ENEE408I FALL 2020
FINAL REPORT

TEAM 2

Yuchen Zhou, Paulo Sousa, Clovis Akobong

Professor Gilmer Blankenship

Levi Burner (TA)

Table of Contents

Introduction

The challenge of this semester was to design and implement an autonomous companion robot that could perform user based tasks, such as communicating to separate robots through a server. Additional intermediate tasks included object avoidance, person following, voice control,and networking, serving as the milestones to guide us to our final robot. In the first month of the semester, we smoothly completed the object avoidance, person following, and partial voice control. The Alexa commands were used to format the directions, initialize recognition routine, and dance the robot. The Robots were also able to initialize each other and send commands over the network. Due to the influence of the COVID-19 pandemic, it became expensive to add some functionalities on all robots of our team. Therefore, we were not able to integrate some of our original ideas, heart rate detection for instance.

Primary materials

- Nvidia Xavier NX Jetson
- 2 x Raspberry Pi Camera V2/Logitech Webcam
- Arduino Nano Microcontroller
- Amazon Alexa Device (Spot, Echo, Alexa App)
- 2 x Pololu 100:1 Metal Gearmotor
- 2 x Pololu VNH5019 Motor Drive Carrier
- 2 x 120x60mm Pololu Knobby Wheels
- 3 x HC-SR04 Ultrasonic Sensors
- 12v LiPo Rechargeable Battery
- Robot Chassis made of acrylic

Github Repository

We used the following github repository for source code maintenance and sharing between members of the team. The repository is organized based on system basic functionality ranging from Arduino functionality, ultrasonic sensors, chat server, facial recognition, and other Jetson support libraries. In the repository, there is a directory called “Final_Code”, which consists of the scripts we used for our final demo. Regarding the control system, there are 5 main scripts:

run_robot.py - main script to initialize and to control the robot. It initializes the serial communication with Arduino Nano board, creates a client object to join the websocket chat server, establishes online communication with Amazon Alexa using Ngrok, and most important, defines the specific functionalities of Alexa commands in low level programming language. **run_robot.py** is running in parallel with **camera_function.py.**

camera_function.py - Python script that processes camera frames repeatedly based on the selected mode. In the initialization process of **run_robot.py**, a thread will call on this script to run in parallel with **run_robot.py**. This script has two modes. The default mode is facial recognition. The second mode is apriltag tracking, which will be turned on if a tracking voice command is received.

robot_chat_server.py - Python script that creates a Websocket chat server for the communication among robots. All registered clients on the server will receive messages sent from other clients simultaneously.

robot_chat_client.py - Python script that defines a client class to the chat_server. **run_robot.py** will import this module to create a client object in the initialization process.

Robot_movement.ino - Arduino script that handles the maneuver part of the robot. It receives and decodes serial commands sent from Jetson NX based on certain Alexa voice commands. Once the serial command is decoded, the Arduino MCU will execute the corresponding Arduino function. In addition, this script contains an automatic obstacle avoidance module that prevents the robot from crashing with an object in its path. The Arduino MCU controls motor drivers and ultrasonic sensors.

System Functionality

Arduino Navigation

Navigation for the robot would be controlled by an arduino nano running code that takes input from the ultrasonic sensors and outputs motor functions accordingly. This design is implemented with three ultrasonic sensors for front, left, and right side distance detection. Below is a mockup of the wiring for the ultrasonic sensors and also shows the power supply input for the sensors.

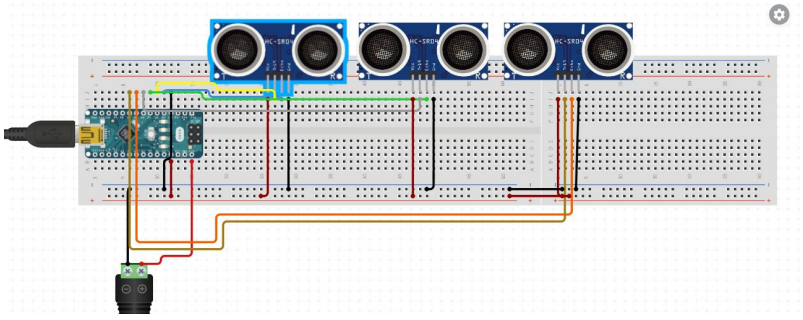


Figure 1. Ultrasonic Sensors Schematic

The ultrasonic sensors have a trig and echo pin, the trig pin is to send an ultrasonic signal while the echo pin is the receiver of the returning signal. With the time it takes the signal to

return the sensor, we are able to discern a distance and furthermore receive an accurate description of the surroundings of the device. Using all three ultrasonic sensors, the robot is able to detect obstacles that are right in front of itself. With the assistance from motor carriers, the robot can navigate itself safely without crashing. Below is the wiring as shown on the robot, it also shows the motor controllers circled in red

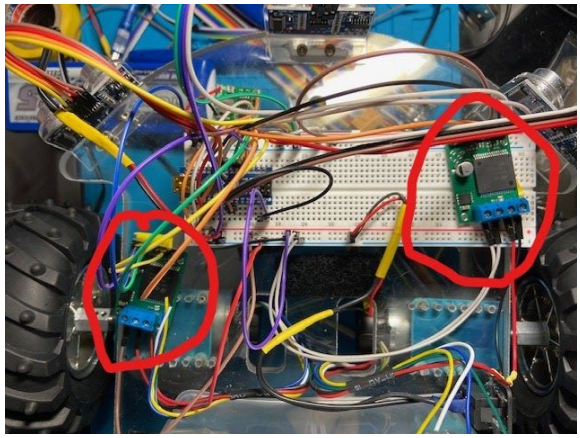


Figure 2. Motor Carriers

Following flowchart shows the basic control flow of the Arduino microcontroller.

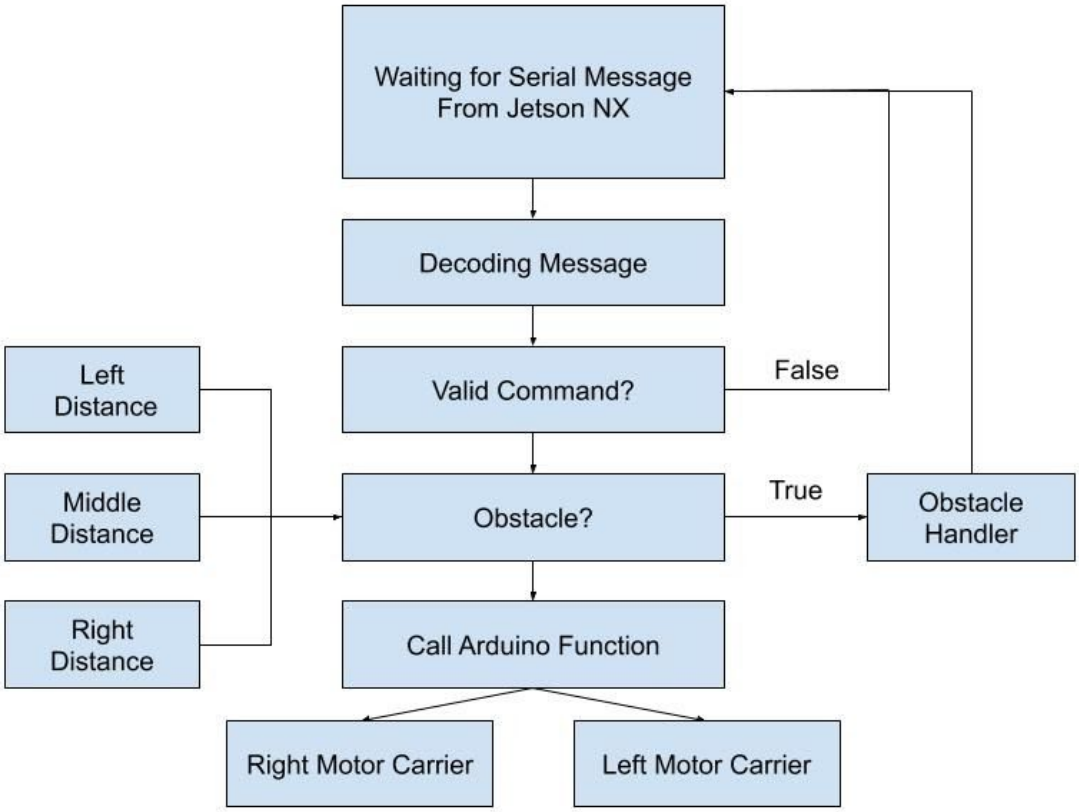


Figure 3. Control Flow of Arduino

This report shows the detailed obstacle handler implementation with black wires representing the default state, and blue and red representing the boolean states.

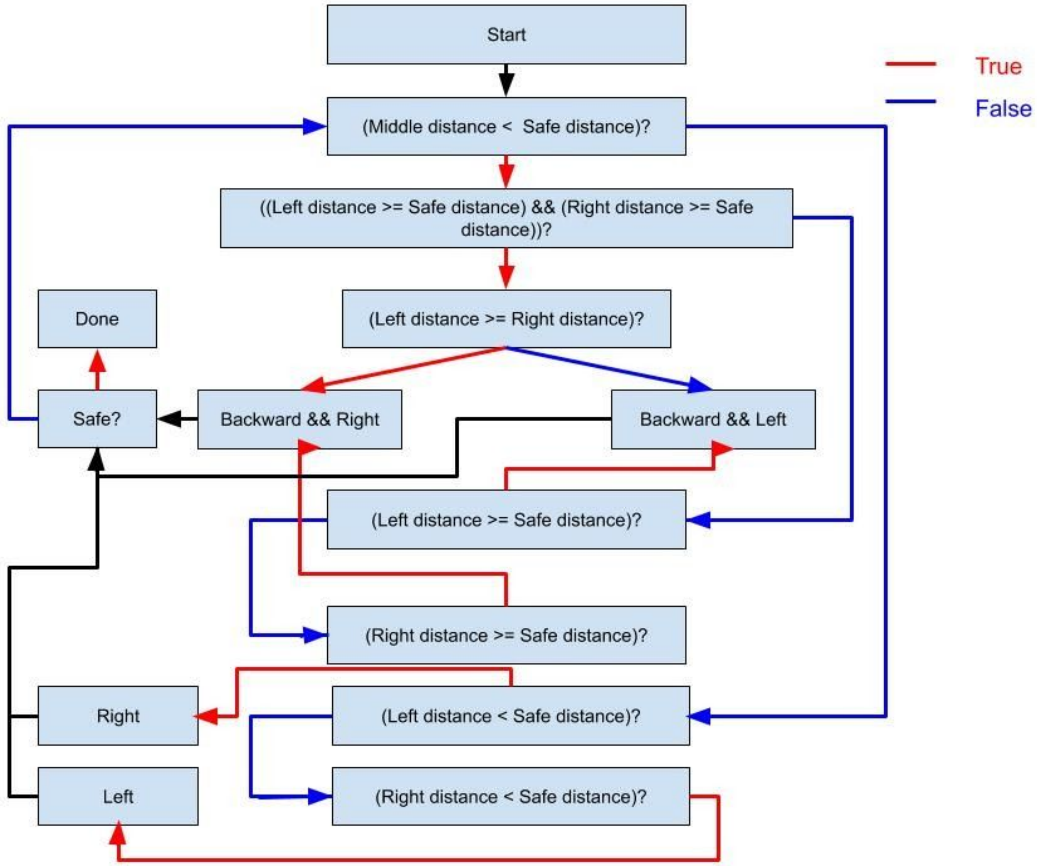


Figure 4. Obstacle Handler Algorithm

Camera Functionality (Facial Recognition and Object Following)

The main function of the camera was to enable group users to use this as the ignition to the robot. Without the same admin level image compared to the live camera image, the robot will not start or take in any voice commands.

Our camera module can be found here.

Users of the robots must first use voice commands to initiate the facial recognition function to verify the identity of the user. No voice commands other than the authentication will be proceeded until the identity of the user is confirmed to be the same as the robot administrator.

The facial recognition function will be initialized and be set to default mode when the robot is turned on. Therefore, there is no delay to call the identification function as long as the camera is not changed.

Regarding the code, we utilized the face-recognition python module that was developed by Mr. Adam Geitgey. face-recognition is a very simple facial recognition api that is able to detect humans inside a frame by using relatively few training images. Our facial recognition

code was built of Adam Geitgey's facerec_from_webcam_faster.py. For detailed algorithms and code implementations, please visit the github repository that is linked above.

Figure 5. Facial Recognition

The second function of the robot camera is to detect and follow tracking objects. The tracking objects will be labelled with different apriltags. Once the robot receives tracking commands from Alexa, it will switch the camera to tracking mode, searching for the corresponding object and moving itself to a relative close position with respect to the object. The shortcoming of mode switch is that we would not be able to use the camera to verify contemporary users, because the user variable will stop updating once the mode is switched.

Before implementing the tracking algorithm, we first built a regression model that approximates the distance between the robot and the object. We placed the robot at fixed distances from the object, then used the camera to measure the average euclidean length of four edges of the tag in pixels (1. Tags should have the same size, otherwise the model must be rebuilt. 2. Coordinates in pixels are extracted using the Apriltags python module developed by pupil-labs). Using these data samples, we built a regression model with the highest R^2 value.

Distance (cm)	Average Edge Length (pixels)
160	12.31
150	12.55
140	14.03
130	15.36
120	16.78
110	17.86
100	19.23
90	21.15
80	25.01
70	27.32
60	34.22

$$Distance\ (cm) \approx \frac{2148.2}{edge\ length^{1.02}} \quad R^2 = .9938$$

Now, we have the distance between the robot and the object. The tracking algorithm in flowchart is shown below. There are few criterions must be noticed. The idea of centered does not mean the center of the object is at exactly halfway of the frame, instead it has a large tolerance. Prev_dir assists the robot to retrieve the object in case the robot loses tracking the object due to overstepping. Finally, the tracking mode can be turned off via voice command "Stop_Follow".

One team member used the Raspberry Pi Camera while two team members used the V2/Logitech Webcam. Very little work was done to code modification to adapt functionality to the different cameras using the GStreamer pipeline. The camera module was written in python using some crucial libraries including Apriltags, OpenCV, and face_recognition were very helpful in completing the camera module. In addition to these fantastic libraries, we also referenced Arian Rosebrock's Apriltag with Python for starting code.

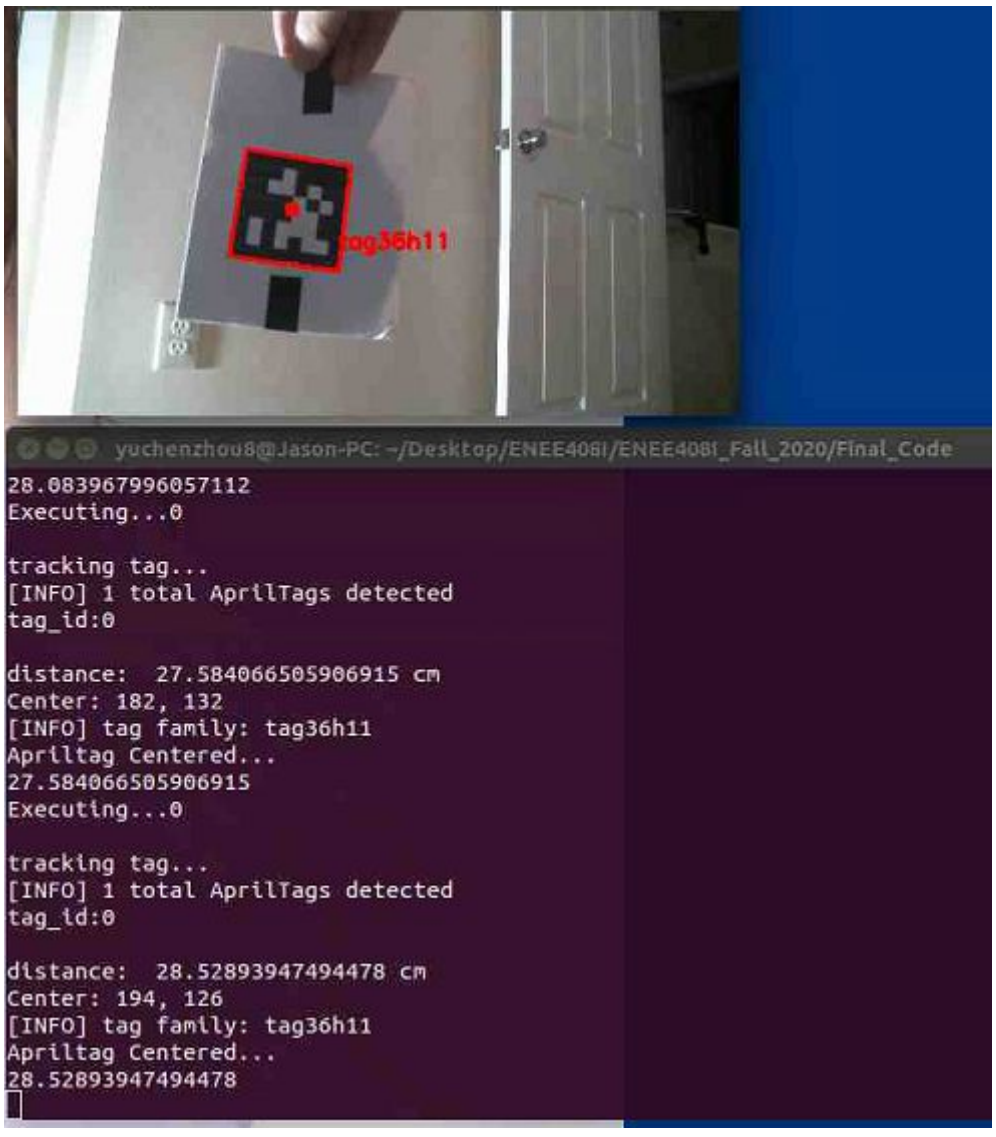


Figure 6. Object Tracking

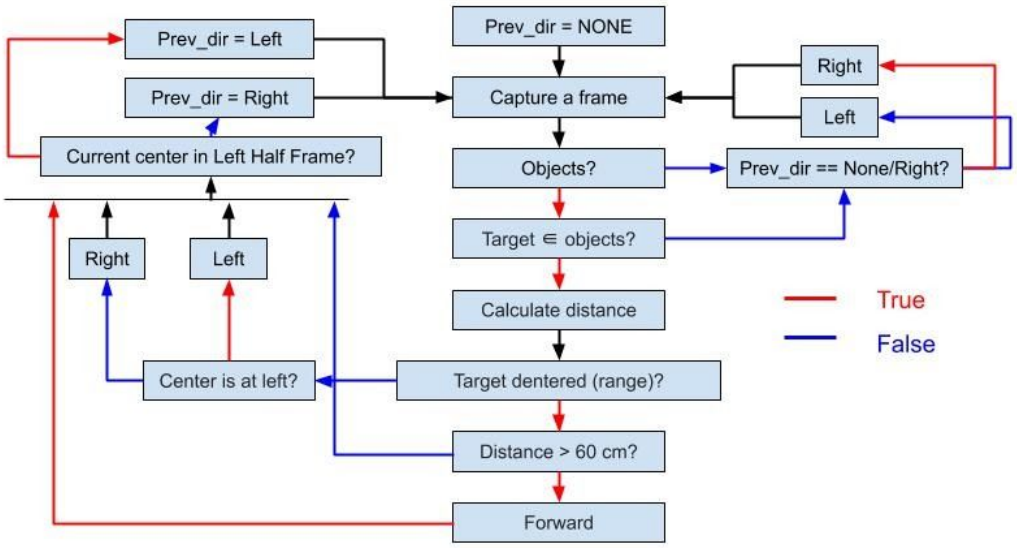


Figure 7. Object Tracking Algorithm

Communication Server (Websockets)

Communication among robots was achieved using the websockets module. Websocket is a computer communication protocol, providing full-duplex communication channels over a single TCP communication. In terms of coding, we initialized a WebSocket connection to a server, sending and receiving data on the connection. The server is created using the following code. Once the server is ready, robots can register as clients to the server via an endpoint generated by ngrok using http channel 5001.

```
client_Yuchen = RobotChatClient('ws://864e487b8730.ngrok.io', callback=test_callback)
```

After that, the robots can send and receive messages via the chat server. There is a default message handler function called test_callback. We use the test_callback function on each robot to interpret the message, as well as to send responses back. Messages include sender, receiver, type of message, and the corresponding contents. Robots will use received commands to complete the corresponding task.

This is the example of our test_callback function. It checks the receiver and the command before executing.

```
def test_callback(message_dict):
    global check_status
    print('Received dictionary {}'.format(message_dict))
    print('The message is type {}'.format(message_dict['type']))

    if message_dict['type'] == 'new_user':
        print('The new user is: {}'.format(message_dict['Username']))

    elif message_dict['type'] == 'users':
        print('Number of users: {}'.format(message_dict['count']))

    elif message_dict['type'] == 'command':
        if message_dict['target'] == default_user or message_dict['target'] == 'all':
            print('Command target: {}\n'.format(message_dict['target']))
            print("The command is: " + message_dict['command_name'] + '\n')

            if message_dict['command_name'] == 'is_online':
                if(group_status[default_user] == 'online'):
                    message = 'Yes'
                else:
                    message = 'No'

                client_Yuchen.send({
                        'type': 'Response',
                        'sender': default_user,
                        'message': message,
                        'receiver': message_dict['sender']
                        })

            elif message_dict['command_name'] == 'follow':
                print("{}'s robot is following him...".format(message_dict['target']))
                cf.function_index = 2

            elif message_dict['command_name'] == 'Dance':
                ser.write("Dance".encode())


    elif message_dict['type'] == 'Response':
        if message_dict['receiver'] == default_user:
            check_status = message_dict['message']
```

Figure 8. Example Test_Callback

Amazon Alexa

The Amazon Alexa played a major role in our core systems implementation. Voice controls ranged from user authentication, robot navigation, robot startup, hibernating or shutting down the robot. The implementation was made possible through a module called flask_ask. We wrote functions to control every aspect of the robot as seen below:

```
import serial
from flask import Flask
from flask_ask import Ask, statement
import threading
import camera_function as cf
from robot_chat_client import RobotChatClient
import time

ser = serial.Serial('/dev/ttyUSB0', 9600)
app = Flask(__name__)
ask = Ask(app, '/')

user_name = "Unknown"
default_user = "jack"
check_status = None
group_status = {
    'jack' : 'offline',
    'Andrew' : 'offline',
    'Clovis' : 'offline',
}

@ask.launch
def launched():
    speech = "Welcome, Harry Potter is now activating..."
    print("launch...")
    group_status['jack'] = 'online'
    return statement(speech)
```

```
@ask.intent('Stop')
def stop():
    speech_text = 'harry potter has stopped. I am dead'
    ser.write('Stop'.encode())
    print(ser.readline())
    return statement(speech_text).simple_card('Muscles', speech_text)

@ask.intent('Self_Driving_MODE')
def self_Driving():
    if(user_name == default_user):
        speech_text = 'harry potter enters self driving mode, I am feeling more energized than ever'
        ser.write('Self_Driving'.encode())
        print(ser.readline())
    else:
        speech_text = 'You are {}, you are not my master'.format(user_name)
    return statement(speech_text).simple_card('Muscles', speech_text)

@ask.intent('Face_rec')
def face_rec():
    global user_name
    user_name = cf.name
    if user_name == default_user:
        speech_text = 'You are {}, welcome!'.format(user_name)
    else:
        speech_text = 'You are {}, emmmmmmm!'.format(user_name)
    return statement(speech_text).simple_card('Muscles', speech_text)

@ask.intent('Lock')
def lock_robot():
    global user_name
    user_name = "Unknown"
    speech_text = 'harry potter has locked itself'
    return statement(speech_text).simple_card('Muscles', speech_text)
```

Figure 9. Alexa Commands

The code snippet above shows multiple functions of which can be separately executed after a good response from the Amazon Alexa development console. After a successful response is received, the module then passes the instruction through serialization from the Jetson to the Arduino and the requested instruction is executed. To send the correct command, users must say the invocation name of the robot and intent's utterances loud and clear.

Communication between the Alexa device and the Amazon Alexa development console is executed through a two way channel. First, initiating a secured port through "./**ngrok http 5000**" on Jetson NX, this opens a secured port communication using an IP address which is fed into the Amazon Alexa development console. Second part is accomplished through the use of

@ask.intent command in the code snippet above. This matched the command and received a response. If the command given in the first step does not match any skills in the amazon alexa development console, it will indicate no match to any skills, otherwise it gives a response the user requested.

For a more detailed tutorial on how to use Amazon Alexa to send voice commands, please visit the following repository.

Demo Video

- Full_Demo

This video demonstrates different functionalities of our robot, including voice command, communication between robots via a chat server, facial recognition, and apriltag tracking. All features are performed as we expected. However, the apriltag tracking algorithm is not optimized because the robot only turns toward right due to the default direction. Therefore, if the apriltag is not detected in the camera frame (this happens frequently because the apriltag algorithm requires all four corners for detection. Therefore, if the robot oversteps due to the default step angle and one corner of the tag vanishes in the frame, the robot will lose track of the tag), the robot has to take a full round to come back, which is inefficient. The optimized algorithm was implemented after recording this video, and its demo is shown in the next video

- Tracking_Demo

This video shows the performance of the optimized apriltag tracking algorithm. The program will save the moving direction of the previous step. If the contemporary step causes the robot to lose track of the apriltag, which is an ordinary situation in the program because the apriltag module is very sensitive to the edges of the tag, the python program will send a serial message to rotate in the saved direction.

Challenges

We faced several challenges during the course of this project ranging from implementation to collaboration. First, Implementation was a little tricky with some modules such as the motor control module written in c++ “MotorControl.h”. For one team member it could only implement for one wheel and for the other the other team members it worked as expected. The team member went through the wiring and software module and it was still uncertain what caused the right wheel to keep rotating after a hult command was passed. Another implementation challenge the team faced was adapting voice navigation to the alexa through serialization. The serialization module was developed separately from the alexa flask_ask. However, the voice commands had to be linked to the commands passed to the arduino and these two modules had to be combined which was very challenging.

Working as a team was by itself very challenging with team members having to work on their separate robots. There were some disconnects at the beginning of the semester on how we as a team were going to work on this project, the scope of the project, timetable of milestones, and division of work. As the course progressed we were able to work out a schedule and a weekly video meeting to track our progress on the milestone which was very helpful in meeting the overall course objectives.

Feedback/Contributions

Yuchen Zhou:

Without a doubt, ENEE408I is for sure one of the most enjoyable classes I took in my college career. In addition to constantly applying what I learned from previous semesters to our robots, I have acquired many new techniques/knowledges related to robotics from lectures given by Dr. Blankenship and the course sponsors, and from the building process. From this class, I learned the importance of procedural processes, how to implement and test tasks separately then integrate them into the final design. In terms of the class itself, the freedom that we can work on our target tasks was also incredible. Most important, the topic of companion robot this semester really enlightens me on what I want to study and why I want to study Robotics in future graduate school. I think that AI robots will play significant roles in our houses in future, replacing humans in many houseworks and the role of family members when they live far apart. For most of the time in this class, I was constantly learning something new, and I really felt appreciated because they are things that I can apply in my later career.

Due to the COVID-19, this course was constructed completely different from the previous semesters. Thus, to make achievements, communication among team members became exceptionally important. I think a meeting that lets members know each other is very crucial at the beginning of the semester, as it helps members to know the skills of each other. Also, there are moments when we are gathered to try to overcome some tasks, but it ended up taking too much time away from other things. I think it is normal in the actual engineering career, and this class experience helped us to learn how to become more efficient at these types of team works. Overall, I think this is a meaningful course for people like me who want to get into the field of robotics, as it is a very entry-level pathway to build a robot from scratch.

As a member of the team, I wrote and tested the scripts for navigation, facial recognition, object following, and communication among robots, including integrating all codes together for the final demo.

Paulo Sousa:

ENEE408I has been a magnificent learning experience that has been a pleasure to take. Through the building process I would gain valuable experience in wiring and building robots, this was the most fun experience in the class. Next we would develop coding skills to both do navigation and Alexa integration. This development of coding was really interesting as I had

never thought about integrating voice commands into a program, while this was difficult at first we were able to eventually send these commands to each robot.

This class would also be a different setting as due to covid we would all work through online video chats. This difficulty was overcome easily with the ability to work together virtually and using online chat sites to keep each other informed.

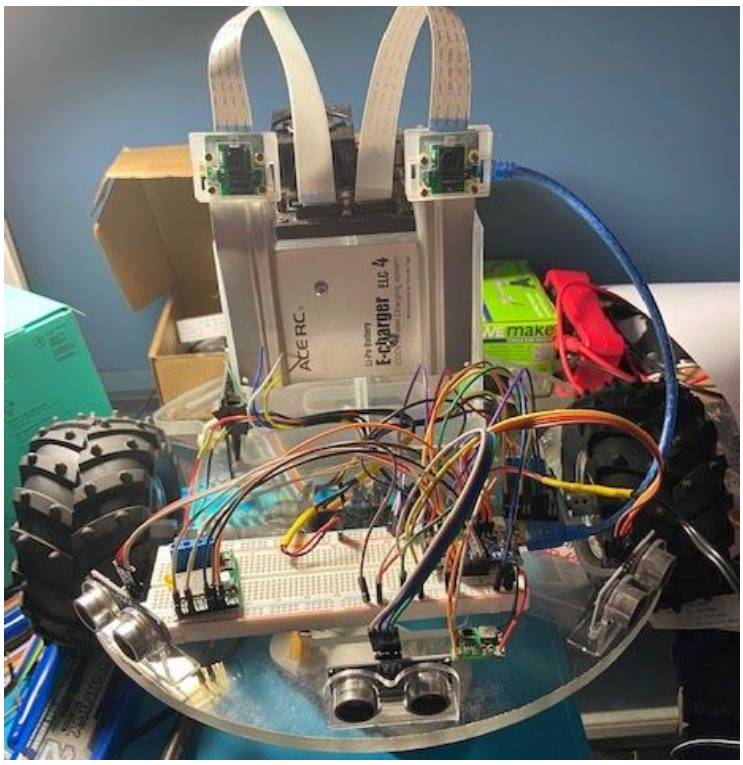
Clovis Akobong:

Fall 2020 has been a really challenging semester due to the COVID-19 pandemic. Despite this, the resources, training, and hands on implementation of ENEE 408I was invaluable. I Had never worked on an Arduino before, so this was a big learning step to implementing knowledge acquired through my engineering studies. The lecture sessions were very educational and despite being remote, the Professor and TA were able to solve all issues students encountered during lecture and laboratory time. Logistics was not an issue for me since all parts I requested were provided on time.

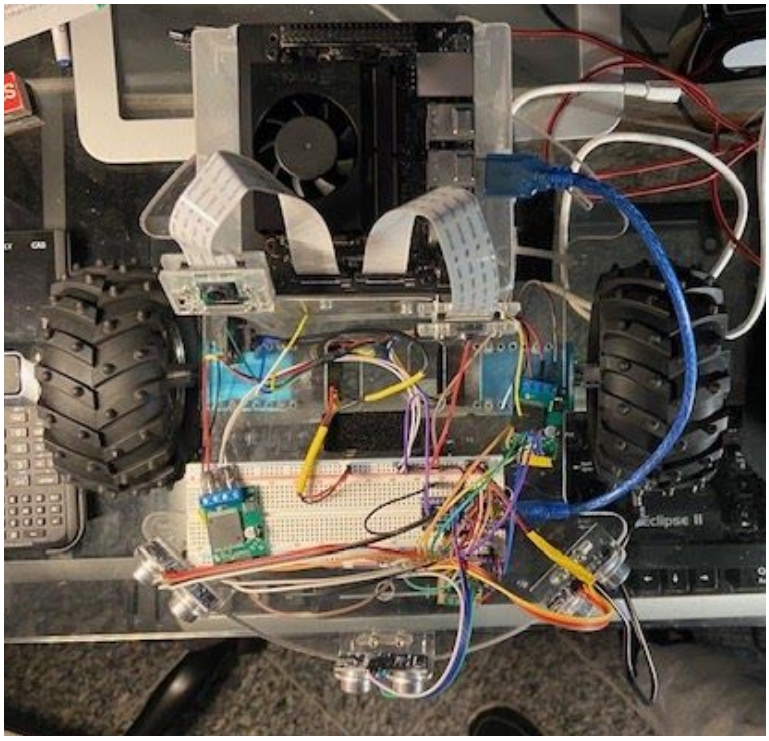
Some minor issues I think if adjusted can really improve the nature of the course include soft skills needed in the lab such as soldering. It was the first time I soldered and with the online nature of the class I made some mistakes that took some time to adjust. Lab time was well utilized and I think was effective through the breakout rooms, we met several times out of the regular lab time to catch up on robot work which i think should be really encouraged since the online nature of the course requires more lab time for each team member to catch up with the other team members.

Robot Final Designs

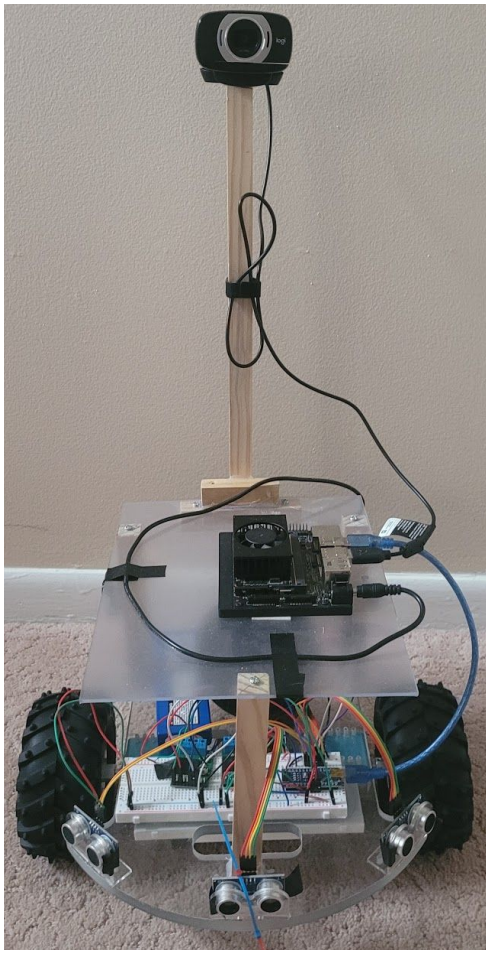
Paulo Sousa’s Robot Front View

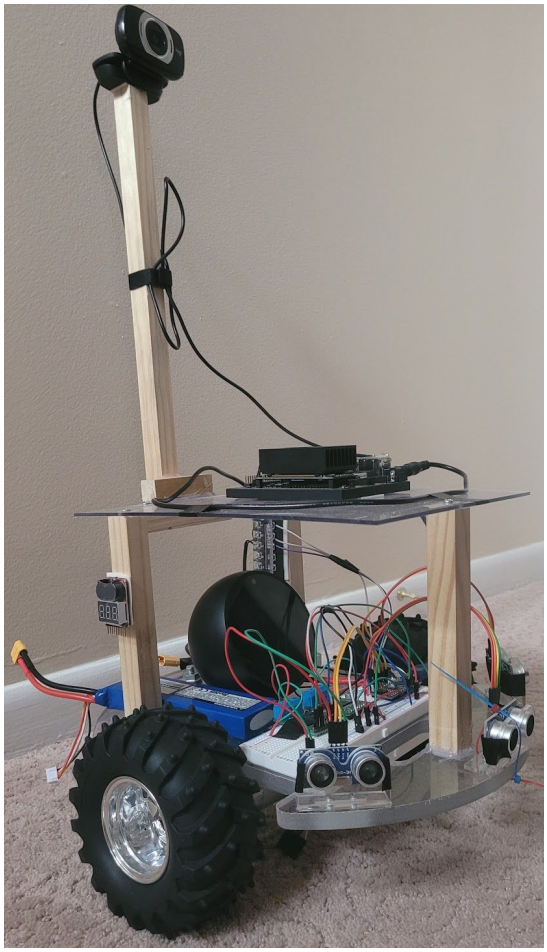


Paulo Sousa’s Robot Top View

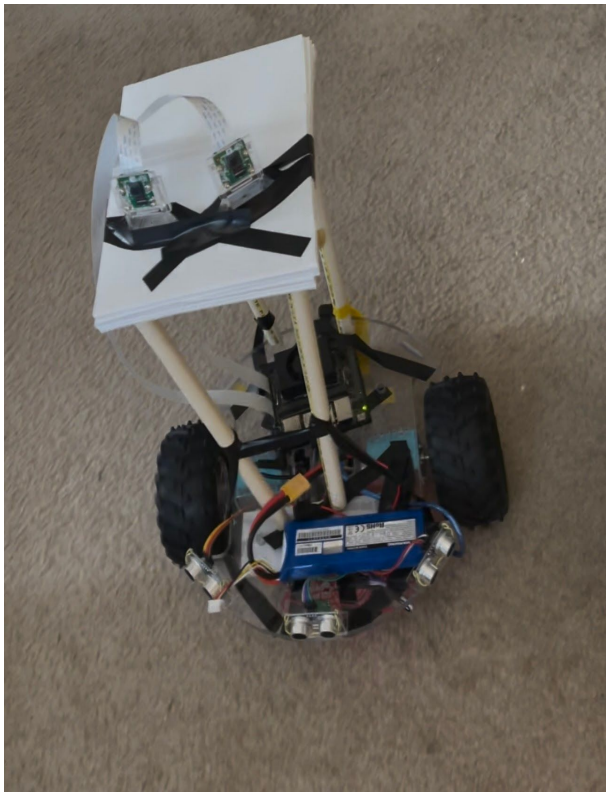


Yuchen Zhou's Robot





Clovis Akobong’s Robot



Conclusion

This robot would challenge the team to expand on programming, computer vision, and practical skills that will help us in our future career as engineers. This build would include understanding how to use an ngrok server, using camera functions and multiple libraries where functions are used. Our team would seem to work well through zoom, but believe the class would work better with in person lab time to help coordinate the robots better. Another recommendation is to have more lectures on facial recognition, they were really interesting and could help future classes in their facial recognition development. Working remotely made the system integration part difficult as our robots would be running slightly modified code and getting the robots to communicate was difficult. Also Paulo was one of the students to receive his robot in the mail so had lost some time to build and develop the robot.

Overall this class was a pleasure to take, even with covid-19 we were able to complete the challenge while adding a few of our own. This project would also teach us how to work in teams efficiently as we would meet for hours helping each other with their problem. This type of problem solving is found in the workplace, and was emulated perfectly with this class.