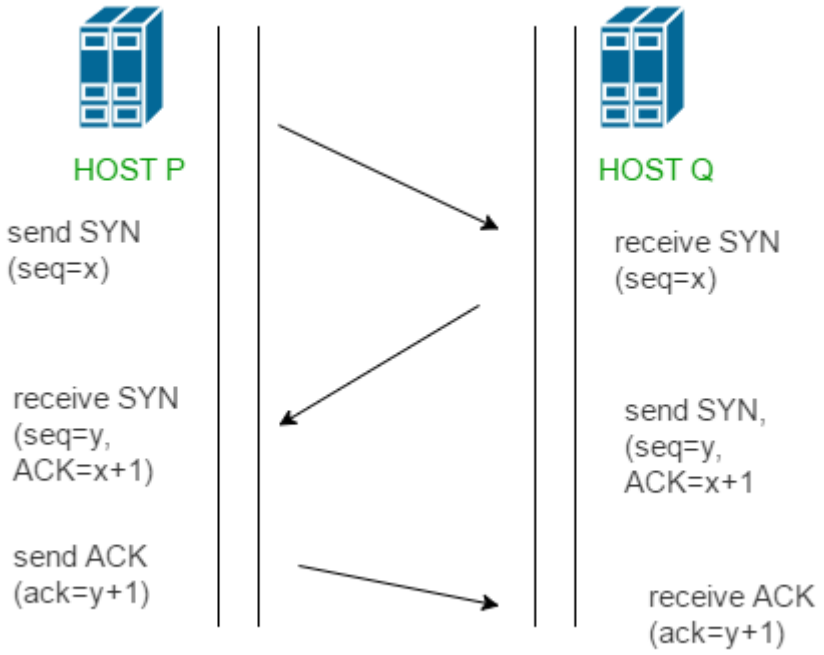


Figure 13: Illustration of 3 Way Handshake

The TCP/IP connection is established by Raspberry Pi using 'socket' package in Python3. The routine for establishing connection and sending and receiving data is shown in Figure 19. It should be noted that every robot is assigned a static IP. The IP numbers are 192.168.1.ID where 'ID' is each groups ID number. Firstly, a socket object is created and bound to a port by the server for client to connect. This machine becomes the host. The host starts to listen to the port for incoming connection requests. On the client side also a socket object is made indicating the socket to be connected and the IP of the host. Then a connection request is made and accepted by the host, resulting in the establishment of TCP connection. Since TCP provides a full-duplex communication, it does not matter which robot is the host after the connection is made. Both

robots should check if there is a message in the buffer to be received and send the necessary replies.

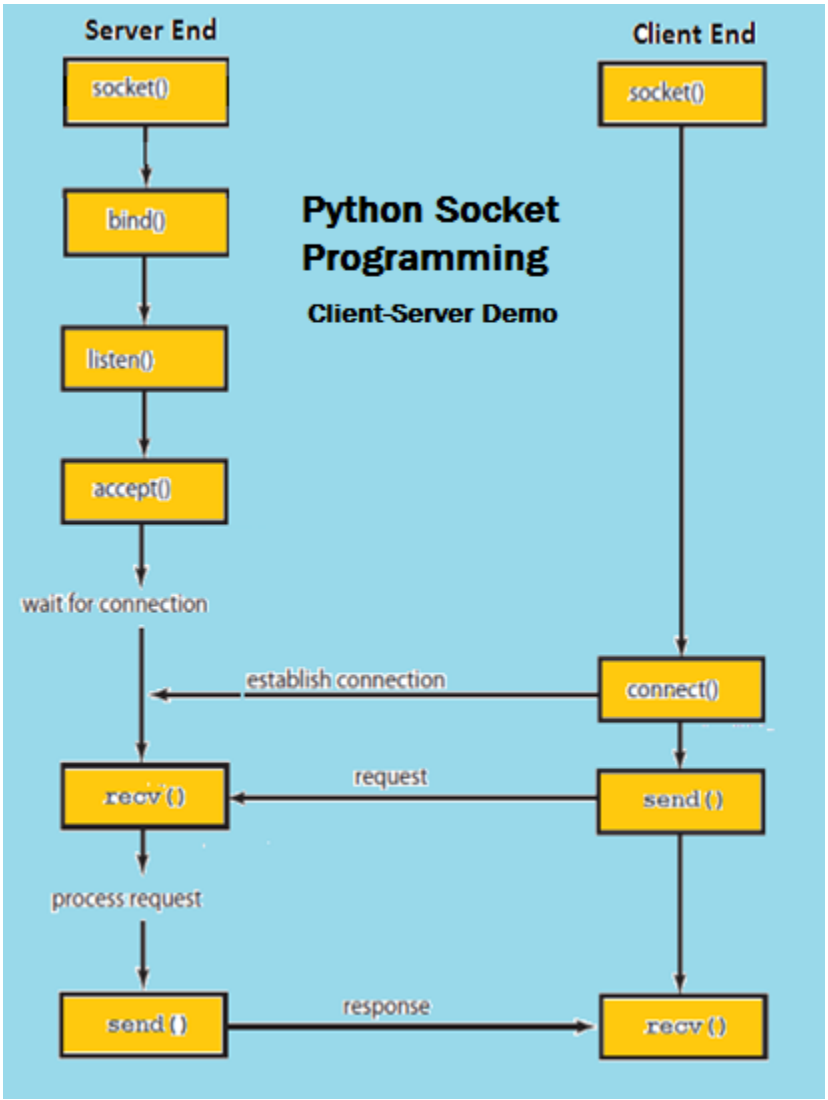


Figure 14: Python3 TCP Connection Routine

The sent messages depend on the position of the robots with respect to each other. These are defined by the standards. The messages to be sent and their meaning are given in Figure 20. Both of the robots should check the buffer periodically if 'ID00' message is waiting in the buffer. This means that the robot was caught by the other one. When this message is received, the rear sensor value is checked to verify this statement. If the statement is true, then 'ID01' message is sent back to the other robot. The race finishes with robots signaling to stop. If the statement is false, then 'ID11' message is sent and the race continues. This communication scheme is very alike to the 3 way handshake method.

First two characters	Last two characters	Respective LED light	Meaning
ID of the sender	00	Red	The sender has caught the other robot
ID of the sender	01	Green	The sender acknowledges that it has been caught
ID of the sender	10	Blue	The sender signals to stop
ID of the sender	11	Yellow	The sender does not acknowledge that it has been caught

Figure 15: Messages to be Sent and Their Meanings with their Corresponding LED Color

3.4 Motor Drive Subsystem

This subsystem is essential in order to control the lane keeping of the robot and includes the Arduino, motor driver and the motors. Due to the standards and the project requirements, some constraints were taken into account when designing:

- Because the thinnest part of the course is 15 centimeters, the motors had to fit in that size; therefore, both for simplicity and requirements, DC micro motors were used.
- The course needs to be finished before 20 seconds, so the speed of the robot had to be arranged to meet this requirement. As Pololu 4.2 cm wide wheels were used and since the path is approximately 293.2 centimeters, the lowest average RPM was calculated as 67. In this product, the covenanted time to complete the course is 11 seconds, therefore the average RPM becomes 120. Corresponding torque calculations are made as follows considering the robot moving at average speed stops within half a second:

$$Force = Mass * Acceleration$$

$$F_{total} = 1kg * 0.5m/s^2 = 0.5\ N$$

$$F_{onemotor} = F_{total}/2 = 0.25N$$

$$Torque = Force * Distance$$

$$Torque = 0.25N * 2.1cm = 0.525\ N.cm$$

One pair of Pololu #2214 (100:1 Micro Metal Gearmotor HP 6V) motors are used to implement this subsystem as it was found out that this model supplies the desired values. The technical drawing and performance specifications of the motor can be observed in Figures 21 and 22 respectively.

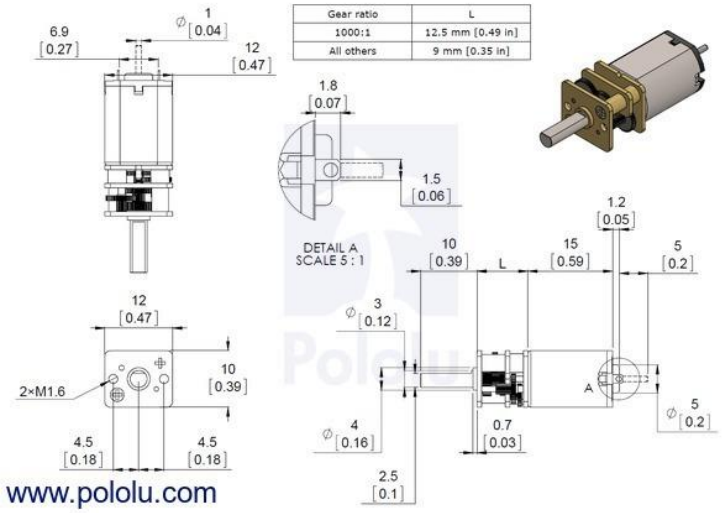


Figure 21 Technical drawing of Pololu #2214

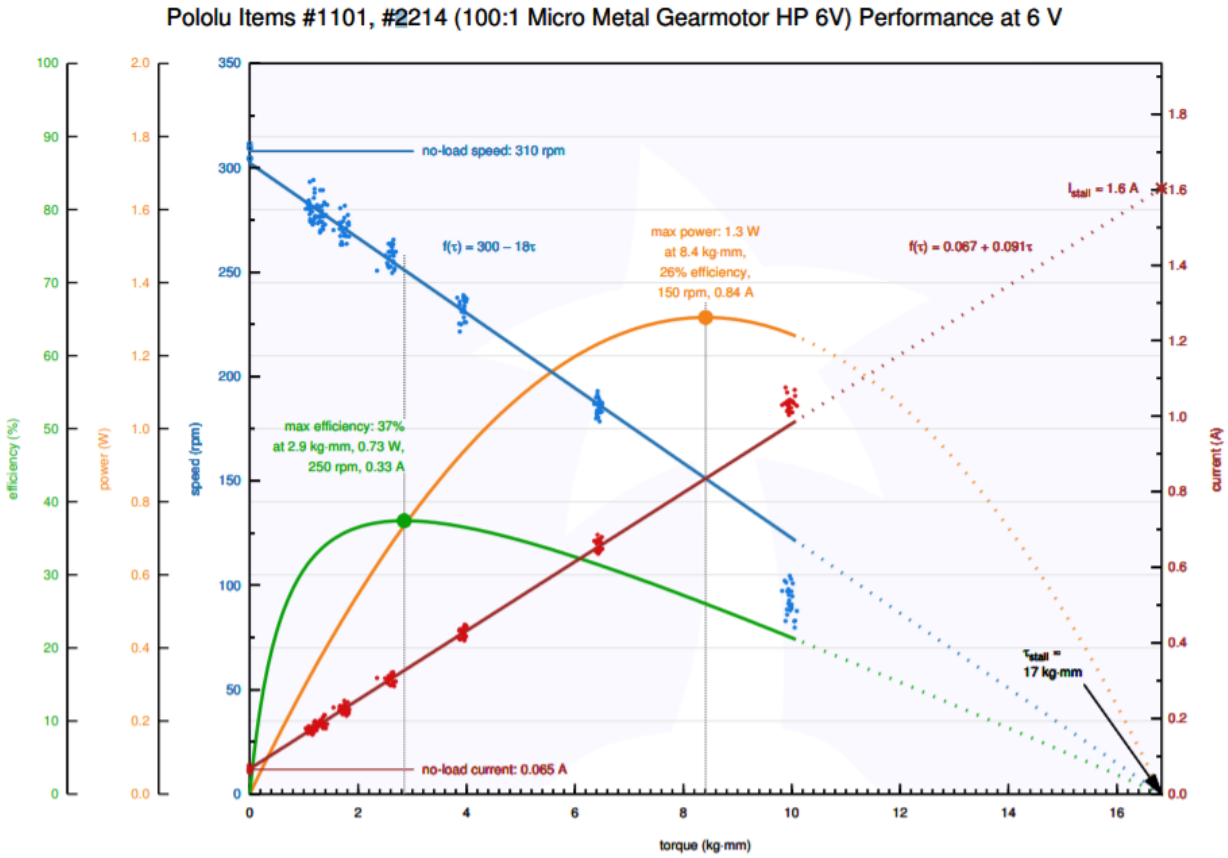


Figure 16 Performance specifications of Pololu #2214

As can be observed from the Figure 22, the motor has approximately 0.9 kg.cm torque (which is almost twice of what was needed) at 120 RPM; however, since the speed is higher at some parts of the course, higher torque is needed to stop the robot, so this model satisfies this condition as well.

- As the used motors operate under 6 volts and has a stall torque of 1.6 amperes, corresponding motor driver was chosen to meet these values. L298N Dual Bridge Motor Driven was used both for its simplicity and low cost and also competence with the robot. This motor driver can be seen in Figure 23.

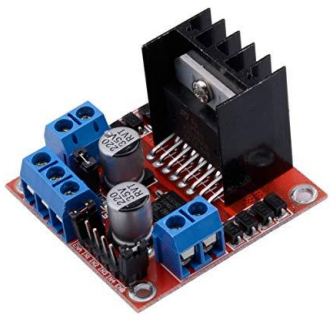


Figure 23 L298N motor driver

3.4.1 Implementation of the Motor Drive

The circuit diagram of this subsystem can be investigated in Figure 24.

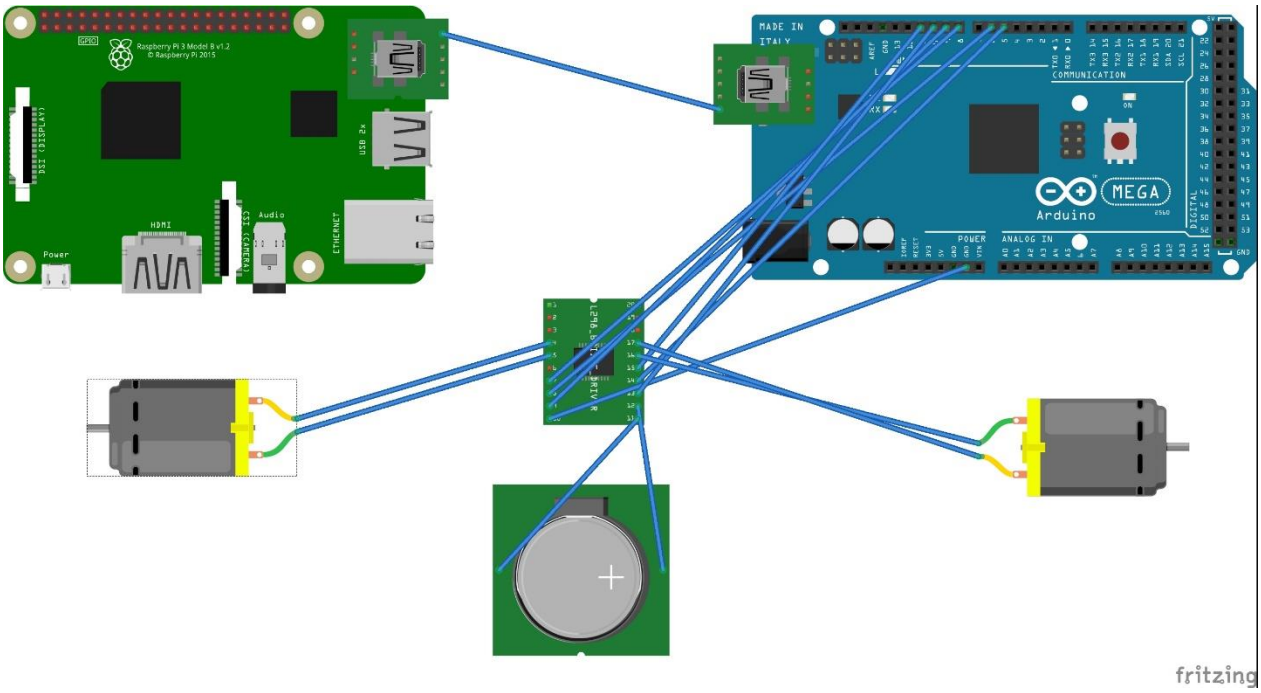


Figure 2417 Circuit diagram for the motor driver subsystem

As can be studied from Figure 24, Raspberry Pi communicates with the Arduino with serial connection, transferring the image processing data obtained from Image Processing subsystem. This data is processed in the Arduino and some states are defined to work with the incoming data. According to these states, some speed values are calculated as outputs within the Arduino and these values are sent to the control system as the input which is also processed in Arduino. The speeds of the motors are measured with encoders and this data is sent as the negative feedback to the control system where the input is the previously calculated speed values. The simple PI controller acts fast to reach the set point by increasing or decreasing the PWM of the motors and the desired speed for each motor is met. These PWM values are sent to the motor driver and both motors are driven with this method.

A state machine approach was defined in order to deal with the different inputs coming from the Raspberry Pi. There are states 1,3,5 and 19-21 to turn left with different approaches and 0,2,4 and 18-20 to turn right. If the middle of the path where the wideness is the highest, the states are followed as 5, 3 and 1 where 5 is the closest to going straight and 1 is the closest to a sharp

turn. These states make the right motor turn faster than the left motor, therefore making a left turn. The same is for the right as 5 corresponds to 4, 3 to 2 and 1 to 0. States 18,19,20 and 21 are defined to center the robot like an emergency call. As soon as it is detected that the robot is close to leaving the path, these states are called and the robot is made to go to the center of the path. What is different with these states are that one motor is turned in the other direction than the other motor, which causes the robot to turn around itself. The over schematic of this subsystem can be observed in Figure 25.

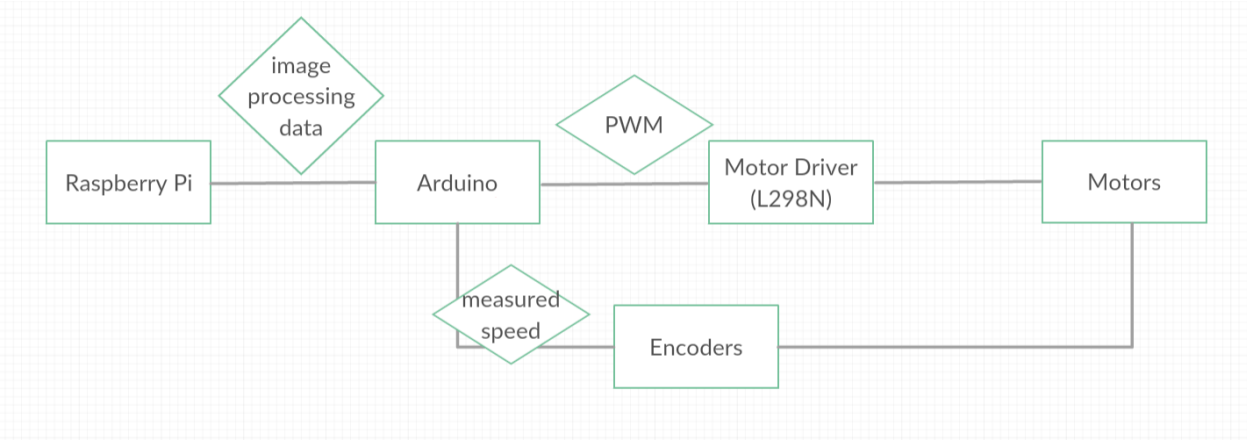


Figure 185 Overall schematic of the Motor Driver Subsystem

4. RESULTS AND ANALYSIS OF PERFORMANCE TESTS

Since Robotz with Attitude team has carefully designed and implemented the subparts, when the subparts connected, they work as intended. Their well design is also can be seen in performance. Our Robot is quite robust towards the disturbances in the road, changes in light and external force.

Our robot can circuit the track in 10 seconds, and can either push away or drive over some small obstacles like ruler, rice, pencils and even a watch. We also did some tests that is mentioned below.

4.1 Test Procedures

We designed several test procedures, including different light conditions and objects on the path as disturbances.

- In the first test we applied heavy illumination and recorded the response of the algorithm. Also, we put an object on the path as a disturbance.
- In the second test we made a big shadow on the path via putting an opaque cardboard on both the camera and the path simultaneously.
- The third test includes an object put on the path in regular conditions. In this test we don’t want this object to interrupt anything in our solution.

4.2 Test Results

In the first test scenario, our solution proved its robustness which can be seen in Figure 26. As it can be seen from Figure 26 the illumination is too large so the color of the path can be considered as cyan rather than dark green even if it's the same cardboard with the earlier figures. Also, the algorithm is capable of not detecting the surrounding green objects such as leaves and trees on the background. In the Figure 27, there is an object, rubber on the line of our interest, however the algorithm is capable of ignoring the object and keep the results true. It must be noted that this is an extreme case because there won't be any disturbance as big as a rubber. Since the path is so narrow there won't be any place for a robot to pass. Another test result in Figure 28 indicates a different location for the rubber. Again, the system is unaffected.

The second test includes a shadow on the path. As in first test result, our solution is able to handle this situation as well. The result can be seen in Figure 29.

The third test results show that the object won't affect the performance of the system in the standard lightning conditions as well.

In all the test cases it is beneficial to note that there is a lot of noise and disturbance on the background, however the robot is supposed to work on the cardboard which is placed on the ground. Hence, there won't be any disturbances as in the test result figures.

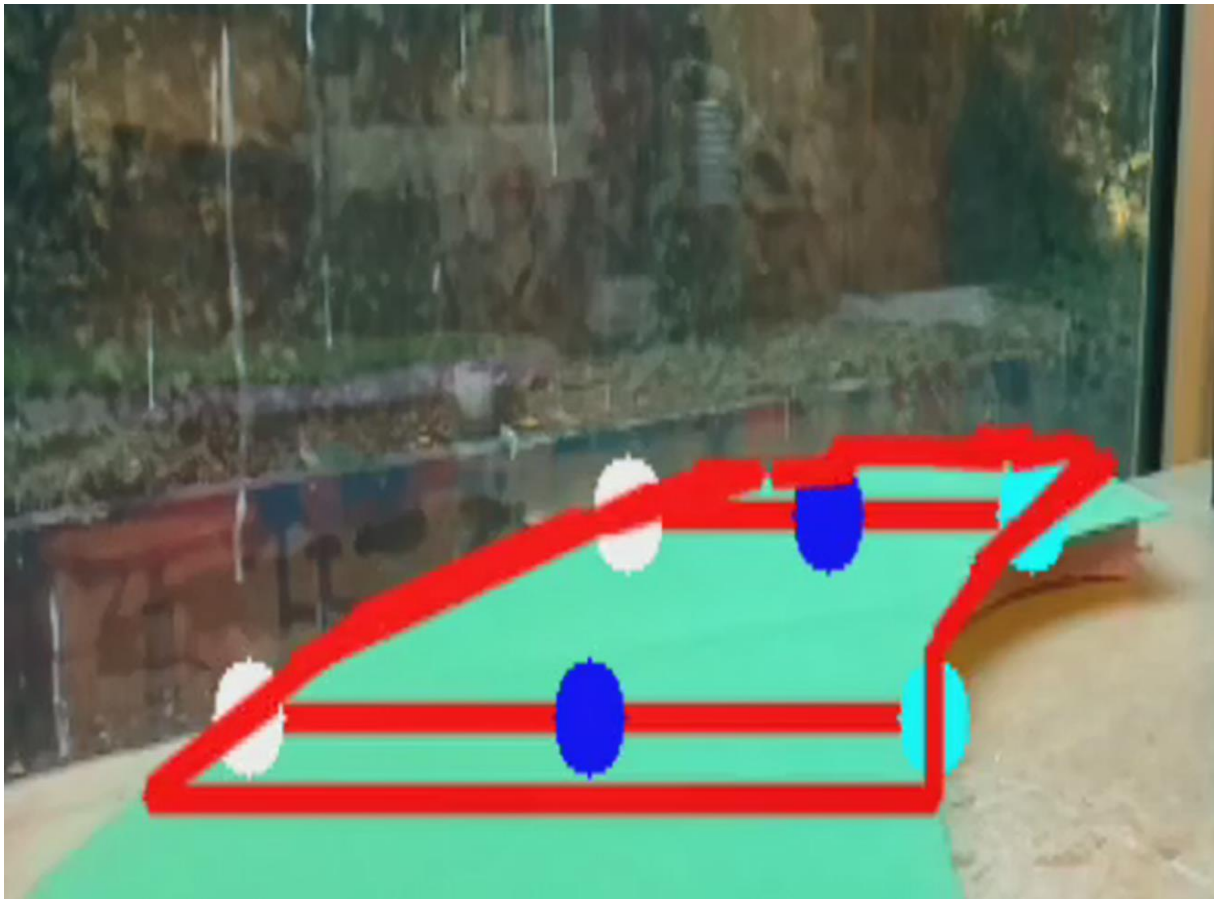


Figure 26: The heavy lightning condition test result

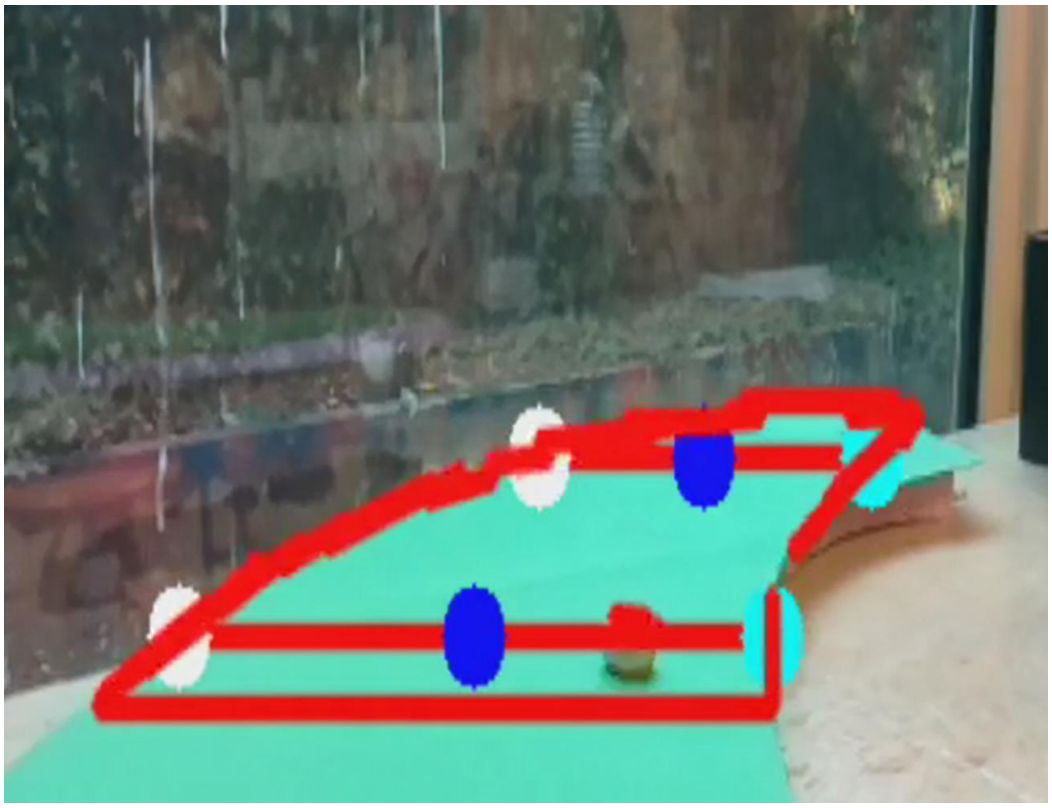


Figure 27: The heavy lightning condition test result with an object on the path

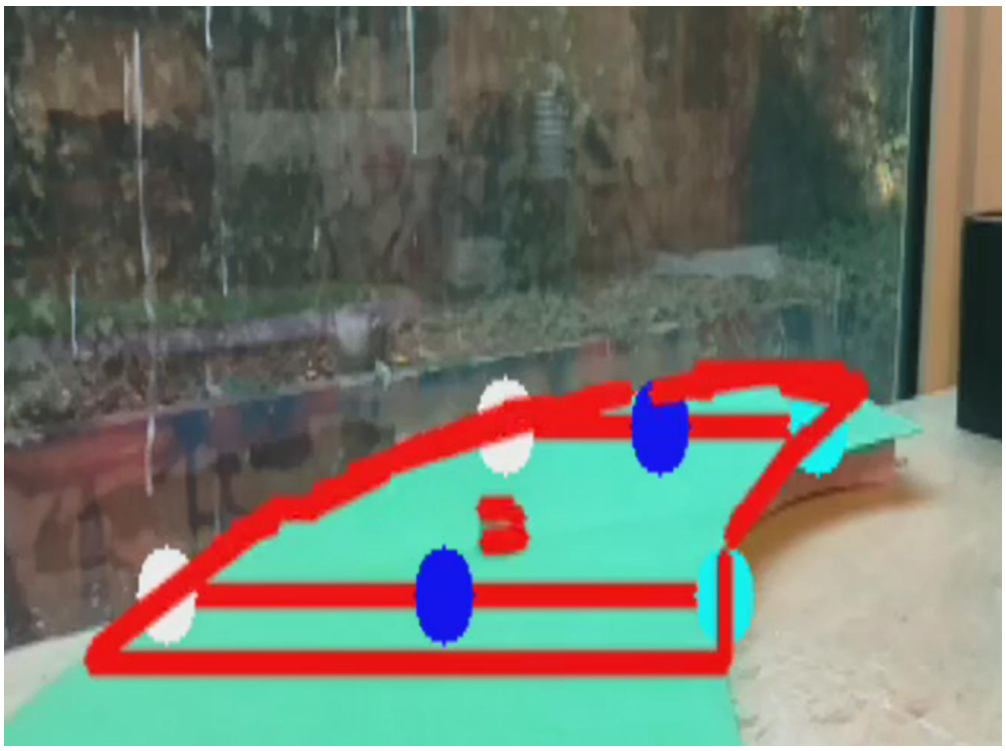


Figure 28*: The heavy lightning condition test result with an object on the path occasion 2*

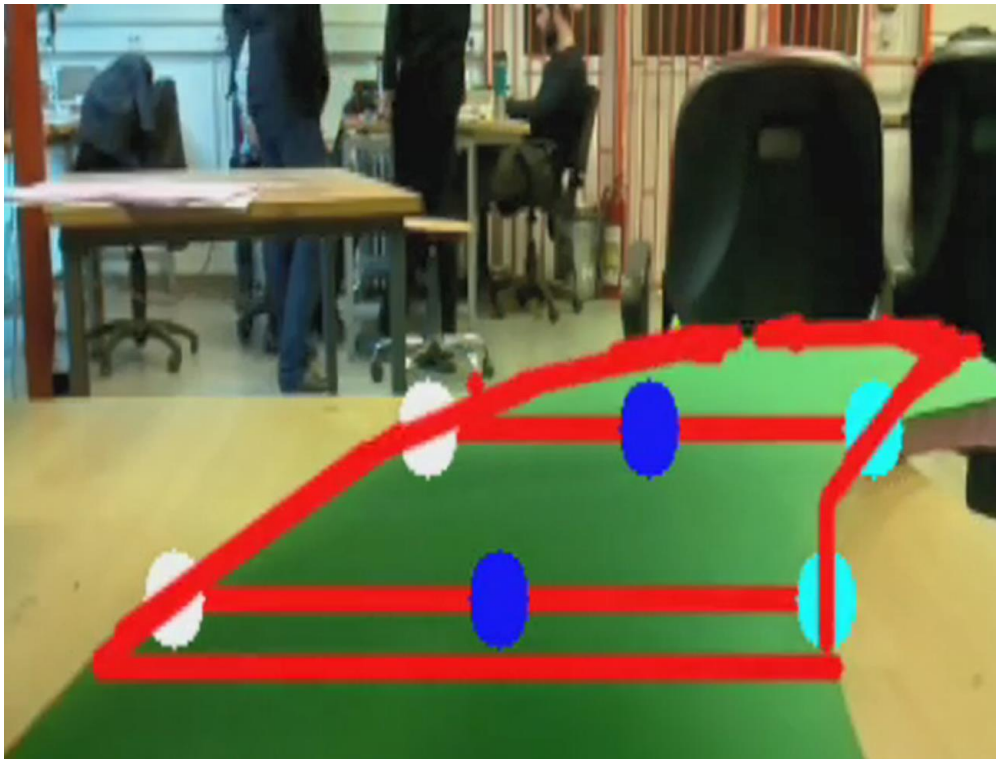


Figure 2919: The shadow on the path won't affect the performance of the system

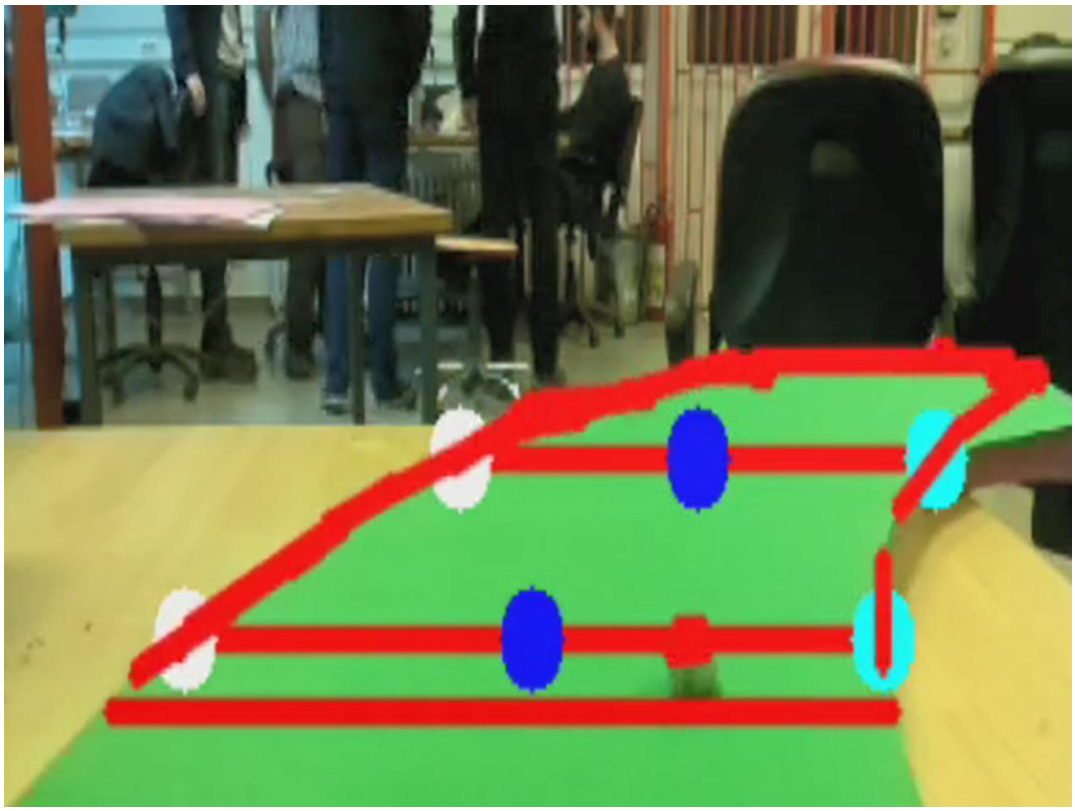


Figure 30: There is an object on the path in regular conditions

5. DELIVERABLES

When a client gives an order, the client is delivered the items below:

- Robot
- Green race track (100 cm x 130 cm)
- 3 cell 11.1 V 1500mAh Li-Po battery
- Li-Po charger
- TTEC 2BB139L AlumiSlim 5.000mAh power bank
- User manual (provided in appendix)

6. COST ANALYSIS

6.1 Actual Expenditures

The components that are planned to use to construct the robot and their prices are shown in Table 1. Note that our total limit is $200.

Table 1. Cost Analysis

Component	# of components	Total cost
Raspberry Pi 3	1	$50
Arduino Uno	1	$20
Raspberry Pi Camera Module	1	$9
HC-SR04 Ultrasonic Distance Sensor	6	$6
Pololu 2215 DC Motors	2	$34
L298N Motor Driver	1	$7
Plastic High Friction Wheels	2	$5
11,1V Li-Po Battery 1350 mAh 25C	1	$10
Powerbank 2500 mAh	1	$10
Ball Caster	1	$1
Plexy Glass	3	$13
3D Printed Parts	2	$8
Foam Board	2	$3
Foam Board Glue	1	$2
Cardboard	5	$1

TOTAL: $189

The cost of other parts such as resistors, capacitors, cables is $5. Hence, our total cost is $194 which is in our limit.

6.2 Total Expenditure

As we have done a lot of trial and errors in the creation process, we have spent a considerable amount of money more from the actual cost. We kept track of this money by using an app called Splitwise. We just register the money we spent in the app and it keeps track of the debt.

Total Money Spent: 2040 TL = 337 $

7. CONCLUSION AND DISCUSSIONS

As Robotz with Attitude team, we were determined to create a tagging robot that can operate autonomously on the racetrack and compete with other robots. Throughout the year, we spent blood, sweat and tears on creating the project plan, coming up with subparts, creating and testing the subparts and actually combining them to create the ultimate machine. At the end, we finally created our robot as we wanted.

During the building process, we have gone through a lot of trial and error and hardships. Overcoming these were quite challenging but made our product much more valuable and satisfying in our eyes.

To sum up, in this report, we mainly discussed what we have designed and created throughout two semesters. We have given an executive summary of the final product, we explained the design details of the robot and the performance of it, list of deliverables and a budget analysis.

Applications of the Product: The final product can be used for autonomous public transportation units which detects the road that it should follow. Moreover, the units can communicate with each other about the status of other units. Also, the infrastructreless Ad-Hoc network may be utilized to carry network for the passengers, or even to the far corners of the city. Moreover, the image processing can be utilized in many areas such as tracking path and objects on it. The image processing algorithm allows the user to detect the shapes in the intersection of line of sight of the robot and the path.

Environmental Impact: With utilization of green energy, the product may provide an efficient and environmental-friendly transportation. The image processing can be modified so that accidents can be prevented, saving many human and animal lives. Also, with recognition of the

obstacles on the road, trash and other materials that are harmful to the nature can be eliminated.

Safety Issues: During the design procedure, common workplace safety issues were addressed and safety was assured. First of all, insulation of DC power supply's output was assured to prevent any danger while conducting tests in the early stages of the design. Also, gloves were used whenever a task is to be done that may injure the hands. These tasks included smoldering the parts of the robot to strengthen the connections and using the hot wire to cut the road from foam. Moreover, any sharp object was put away as soon as they are done with. Overall, a safe work environment was preserved.

8. APPENDICES

APPENDIX A: USER MANUAL

SETTING UP THE SYSTEM

Before starting the robot, make sure the power bank and Li-Po battery is connected to the system properly.

After connecting the power supplies, connect a monitor, keyboard and a mouse to Raspberry Pi. Open /etc/rc.local with a text editor. Follow the instructions written in the file to set up the ad-hoc network or to connect to an existing ad-hoc network. Next, open /Documents/karar.py with a text editor. To connect to an existing TCP connection, change the rival_ip to the IP of the other robot and make sure if_server variable is set to False. To set up a TCP connection, set if_server to True, rival_ip variable is not used to open a TCP server. If you wish to use the robot without pairing with another robot, open /Documents/ top_module.py with a text editor and comment out the indicated lines.

Now that the system is prepared, place the robot on the race track and press the push button once. This will start the software of the robot. Once the LED turns off, meaning that connection is established and the robot is ready to go, close the switch to activate the motors. The robot will follow the track until it gets closer than 5 cm to another paired robot and stops.

WARNINGS

- For best performance, do not to operate the robot when Li-Po battery charge is lower than 11V.
- For best performance, do not to operate the robot when power bank charge is lower than 50%.
- Do not connect a power bank with output current less than 2.1A.

- For best performance, make sure the camera angle is 7°.
- We strongly advise you against modifying the source codes of the system other than the ones that are required for pairing.