– TANK BOT –
PROJECT REPORT

BY: NEBIYU TADESSE

MCEN 4115: Mechatronics

Date: 05/04/2020

Mechanical Engineering
University of Colorado Boulder

Table of Contents

Introduction and Project Requirements --- [3]

Code -- [4]

a. Flow Diagram -- [4]
b. Libraries Explanation --- [5]
c. Systems Integration -- [6]

Results -- [7]

a. Progress and Results --- [7]
b. Future Work -- [7]
c. Advice for future kits users --- [7]

Citation -- [8]

Appendices --- [8]

a. Code --- [8]
b. Other technical support -- [8]

I. Introduction and Project Requirements

The original project for this course was to build and program a Police Academy Robot that can travel through a custom-made map and strike down a simple enemy target, by shooting it down with NERF foam bullets or by other means that fit the scenario and were safe. However, considering the recent pandemic, and school shutting down, the project was changed such that it was more feasible for student to work from home.

I opted in for the individual robotics kit project where the professor bough us a robot kit from online, and we tried to do closely approximate the goal of the original project but with the hardware at hand. The requirements of this project, as presented by the professor, go as follow:

1. Lay out a course with an intersection or two using tape on the floor or cardboards to make walls
2. Get you robots to navigate through your course, using vision or sensors (not timed or hardcoded)
3. Find an object you think your robot can easily identify. Place it at the end of the course.
4. Put a lance (stick/ ruler/ extension) on the front of your robot and have your robot drive up to the object and knock it down with the lance.

The robot that I got was the Raspberry Pi Tank Smart Robot Kit by Yahboom, **Fig.1,** and the first step of this project was to assemble to robot since it came disassembled, **Fig.2**, then make a timeline, **Fig.3**, in order to complete the project on time.

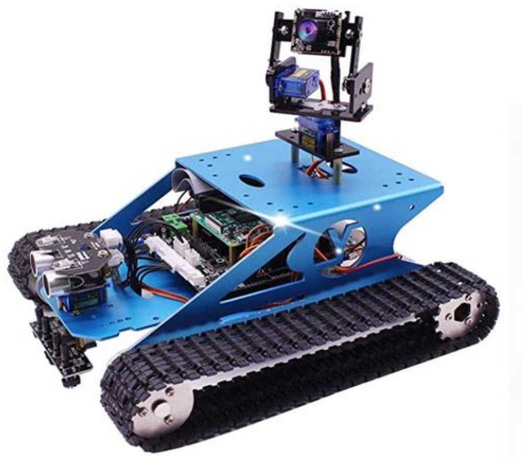


Figure 1: Robot Kit from Yahboom

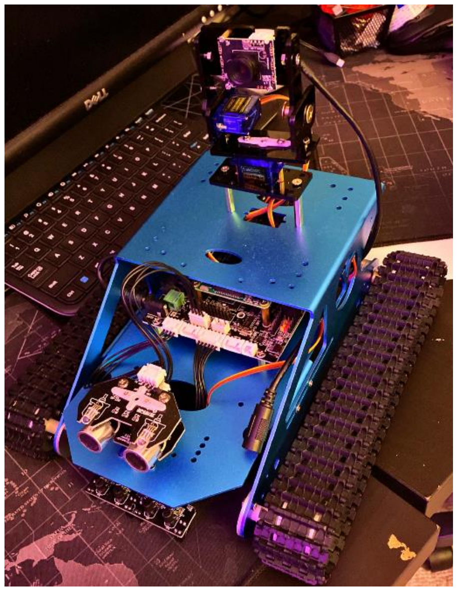
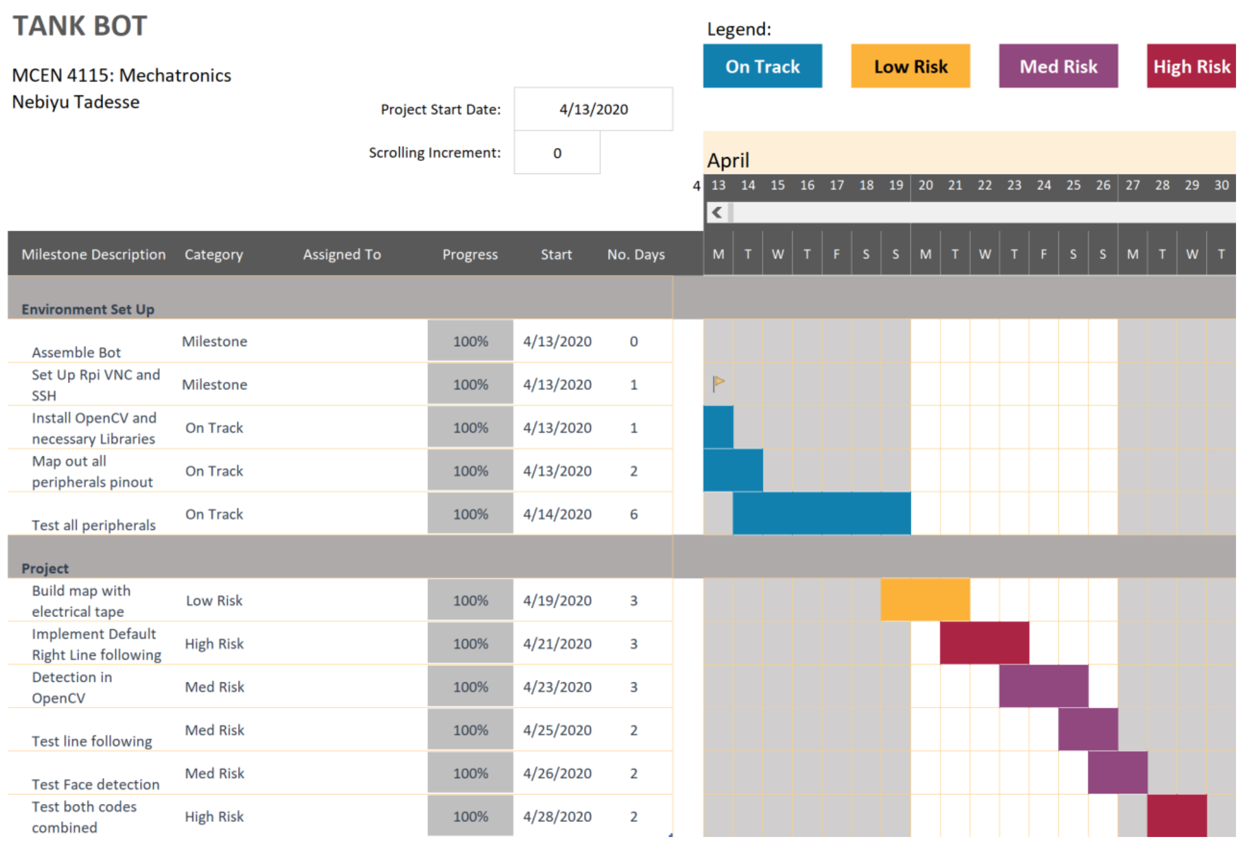


Figure 2: Assembled Robot Kit



TANK BOT

MCEN 4115: Mechatronics
Nebiyu Tadesse

Project Start Date: 4/13/2020

Scrolling Increment: 0

Legend: On Track, Low Risk, Med Risk, High Risk

April

Milestone Description	Category	Assigned To	Progress	Start	No. Days
Environment Set Up					
Assemble Bot	Milestone		100%	4/13/2020	0
Set Up Rpi VNC and SSH	Milestone		100%	4/13/2020	1
Install OpenCV and necessary Libraries	On Track		100%	4/13/2020	1
Map out all peripherals pinout	On Track		100%	4/13/2020	2
Test all peripherals	On Track		100%	4/14/2020	6
Project					
Build map with electrical tape	Low Risk		100%	4/19/2020	3
Implement Default Right Line following	High Risk		100%	4/21/2020	3
Detection in OpenCV	Med Risk		100%	4/23/2020	3
Test line following	Med Risk		100%	4/25/2020	2
Test Face detection	Med Risk		100%	4/26/2020	2
Test both codes combined	High Risk		100%	4/28/2020	2

Figure 3: Project Timeline

II. Code

The coding for this project was done in python and is broken into 3 separate files, which are:

1. Module that includes functions for all low-level interfaces
2. Module that includes functions for all high-level Algorithms and main function
3. Module that sets up all functions and uses them as they were intended to be used to allow easily running tests on all modules and functions created.

a. Flow Diagram

The flow diagram of the project is quite straight forward,

Flow Diagram Posted on my GitHub: https://github.com/N3b3x/TankBot

All it does, is follow the line it is on until it arrives at an intersection. Once it is at the intersection, the robot uses its camera to check all three directions, forward, left, and right to see if the enemy can be seen. If the enemy is seen in a certain direction, the robot turns toward that direction till it knocks it down. However, if the enemy is not seen, it will always try to default to a right turn, if not possible, go forward, and if that is not possible either, it will take a left turn. If we reach a dead end, the robot will just turn 180° and continue its quest to eliminate the enemy.

b. Libraries Explanation

The external libraries used for this project are:

1. Open-CV – which is "a library of programming functions mainly aimed at real-time computer vision"[1]
2. Pigpio – which is a "library for the Raspberry Pi which allows control of the General Purpose Input Outputs (GPIO)"[2]

Open-CV was chosen because it is open source and has functions that allow to easily implement computer vision into the project. It also has a big user community behind it.

I chose Pigpio, pronounced as pi gpio and not pig pio, because of strong focus on meeting timing requirements for PWM pulses and others which is very useful for the servos and ultrasonic sensor. More can be found online for both.

c. Modules Explanation

i. Robot_control.py

The robot_control.py file wraps up all the low-level function into easy to use functions. The following functions are what it holds:

1. Init()
2. moveLeftForward(dutyCycle)
3. moveLeftBackward(dutyCycle)
4. moveRightForward(dutyCycle)
5. moveRightBackward(dutyCycle)
6. moveForward(dutyCycle)
7. moveBackward(dutyCycle)
8. rotate(dutyCycle)
9. stop()
10. ping(angle)
11. setLED(R_DUTY,G_DUTY,B_DUTY)
12. setUltraServo(angle)
13. setCameraHorzServo(angle)
14. setCameraVertServo(angle)
15. setCameraServos(horz_angle,vert_angle)

16. readButton()
17. readFarLeftIR()
18. readLeftIR()
19. readRightIR()
20. readFarRightIR()
21. setBuzzer(duty,frequency)
22. setServoAngle(servo_pin,angle)

The naming should make sense on their own but if curious, all of the functions are well commented and available on my github account.

ii. Camera_test.py

The Camera_test.py file runs a test script that shows you if your camera is working and can detect faces. The face detection is done using haarcascades that can be found on Open-CV's github page. I am using the default frontal face detection cascades.

iii. Ultra_test.py

The Ultra_test.py file runs a test script to see if the ultrasonic sensor is working. It should continuously output a distance read as long as the file is running.

iv. Main.py

The main.py file holds all of the high level algorithm and the main function that runs everything together. The following functions are what it holds:

1. readIR()
2. followLine(MODE)
3. turn(dir)
4. detectFace(angle)
5. lookForFace()
6. detectForwardLine()
7. main()

All of the names of the function should give a quick overview of what they do but if more is needed to be known, the code itself holds an in depth commentary on almost all of the lines, thus that should be where to go first.

v. Main_functions_test.py

The main_functions_test.py file runs a script that allows the different functions in the other files to be run as how they are intended to in order to guarantee their functionality is expected. This is great for unit testing each as that is an important step in every library writing in order to catch mistakes early and make debugging much easier.

vi. Stopit.py

The stopit.py file runs a one line script that just stops the motors. This was implemented in case you stop you're code by force without it fully finishing and the motors were left running. It came in pretty handy during the test phases of the project

d. Systems Integration

System integration was not terrible at all. I was able to most of the code without even laying my eye on the robot once. The biggest concern that comes when integrating the code with the robot is that the intended speeds for each wheel may not be reached for a multitude of factors, and there is no feedback loop to check on em. Adjusting the wheel speeds was one of the biggest time-consuming part of all.

III. Results

a. Progress and Result

The code is currently fully working, and a video has been made and uploaded in conjunction to this report, to show its capabilities. I will also post it on my YouTube channel, "Nebiyu Tadesse".

b. Future Work

This project can be greatly expanded into traveling through a map made with walls using the ultrasonic sensors and camera, instead traveling through a map using line following. A custom-made robotic arm can also be attached to increase it's degree of freedom and open up more project possibilities.

c. Advice for Future Kit Users

My biggest advice for future kit users would be to add a hall sensor or some kind of sensor to measure the rotation of the wheels easily in order to have a feedback control system which could increase the kits reliability on different surfaces greatly.

Another large one would be to use the pigpio library for GPIO library, due to its better timing in order to control servos and other timings critical sensors well. The drawback of using that library is that you can't use raspberry pi BCM pin 0 and 1 which is where the ultrasonic sensor plug is attached to (It is also heavily recommended to not use those pins by raspberry pi anyways due to what they're actual use is for so it's a lowkey win for you). Therefore, get some wire jumpers and connect the trigger and echo pins to unused pin next to where the servo's wired are meant to be plugged in.

IV. Citations

[1] https://en.wikipedia.org/wiki/OpenCV

[2] http://abyz.me.uk/rpi/pigpio/

[3] D. Reamon, *MCEN 4115: Individual Kit Final Report*, University of Colorado Boulder, 2020.

V. Appendices

a. Code

Link to my github: https://github.com/N3b3x/TankBot

b. Other Technical Support

To install the two main libraries, and clone my repository just run the following:

1. pip3 install opencv-python
2. pip3 install opencv-contrib-python
3. pip3 install pigpio
4. git clone https://github.com/N3b3x/TankBot.git