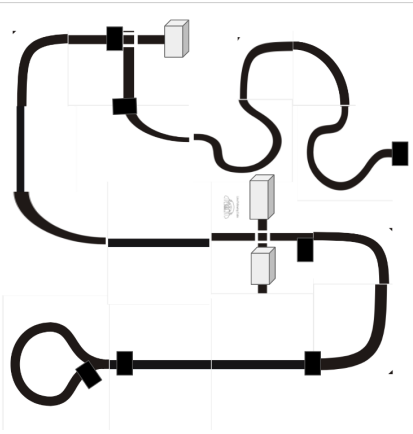
Davis 109006201
https://github.com/davizjc/Embedded-System-Lab-Final_Project.git
周志偉

Final project

- Introduction
    - There are basically four part in this project map, marker , ping and MQTT

- Set up
    - PwmIn servo0_f(D9), servo1_f(D10); for the servo
    - PwmOut servo0_c(D11), servo1_c(D12);
    - BusInOut qti_pin(D4,D5,D6,D7); for QTI
    - DigitalInOut pin8(D8); parallax_laserping ping1(pin8); For the ping
    - BBCAR and pwm library will be included in this lab
        - Since I wire the servo and the QTI sensor the opposite way from lab 13 my code will be kinda different a little bit. The four qti pins will be read and will output a pattern. When the pattern is 8 it moves right, when it 1 it goes left, and when both of the middle sensors read 6 it goes forward. Since my servo is wired in the opposite direction for it to go forward I use car.goStraight(-50); instead of car.goStraight(50);. To fix when the pattern (0) doesn't read anything or when the 4 sensor is completely out of the track I set the bb car to move backward.

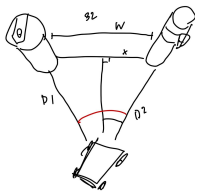
- Map design



    - This is the map that I have designed. It will first begin with a circle and along the way there will be black marker to tell where we are. There are 3 main tasks that my project includes. The first part is the circle after the car goes around for one time it will reach marker one which then controls the car to turn left and continue to the other checkpoint in the map. The second part is when it gets to checkpoint 4, which will have a marker that is 7, the car will scan for the object on the left and right and output the distance between them. The third part is the branch where it will first go forward and detect and if it detects an object it moves backward until the marker in front and turns right after word. The program will end when it reaches the final marker and print the total distance.
        - The distance is calculated by using this formula = abs(((car.servo0.angle)*6.5*3.14/360))

- Marker
    - There are many markers in this map. There are a total of 8 markers, and some of them are special markers like the pattern 7 marker which will make it scan left and right.

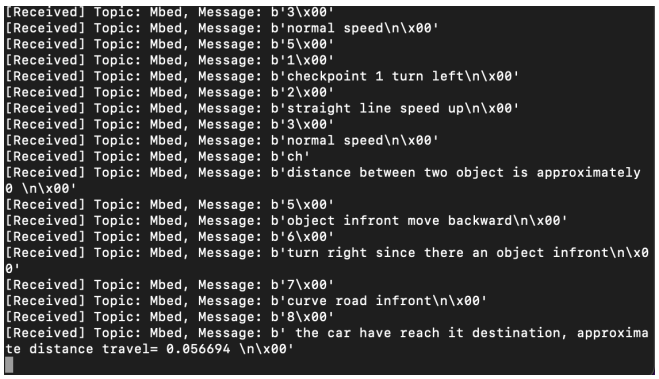
- Ping
    - At pattern 7 the car function checking_object() will be called. The car will then turn left first and scan for the object. At this marker the BB car will be around the middle or in between the two objects. The ping will be read, which will read the background. I wrote my code so that the BB car will turn left and when it detects an object closer than 50 it will stop and print out the actual distance. Then it will turn right after, but during this the thread will sleep for 2 seconds so that it will turn the car until the ping detects the background instead of an object. It will continue to read the ping until the car stops at a second object and prints the second object distance. Then the car turned left so it was on the track properly before going forward again.
        - To calculate the distance apart I measure the red angle when the car detect an object on the left until the right. Then I take that angle and divide by 2 assuming that the car is in them middle if i split the angle in half I can use sin { black angle) multiply by d2 to get the X and the W should be x multiple by 2



        -

    - The second ping will be when it reaches the branch object_infront() will be called and the car will go forward first but when it detects an object close to it. Instead of moving forward it moves backward until it gets to the last marker and turns right.

- Mqtt
    - 2 library will be added and use
        - Import ISM43362 library for MQTT part
        - Import MQTT library for MQTT part
    - Command needed are
        - python3 final.py (to run python run in /Mbed Programs/final/)
        - /usr/local/opt/mosquitto/sbin/mosquitto -c ~/Mbed\ Programs/mosquitto.conf - To run mosquitto (run in from the root directory)
    - I implement the program so that it will connect to SSID, then connect to TCP network…, then IP ADD, and Port. If everything works properly then Successfully connected! Will be printed. When the car reaches every marker I will let the function sendmsg() send the checkpoint over to the python program. Some messages like distance travel or car instruction will also be sent over.



- The message will be something like this

- Problem facing
  - The main problem I faced is that the car is not really consistent, most of the time it would work but sometimes the car would get stuck for no reasons and a little touch on it would make it continue moving. Track also plays an important role in this project since the car can read a wrong sensor and make a different move. Sometimes the car doesn't move how I wanted to miss the sensor by a little bit. This is the main problem and I can't manage to completely fix this problem.
  - Ping is not consistent; sometimes it reads a weird number and sometimes it doesn't.

VDO will be added to the drive to show that it work
https://drive.google.com/drive/folders/15Yi-zfVJ5OpmTZ3TurnlLc9V071qkRZP?usp=share_link