CZ3004-Multi-Disciplinary Project

Log Report 2 (Implementation)

Submitted by

Team 2

Member 1: Aye Pwint Phyu Matric No. : U1920903F
Member 2: Lin Yan Matric No. : U1920925F
Member 3: Jesline Ng Matric No. : U1920213C
Member 4: Phone Myint Thu Mya Min Matric No. : U1920225E

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

	Member	Section
1	Aye Pwint Phyu	Robot movement and Rpi communication
2	Lin Yan	Image recognition
3	Jesline Ng	Android development
4	Phone Myint Thu Mya Min	Algorithm

1) Robot movement and Rpi communication

Implementation of the design proposed earlier

Basic movements such as Forward, Reverse, Turn Right and Turn Left are implemented using timed loops. Specific data packets are written into the serial port accordingly. An example of a data packet with speed 130 forward movement is [0x5a, 0x0c, 0x01, 0x01, ***0x00, 0x82***, 0x00, 0x00, ***0x00, 0x00***, 0x00, 0xff]. The first bolded pair of Hex values are considered as 16-bit data for speed and the second as turning angle. For the speed, positive values indicate forward, and negative values indicate reverse. For the turning angle, positive values indicate left and negative values indicate right. The functions will then be called into counted loops to move about depending on the nature of the call. If faster speeds are required, the time set for the timed loop can be reduced to match the same move distance/angle.

For easier understanding, written below is the sudo code explanation for such movements:

```
def forward/reverse10cm ()
    move = [specific speed with zero turning angle]
    start = start time
    while True:
        write move into the port using serial communication
        if (current time - start) >= certain value (in seconds): break
def turning90deg ()
    turn = [positive speed and max turning angle]
    reverse = [negative speed and max turning angle]
    turnv2 = [positive speed and very low turning angle]
    start = start time

    # first turn with max turning angle
    while True:
        write turn into the port using serial communication
        if (current time - start) >= certain value (in seconds): break

    # reverse with same turning angle
    while True:
        write reverse into the port using serial communication
        if (current time - start) >= certain value (in seconds): break

    # turn with reduce angle to adjust the full 90 degree turn
    while True:
        write turnv2 into the port using serial communication
        if (current time - start) >= certain value (in seconds): break
```

For Rpi communication, there are software communication via remote wifi access and Bluetooth to communicate with other devices and hardware communication with the camera.

- Remote wifi access was established after setting the static ip address as 192.168.2.1 as our group is group (2). Other devices can connect to Rpi's wifi or remote access via VNC or SSH directly. Here are the details needed for other devices to know for successful access.
    - Wifi name - MDPGrp2
    - Wifi password - 2021Grp2
    - Login username - pi
    - Login user password - group2
- Bluetooth connection can be established by running a Bluetooth program. Upon a successful connection, Rpi and other devices can exchange messages to and fro. In this Bluetooth program,
    - Python's standard socket module is utilized to establish both server and client in one program.
    - Incoming messages are decoded into 'utf-8' format to display/store as string messages.
    - Outgoing messages are sent as string messages as well.
    - The reliability of these message exchanges is going to be crucial for the next step of the project: *Integration*.
- Camera connection was achieved by temporarily attaching it to the robot to test the OpenCV program written by my teammate: Lin Yan. After successful connection and multiple testings with the robot for successful detection of the images provided, the camera was then detached again for Lin Yan to improve her program further for the next phase of the project: *Integration*.

Unexpected issues and changes incorporated

For Robot movement, as we are testing the robot in different places with different floor textures, the inaccuracy of the displacements is encountered due to the different friction level. As of this phase, we are unable to fix the issue. However, in the next phase with integration, we will be able to use camera and adjust the offset of the displacements by differentiating the size of the pictures taken (i.e., bigger picture indicates robot is closer to the object and if the size of the object is not within desired frame, movements can be adjusted)

For communication, Bluetooth connection via listening to a specific channel of the device connected was complicating the service ports registered. Instead, we have resorted to using the server-client socket method by using one static channel for Bluetooth communication.

2) Image recognition

Implementation of Image recognition

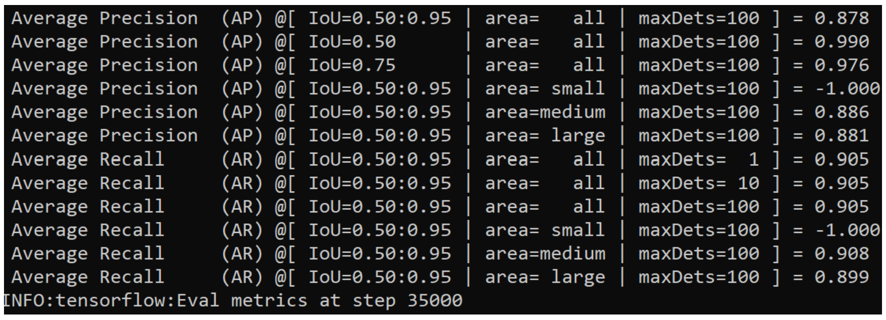
1. <u>Image acquisition and pre-processing.</u>
   Use the Rpi camera to take photos through program and label the photos by labelling to generate xml file for each of them. (150 photos for each image)

```
for label in labels:
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

2. <u>Image recognition model training & evaluation</u>

   Divide the processed pictures into a test set and a train set at a ratio of 2:8. Download TF Models Pretrained Models from Tensorflow Model Zoo and install models. Considering Rpi's limited computing power, I chose the "SSD MobileNet V2 FPNLite 320x320" model here, which can maintain a faster computing speed on the mobile terminal. The main idea is to perform dense sampling evenly at different positions of the picture, and different scales and aspect ratios can be used when sampling. Then use CNN to extract features, directly perform classification and regression. The whole process only requires one step, so its advantage is that it is fast. However, an important disadvantage of uniform dense sampling is that training is more difficult, which leads to slightly lower model accuracy. In order to improve the accuracy of the model, I conducted 35,000 training.

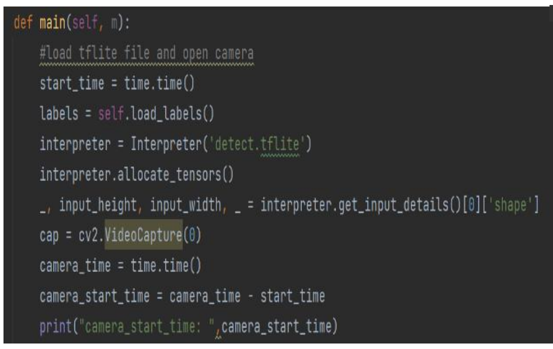
   In the end, the average recognition rate of the model can reach 0.878. After testing, as long as the complete picture appears in the camera within a distance of 50cm, the recognition accuracy rate can exceed 90%.



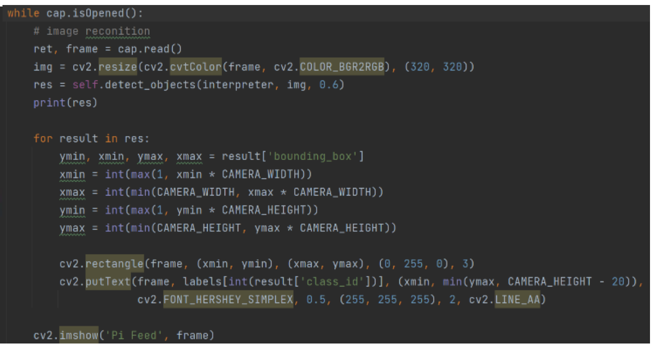
```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.878
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.976
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.886
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.881
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.905
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.905
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.905
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.908
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.899
INFO:tensorflow:Eval metrics at step 35000
```

3. <u>To deploy the trained model into the Raspberry Pi</u>

   Export the trained model and convert the model to tflite format. The tflite is a toolkit for deploying deep learning models to mobile terminals and embedded devices. It can transform, generate, and optimize the trained models to increase computing speed and reduce memory and video memory usage. The deployment of the target recognition model is mainly divided into two parts, the first is to load the tflite model, and then use the OpenCV package for real-time photo processing, and put the photo results into the target recognition model for recognition. Program will try catch 5 photos whose accuracy are not less than 80% within 5 seconds, and use the most frequent image_id as the final recognized image_id.



```python
def main(self, m):
    #load tflite file and open camera
    start_time = time.time()
    labels = self.load_labels()
    interpreter = Interpreter('detect.tflite')
    interpreter.allocate_tensors()
    _, input_height, input_width, _ = interpreter.get_input_details()[0]['shape']
    cap = cv2.VideoCapture(0)
    camera_time = time.time()
    camera_start_time = camera_time - start_time
    print("camera_start_time: ",camera_start_time)
```



```python
while cap.isOpened():
    # image reconition
    ret, frame = cap.read()
    img = cv2.resize(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB), (320, 320))
    res = self.detect_objects(interpreter, img, 0.6)
    print(res)

    for result in res:
        ymin, xmin, ymax, xmax = result['bounding_box']
        xmin = int(max(1, xmin * CAMERA_WIDTH))
        xmax = int(min(CAMERA_WIDTH, xmax * CAMERA_WIDTH))
        ymin = int(max(1, ymin * CAMERA_HEIGHT))
        ymax = int(min(CAMERA_HEIGHT, ymax * CAMERA_HEIGHT))

        cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 255, 0), 3)
        cv2.putText(frame, labels[int(result['class_id'])], (xmin, min(ymax, CAMERA_HEIGHT - 20)),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.imshow('Pi Feed', frame)
```

Unexpected issues and changes incorporated

Since a model with a faster speed but a lower accuracy is selected, the training of the model is more difficult. In order to improve the accuracy of the model, I made improvements in two aspects:

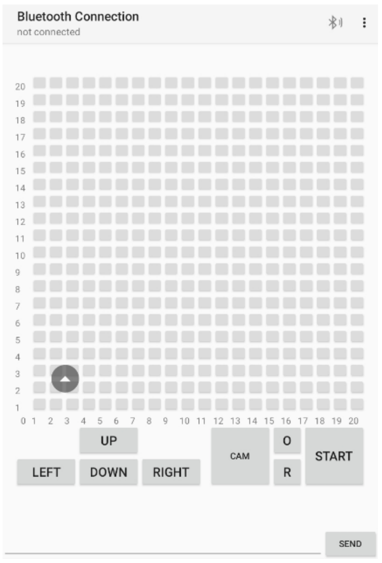
1. Improve the picture collection for training
   - All pictures are taken with the Rpi camera
     Different camera will cause different pixel and different colour, which will increase the recognition error.
   - Try to simulate the scene as much as possible to take photos
     Consider all the possible scenarios and remove unnecessary interruptions.
   - More angles, different distances photos
     Expand the scope of recognition as much as possible.

2. More training times
   I have tried to train the model by 2,000 times, 5,000 times and 35,000 times, the more training times, the better the recognition loss will converge, eventually converging to 0.16.

3) Android development

Implementation of the Design Proposed

The overall UI design of the application remains the same with the addition of a “CAM” button to activate the camera. Other that that the layout and remains same with the design stage.



Bluetooth Connection

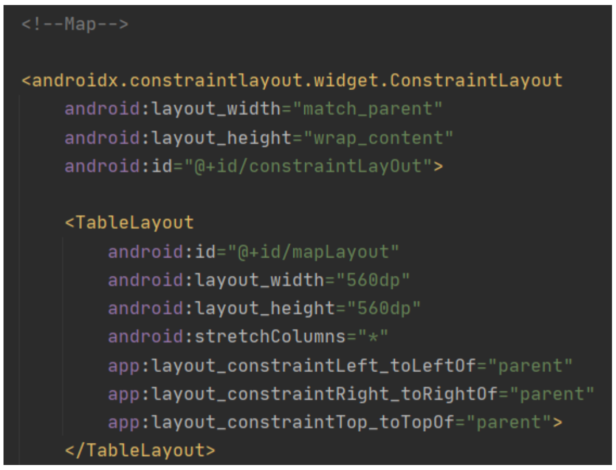
There are four major task that needed to be implemented to have a working Bluetooth connection.

- Setting up Bluetooth
- Finding nearby devices
- Connecting to the device
- Transfer data between devices

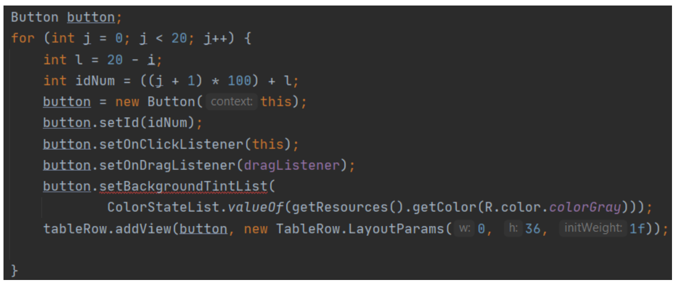
This was implemented following the documentation available at the Android Developer.

Functional Requirement

The map that features prominently in the application is implemented using buttons. This is implemented by setting the area in the activity_main.xml with a table view. It is then filled programmatically with button when the application is started.

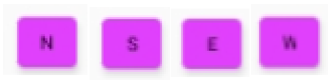


```xml
<!--Map-->

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/constraintLayOut">

    <TableLayout
        android:id="@+id/mapLayout"
        android:layout_width="560dp"
        android:layout_height="560dp"
        android:stretchColumns="*"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </TableLayout>
```

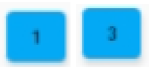


```java
Button button;
for (int j = 0; j < 20; j++) {
    int l = 20 - i;
    int idNum = ((j + 1) * 100) + l;
    button = new Button( context: this);
    button.setId(idNum);
    button.setOnClickListener(this);
    button.setOnDragListener(dragListener);
    button.setBackgroundTintList(
            ColorStateList.valueOf(getResources().getColor(R.color.colorGray)));
    tableRow.addView(button, new TableRow.LayoutParams( w: 0,  h: 36,  initWeight: 1f));

}
```

The obstacle is indicated by changing the colour of the button and the face of each obstacle is being indicated by the letter N, S, E, W on the button, as shown.



When the image has been sensed. The button will change its colour to blue and the number ID of the image can be display in the button.



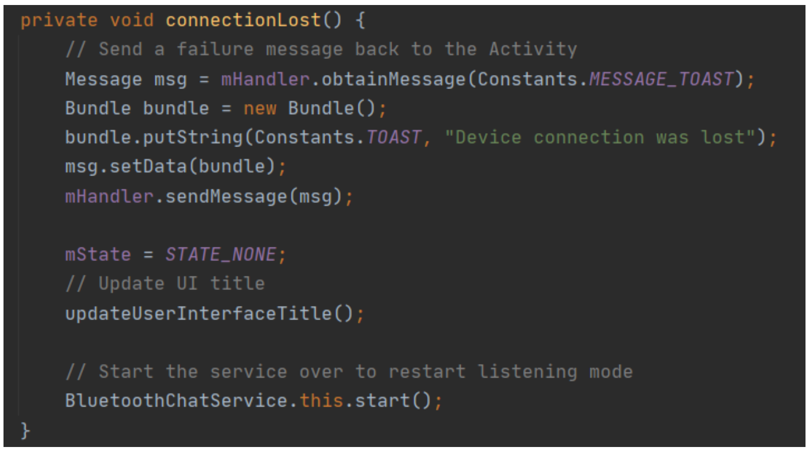
Additional Function

We added the ability to show the location indicated in the map to allow the user to know where the obstacle is positioned at. This will be indicated on top of the map.

Map Location 17, 6

Unexpected Issues

The Bluetooth connection was unable to maintain a stable connection. When the client was disconnected from the android, the android will close the connection instead of trying to reconnect again. In order to overcome this issue, below code is added. With this addition, the android will try to listen again to any devices that is trying to reconnect instead of dropping the connection.



```java
private void connectionLost() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(Constants.MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(Constants.TOAST, "Device connection was lost");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    mState = STATE_NONE;
    // Update UI title
    updateUserInterfaceTitle();

    // Start the service over to restart listening mode
    BluetoothChatService.this.start();
}
```

Implementation Strategies

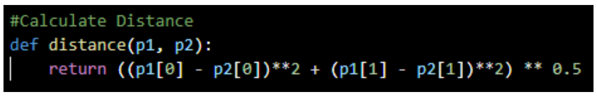
The implementation has been divided into 3 stages: connection, functionality, aesthetic. As the Bluetooth connection can be considered as the backbone of the application, it will need to be implemented first before the rest could be implemented. Once the connection has been established, the required functionality can be added. Finally, if there is time left, the aesthetic of the application will be modified.

4) Algorithm

Implementation of the algorithms

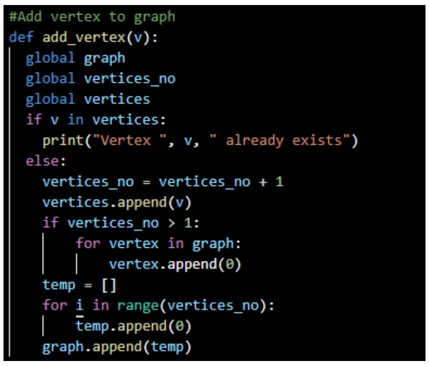
Hamiltonian Path implementation

Since android app send the coordinates of the obstacles, a variable is created to store the list of obstacles with coordinates and directions. However, the distance between the obstacles and the robot are still unknown. Therefore, distance function is created to calculate the approximate distances between each obstacle.

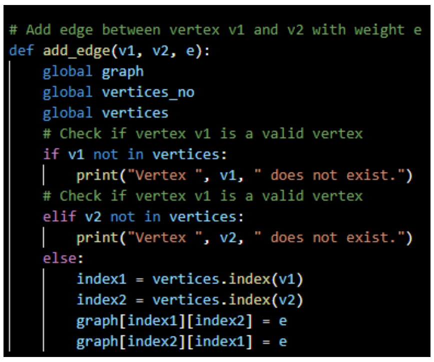


```python
#Calculate Distance
def distance(p1, p2):
    return ((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2) ** 0.5
```

The distances of obstacles and number of obstacles are known. For easier calculation, the graph function is created as below.

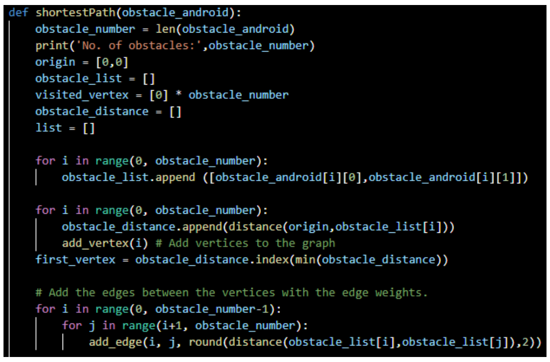


```python
#Add vertex to graph
def add_vertex(v):
  global graph
  global vertices_no
  global vertices
  if v in vertices:
    print("Vertex ", v, " already exists")
  else:
    vertices_no = vertices_no + 1
    vertices.append(v)
    if vertices_no > 1:
        for vertex in graph:
            vertex.append(0)
    temp = []
    for i in range(vertices_no):
        temp.append(0)
    graph.append(temp)
```

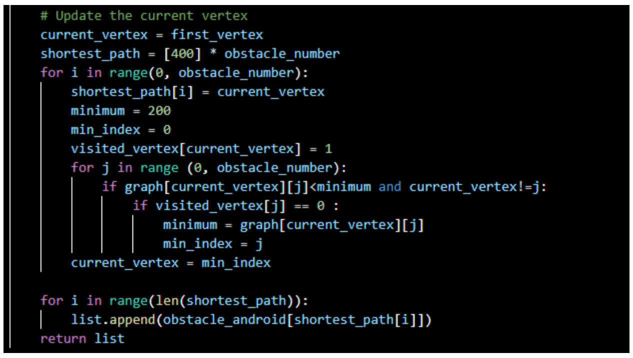


```python
# Add edge between vertex v1 and v2 with weight e
def add_edge(v1, v2, e):
    global graph
    global vertices_no
    global vertices
    # Check if vertex v1 is a valid vertex
    if v1 not in vertices:
        print("Vertex ", v1, " does not exist.")
    # Check if vertex v1 is a valid vertex
    elif v2 not in vertices:
        print("Vertex ", v2, " does not exist.")
    else:
        index1 = vertices.index(v1)
        index2 = vertices.index(v2)
        graph[index1][index2] = e
        graph[index2][index1] = e
```

Using graph and distance functions, the vertexes and edge with weights are added by iterating through a list. Then, using the nearest neighbour algorithm, the path function will return a list of sorted obstacles. The implemented function is shown below.



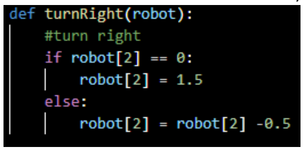
```python
def shortestPath(obstacle_android):
    obstacle_number = len(obstacle_android)
    print('No. of obstacles:',obstacle_number)
    origin = [0,0]
    obstacle_list = []
    visited_vertex = [0] * obstacle_number
    obstacle_distance = []
    list = []

    for i in range(0, obstacle_number):
        obstacle_list.append ([obstacle_android[i][0],obstacle_android[i][1]])

    for i in range(0, obstacle_number):
        obstacle_distance.append(distance(origin,obstacle_list[i]))
        add_vertex(i) # Add vertices to the graph
    first_vertex = obstacle_distance.index(min(obstacle_distance))

    # Add the edges between the vertices with the edge weights.
    for i in range(0, obstacle_number-1):
        for j in range(i+1, obstacle_number):
            add_edge(i, j, round(distance(obstacle_list[i],obstacle_list[j]),2))
```



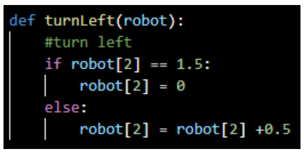
```python
    # Update the current vertex
    current_vertex = first_vertex
    shortest_path = [400] * obstacle_number
    for i in range(0, obstacle_number):
        shortest_path[i] = current_vertex
        minimum = 200
        min_index = 0
        visited_vertex[current_vertex] = 1
        for j in range (0, obstacle_number):
            if graph[current_vertex][j]<minimum and current_vertex!=j:
                if visited_vertex[j] == 0 :
                    minimum = graph[current_vertex][j]
                    min_index = j
        current_vertex = min_index

    for i in range(len(shortest_path)):
        list.append(obstacle_android[shortest_path[i]])
    return list
```

Path navigation implementation

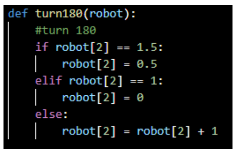
3 functions for turning are created for easier integration with robot movement. To keep track of robot facing direction, the turning angle is added once the function is called. The coefficient of pi is marked as theta hence 0 as east, 0.5 as north, 1 as west and 1.5 as south.



```python
def turnRight(robot):
    #turn right
    if robot[2] == 0:
        robot[2] = 1.5
    else:
        robot[2] = robot[2] -0.5
```

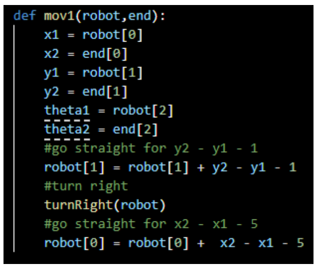


```python
def turnLeft(robot):
    #turn left
    if robot[2] == 1.5:
        robot[2] = 0
    else:
        robot[2] = robot[2] +0.5
```

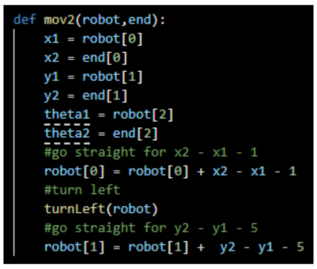


```python
def turn180(robot):
    #turn 180
    if robot[2] == 1.5:
        robot[2] = 0.5
    elif robot[2] == 1:
        robot[2] = 0
    else:
        robot[2] = robot[2] + 1
```

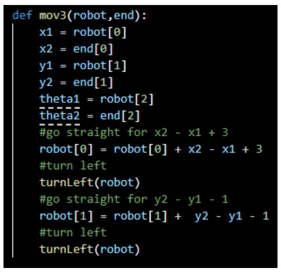
16 movement functions for the robot are implemented as the flowchart mentioned in log report 1. Few samples of the functions are shown below.



```python
def mov1(robot,end):
    x1 = robot[0]
    x2 = end[0]
    y1 = robot[1]
    y2 = end[1]
    theta1 = robot[2]
    theta2 = end[2]
    #go straight for y2 - y1 - 1
    robot[1] = robot[1] + y2 - y1 - 1
    #turn right
    turnRight(robot)
    #go straight for x2 - x1 - 5
    robot[0] = robot[0] +  x2 - x1 - 5
```

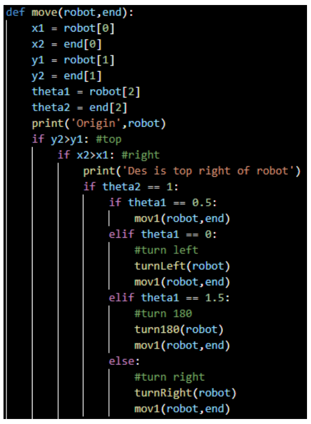


```python
def mov2(robot,end):
    x1 = robot[0]
    x2 = end[0]
    y1 = robot[1]
    y2 = end[1]
    theta1 = robot[2]
    theta2 = end[2]
    #go straight for x2 - x1 - 1
    robot[0] = robot[0] + x2 - x1 - 1
    #turn left
    turnLeft(robot)
    #go straight for y2 - y1 - 5
    robot[1] = robot[1] +  y2 - y1 - 5
```

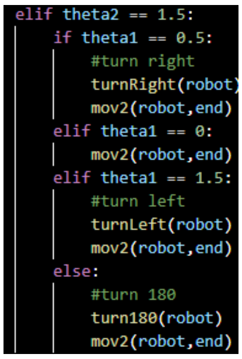


```python
def mov3(robot,end):
    x1 = robot[0]
    x2 = end[0]
    y1 = robot[1]
    y2 = end[1]
    theta1 = robot[2]
    theta2 = end[2]
    #go straight for x2 - x1 + 3
    robot[0] = robot[0] + x2 - x1 + 3
    #turn left
    turnLeft(robot)
    #go straight for y2 - y1 - 1
    robot[1] = robot[1] +  y2 - y1 - 1
    #turn left
    turnLeft(robot)
```

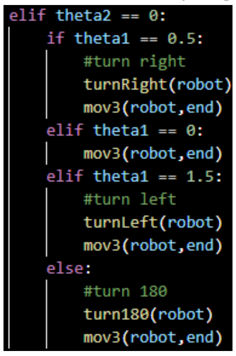
The main function for path navigation algorithm (for obstacle at top right) is attached below.



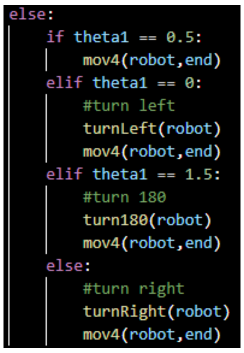
```python
def move(robot,end):
    x1 = robot[0]
    x2 = end[0]
    y1 = robot[1]
    y2 = end[1]
    theta1 = robot[2]
    theta2 = end[2]
    print('Origin',robot)
    if y2>y1: #top
        if x2>x1: #right
            print('Des is top right of robot')
            if theta2 == 1:
                if theta1 == 0.5:
                    mov1(robot,end)
                elif theta1 == 0:
                    #turn left
                    turnLeft(robot)
                    mov1(robot,end)
                elif theta1 == 1.5:
                    #turn 180
                    turn180(robot)
                    mov1(robot,end)
                else:
                    #turn right
                    turnRight(robot)
                    mov1(robot,end)
```



```python
elif theta2 == 1.5:
    if theta1 == 0.5:
        #turn right
        turnRight(robot)
        mov2(robot,end)
    elif theta1 == 0:
        mov2(robot,end)
    elif theta1 == 1.5:
        #turn left
        turnLeft(robot)
        mov2(robot,end)
    else:
        #turn 180
        turn180(robot)
        mov2(robot,end)
```

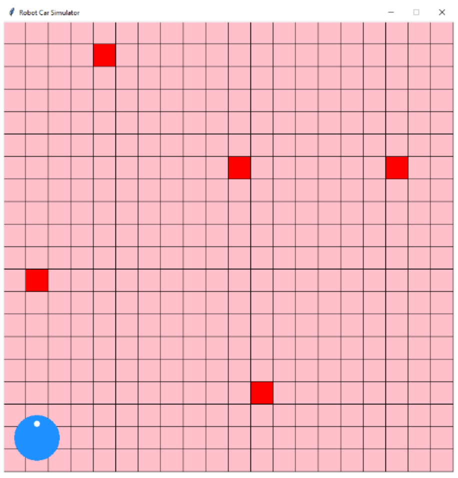


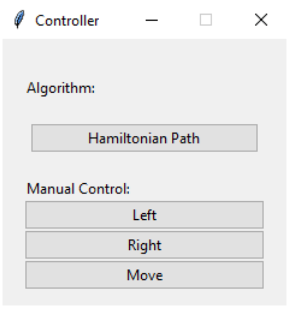
```python
elif theta2 == 0:
    if theta1 == 0.5:
        #turn right
        turnRight(robot)
        mov3(robot,end)
    elif theta1 == 0:
        mov3(robot,end)
    elif theta1 == 1.5:
        #turn left
        turnLeft(robot)
        mov3(robot,end)
    else:
        #turn 180
        turn180(robot)
        mov3(robot,end)
```



```python
else:
    if theta1 == 0.5:
        mov4(robot,end)
    elif theta1 == 0:
        #turn left
        turnLeft(robot)
        mov4(robot,end)
    elif theta1 == 1.5:
        #turn 180
        turn180(robot)
        mov4(robot,end)
    else:
        #turn right
        turnRight(robot)
        mov4(robot,end)
```

It is difficult to check whether the algorithm is working or not. Therefore, a simulator for robot is created to test out the algorithm. GUI is showned below.





Unexpected issues

Forward function is iterated according to the calculated distance. The issue occurs once the result is negative. This issue is solved by calling reverse function. Another issue is that the algo does not check whether the obstacle is in front or not. The robot moves as the created move functions even if there is an obstacle in front. It is solved by creating one function to check the moving path is clear or not. If it is not clear, robot will move out to the free space and check again till there is no obstacle.

Implementation strategies

The flow of the algorithms is thoroughly analysed in the design stage. A few functions are created in the purpose of reusing during the implementation to improve modularity and optimization. The comments are also added to improve the readability.