**Report for CENG2400 project**

**Abstract**

**This project is aimed to program the Tiva C LaunchPad robot car in order to (test 1)run a straight line from any location and (test 2)avoiding obstacles and wall automatically. Peripherals like step motor, voltage regulator, DC motors are used and introduced in the project. It is required to integrate the system into the final product by myself, including assembling car base, soldering components on main board and connecting wires. The ccstudio and Realterm software is used. In result, test 1 is achieved and the test 2 is achieved but not smoothly. After discussion and evaluation, the problem is solved but several improvements can be made. First, the IR sensor should be used to sense all direction in every stop of the robot car. Second, higher speed should be tuned by tuning the voltage regulator, increasing number of battery and changing the PID values.**

**Chapter 1: Introduction**

The problem addressed here in the project is to program a robot car that can

Test 1 : run a straight line from any location and

Test 2 : avoiding obstacles and wall and then run maze by itself automatically.

The motivation for solving this problem is to help me learn how to use Tiva C launchPad to produce an autonomous robot and understanding almost all the basic functions of the launchpad and other peripherals like step motor, voltage regulator etc. which is vital for my further study and learning in engineering field.

My previous solution is to address the PID to expected value while both motor for wheels is running at the same speed. Besides, I address the IR sensor to sense the left and right wall, if there is no wall sensed, it will turn to left or right respectively. Instead, the first straight route used the IR sensor to sense obstacle in front. If there is an obstacle sensed, it will avoid. The last route is hardcoded. Compared with others, some of them choose to run the car faster to avoid friction on the floor and some use the IR sensor very well which can sense all wall from all directions which I think is excellent.

I achieved test 1 and I achieved test 2 but not smoothly. Compared with other approach, I address my PID value into a realistic value which eliminate the error made for the different between the left and right motor speed. Besides, full charged battery is needed for the whole project in order to attain the expected speed of the motor for the robot car. I addressed the last route from hardcode to using IR sensor to detect the block in the “END zone” (figure I) such that it makes the solution more automatic. Compared with others, my result is satisfactory because tests can be achieved in some extent. However, compared to others, technique like coding the IR sensor to test all directions and tuning the car to run in a faster speed is not applied in my solution. There are areas that can be improved in order to pass all tests smoothly and beautifully. Besides, the floor on the right-hand side of the testing path have some extent of slanting. The traction and speed of the car should be adjusted too in order to improve the result optimistically to pass the test 2 smoothly.

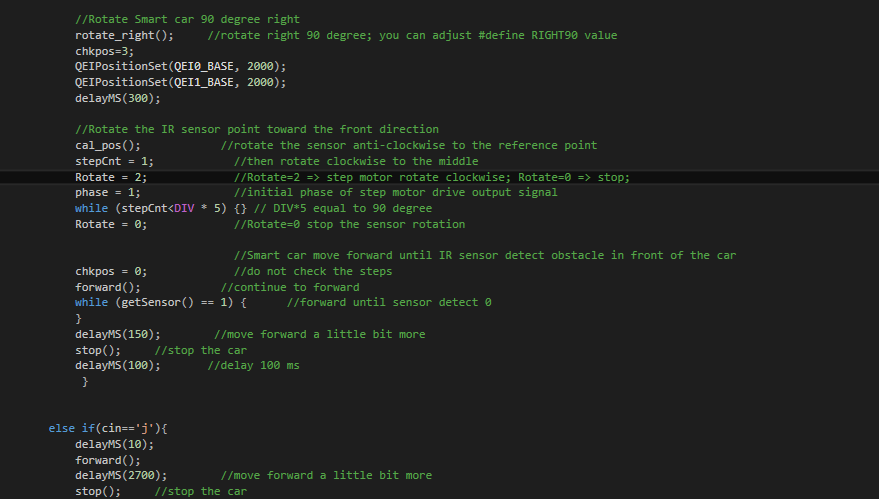


figure I

The report will be structured as follows:

**Abstract (at the very beginning of the report)**

**Chapter 1: Introduction (this chapter)**

**Chapter 2: Theory and design**

**Chapter 3: Implementation and experimental result**

**Chapter 4: Discussions**

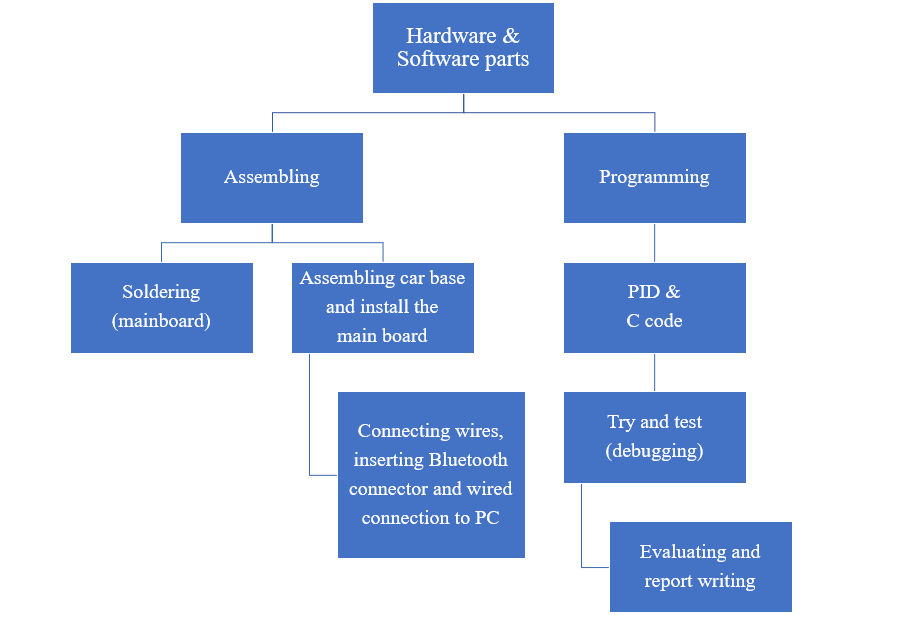
**Chapter 5: Conclusion**

**Chapter 6: References**

**Chapter 7: Appendix**

**Chapter 2: Theory and design**

Overview:



Architecture and design of the robot:

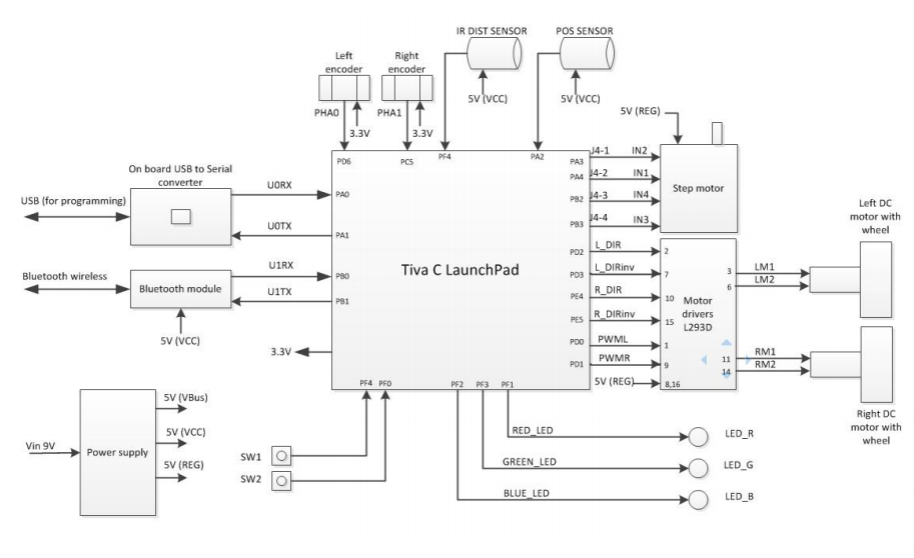


figure X

Input : POS SENSOR, IR DIST SENSOR, Left(Right) encoder (motion control

Feedback(QEI)), Bluetooth module(U1RX), On board USB to Serial

converter(U0RX), Power supply part(5V regulator & battery)

Output: On board USB to Serial converter(U0TX), Bluetooth module(U1TX), output

voltage (3.3V) , step motor values(In2, In1, In4, In3), motor speed values

(PWML, PWMR, PID, left(right) direction(DIR & DIRinv) values)

(Unused in my solution : TIVA C mainboard (SW1, SW2, LED)

Module Description:

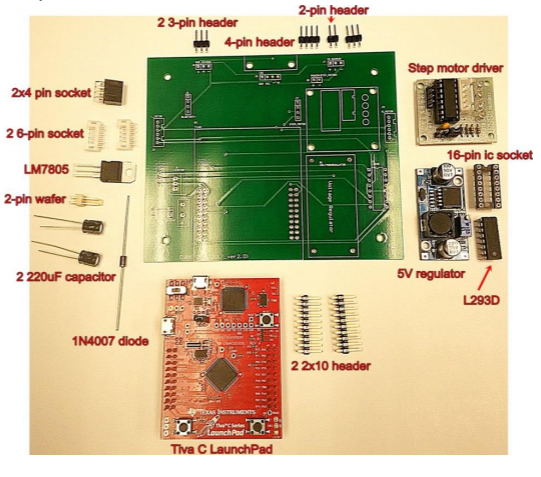


figure 2

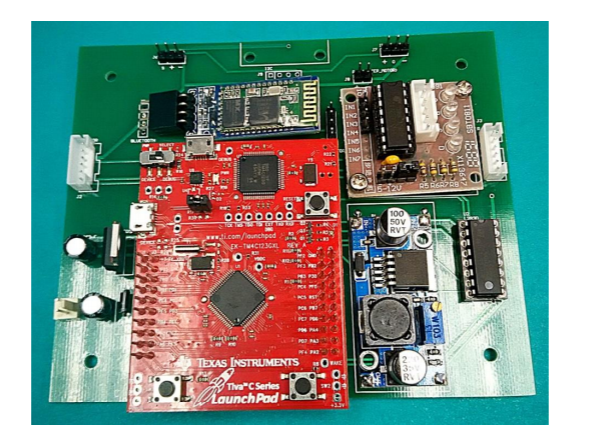


figure 3

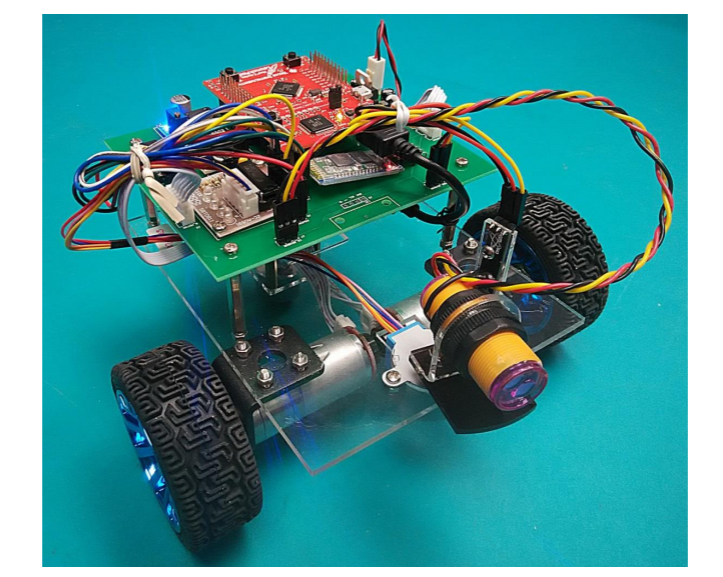


figure 4

Hardware part: in figure 2 (not yet assembled) and in figure 3 (assembled)

Step motor driver module: position control, rotation of IR DIST sensor

Voltage regulator module: (5V regulator, LM9805): maintain voltage level and adjust the amount of voltage

Motor driver IC module: (L293D) : convert microcontroller signal to corresponding output to 2 DC motors

Capacitors and diode: 2 220uF capacitor are used to store electrical energy, 1N4007 diode is used to convert AC current to DC current.

Bluetooth and USB module: to connect between the car and PC by Bluetooth and USB port wires and install program onto the board by USB port wire.

PWM & QEI module (Tiva C board): motor speed controlling

Mechanical part: in figure 4 (the car base, POS sensor and IR DIST sensor).

Car base: 2 DC motors is used to drive 2 wheels, a ball wheel is also inserted.

POS sensor: sense the direction pointing by the IR DIST sensor.

IR DIST sensor: sensing obstacles and walls.

Software: Code Composer Studio 7.1.0 Properties(ccstudio) & Realterm

ccstudio: used to program the Tiva C board on the robot car.

Realterm: used for keyboard input to command the robot car.

B1: voltage regulator

B2: Tiva C board

D1: 1N4007 diode

CX: uF220F Capacitor 1, uF220F capacitor 2 (X: 1 or 2)

U1: LM7805 IC

U2: L293D IC

S1: step motor

iii.

no assumption declared yet

iv.

See figure X

v.

Pseudo Code

Test 1:

Set IR sensor to the reference point(detected by POS sensor);

Rotate the sensor clockwise;

Initialize phase of the step motor;

Count the step when the angle = 90 degree;

Stop the rotation;}

Unable the step checking;

Move the car forward until the sensor sense obstacles in front of it;

Delay for a while to move for a while;

Stop the car;

Delay for a while;

Test2:

Move the IR sensor to the middle by step motor;

Move forward until the sensor detect an obstacle in front;

Move forward a little bit;

Turn right in 90 degree;

Move the IR sensor to the left (the reference point (~90 degree));

Move forward until no wall on the left;

Turn left in 90 degree;

Move forward a little bit;

Move forward until no wall on the left;

Move forward a little bit;

Turn left in 90 degree;

Move the IR sensor the right in 90 degree by step motor;

Move forward until no wall on the right;

Turn right in 90 degree;

Move the IR sensor to the middle by step motor;

Move forward until the sensor detect an obstacle in front;

**Chapter 3: Implementation and experimental result**

Implementation:

First, for the hardware part, I assembled all the components, including B1, D1, C1, C2, U1, U2, and all the headers, sockets and a wafer connector (11 connectors in total) onto the green circuit board. I soldered them all. And then, I inserted S1 by screws, and assembled the B2 onto two 2x10 pins header. I connected the Bluetooth chip to the corresponding socket.

Second, for the mechanical part, I assembled the 2 DC motors with encoders, 2 wheels, the ball wheel and the IR sensor on the base by M3x30, M3x35 stands & M3x5, M3x8, M3x10 screws. The position sensor was fixed on the car base by hot melting glue. The battery box was fastened by magic tape.

Third, I installed the green main board to the car base by 4 M3x5 screws and connected all the wires to the main board including the DC motor cables, the step motor cable, step motor driver 5V power and the IR and position sensor cables. the completed car is shown in figure 3.

Last, using ccstudio, I used the source code and modified the PID values and added two keyboard input commands ‘j’(for test 1) and ‘k’ (for test 2). I wrote the code for the commands according to the pseudo code part provided in Chapter 2 (Module description part v.) And then. I connected the robot car to the PC by USB port wire to install the code onto the Tiva C board. At the same time, I connected the Bluetooth chip to the PC and also another chip to the main board of the robot car. I tested many times for the PID values and did many debugging and improvement. And, finally I disconnected the USB wire and open the Realterm to use the keyboard input and the Bluetooth function to do both test 1 and test 2.

Experimental result:

Procedure:

A testing path is set on the floor. (see figure 5)

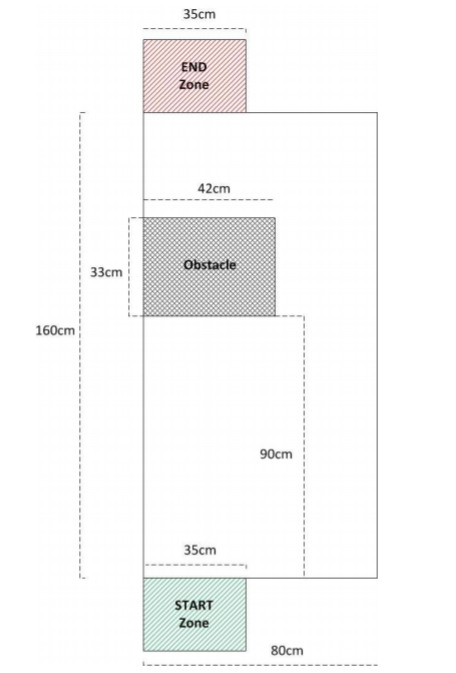


figure 5

Pressing the button ‘j’, the test 1 demonstration goes, times used is recorded.

Pressing the button ‘k’, the test 2 demonstration goes, times used is recorded.

The result in test 1 can meet my expectation. Although at the very first, I expected a very straight line run by the robot car as the PID value was set in expected speed and the two motor speeds were nearly the same. However, due to the friction on the floor and also the different between motor speeds , the PID value is set to a realistic value

#define initPWML 400

#define initPWMR 500

#define P\_left 1000

#define I\_left 400

#define D\_left 200

#define P\_right 2300

#define I\_right 400

#define D\_right 200



now. Although not a very straight line in a long run, it still achieved test 1 (160 cm) imperfect straight-line run.

The result in test 2 cannot meet my expectation. I expected the robot car can run smoothly on the right-side path of the testing path. The robot car cannot overcome the challenge of slanting of the floor very smoothly. So, it crashed into the obstacles 4 times in 5 times demonstration when running the right-side path.

Some errors of turning will occur when delayMs() is not in appropriate value.

From the result, I learnt that the program should be coded with IR sensor sensing all the directions. So, chances for colliding to the wall will be reduced. Also, the traction of the wheel should be increased to avoid slanting problem. PID values should be set beautifully (should be in the expected value if traction is enough) in order to run well in both test 1 and 2.

**Chapter 4: Discussions**

Expectations:

Test 1 is achieved. Test 2 is achieved but not smoothly.

Discussion:

I have discussed with my course mates about the results, as we need to keep changing the PID values in different times and the result is that the robot car cannot run a straight line with the expected result (expected speed and same motor speed for left and right). It is discussed and we believe that it is because of mainly three reasons.

1. the different speed between each motors
2. the different degree of slanting and friction of the floor
3. the battery level\*(the most vital reason)

Besides, for the turning parts, the degree of turning is different and some errors also occurs. We believe that it is related to the delayMs() time as each operation need different time to finish. Unexpected finishing time can result in weird situation.

Difficulties and limitations:

Hardware and mechanical part limitation: As the hardware is not standardized and cheap, it is easy to figure out errors from the hardware part and need to redo the project again and again with different hardware, say same motor model but great variety speed of motors.

Difficulties: PID values vary with different factors. So, it needed to be changed time by time or different solutions is needed to be compatible with great deviations of speeds (PID values) of the motors on left and right.

Others:

The cost of production of the design is low and is suitable for mass production indeed.

Further improvement:

1. The speed can be tuned to be faster to get better result.
2. The IR sensor should be in sensing in all direction in every stop of the car in order to make a real solution of the problem instead of hardcoding.
3. More standardized hardware should be used.

**Chapter 5: Conclusion**

The problem addressed here in the project is to program a robot car that can

Test 1 : run a straight line from any location and

Test 2 : avoiding obstacles and wall and then run maze by itself automatically.

I have achieved both tests but not very smoothly in test 2. Besides, I hardcode some parts which means it is just usable in the test but cannot be widely used. In order to have better performance and compared to others, the IR sensor should be in sensing in all direction in every stop of the car. And also the speed should be tuned faster by using the voltage regulator and more batteries.

**Chapter 6 References:** **https://www.sparkfun.com/datasheets/Components/LM7805.pdf**

<https://en.wikipedia.org/wiki/Voltage_regulator>

<https://www.pololu.com/category/120/stepper-motor-drivers>

<https://www.instructables.com/id/Using-Motors-With-L293D-IC/>

<https://www.google.com.hk/search?q=capacitor&oq=capacitor&aqs=chrome..69i57.1814j0j1&sourceid=chrome&ie=UTF-8>

<https://www.diodes.com/assets/Datasheets/ds28002.pdf>

<https://www.theengineeringprojects.com/2017/07/introduction-to-1n4007.html>

<https://www.google.com.hk/search?safe=strict&biw=898&bih=630&ei=648CXOmwKZrr-QbziKqoCw&q=wafer+battery&oq=wafer+battery&gs_l=psy-ab.3..0i203j0i30l2j0i8i30j0i5i30.6182.9632..9841...0.0..0.61.471.9......0....1..gws-wiz.......0j35i39j0i67j0i10i67j0i7i30j0i10j0i10i30.xsib1M0vem4>

<https://forum.43oh.com/topic/7170-using-harware-qei-on-tiva-launchpad/>

<https://www.engineersgarage.com/electronic-components/7805-voltage-regulator-ic>

<https://www.google.com.hk/search?q=bluetooth+noise&oq=bluetooth+noise&aqs=chrome..69i57.12751j0j1&sourceid=chrome&ie=UTF-8>

<https://www.quora.com/What-is-the-maximum-input-voltage-for-the-7805-and-7812>

**Chapter 7: Appendix**

**Attached together in folder (Demo.c) (PID)**