# CZ3004-Multi-Disciplinary Project

# Log Report 3 (Integration)

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

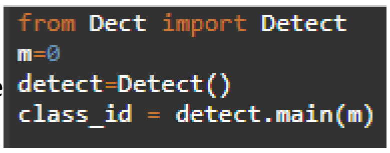
NANYANG TECHNOLOGICAL UNIVERSITY

| | 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

### Integration with the Camera

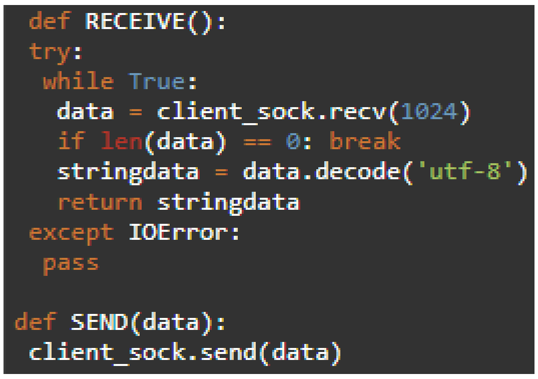
This task is fairly easy, as it was to be done by installing the camera onto the Robot and calling the program into the main and the algorithm from the image detection module. As the image detection module was written to be used as a class, any other module can use this class



```
from Dect import Detect
m=0
detect=Detect()
class_id = detect.main(m)
```

### Integration with the Android

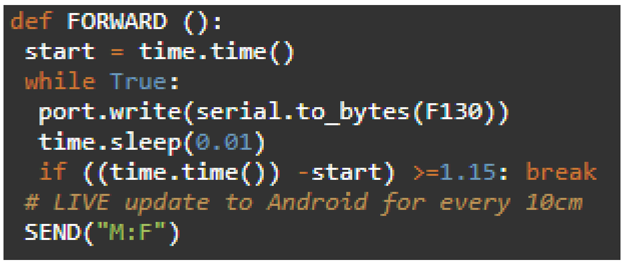
The bluetooth connection was already established in the previous phase of the project. Thus, integration was done by exchanging messages properly. First the bluetooth module was separated from the main file as a module with separate functions. The functions are then called into necessary places that require communication with the robot either for robot movement update or the scanned images ID.



```
 def RECEIVE():
 try:
  while True:
   data = client_sock.recv(1024)
   if len(data) == 0: break
   stringdata = data.decode('utf-8')
   return stringdata
 except IOError:
  pass

def SEND(data):
 client_sock.send(data)
```

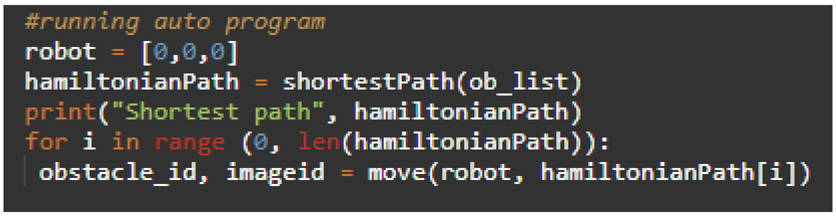
An example of LIVE communication with Android



```
def FORWARD ():
 start = time.time()
 while True:
  port.write(serial.to_bytes(F130))
  time.sleep(0.01)
  if ((time.time()) -start) >=1.15: break
 # LIVE update to Android for every 10cm
 SEND("M:F")
```

### Integration with the Algorithm

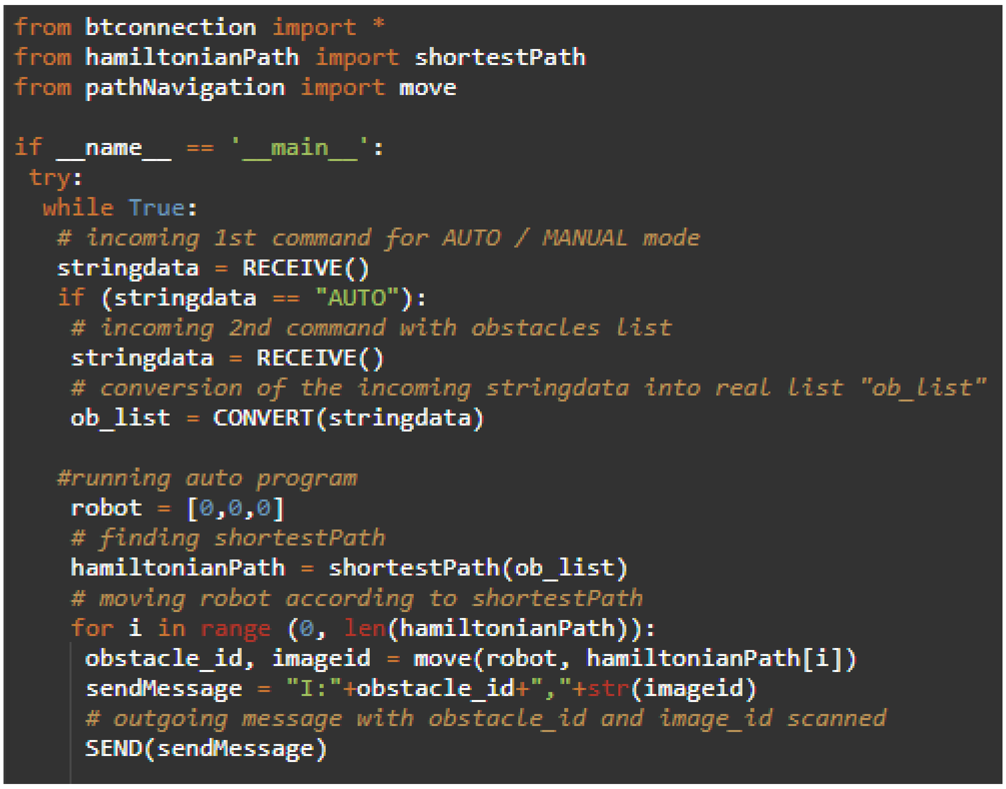
The algorithm was integrated into the main program and the pathNavigation program was integrated with image detection and movement functions inside. Next it was integrated with bluetooth to feedback return values to the Android.



```
#running auto program
robot = [0,0,0]
hamiltonianPath = shortestPath(ob_list)
print("Shortest path", hamiltonianPath)
for i in range (0, len(hamiltonianPath)):
 obstacle_id, imageid = move(robot, hamiltonianPath[i])
```

## Integration with all components for seamless communication

Auto operation involves most communication amongst all components. Here is the code for the auto operation. Everything imported is being used in this integration. Once the Android sends out the command “Auto”, everything will be autonomous.



```
from btconnection import *
from hamiltonianPath import shortestPath
from pathNavigation import move

if __name__ == '__main__':
 try:
  while True:
   # incoming 1st command for AUTO / MANUAL mode
   stringdata = RECEIVE()
   if (stringdata == "AUTO"):
    # incoming 2nd command with obstacles list
    stringdata = RECEIVE()
    # conversion of the incoming stringdata into real list "ob_list"
    ob_list = CONVERT(stringdata)

   #running auto program
    robot = [0,0,0]
    # finding shortestPath
    hamiltonianPath = shortestPath(ob_list)
    # moving robot according to shortestPath
    for i in range (0, len(hamiltonianPath)):
     obstacle_id, imageid = move(robot, hamiltonianPath[i])
     sendMessage = "I:"+obstacle_id+","+str(imageid)
     # outgoing message with obstacle_id and image_id scanned
     SEND(sendMessage)
```

## Integration Strategies

First robot movements and scanning modules are integrated into algorithm “pathNavigation” for autonomous movement and detection. Next algorithm was tested with movement and dectection. Once it was working, Rpi and android communication procedure was created to integrate auto and manual operations respectively.

## Experience of the final task (during week 5)- what we have learned.

I have learnt that for the integration to work, teamwork and unity is utmost important. Everyone in our team is an active member making alternate compromises with full time work to get the integration done properly. With weekly tasks checklist leading up to week 5, final task of integration was made easier for testing and improving. It was a great learning experience for the project itself as well as a great experience learning to co-operate to create a wholesome project with different members with different tasks.

## 2) Image recognition

### Integration of various subcomponents within the section

Almost all of our programs are on Rpi, and Rpi will connect with the Android App via Bluetooth to communicate.

Android App is our initial data receiving end which is responsible for inputting the location of obstacles and the location of the image that needs to be recognized. Besides, after the car starts to move, Rpi will continuously send back its movement instructions so that the user can observe the movement trajectory of the nano car on the App's map.

The algorithm program will generate a suitable route after receiving the information sent by the Android App, and the car will move according to the generated route.

When the car moves to the position before the image which need to be recognized, the camera will turn on and start to recognize. The image recognition program will return a class_id after the recognition is completed, and Rpi will send the id back to the Android app and display it on the app's map.

### Integration and complete seamless communication for the final task

The image recognition program is written as a class that can be imported by the main program. Once the car moves to a given position in the image, the main program will call the image recognition module and turn on the camera to recognize the image.

The camera will collect five recognition results with a recognition accuracy of not less than 0.6, and finally return the result with the highest frequency as the class_id. At the same time, the recognition time of each picture must not exceed 5s. Once it exceeds, program will return to the currently collected result with the highest frequency. After returning the class_id, the camera will automatically turn off.

In addition to specifying the class_id of each image, we also added class_id "0" for the bullseye and class_id "16" for undetected targets for users' reference. When the main program receives the class id, it will start moving to the next target.

### Integration strategies

Since need to return the coordinate position of the picture and in order to avoid identifying the wrong target, we choose to turn on the camera and start the recognition when the car reaches the specified position. The recognition result will be returned to the main program as integer, then send to the app via Bluetooth. Finally, the image calss_id will be displayed on the map of the app.

### Experience of the final task

The main task in week 5 is integration and testing. Based on the good communication in the early stage, the integration between the members is successful. Although we encountered many problems during the testing phase, such as program bugs, sudden camera failure, etc., my team members are very active in trying to solve the problems.  I am very thankful that my team members are very serious, responsible and willing to communicate actively. My biggest experience in week 5 is that everyone’s active participation is the most important thing in teamwork. Everyone in our group has no relevant experience in this project, but because everyone is working hard, the originally arduous task is no longer out of reach.

## 3) Android development

### Integration of various subcomponents within the section

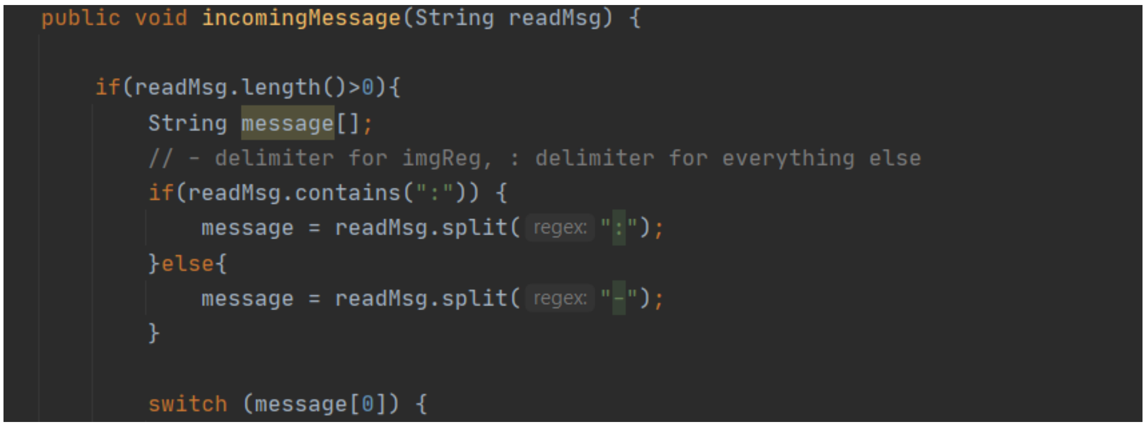
The integration between the algorithm and the android happened in the auto mode. It is made by passing the information of the obstacle to on the map to the algorithm. We keep track of each obstacle with its coordinate to allow for easy tracking. When the algorithm is navigating the robot, the movement will send the status update to the android, and this is how we update the position of the robot on the map. Once the robot has reached the obstacle and the camera has scanned the image, the obstacle coordinate & image ID is passed back to the android to be updated in the map.

In the manual mode, the map updating has been changed from the previous implementation. Instead of updating the position when the button is pressed, the position is only updated once the robot has made the moves. This will allow the user to know that the robot has made the movement and prevent the user to press the button multiple times and caused a mistake in the movement.

### Integration and complete seamless communication for the final task

The communication between android and the robot is made via Bluetooth. As mentioned previously a robust communication is the backbone of the application. Since it has been established, it allows for a simple integration for the final task.

To allow for a seamless communication between android and the robot, a function is added to parse the incoming message. This allows the android to ignore any irrelevant message and convey the important message to the user.

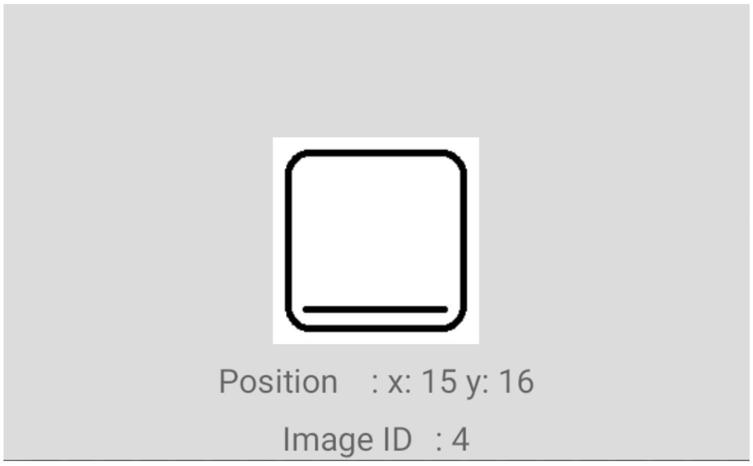


```
public void incomingMessage(String readMsg) {

    if(readMsg.length()>0){
        String message[];
        // - delimiter for imgReg, : delimiter for everything else
        if(readMsg.contains(":")) {
            message = readMsg.split( regex: ":");
        }else{
            message = readMsg.split( regex: "-");
        }

        switch (message[0]) {
```

Another feature that has been added to the android for the final task is the pop-up screen that shows the information of the obstacle. This allows the inspector to see more clearly the information of the obstacle.



Position : x: 15 y: 16

Image ID : 4

## Integration strategies

The strategy is having a main file that will call the different function hence making the integration process easier to implement. This also allow us to easily figure out which section is having trouble and how to solve it easily.

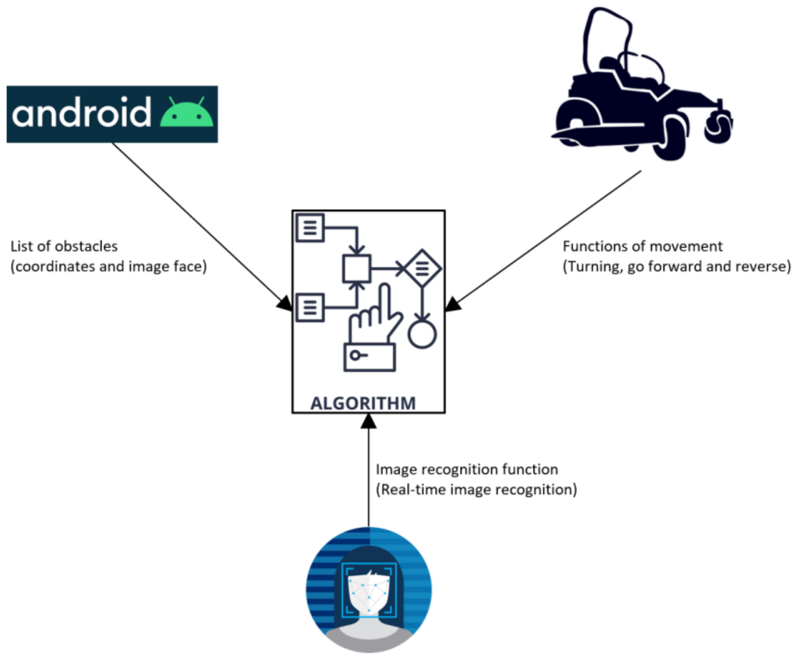
## Experience of the final task

Even though our team faces a lot of issue during the implementation stage, we are able to work together and figure out the solution to those issue. One of the issues was only solved on the day of the final task. It’s due to the fact that our team’s member never gave up and continue to persevere that we were able to finally complete the task.

## 4) Algorithm

### Integration of various subcomponents within the section

Once all subcomponents are working, we plan to bring all together. Android app will send coordinates of obstacles to the algo. With the help of Bluetooth connection with pi, the list of coordinates can be easily stored as a variable in algorithm. Then algorithm start with the Hamiltonian path calculation followed by the path navigation. In the path navigation, movement functions of robot are needed to move the robot to the obstacle according to algorithm. So, the integration with pi is done by calling movement functions like turn left, turn right, go straight from the robot movement part into the path navigation algorithm. Once the robot stops in front of the obstacle, robot needs to start camera and scan the image. The integration with image recognition is also straight forward as we can call function from image recognition that can open camera and perform real time image recognition.


android
List of obstacles
(coordinates and image face)
Functions of movement
(Turning, go forward and reverse)
ALGORITHM
Image recognition function
(Real-time image recognition)

### Integration and complete seamless communication for the final task

The integration was quite straight forward as we all created functions for each part. We just call functions when we started integration of sub-systems into unified single system. When we tested out the integrated system, it worked as desire though robot movements and algorithm are needed to be calibrated.

### Integration strategies

We first simplify the infrastructure to eliminate the extra steps. We try to integrate with one part at a time for easier troubleshooting. Since we all use single responsibility principle to write code for each part, it is easier for us to modify. I use GitHub to track changes in the code across versions and it also offers Integration options.

## Experience of the final task (during week 5)- what you learned.

During the final run, I see a lot of different algorithm approaches and different style of movement. Glad to find out how to use all these prior knowledges and skills from the courses and apply them into the project. Getting to know all these new team members and work together as a team. Overall, it is a great experience.