Department of Electrical & Electronic Engineering

University of Nottingham Ningbo China

EEE1039 Report2:

Motion sensing and remote control

&

Line Following

Shun YAO

20321234

ABSTRACT:

This experiment basically aims to realize advance vehicle control using Arduino NANO and other related components. The advance vehicle control could be general divided into the following three parts: Automatically path tracking with MPU6050, remote control with fr24 module and HC-60 bluetooth module, line following with TRCT5000 module and SF-8CHIDTS module. In addition, there is another individual test that does not relate with vehicle control in this session, that is logic board designing and soldering task with 74 series IC. All the above tasks were completed in this session except the automatic path tracking task. All components of tasks were studied and tested before applying onto vehicle. All the code using in this experiment is programmed by Arduino IDE.

INTRODUCTION:

Task1: Study wireless control, motion sensor, and logic gates

Wireless communication in this session including nRF24L01+ 2.4GHz RF wireless module and HC-06 Bluetooth module. They both need to connect with Arduino and programming for wireless communication.

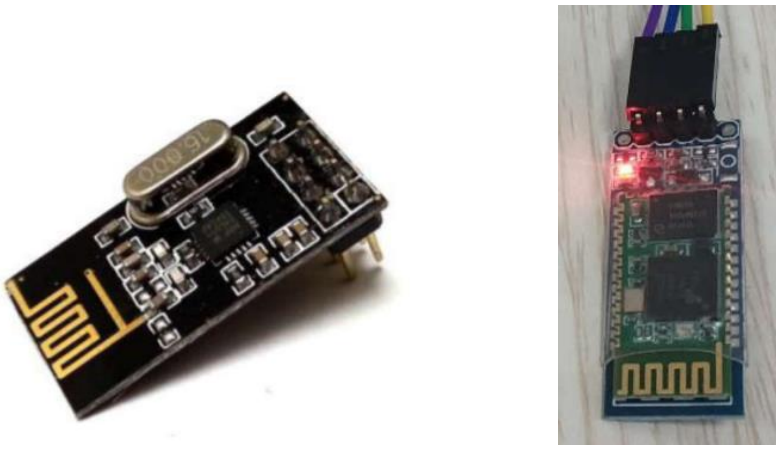


Figure 1. The figure of nRF24L01 (left) and HC-06 (right)

Motion sensor is a component that could sense the vehicle stage and transmit it into digital signal. In this session, MPU6050: The world's first integrated 6-axis Motion Tracking device was used. An accelerometer and a gyroscope are combined in this component, the acceleration ranges: +/- 2g, +/- 4g, +/- 8g, +/- 16g, the Gyroscopes ranges: +/- 250, 500, 1000, 2000 degree/sec(GU and Wang 2021). The movement of it could be sensed and transmit after connecting with a programmed Arduino board.

Figure 2. The figure of MPU6050

Digital systems can have two states, which have various descriptions including: 1 / 0, True / False, and High / Low. A logic gate is an idealized model of computation or physical electronic device implementing a Boolean function, a logical operation performed on one or more binary inputs that produces a single binary output in digital system. In this session, 74 series IC was used to build a logic board which work same as a CPLD of the baseboard.


Figure3. A figure of an example 74 series IC

Task2: Implement of system hardware soldering and debugging

A remote controller was needed for wireless control. Wire connection and working principle of joystick and LCD were studied before the designing a remote controller.


Figure 4. The figure of a joystick and an LCD

Two requirements of the controller:

1. Collect data from joystick and sent it to vehicle using rf24 module.
2. Receive data from vehicle and display it in the LCD.

In that case, the remote controller should content a power supply with battery, an Arduino, a joystick, an LCD, and a rf24 module.

To sense the movement of vehicle, the MPU6050 module should be stably combined with vehicle, otherwise the movement of vehicle can't synchronize with MPU6050 module.

It is required to design and solder a logic board on a stripboard as an individual task.

Task3: Accomplish challenge task A and B

Two tasks in total:

Task A:
A simple path leaves to a ramp on any straight segment

- It is required to pause at the top of the ramp for 2 seconds before rotating 360˚ and continue running
- Automatic path tracking with motor encoder & MPU-6050
- Run two times with the ramp placed at different points
- Vehicle status must be displayed on the remote LCD all the time (run/stop, inclination, distance from start)
- 0.8% marks (bonus up to 0.2% if run faster than median) By the average duration of the two trials

Task B: B-1 & (B-2 or B-3)

- Task B-1: Remote control with nRF24L01+ and joystick (with the remote handle) [upto 0.6%]
  Basic marks [0.4%]: Unidirectional communication
  Full marks: Bi-directional communication with status on the LCD [+0.2%]
  (run/stop, speed, distance from start)
- Task B-2: Remote control with nRF24L01 and MPU-6050 (on the remote handle) [0.4%]
- Task B-3: Remote control with HC-06 (with cellphone) [0.4%]

Task4: Study about basic knowledge of optical sensors and line following

In this session, TCRT5000 was studied. This module contains an infrared light-emitting diode and an Infrared receiving diode. Infrared receiving diode are covered with materials that block visible light and can filter out most of the visible light(Ma 2021).

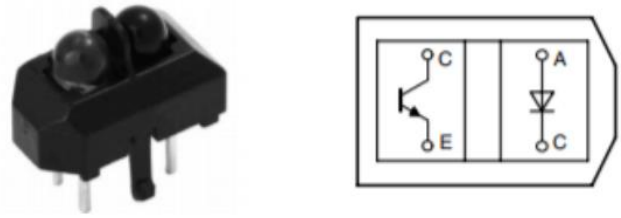


Figure5. The real and schematic view of TCRT5000

It is required to record the output voltages (at A0 and D0) with a certain distance and darkness/colour of the reflection surface using TCRT5000.

Individual task was required to Record the output voltages (at A0) with different distance from black + white and one of yellow / red / green / blue surfaces.

Using the above knowledge, it is possible to implement a line following vehicle with two TCRT5000 module and a bang-bang control or PID control program.

Task5: Proportional-Integral Derivative control and 8 sensors

Study about SF-8CHIDTS, Understand the application of the weighted average algorithm. Design a weighted average algorithm and calibration method for the sensor based on given requirement and record the output position with difference placement of the vehicle to verify this algorithm.

Figure6. The real view of SF-8CHIDTS module

Study the basis PID (Proportional-Integral Derivative) control method and design an HMI to set the parameter of PID control system with LCD, button, encoder.

Task6: Challenge task of line following (HMI, parameter tuning, competition)

This task required to combine SF-8CHIDTS module, vehicle, designed HMI, and programmed code in Arduino to meet the following requirements.

- Complete the advanced line following task with SF-8CHIDTS and PID control (two times continuously through any colour, - 0.1% for touching each roadblock)
- RT control with a certain frequency (>100Hz)
- PID parameters tuned with soldered HMI (LCD + encoder)
- System status display on LCD (run/stop, L&R speeds, PID parameter & error)

METHODS:

Task1: Study wireless control, motion sensor, and logic gates

This task requires to build a wireless communication between two Arduino board using nRF24L01 module. It will need to use RF24 library for programming.

Firstly, connect rf24 module with Arduino as figure7 shows. Prepare two rf24 with two Arduino board and upload the transmit program and receive program that from Moodle page respectively to the two Arduino boards.

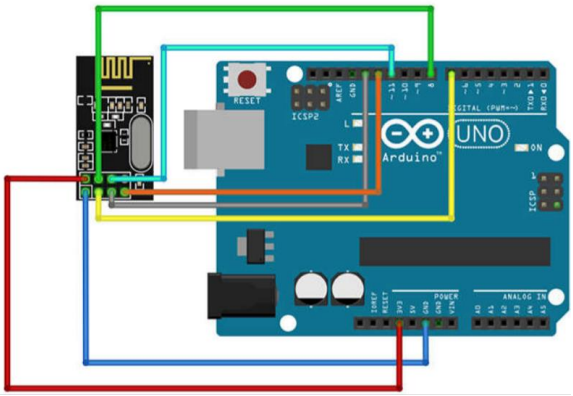


Figure7. The connection between Arduino and rf24

The first code uploaded to Arduino was a unidirectional communication code with rf24 library. One address and a unique channel are required to transmit data from sender rf24 to the reader rf24.

The channel is calculated using the following formula:

$$channel\ number\ =\ 129\ -\ (2\ \times\ group\ number) \qquad Equ.1$$

The channel of group 13 was “103” and address was set as “0001”. The value sent from sender rf24 could be successfully received by the target rf24 and print it on serial monitor with its Arduino board.

The bidirectional communication was followed by this success. Bidirectional example code was studied at the beginning, it is required to set two addresses to make two rf24 become sender and reader at the same time. The two addresses were set in “0001” and “0002”, open serial monitor on both Arduino board, the sender Arduino could receive the feedback value from its reader and the reader Arduino could receive the sent value from its sender. This makes the bidirectional communication succeed.

Another wireless communication was using HC-06 Bluetooth module with cellphone. Connect HC06 module with Arduino and upload Bluetooth code from moodle page and download an app named “Arduino Bluetooth HC-06”. Using this app search and connect with HC06 module. In that case, cellphone is able to send data to hc06 module through Bluetooth wireless communication.

MPU6050 has also been studied in this session. This module is connected in I2C communication which means only SCL and SDA pins are needed to transmit data. It consists of three-axis accelerometer and three-axis gyroscope. It could measure the acceleration, gyroscope and other motion like features. The base circuit of it shows in figure8.

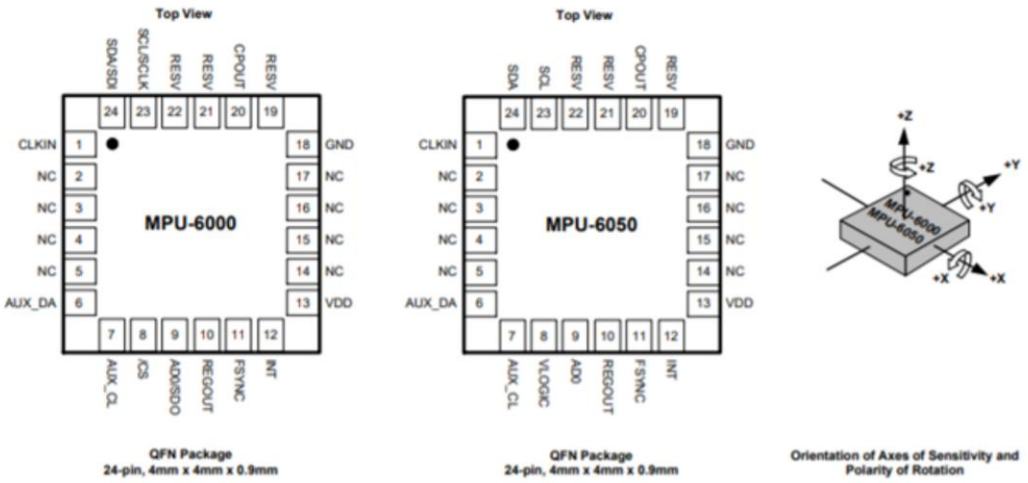


Figure8. The base circuit of MPU6050

Connect this module with Arduino board and upload related code to it, this module could print the acceleration and rotational velocity in each axis “X”, “Y”, and “Z” on serial monitor. The angle that MPU6050 has rotated could be calculated by the following formula:

$$\theta = tan^{-1}\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z} \qquad Euq2$$

Add those calculations to the code, the angle that MPU6050 has rotated could be detected and print on serial monitor.

The individual task requires using 72 series IC of logic gates build a logic board that work same as logic inside CPLD of the baseboard shows in table1. It is easy to tell that this logic system has three inputs: MODE; PWMx, CTLx and two outputs: Gate_x_F, Gate_x_B.

Table1. Logic inside CPLD of the baseboard (each channel)

Logic	Unipolar	Bipolar
	MODE = “0”	MODE = “1”
Run forward CTLx = “0”	Gate_x_F = PWMx Gate_x_B = “0”	Gate_x_F = PWMx Gate_x_B = not PWMx
Run backward CTLx = “1”	Gate_x_F = “0” Gate_x_B = PWMx	Gate_x_F = PWMx Gate_x_B = not PWMx

In this case, three kinds of 74 series IC chips were used in this logic board: SN74LS32N, SN74LS08N, SN74LS04N. They are OR gate, AND gate, and NOT gate.

Task2: Implement of system hardware soldering and debugging

In task B, a remote vehicle controller needs to be designed and soldered. This remote controller should content a battery power supply, a joystick, an LCD, and rf24 module combine with an Arduino board.

Firstly, the battery power supply board was designed and soldered, two batteries would supply voltage of 7.4V while Arduino working voltage is around 5V, so a voltage converter was needed. Switch and two LED indicator with two protect resistors were also part of the power supply board. In order to protect the circuit, a fuse was connected in series. The joystick, LCD, rf24 were combined on another board.

Task A requires to using encoder and MPU6050 to automatically track the path and run two times with ramp placed at different position, which means no remote control allowed in the task. In that case, a sensor that could detect the ramp must be used as the ramp could be different position of the path.

The MPU6050 could detect its own acceleration and rotation velocity, which could calculate the angle of itself. So, to detect the ramp on the path, it is necessary to fix the MPU module on the vehicle stably. It is decided to use gummed tape fix this module on the vehicle.

Before the soldering work of individual task starts, the truth table of this logic system was drawing in table2.

Table2. The truth table of Logic inside CPLD of the baseboard

MODE	PWMx	CTLx	Gate_x_F	Gate_x_B
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

There were many attempts before the final success of individual task as this was a kind complex soldering work that were very likely to make a tiny mistake which leads to the final failure. To prevent the high temperature during soldering damage the 74 series IC, it is decided to solder a package of it instead of a 74 series IC chip.

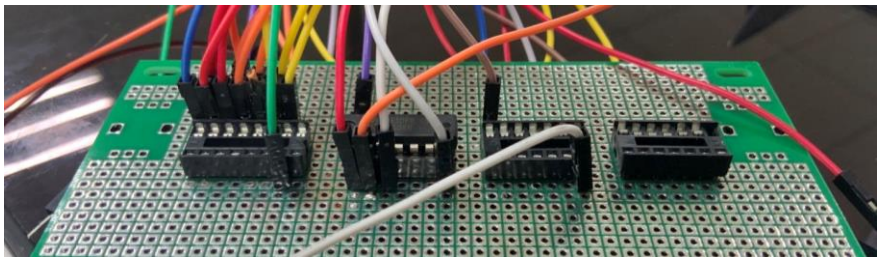


Figure9. A real view of 74 series IC package (soldered)

Task3: Accomplish challenge task A and B

Task A was automatically path tracking with encoder and MPU6050 module and using a remote LCD to display: run/stop, inclination, distance from start.

The travel distance and turning degrees were count using encoders on motor, in that case, it is necessary to be familiar with working principal of encoder on the motor and the diameter of the tyre of vehicle. The diameter of tyre was 0.067m measured by ruler and the encoder value of one cycle turn was given in 234.256.

Figure10. A figure of the ramp

The ramp will randomly appear on the path, so the MPU6050 sensor should keep working at every point of the path. It is designed to detect the angle of MPU module, the code could calculate

the angle. When the angle of this module is detected higher than 15 degrees, the program would start the “upslope” mode that contains three motion control parts: run straight 0.7m, spin 720 degrees, run straight 0.5m and one encoder minus process to insurance the main program could still complete one cycle of the path. As this motion sensor module could detect “uphill” only if the vehicle running, it is suitable to put the sensor code into the external function “readEncoder”.

The code using to transmit data should running constantly in a loop. Using real time control function: “MsTimer2” could realize this kind of loop.

Task B contains three specific tasks: B1, B2, and B3. Choose any two out of the three, it was decided to choose B1: remote control with rf24, LCD and joystick and B3: remote control with HC06 bluetooth module and cellphone.

Both tasks require to complete one cycle ride of the same path in task A. Task B1 requires using bidirectional communication between two rf24 module to control vehicle and display on LCD. So the code in task1 should be combined with the code motion control, joystick read, and LCD display. Using the remote controller and vehicle built in task2 for this remote control task.

Task B3 is based on HC-06 bluetooth module. Combine the Bluetooth communication code and the vehicle motion code together and upload this code to Arduino board. Open the serial monitor of this Arduino, rename this HC06 module to distinguish from other groups. Download an app named “Arduino Bluetooth HC-06” on the cellphone, open and connect it with the renamed HC-06 module. The cellphone could send data in ASCII to HC-06 module, in that case, set different ASCII code of different button in the app could order the vehicle running in different stage.

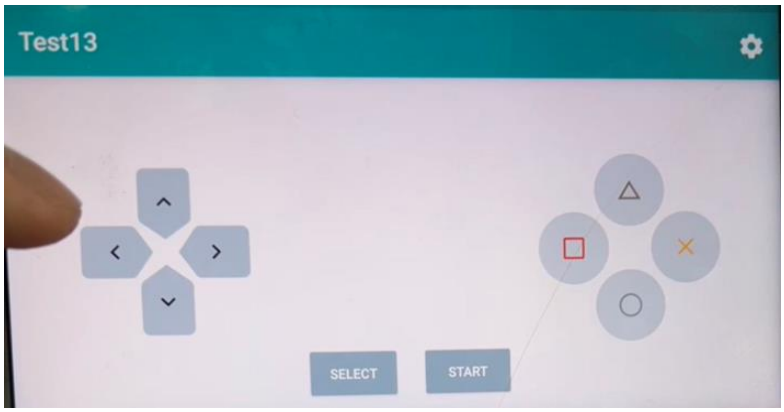


Figure11. The interface of “Arduino Bluetooth HC-06”

Task4: Study about basic knowledge of optical sensors and line following

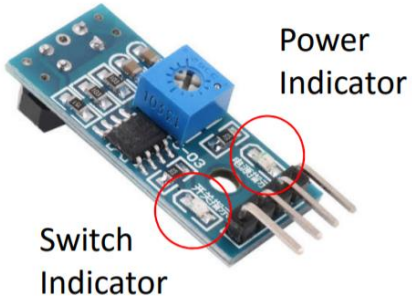


Figure12. A real view of TCRT5000 module

In this session, it is required to use TCRT5000 module detect different surfaces. The TCRT5000's working principle is the same of the common infrared sensor. It has an infrared emission tubes and

an infrared receiver. The resistance of receiver will change when the infrared signal is received. Changes of the resistance depend on received signal strength. Colors of reflector surface and distances of receiving tubes are their general manifestation(Liang and Rao 2011).

Individual task is aims to detect and record the output voltages (at A0) with different distance from black + white and one of yellow / red / green / blue surfaces. The distance between this module and target surface should been recoded from 1mm to 16mm (incr 1mm) which means there are 16 values of each color needs to be measured, 96 values in total. Using ruler to measure the distance, equipment shows in figure14 are all used to detect the output of TCRT5000 module.

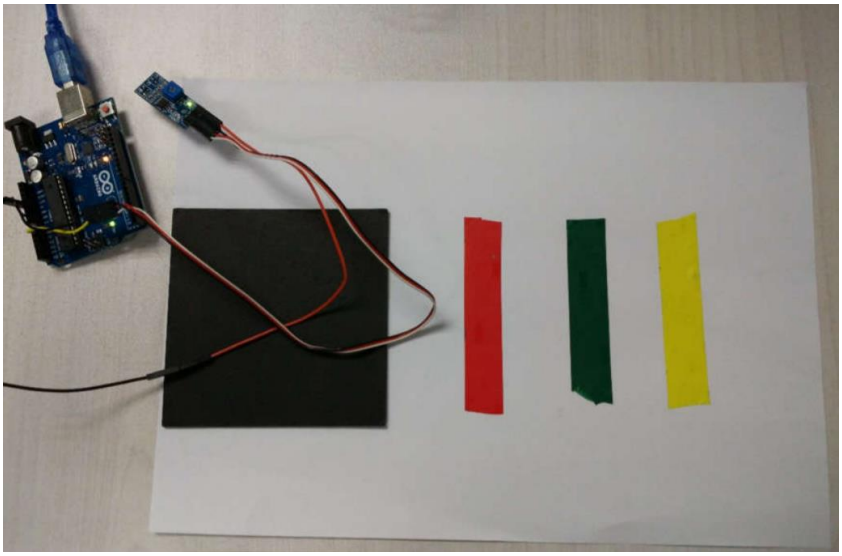


Figure13. Arduino and different color tapes

Based on different surface could have different feedback, it is feasible to realize basic line following task with two TCRT5000 module and bang-bang control or PID control. Connect two TCRT5000 module on the head of vehicle as figure15 and upload the code onto Arduino board. The logic of bang-bang control program could be divided into three parts: “straight”, “left spin” and “right spin”. When both sensors detect white surface on the path, the vehicle would go straight; when the left sensor detect black surface and right sensor detect white surface, the vehicle would spin clockwise; when the right sensor detect black surface and left sensor detect white surface, the vehicle would spin anticlockwise.

This program is based on bang-bang control theory.

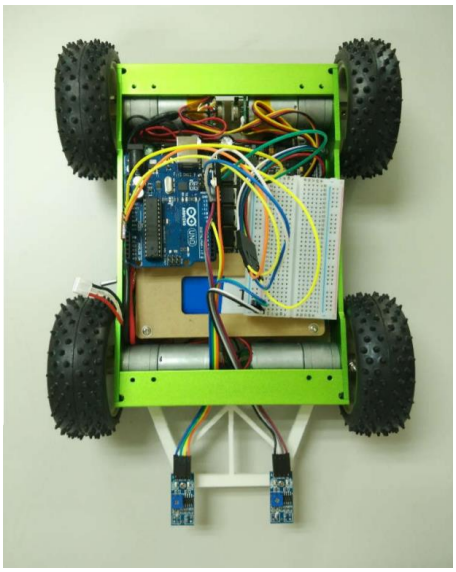
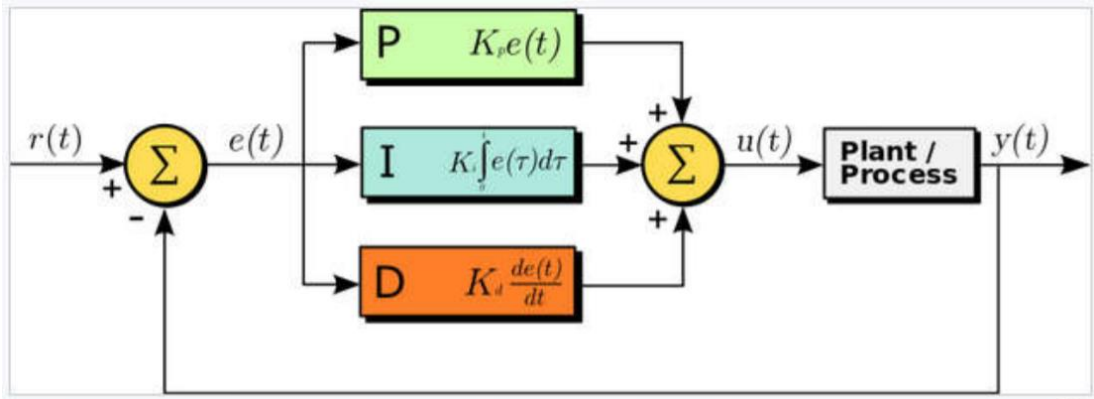


Figure14. Vehicle that connects with two TCRT5000 module

Task5: Proportional-Integral Derivative control and 8 sensors

The PID control of line follower is a method consisting of Proportional, Integral & Derivative functions to improve the movement of the robot. The robot uses several sensors to identify the line thus assisting the bot to stay on the track(Jibrail and Maharana 2013). This control method could make the vehicle following the given line smoothly while bang-bang control will make the vehicle shaking.

PID control has three parameters: Kp, Ki, and Kd. Those parameters determine the line follow vehicle running stage. Kp is related to how hard of the motor draw back vehicle back to path when the vehicle deviates from the line. Ki and Kd are both used to adjust the Kp related motion.



A block diagram of a PID controller in a feedback loop. r(*t*) is the desired process value or setpoint (SP), and y(*t*) is the measured process value (PV).

Figure15. A block diagram of a PID controller

SF-8CHIDTS module contains 8 optical sensors, it could give digital and analogy feedback of the surface. In that case, using weight average method to calculate the 8 analogy read values to figure out the distance between the centre line of the vehicle and the black line. Using this distance as the error of a PID control system, it will work properly.

The distance could be calculated by the following equation:

$$Distance = \frac{\sum_{i=1}^{8} sensor\ value_i . weight_i}{\sum_{i=1}^{8} sensor\ value_i} \qquad Equ.3$$

The weight of each sensor value is determined by the distance between this sensor and the centre line of the vehicle. If one sensor is 6.5mm from the centre line, then the weight of this sensor is 6.5.

Task6: Challenge task of line following (HMI, parameter tuning, competition)

Advanced line following task requires to use PID control on the vehicle and the three parameters of the PID system should be set using a soldered HMI (encoder and LCD).

The HMI part and the PID control part were processing separately. An encoder and LCD combined system on a breadboard was built for the previous programming and testing. Using three “while” loop to set the value of three parameters, push the encoder button once to break one

“while” loop. The LCD display code on this setting part was putted on the external function “doEncoder” which means once the encoder value changes, the LCD display will be refreshed.

The PID motion control code was programmed based on the example code from moodle page. A weight average algorithm, a calibration method and some motion control instructions were added to this program, so the program could get output from PID control system and apply this output to vehicle motion control. The calibration of sensor values are aims to limit those values in a range by using “map”. It is required to use RT control with a certain frequency that greater than 100Hz, in that case, the function “MsTimer2” should be used instead put everything in function “loop”.

When both parts of the program completed, combine them together to meets all requirements. The LCD display on running stage should be added to function “loop”. The vehicle with HMI and programmed code should be able to complete two cycle of the given path.

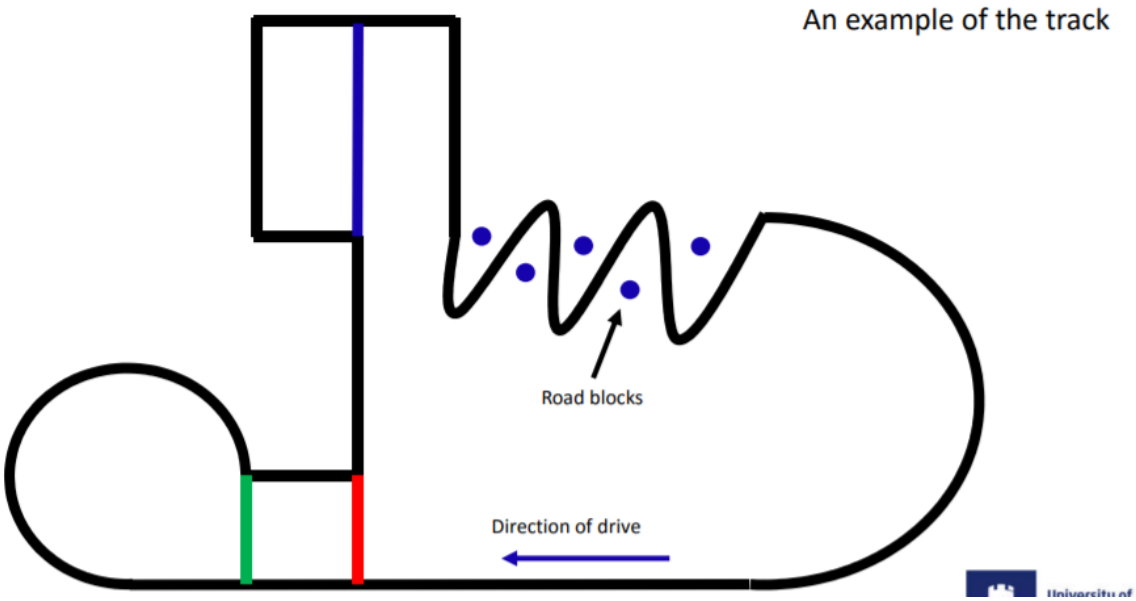


Figure16. An example of the path

RESULTS:

Task1: Study wireless control, motion sensor, and logic gates

Unidirectional communication and bidirectional communication were both successfully established between two rf24 module with their own Arduino board. The transmit program could sent data in different data type to the receive program including int, float, char, etc. When it comes to bidirectional communication, the receive program is required to giving feedback message to the transmit module, this feedback needs a different address in wireless communication to distinguish the receiving and sending messages.

Figure17. Received message through rf24

The Bluetooth wireless communication base on HC06 module should working with a cellphone that connected with this HC06 module. To distinguish the target HC06 module from other groups, this module was renamed as “group 13 test” using related code and serial monitor. Open app “Arduino Bluetooth HC-06” on cellphone find the device named “group 13 test” and connect with it. Then, the data sent by from cellphone could be received by the HC06 module, the wireless communication based on HC06 bluetooth module has successfully established.

MPU6050 is used to measure the movement of vehicle, some previous test of this module was processed and the acceleration with corresponding tilt angle results were recorded in table3. The acceleration vs. corresponding tilt angle plot diagram of upslope and downslope shows in figure18 and figure19 respectively.

Table3. Acceleration with corresponding tilt angle results

Serial Number	ax(g)	ay(g)	az(g)	a(g)	θ (˚)
1	0.009	0.24	0.80	0.835	16.71
2	0.007	0.37	0.76	0.845	25.96
3	0.029	0.45	0.73	0.852	31.13
4	0.006	0.56	0.66	0.866	40.32
5	0.016	0.68	0.57	0.887	50.03
6	-0.041	-0.66	0.55	-0.849	-51.56
7	-0.011	-0.59	0.61	-0.838	-44.05
8	-0.0089	-0.46	0.69	-0.829	-33.69
9	-0.011	-0.34	0.75	-0.823	-24.39
10	-0.010	-0.22	0.79	-0.820	-15.62

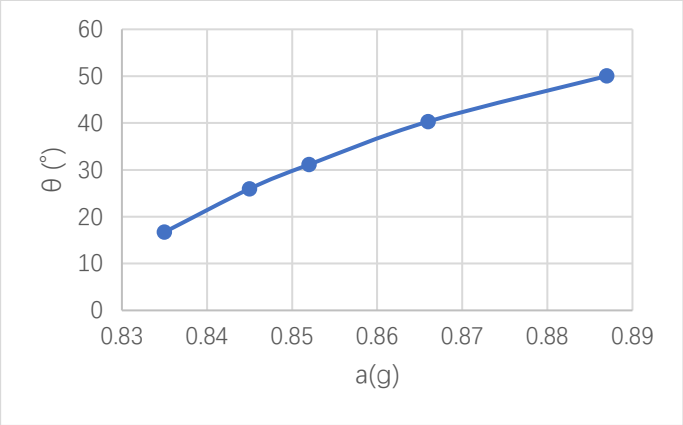


Figure18. Acceleration vs. tilt angle of upslope

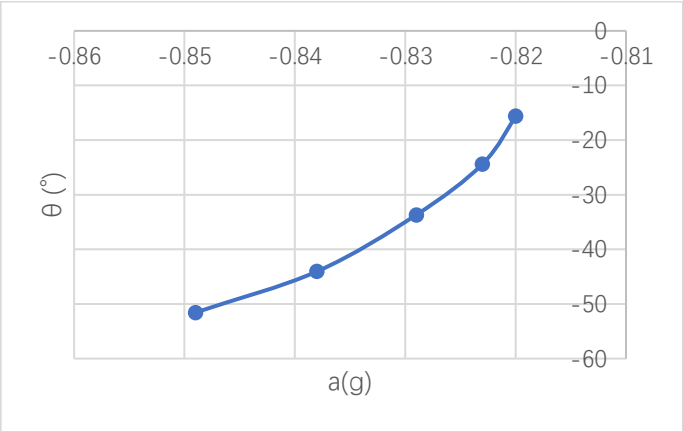


Figure19. Acceleration vs. tilt angle of downslope

This logic gate individual task requires to using 72 series IC of logic gates build a logic board that work same as logic inside CPLD of the baseboard. So it is necessary to get famous with the types of 72 series IC. Based on the truth table shows in table2 the following three figure shows the chips that should be used in the logic system.

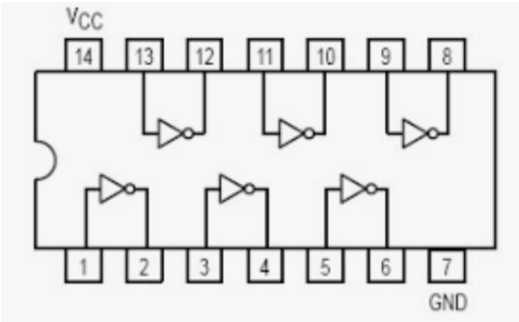


Figure20. A diagram of SN74LS04N

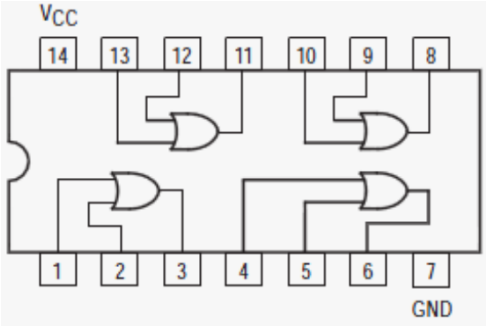


Figure21. A diagram of SN74LS32N

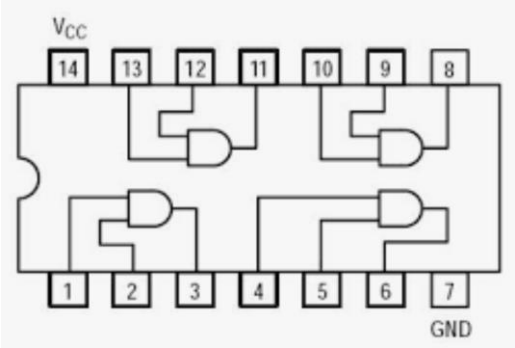


Figure22. A diagram of SN74LS08N

Task2: Implement of system hardware soldering and debugging

The remote controller was combined by two board, one power supply board and one control board. Power supply board use two 3.6V battery as voltage source and convert the voltage from 7.2V to 5V to supply Arduino board. A fuse, two LED with resistor were all combined on this board. Control board contain a joystick, an LCD and a rf24 module with Arduino. All of them were fixed together.

Final controller shows in figure23, the data generated by joystick could be sent to the vehicle and the feedback from vehicle (run stage, speed, distance) could be received and displayed on the LCD. All of above was based on the bidirectional communication between two rf24 module.

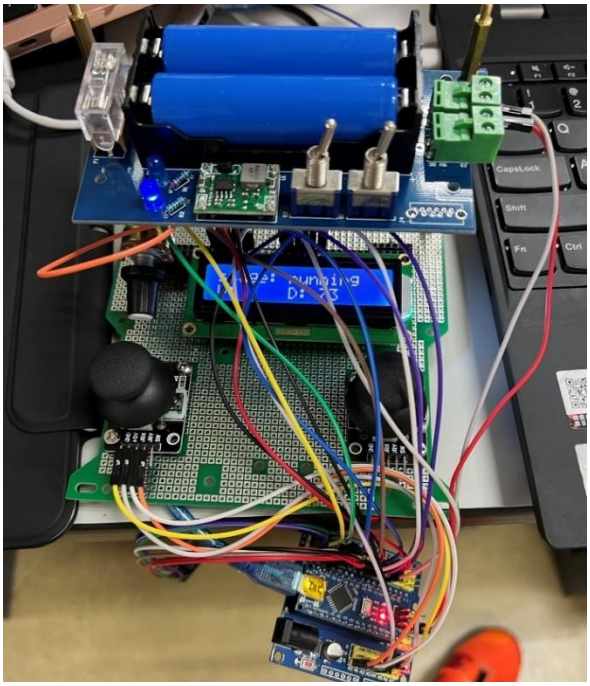


Figure23. A real view of remote controller

The MPU6050 module was gummed on the top of vehicle, in that case, the movement of this module and vehicle could be synchronized. When the vehicle running to the upslope of the ramp, the acceleration of gravity on MPU changes, the detected angle value will rise.

To build the logic board which follow the truth table shows in table2, draw a logic gate circuit first. Based on truth table, the final circuit shows in figure24. As it shows in this diagram, three NOT gates, four AND gates, and two OR gates are needed to build this logic system.

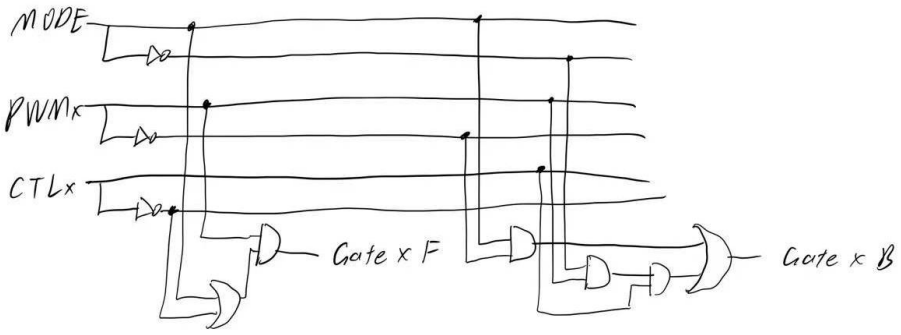


Figure24. The logic gate circuit diagram

Soldering starts immediately when the circuit confirmed on a breadboard. Three packages of 74 series IC chip and the wires of this system were soldered on a stripboard. This is a high accuracy required soldering work that is easy to make mistake, so before install the chips on this circuit, it is decided to test it first. In that case, if this logic system cannot work properly, should be something wrong with soldering but the chips. The soldered logic system was finally completed at the third soldering work.

Task3: Accomplish challenge task A and B

The testing path shows in figure25. In task A, it could divide the path into the following 14 parts: run straight 2.5m, turn left 45 degrees, go straight 1.414m, turn left 45 degrees, run backward 0.5m, turn left 63.5 degrees, run straight 1.2m, turn left 23.5 degrees, run straight 2.5m, turn left 45 degrees, run straight 0.8m, turn left 45 degrees, run a quarter of circle, run straight 0.5m. The code was programmed based on the 14 different running stage.

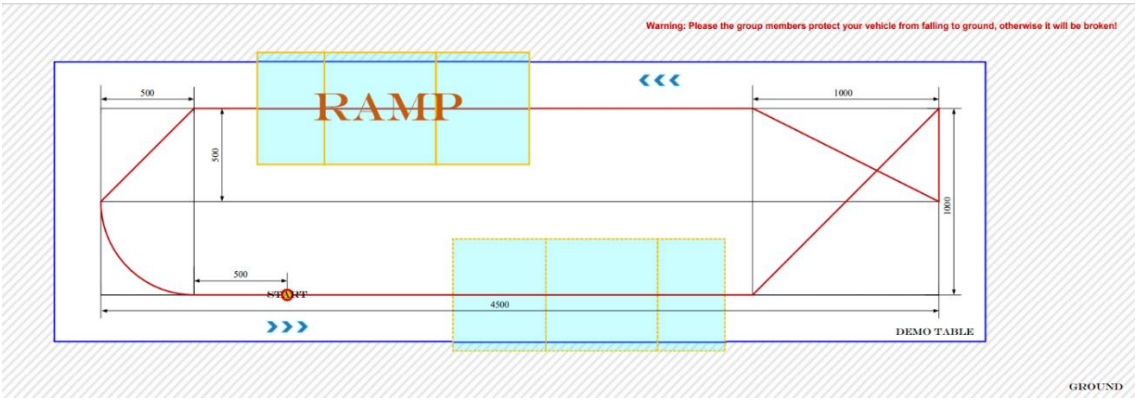


Figure25. A diagram of the path

The theory encoder value of each part was used in the previous experiments. After several experiments, the final encoder values of each running stage are as follow: {2760, 135, 1550, 160, 450, 179, 1100, 100, 3000, 60, 700, 60, 400, 500}. When MPU detect the ramp of the path, program will start “upslope” mode.

However, this is a long-distance travel, any small mechanic error are able to disturb the whole system and leads to the final failure of this task. To make the vehicle running a straight line, each motor was set with different running speed as the mechanic resistance on different motor were different. The two motors on right hand side were set in “42” and other two motors were set in “43”. In the turning stage, right hand side motors were set in “58” and left hand side in “50”.

Due to the mechanic error is far bigger than the expectation, each running always cannot accurately back to the start point. The task A was failed.

Task B1 is aims to using two rf24 modules to establish bidirectional communication between the remote controller and the vehicle. The controller should collect the data from joystick and sent it to vehicle, the vehicle should send its running stage, speed, and distance from start back to the controller.

The joystick could generate an X and a Y value between 0 and 1023 as figure26 shows. In that case, use the instruction: “map”, map both X and Y value on the range between -80 to 80 and using this value set as the speed of vehicle.

Using X axis value as the speed of forward and backward, when the joystick was on the middle point, the mapped value of X should be 0 which means the speed of vehicle is zero. Using the Y axis value as the speed of spin clockwise and spin anticlockwise, the vehicle will also be static when joystick was on the middle point. Programming was based on the above logic, and the vehicle was work properly with the LCD display.

Task B1 was successfully completed.

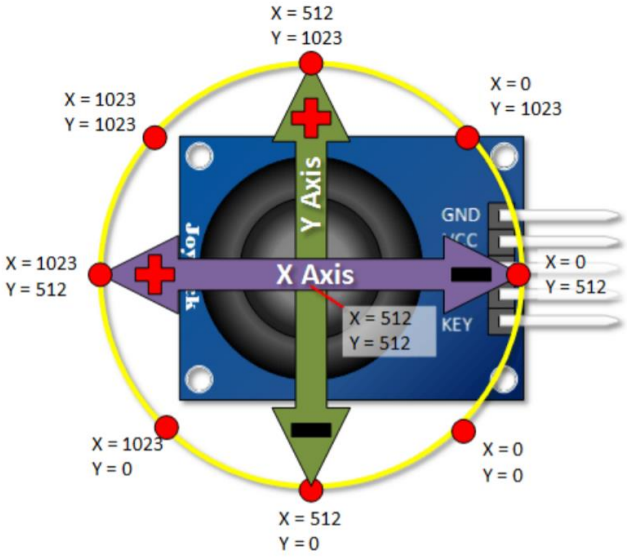


Figure26. The values on X axis and Y axis of joystick

Task B3 is a HC06 base Bluetooth wireless control, using HC06 module control the vehicle. When the HC06 module connected with cellphone, it will be able to receive data in ASCII form from cellphone. In that case, set different related value of each button in the app to represent different running stage should be feasible to control the vehicle motion.

If the received value is “43”, then running forward; if is “49”, then spin anticlockwise; if is “51”, then spin clockwise; if is “52”, then running backward; if is “37”, then stop. The program was based on this logic, and it was working properly.

Task B3 was successfully completed.

Task4: Study about basic knowledge of optical sensors and line following

Upload the related code onto Arduino board, open serial monitor, the feedback of different color was printed there. The digital read result on black is “1” and on all other colors the result becomes “0”. The analog read result on each surface was changing from 0 to 1023, but only the result on black surface could reach 200 even higher. The result on white/bule/green/yellow/red surface was all smaller than 200. It is considered that black can absorb most infrared rays while others cannot.

The inside circuit of TCRT5000 shows in figure27 and the relative collector current vs. distance diagram shows in figure28. It is easy to figure out that the least sensitive working distance of this module is 2mm. Because when the current on the circuit rise, the voltage on the potential divider resistor will rise, in that case, the output analog voltage that between A0 and ground will decrease as the supply voltage are constant in 5V.

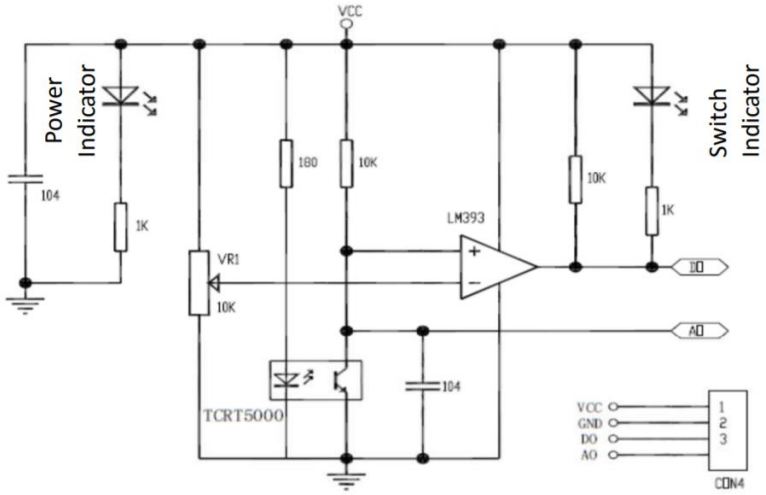


Figure27. The inside circuit of TCRT5000

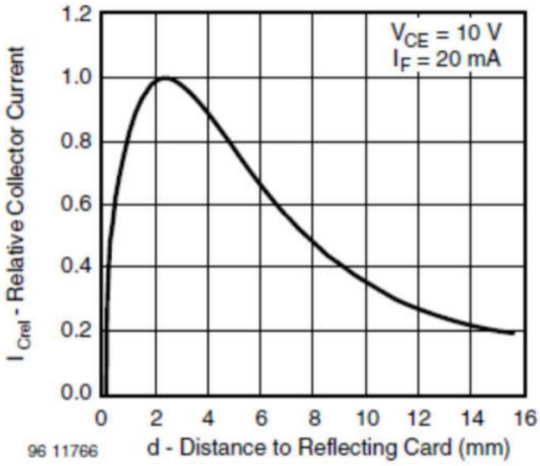


Figure28. Relative collector current vs. distance diagram

The individual task requires to detect and record the output voltages (at A0) with different distance from black + white and one of yellow / red / green / blue surfaces. To increase the accuracy of this experiment, a multimeter was connect in parallel with the TCRT5000 module and all measured value were recorded in table4.

Table4. The measured voltages at A0 with different distance and colors

Distance (mm)	black (V)	White (V)	Blue (V)	Green (V)	Yellow (V)	Red (V)
1	3.632	0.163	0.168	0.156	0.136	0.143
2	0.271	0.145	0.151	0.14	0.124	0.134
3	0.224	0.142	0.147	0.136	0.123	0.131
4	0.228	0.143	0.147	0.137	0.13	0.132
5	0.247	0.145	0.15	0.139	0.131	0.133
6	0.293	0.148	0.156	0.142	0.134	0.136
7	0.531	0.151	0.159	0.145	0.136	0.139
8	1.132	0.154	0.161	0.148	0.141	0.142
9	1.812	0.159	0.166	0.15	0.143	0.145
10	2.153	0.162	0.17	0.153	0.147	0.148
11	2.571	0.166	0.174	0.156	0.148	0.151
12	2.742	0.166	0.177	0.159	0.152	0.155
13	2.823	0.167	0.18	0.163	0.155	0.159
14	3.015	0.17	0.183	0.166	0.156	0.163
15	3.223	0.175	0.192	0.169	0.158	0.168
16	3.321	0.18	0.197	0.174	0.161	0.172

The analog read value from Arduino could be calculated by the following formula:

$$\text{analogread} = 5 \times \frac{\text{voltage\_at\_A0}}{1024}\ (V) \qquad Equ.4$$

Basic line following with bang-bang control has successfully realized using two TCRT5000 module. The working principal of this system shows in figure. The vehicle could go through most of the turns on the path with this control system, but it is not always working properly on the right-angle turn. So, a short “dalay()” for 0.1 second was added to the program, this delay allow the vehicle spin 15 degrees before the next round detect to avoid the case that both sensors detected the black line.

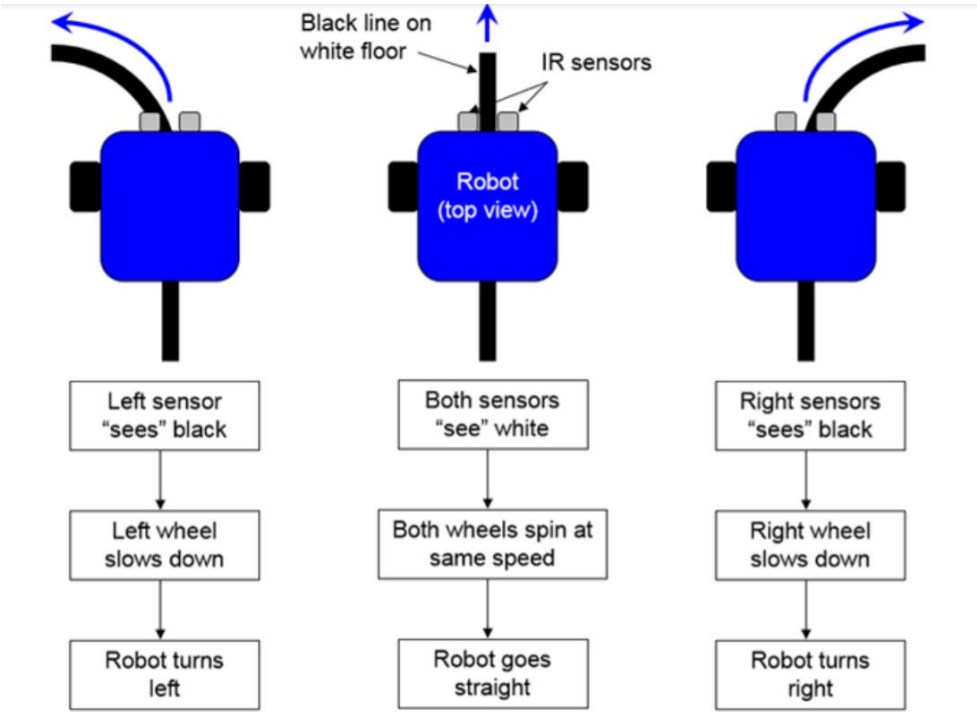


Figure29. Working principal of bang-bang control basic line following task

Task5. Proportional-Integral Derivative control and 8 sensors

The PID control program was completed based on the example code on moodle. The error (distance between the vehicle centre line and black line) is the input of the PID control system, this system will give the related output based on the input distance. This output of PID system will be used to control the speed of motor to change the motion of this vehicle. When the sensor detects the black line is on the right side, the PID control system will gives an output, and this output value will add to the left side motor speed, the right side motor will subtract an output value.

The sensor value will also change as the motion of vehicle changes. In that case, the program running constantly then the motion of vehicle will also be changing constantly.

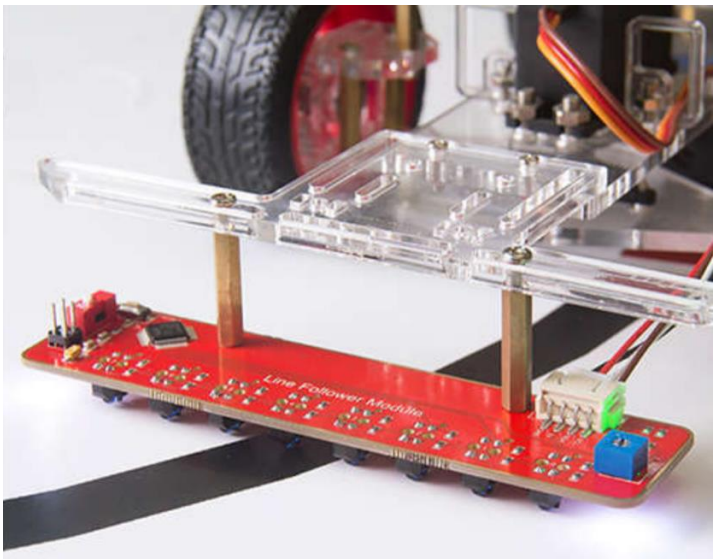


Figure30. A working SF-8CHIDTS module on a vehicle

It is important to get the 8 sensor values from SF-8CHIDTS module to calculate the distance. The analogy read of this module is from 0 to 1023, basically the same as TCRT5000, so the point is to

calculate the weight average of 8 sensor values to gain the distance. Weight of each sensor value in the advance line following task was listed in the table5.

Table5. Weight of each sensor

Sensor ID	1	2	3	4	5	6	7	8
Weight	-45.5	-32.5	-19.5	-6.5	6.5	19.5	32.5	45.5

Task6: Challenge task of line following (HMI, parameter tuning, competition)

The HMI part that contains an LCD and an encoder was soldered, a real view of this HMI shows in figure31. Upload the related program to the Arduino that connected with this HMI, it was working properly of setting three parameters of PID control system.

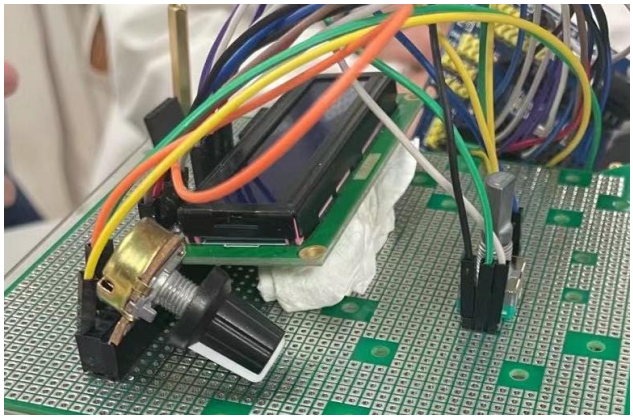


Figure31. A real view of soldered HMI

Program of PID control part was processed with many experiments of different parameters set. The parameters of a PID control system will affect vehicle performance on the path significantly. The final set of three parameters were: Kp=3; Ki=0; Kd=0.5, which decided by experiments.

RT control also processed in the program using function “MsTimer2”. It was decided to set the frequency of sensor reading and motion control process at 200 Hz, which means it will processing once per 0.05s.

However, this PID system cannot always working properly on the right angle turn on the path. If solving this issue by adjusting parameters, new issues come out on other turns of this path. In that case, it is decided to modify a “turn mode” that will active when the sensor detect right angle turn. The “turn mode” was a spin process that will working for 0.45s and the sensor value on a right angle was recorded as the condition to active this mode.

In the final challenge, turn on the switch on vehicle, then the LCD on HMI will display “Kp= ”, turn the encoder, the value of Kp will rise if turning clockwise decrease if turning anticlockwise. Push the button on this HMI program will store the Kp value and start the setting of Ki, push the button again for Kd. When the last Kd value was set, push the button again, the vehicle will start running and the LCD will display: run/stop, L&R speeds, PID parameter & error.

The advanced line following task was successfully completed.

DISCUSSION:

Task 1: This task is mainly about the study of components that will be used in the coming tasks. All the codes in wireless communication needs to be fully understood to combining the communication code with the motion control code.

The data recorded in table3 was not exactly matched with the theory data, possibly this is caused by the imprecise of MPU measured tilt angle as this process was hold by hand. It is feasible to improve the accuracy of measurement by fix this module on a protractor.

Task 2: This task is focusing on the hardware designing and building for the following tasks. The remote controller could have many different combing methods, there are all acceptable if it can work properly. The same as the fixation of MPU6050 and vehicle.

Soldering was the hardest part of the building of logic board as it requires high accuracy of wire connection. It is more convenient to using connected metallic tin instead of a wire, because there are already too many wires on this board, it will make the circuit more confusing.

Task 3: In task A, the mechanic errors easily occurs when there are sliding friction between the tire and the ground. Especially on the ramp, when the vehicle run upslope or downslope, sliding friction usually occurs, which leads the encoder value cannot represent the travelled distance accurately. This error cannot be avoided on the current logic of running process. To running properly on the path, MPU6050 should be used not only detect the ramp, also to correct the position of vehicle. When MPU detected the angle of vehicle is wrong, start a process to correct it until MPU detect a matched angle.

For task B1, there will issue occurs if the joystick is in a position where X axis value and Y axis value are both 1023 or both 0. It will be better to program a turning mode when the joystick values not allow vehicle running straight or spin.

Task 4: The results in table4 was not exactly following the theory data based on TCRT5000 working principal, this could be considered as the influence caused by the environment light. To improve the accuracy of this measurement, a dark measuring environment is in need.

Using bang-bang control for the basic line following task cannot working properly when the vehicle is turning a right angle or a sharper one. This is because two sensors on the head of vehicle could possibly detect the black line at the same time, in that case, there is no suitable process in its program so the vehicle will rapidly switching between “right spin” mode and “left spin” mode or the second sensor did not detect the black line the then vehicle straight go through the black line. To fix this issue, it is feasible to add a short “delay()” on the program but is not good as this delay could influent vehicle performance on the straight line of the path. It is also feasible to fix this issue by adding a mode for both sensors detected black line.

Task 5 & 6: PID control in line following task could makes the vehicle running in a much smoother way than bang-bang control if all parameters were set in a suitable value. Adjust the parameters of PID is a time-consuming project, so it is better to recode the parameters in each experiment and analysis it to get the final value instead of only experiment. It was not the best solution to set a “turn mode” for right angle turn because it possible to meets a sharper angle at some line following task. Adjustment should be made in a way that combined with PID control.

CONCLUSION:

In conclusion, this experiment is mainly about some advanced vehicle control method and some knowledge of related components. Each component was studied in advance before applying it on the vehicle.

The first task is using MPU6050 and encoder to complete a one cycle automatically path tracking. MPU and encoder was studied in advance, using this related knowledge, programming a code that could detect ramp and read the encoder values to fulfil the task. Then, the study of wiseless communication started, two kind of modules was studied: rf24 module and the HC-06

bluetooth module. Apply the wireless communication to a controller and the vehicle to realize remote control of vehicle.

Second task is focusing on line following. To make a line follower, TCRT5000 module was used to detect the black line on white ground and bang-bang or PID control were used to control the motion of vehicle. The basic line following was realized using the TCRT5000 module and bang-bang control method, vehicle in this task always shacking because of bang-bang control working principal. Advanced line following was realized based on the 8 sensors SF-8CHIDTS and PID control. Vehicle controlled by PID method was much smoother than bang-bang control.

Some further study of PID control and their parameters are essential.

References:

1) GU, Chunyang, and Jing Wang
   2021 Session III: Motion sensing and remote control & Session IV: Line Following.
2) Liang, Jiahai, and Zhiqiang Rao
   2011 Research on running state control for the AGV based on reflected infrared photoelectric sensor. 2011 International Conference on Electric Information and Control Engineering, 2011, pp. 694-697. IEEE.
3) Ma, Ruichen
   2021 Line following and beacon tracking robot based on Arduino Mega 2560. 2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT), 2021, pp. 32-36. IEEE.
4) Jibrail, Sheikh Farhan, and Rakesh Maharana
   2013 PID Control of Line Followers.

Appendix:

rf24 receive code:

```
#include <SPI.h>
#include "RF24.h"

RF24 rf24(9,10); // CE, CSN

const byte addr[] = "1Node";
const byte pipe = 1;

void setup() {
  Serial.begin(9600);
  rf24.begin();
  rf24.setChannel(103);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_1MBPS);
  rf24.openReadingPipe(pipe, addr);
  rf24.startListening();
  Serial.println("nRF24L01 ready!");
}

void loop() {
  if (!rf24.available(&pipe)) {
    char msg[32] = "";
    //int dog;
    //rf24.read(&dog, sizeof(dog));
    rf24.read(&msg, sizeof(msg));
    Serial.println(msg);
    Serial.println("good");
  }
}
```

rf24 transmit code:

```
#include <SPI.h>
#include "RF24.h"

RF24 rf24(9,10); // CE, CSN

const byte addr[] = "1Node";
const char msg[] = "Happy Hacking!";

void setup() {
  rf24.begin();
  rf24.setChannel(103);
  rf24.openWritingPipe(addr);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_1MBPS);
  rf24.stopListening();
}
```

```
void loop() {
  rf24.write(&msg, sizeof(msg));
  delay(1000);
  Serial.println("sent");
}
```

MPU6050 test code:

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

MPU6050 accelgyro;

unsigned long now, lastTime = 0;
float dt;

int16_t ax, ay, az, gx, gy, gz;
float aax=0, aay=0,aaz=0, agx=0, agy=0, agz=0;
long axo = 0, ayo = 0, azo = 0;
long gxo = 0, gyo = 0, gzo = 0;

float pi = 3.1415926;
float AcceRatio = 16384.0;
float GyroRatio = 131.0;

uint8_t n_sample = 8;
float aaxs[8] = {0}, aays[8] = {0}, aazs[8] = {0};
long aax_sum, aay_sum,aaz_sum;

float a_x[10]={0}, a_y[10]={0},a_z[10]={0} ,g_x[10]={0} ,g_y[10]={0},g_z[10]={0};
float Px=1, Rx, Kx, Sx, Vx, Qx;
float Py=1, Ry, Ky, Sy, Vy, Qy;
float Pz=1, Rz, Kz, Sz, Vz, Qz;

void setup()
{
   Wire.begin();
   Serial.begin(115200);

   accelgyro.initialize();

   unsigned short times = 200;
   for(int i=0;i<times;i++)
   {
      accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
      axo += ax; ayo += ay; azo += az;
      gxo += gx; gyo += gy; gzo += gz;

   }

   axo /= times; ayo /= times; azo /= times;
   gxo /= times; gyo /= times; gzo /= times;
```

```
}

void loop()
{
  unsigned long now = millis();
  dt = (now - lastTime) / 1000.0;
  lastTime = now;

  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  float accx = ax / AcceRatio;
  float accy = ay / AcceRatio;
  float accz = az / AcceRatio;

  aax = atan(accy / accz) * (-180) / pi;
  aay = atan(accx / accz) * 180 / pi;
  aaz = atan(accz / accy) * 180 / pi;

  aax_sum = 0;
  aay_sum = 0;
  aaz_sum = 0;

  for(int i=1;i<n_sample;i++)
  {
    aaxs[i-1] = aaxs[i];
    aax_sum += aaxs[i] * i;
    aays[i-1] = aays[i];
    aay_sum += aays[i] * i;
    aazs[i-1] = aazs[i];
    aaz_sum += aazs[i] * i;
  }

  aaxs[n_sample-1] = aax;
  aax_sum += aax * n_sample;
  aax = (aax_sum / (11*n_sample/2.0)) * 9 / 7.0;
  aays[n_sample-1] = aay;
  aay_sum += aay * n_sample;
  aay = (aay_sum / (11*n_sample/2.0)) * 9 / 7.0;
  aazs[n_sample-1] = aaz;
  aaz_sum += aaz * n_sample;
  aaz = (aaz_sum / (11*n_sample/2.0)) * 9 / 7.0;

  float gyrox = - (gx-gxo) / GyroRatio * dt;
  float gyroy = - (gy-gyo) / GyroRatio * dt;
  float gyroz = - (gz-gzo) / GyroRatio * dt;
  agx += gyrox;
  agy += gyroy;
  agz += gyroz;

  /* kalman start */
  Sx = 0; Rx = 0;
  Sy = 0; Ry = 0;
  Sz = 0; Rz = 0;
```

```
for(int i=1;i<10;i++)
{
   a_x[i-1] = a_x[i];
   Sx += a_x[i];
   a_y[i-1] = a_y[i];
   Sy += a_y[i];
   a_z[i-1] = a_z[i];
   Sz += a_z[i];
}

a_x[9] = aax;
Sx += aax;
Sx /= 10;
a_y[9] = aay;
Sy += aay;
Sy /= 10;
a_z[9] = aaz;
Sz += aaz;
Sz /= 10;

for(int i=0;i<10;i++)
{
   Rx += sq(a_x[i] - Sx);
   Ry += sq(a_y[i] - Sy);
   Rz += sq(a_z[i] - Sz);
}

Rx = Rx / 9;
Ry = Ry / 9;
Rz = Rz / 9;

Px = Px + 0.0025;
Kx = Px / (Px + Rx);
agx = agx + Kx * (aax - agx);
Px = (1 - Kx) * Px;

Py = Py + 0.0025;
Ky = Py / (Py + Ry);
agy = agy + Ky * (aay - agy);
Py = (1 - Ky) * Py;

Pz = Pz + 0.0025;
Kz = Pz / (Pz + Rz);
agz = agz + Kz * (aaz - agz);
Pz = (1 - Kz) * Pz;

/* kalman end */

Serial.print(agx);Serial.print("(X)");
Serial.print(agy);Serial.print("(Y)");
Serial.print(agz);Serial.print("(Z)");Serial.println();
```

```
}
```

tcrt5000 8 sensor test code:

```
#include <Wire.h>          // Include the Wire library for I2C communication

unsigned char dataRaw[16];
unsigned int sensorData[8];

double distance = 0;
double sensorValue = 0;

void setup()
{
  Wire.begin();       // Join the I2C bus as a master (address optional for master)
  Serial.begin(57600);  // Start serial for output to PC
}

void loop()
{
  readSensorData();
  Serial.println("---------------");
  Serial.println(sensorData[0]);
  Serial.println(sensorData[1]);
  Serial.println(sensorData[2]);
  Serial.println(sensorData[3]);
  Serial.println(sensorData[4]);
  Serial.println(sensorData[5]);
  Serial.println(sensorData[6]);
  Serial.println(sensorData[7]);
  int i = 0;
  for(i = 0; i < 8; i++){
    sensorValue += sensorData[i];
  }
  distance = ((-45.5)*sensorData[0] + (-32.5)*sensorData[1] + (-19.5)*sensorData[2] + (-
6.5)*sensorData[3]
  + 6.5*sensorData[4] + 19.5*sensorData[5] + 32.5*sensorData[6] + 45.5*sensorData[7]) /
sensorValue;
  Serial.print("Distance = ");
  Serial.println(distance);
  delay(300);

}



void readSensorData(void)
{
  unsigned char n;          // Variable for counter value
  unsigned char dataRaw[16];  // Array for raw data from module

  // Request data from the module and store into an array
  n = 0;  // Reset loop variable
```

```
 Wire.requestFrom(9, 16);  // Request 16 bytes from slave device #9 (IR Sensor)
 while(Wire.available())  // Loop until all the data has been read
 {
  if (n < 16)
  {
   dataRaw[n] = Wire.read();  // Read a byte and store in raw data array
   n++;
  }
  else
  {
   Wire.read();  // Discard any bytes over the 16 we need
   //n = 0;
  }
 }

 // Loop through and covert two 8 bit values to one 16 bit value
 // Raw data formatted as "MSBs 10 9 8 7 6 5 4 3", "x x x x x 2 1 LSBs"
 for(n=0;n<8;n++)
 {
  sensorData[n] = dataRaw[n*2]<< 2;   // Shift the 8 MSBs up two places and store in array
  sensorData[n] += dataRaw[(n*2)+1];  // Add the remaining bottom 2 LSBs
 }
}
```

bluetooth HC 06 module test code:

```
#include <SoftwareSerial.h>

SoftwareSerial hc06(2,3);

void setup(){
//Initialize Serial Monitor
Serial.begin(9600);
Serial.println("ENTER AT Commands:");
//Initialize Bluetooth Serial Port
hc06.begin(9600);
}

void loop(){
//Write data from HC06 to Serial Monitor
if (hc06.available()){
Serial.write(hc06.read());
}

//Write from Serial Monitor to HC06
if (Serial.available()){
hc06.write(Serial.read());
}

}
```

joystick test code:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // CE, CSN
const byte address[6] = "00001";

#define joyX A0
#define joyY A1

float xValue, yValue;

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}

void loop() {

  xValue = analogRead(joyX);
  yValue = analogRead(joyY);

  //print the values with to plot or view
  Serial.print(xValue);
  Serial.print("\t");
  Serial.println(yValue);

  radio.write(&xValue, sizeof(xValue));

}
```

task b1 bidirectional communication with rf24 receiver code:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>

RF24 radio(9, 10); // CE, CSN
byte A_address[6] = "00001";
byte B_address[6] = "00002";

long unsigned int encoder1Value = 0;
long unsigned int encoder2Value = 0;
long unsigned int encoder3Value = 0;
long unsigned int encoder4Value = 0;

char* stage[20];
int reader[2];
int distance;
```

```
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(103);
  radio.openWritingPipe(A_address);
  radio.openReadingPipe(1, B_address);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening();
}

void loop() {

  int x = 0, y = 0;

  if (radio.available()) {
    radio.read(&reader, sizeof(reader));

    x = reader[0];
    y = reader[1];

    Serial.print("x value: ");
    Serial.print(x);
    Serial.print("    y value:");
    Serial.println(y);
  }

  radio.stopListening();

  if (x<10 && x>-10 && y<10 && y>-10)
  {
  Wire.beginTransmission(42);
     Wire.write("baffff");
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
       Wire.write(0);
  Wire.endTransmission();
  stage[20] = "stop";
  }

  if( x > 10)
  {
     Wire.beginTransmission(42);
     Wire.write("baffff");
       Wire.write(x);
       Wire.write(0);
       Wire.write(x);
       Wire.write(0);
```

```
    Wire.write(x+6);
    Wire.write(0);
    Wire.write(x+6);
    Wire.write(0);
  Wire.endTransmission();

  readEncoder();
} //foward

if( x < -10 )
{
  Wire.beginTransmission(42);
  Wire.write("barrrr");
    Wire.write(-x);
    Wire.write(0);
    Wire.write(-x);
    Wire.write(0);
    Wire.write(-x+6);
    Wire.write(0);
    Wire.write(-x+6);
    Wire.write(0);
  Wire.endTransmission();
  stage[20] = "running";
  readEncoder();
} //backwards


 if( y < -10 )
{
  Wire.beginTransmission(42);
  Wire.write("barrff");
    Wire.write(-y);
    Wire.write(0);
    Wire.write(-y);
    Wire.write(0);
    Wire.write(-y+6);
    Wire.write(0);
    Wire.write(-y+6);
    Wire.write(0);
  Wire.endTransmission(); //left
  stage[20] = "running";
  readEncoder();
 }

 if( y > 10 )
 {
  Wire.beginTransmission(42);
  Wire.write("baffrr");
    Wire.write(y);
    Wire.write(0);
    Wire.write(y);
    Wire.write(0);
    Wire.write(y+6);
```

```
      Wire.write(0);
      Wire.write(y+6);
      Wire.write(0);
    Wire.endTransmission();//right
    stage[20] = "running";
    readEncoder();
  }

  Serial.println(encoder1Value);

  distance = encoder1Value/100;


  radio.write( &stage, sizeof(stage) );
  radio.write( &distance, sizeof(distance));

  radio.startListening();

}



void readEncoder()
{
  long unsigned int encoder1 = 0;
  long unsigned int encoder2 = 0;
  long unsigned int encoder3 = 0;
  long unsigned int encoder4 = 0;
  Wire.beginTransmission(42);
  Wire.write("i");
  Wire.endTransmission();
  delay(1);
  Wire.requestFrom(42,8);
  delay(10);
  //if(Wire.available()==8)
  {
    encoder1 = (long unsigned int) Wire.read();
    encoder1 += ((long unsigned int) Wire.read() <<8);
    encoder1 += ((long unsigned int) Wire.read() <<16);
    encoder1 += ((long unsigned int) Wire.read() <<24);
    encoder2 = (long unsigned int) Wire.read();
    encoder2 += ((long unsigned int) Wire.read() <<8);
    encoder2 += ((long unsigned int) Wire.read() <<16);
    encoder2 += ((long unsigned int) Wire.read() <<24);
  }
  encoder1Value = encoder1;
  encoder2Value = encoder2;
  Wire.requestFrom(42,8);
  delay(10);
  //if(Wire.available()==8)
  {
    encoder3 = (long unsigned int) Wire.read();
    encoder3 += ((long unsigned int) Wire.read() <<8);
```

```
   encoder3 += ((long unsigned int) Wire.read() <<16);
   encoder3 += ((long unsigned int) Wire.read() <<24);
   encoder4 = (long unsigned int) Wire.read();
   encoder4 += ((long unsigned int) Wire.read() <<8);
   encoder4 += ((long unsigned int) Wire.read() <<16);
   encoder4 += ((long unsigned int) Wire.read() <<24);
  }
  encoder3Value = encoder3;
  encoder4Value = encoder4;
}
```

task b1 bidirectional communication with rf24 transmit code:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <LiquidCrystal.h>

const int rs = 17, en = 16, d4 = 4, d5 = 6, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

RF24 radio(9, 10); // CE, CSN
byte A_address[6] = "00001";
byte B_address[6] = "00002";

#define joyX A0
#define joyY A1

int xValue = 0, yValue = 0, distance = 0;
int sent1[2];
char* stage[20];

void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(103);
  radio.openWritingPipe(B_address);
  radio.openReadingPipe(1, A_address);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening();
}

void loop()
{
  radio.stopListening();

  xValue = analogRead(joyX);
  yValue = analogRead(joyY);

  xValue = map (xValue, 0, 1023, -80, 80);
  yValue = map (yValue, 0, 1023, -80, 80);
```

```
  //print the values with to plot or view
  Serial.print(xValue);
  Serial.print("\t");
  Serial.println(yValue);

  if (xValue>10 || xValue<-10 || yValue>10 || yValue<-10 )
  {
    stage[20] = "running";
  }
  else
  {
    stage[20] = "stop";
  }

  sent1[0] = xValue;
  sent1[1] = yValue;

  radio.write(&sent1, sizeof(sent1));

  radio.startListening();

  radio.read( &stage, sizeof(stage) );
  radio.read( &distance, sizeof(distance) );

  lcd.setCursor(0, 0);
  lcd.print("stage:");
  lcd.setCursor(7, 0);
  lcd.print(stage[20]);
  lcd.setCursor(0, 1);
  lcd.print("V:");
  lcd.setCursor(3, 1);
  lcd.print(xValue-1);
  lcd.setCursor(7, 1);
  lcd.print("D:");
  lcd.setCursor(10, 1);
  lcd.print(distance);

  delay(10);
  lcd.clear();
}
```

Task A code:

```
#include <Wire.h>
#include <PinChangeInterrupt.h>
#include<I2Cdev.h>
#include <MPU6050.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // CE, CSN
const byte address[6] = "00003"; //rf24 init
```

```
char sent[3]; //rf24 init

 void foward();
 void left();
 void right();
 void back();
 void upmode();
 void stopmotor();
 void downmode();

 int cnt = 0;
 long unsigned int m = 0;


//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&
MPU6050 accelgyro;

unsigned long now, lastTime = 0;
float dt;                          //微分时间

int16_t ax, ay, az, gx, gy, gz;           //加速度计陀螺仪原始数据
float aax=0, aay=0,aaz=0, agx=0, agy=0, agz=0;    //角度变量

long axo = 0, ayo = 0, azo = 0;           //加速度计偏移量
long gxo = 0, gyo = 0, gzo = 0;           //陀螺仪偏移量

float pi = 3.1415926;
float AcceRatio = 16384.0;              //加速度计比例系数
float GyroRatio = 131.0;                //陀螺仪比例系数

uint8_t n_sample = 8;                   //加速度计滤波算法采样个数
float aaxs[8] = {0}, aays[8] = {0}, aazs[8] = {0};        //x,y 轴采样队列
long aax_sum, aay_sum,aaz_sum;                   //x,y 轴采样和

float a_x[10]={0}, a_y[10]={0},a_z[10]={0} ,g_x[10]={0} ,g_y[10]={0},g_z[10]={0}; //加速度计协方
差计算队列
float Px=1, Rx, Kx, Sx, Vx, Qx;           //x 轴卡尔曼变量
float Py=1, Ry, Ky, Sy, Vy, Qy;           //y 轴卡尔曼变量
float Pz=1, Rz, Kz, Sz, Vz, Qz;           //z 轴卡尔曼变量
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

long unsigned int encoder1Value = 0;
long unsigned int encoder2Value = 0;
long unsigned int encoder3Value = 0;
long unsigned int encoder4Value = 0;

int* p;
long unsigned int* q;
```

```
int target[] = {2760, 135, 1550, 160, 450, 179, 1100, 100, 3000, 60, 700, 60, 400, 500};  //add stage 13
//          0    1   2    3   4   5    6   7    8   9   10   11   12   13

void setup(){
  Wire.begin();
  Serial.begin(115200);

  radio.begin();
  radio.setChannel(103);
  radio.openWritingPipe(addr);
  radio.setPALevel(RF24_PA_LOW);
  radio.stopListening();            //24 init

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
//   Wire.begin();
//   Serial.begin(115200);
//
    accelgyro.initialize();              //初始化

    unsigned short times = 200;          //采样次数
    for(int i=0;i<times;i++)
    {
      accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); //读取六轴原始数值
      axo += ax; ayo += ay; azo += az;     //采样和
      gxo += gx; gyo += gy; gzo += gz;

    }

    axo /= times; ayo /= times; azo /= times; //计算加速度计偏移
    gxo /= times; gyo /= times; gzo /= times; //计算陀螺仪偏移
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

//  int target[] = {2900, 230, 1246, 230, 556, 250, 900, 104, 2900, 230, 790, 230, 700};  //add stage 13
////           0    1    2   3   4   5   6   7    8   9   10  11   12

  int spintarget = 1400 ;
  int uptarget = 1150  ;
  int downtarget = 800;
  int target_left = 0;

  int a = 0;
  int extra_distance = 100;
  int t = 0;

//  long unsigned int m = 0;
  long unsigned int p = 0;
  long unsigned int x = 0;
  long unsigned int y = 0;
  long unsigned int z = 0;
```

```
  long unsigned int r = 0; //save encoder value

  unsigned char i=0; //motor number

 //  int cnt = 0;

  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  digitalWrite(13,LOW);
  delay(100);

 //*****************************************************
  cnt = 0;
  p = &cnt;
  foward();
 //  sent[1] = "Run";
  a = target[cnt];
  readEncoder();
  m = encoder1Value;
  q = &m;
  while(encoder1Value < m + target[cnt]){
   readEncoder();
  }
  stopmotor();
  delay(500);
  t = target[cnt];


 //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$

  if(target[cnt] < a){
   readEncoder();
   p = encoder1Value;
   upmode();
   while(encoder1Value < p + uptarget){
    readEncoder();
   }
   stopmotor();  //up
   delay(2000);


   readEncoder();
   x = encoder1Value;
   right();
   while(encoder1Value < x + spintarget){
    readEncoder();
   }
   stopmotor();  //spin
   delay(500);

   readEncoder();
   y = encoder1Value;
```

```
      downmode();
      while(encoder1Value < y + downtarget){
        readEncoder();
      }
      stopmotor();  //down
      delay(2000);


      readEncoder();
      z = encoder1Value;

    //    target_left = a - ((z - m) - extra_distance);
     //   target_left = a - (target[cnt] - extra_up_distance + foward + down);
       target_left = a - t - 1500;
       foward();

      while(encoder1Value < z + target_left){
        readEncoder();
      }
      stopmotor();
      delay(2000);  //foward_left
    }
    //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$

    cnt = 1;
    left();
    readEncoder();
    r = encoder1Value;
    while(encoder1Value > r - target[cnt]){
      readEncoder();
    }
    stopmotor();
    delay(500);

    cnt = 2;
    foward();
    readEncoder();
    r = encoder1Value;
    while(encoder1Value < r + target[cnt]){
      readEncoder();
    }
    stopmotor();
    delay(500);


    cnt = 3;
    left();
    readEncoder();
    r = encoder1Value;
    while(encoder1Value > r - target[cnt]){
      readEncoder();
    }
```

```
  stopmotor();
  delay(500);


  cnt = 4;
  back();
  readEncoder();
  r = encoder1Value;
  while(encoder1Value > r - target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

  cnt = 5;
  left();
  readEncoder();
  r = encoder1Value;
  while(encoder1Value > r - target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

  cnt = 6;
  foward();
  readEncoder();
  r = encoder1Value;
  while(encoder1Value < r + target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);


  cnt = 7;
  left();
  readEncoder();
  r = encoder1Value;
  while(encoder1Value > r - target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

//*****************************************
  cnt = 8;
  p = &cnt;
  a = target[cnt];
  foward();
  readEncoder();
  m = encoder1Value;
  q = &m;
```

```
  while(encoder1Value < m + target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);
  t = target[cnt];

//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
  if(target[cnt] < a){

    readEncoder();
    p = encoder1Value;
    upmode();
    while(encoder1Value < p + uptarget){
      readEncoder();
    }
    stopmotor();  //up
    delay(500);


    readEncoder();
    x = encoder1Value;
    right();
    while(encoder1Value < x + spintarget){
      readEncoder();
    }
    stopmotor();  //spin
    delay(500);

    readEncoder();
    y = encoder1Value;
    downmode();
    while(encoder1Value < y + downtarget){
      readEncoder();
    }
    stopmotor();  //down
    delay(500);


    readEncoder();
    z = encoder1Value;
    foward();

//    target_left = a - ((z - m) - extra_distance);
//    target_left = a - (target[cnt] - extra_up_distance + foward + down); //choose one
     target_left = a - t - 1500;

    while(encoder1Value < z + target_left){
      readEncoder();
    }
    stopmotor();
    delay(500);  //foward_left
  }
```

```
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  cnt = 9;
  readEncoder();
  r = encoder1Value;
  left();
//  readEncoder();
//  r = encoder1Value;
  while(encoder1Value > r - target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

  cnt = 10;
  readEncoder();
  r = encoder1Value;
  foward();
//  readEncoder();
//  r = encoder1Value;
  while(encoder1Value < r + target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

  cnt = 11;
  left();
  readEncoder();
  r = encoder1Value;
  while(encoder1Value > r - target[cnt]){
    readEncoder();
  }
  stopmotor();
  delay(500);

  cnt = 12;
  readEncoder();
  r = encoder1Value;
  Wire.beginTransmission(42);//left forward 0.4m
  Wire.write("sa");
    Wire.write(20);
    Wire.write(0);
    Wire.write(20);
    Wire.write(0);
    Wire.write(50);
    Wire.write(0);
    Wire.write(50);
    Wire.write(0);
  Wire.endTransmission();
//  readEncoder();
//  r = encoder1Value;
  while(encoder1Value < r + target[cnt]){
```

```
  readEncoder();
 }
 stopmotor();
 delay(500);



 cnt = 13;
 readEncoder();
 r = encoder1Value;
 foward();
 while(encoder1Value < r + target[cnt]){
  readEncoder();
 }
 stopmotor();
 delay(500);
}


void loop(){

}


void readEncoder(){
 long unsigned int encoder1 = 0;
 long unsigned int encoder2 = 0;
 long unsigned int encoder3 = 0;
 long unsigned int encoder4 = 0;
 Wire.beginTransmission(42);
 Wire.write("i");
 Wire.endTransmission();
 delay(1);
 Wire.requestFrom(42,8);
 delay(10);
 //if(Wire.available()==8)
 {
  encoder1 = (long unsigned int) Wire.read();
  encoder1 += ((long unsigned int) Wire.read() <<8);
  encoder1 += ((long unsigned int) Wire.read() <<16);
  encoder1 += ((long unsigned int) Wire.read() <<24);
  encoder2 = (long unsigned int) Wire.read();
  encoder2 += ((long unsigned int) Wire.read() <<8);
  encoder2 += ((long unsigned int) Wire.read() <<16);
  encoder2 += ((long unsigned int) Wire.read() <<24);
 }
 encoder1Value = encoder1;
 encoder2Value = encoder2;
 Wire.requestFrom(42,8);
 delay(10);
 //if(Wire.available()==8)
 {
  encoder3 = (long unsigned int) Wire.read();
```

```
  encoder3 += ((long unsigned int) Wire.read() <<8);
  encoder3 += ((long unsigned int) Wire.read() <<16);
  encoder3 += ((long unsigned int) Wire.read() <<24);
  encoder4 = (long unsigned int) Wire.read();
  encoder4 += ((long unsigned int) Wire.read() <<8);
  encoder4 += ((long unsigned int) Wire.read() <<16);
  encoder4 += ((long unsigned int) Wire.read() <<24);
 }
 encoder3Value = encoder3;
 encoder4Value = encoder4;

//  refresh 6050 valule
  unsigned long now = millis();          //当前时间(ms)
  dt = (now - lastTime) / 1000.0;        //微分时间(s)
  lastTime = now;                    //上一次采样时间(ms)

  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); //读取六轴原始数值

  float accx = ax / AcceRatio;           //x 轴加速度
  float accy = ay / AcceRatio;           //y 轴加速度
  float accz = az / AcceRatio;           //z 轴加速度

  aax = atan(accy / accz) * (-180) / pi;   //y 轴对于 z 轴的夹角
  aay = atan(accx / accz) * 180 / pi;      //x 轴对于 z 轴的夹角
  aaz = atan(accz / accy) * 180 / pi;      //z 轴对于 y 轴的夹角

  aax_sum = 0;                     // 对于加速度计原始数据的滑动加权滤波算法
  aay_sum = 0;
  aaz_sum = 0;

  for(int i=1;i<n_sample;i++)
  {
    aaxs[i-1] = aaxs[i];
    aax_sum += aaxs[i] * i;
    aays[i-1] = aays[i];
    aay_sum += aays[i] * i;
    aazs[i-1] = aazs[i];
    aaz_sum += aazs[i] * i;

  }

  aaxs[n_sample-1] = aax;
  aax_sum += aax * n_sample;
  aax = (aax_sum / (11*n_sample/2.0)) * 9 / 7.0; //角度调幅至 0-90°
  aays[n_sample-1] = aay;                //此处应用实验法取得合适的系数
  aay_sum += aay * n_sample;               //本例系数为 9/7
  aay = (aay_sum / (11*n_sample/2.0)) * 9 / 7.0;
  aazs[n_sample-1] = aaz;
  aaz_sum += aaz * n_sample;
  aaz = (aaz_sum / (11*n_sample/2.0)) * 9 / 7.0;

  float gyrox = - (gx-gxo) / GyroRatio * dt; //x 轴角速度
```

```
float gyroy = - (gy-gyo) / GyroRatio * dt; //y 轴角速度
float gyroz = - (gz-gzo) / GyroRatio * dt; //z 轴角速度
agx += gyrox;                    //x 轴角速度积分
agy += gyroy;                    //x 轴角速度积分
agz += gyroz;

/* kalman start */
Sx = 0; Rx = 0;
Sy = 0; Ry = 0;
Sz = 0; Rz = 0;

for(int i=1;i<10;i++)
{          //测量值平均值运算
   a_x[i-1] = a_x[i];            //即加速度平均值
   Sx += a_x[i];
   a_y[i-1] = a_y[i];
   Sy += a_y[i];
   a_z[i-1] = a_z[i];
   Sz += a_z[i];
}

a_x[9] = aax;
Sx += aax;
Sx /= 10;                    //x 轴加速度平均值
a_y[9] = aay;
Sy += aay;
Sy /= 10;                    //y 轴加速度平均值
a_z[9] = aaz;
Sz += aaz;
Sz /= 10;

for(int i=0;i<10;i++)
{
   Rx += sq(a_x[i] - Sx);
   Ry += sq(a_y[i] - Sy);
   Rz += sq(a_z[i] - Sz);
}

Rx = Rx / 9;                    //得到方差
Ry = Ry / 9;
Rz = Rz / 9;

Px = Px + 0.0025;                // 0.0025 在下面有说明...
Kx = Px / (Px + Rx);             //计算卡尔曼增益
agx = agx + Kx * (aax - agx);        //陀螺仪角度与加速度计速度叠加
Px = (1 - Kx) * Px;              //更新 p 值

Py = Py + 0.0025;
Ky = Py / (Py + Ry);
agy = agy + Ky * (aay - agy);
Py = (1 - Ky) * Py;
```

```
    Pz = Pz + 0.0025;
    Kz = Pz / (Pz + Rz);
    agz = agz + Kz * (aaz - agz);
    Pz = (1 - Kz) * Pz;

    /* kalman end */

    Serial.print(agx);Serial.print(",");
    Serial.print(agy);Serial.print(",");
    Serial.print(agz);Serial.println();
//  refresh 6050 valule

  if(agx > 10){
    target[cnt] = encoder1Value - m;
  }
}




// moving functions
void foward(){
  Wire.beginTransmission(42);//foward1 3m
  Wire.write("sa");
    Wire.write(42);
    Wire.write(0);
    Wire.write(42);
    Wire.write(0);
    Wire.write(43);
    Wire.write(0);
    Wire.write(43);
    Wire.write(0);
  Wire.endTransmission();
}

void left(){
  Wire.beginTransmission(42);//FOUR_FIVE_TURN1 45degree
  Wire.write("barrff");
 Wire.write(43);
    Wire.write(0);
    Wire.write(43);
    Wire.write(0);
    Wire.write(43);
    Wire.write(0);
    Wire.write(43);
    Wire.write(0);
  Wire.endTransmission();
}

void right(){
  Wire.beginTransmission(42);//FOUR_FIVE_TURN1 45degree
  Wire.write("baffrr");
```

```
    Wire.write(50);
   Wire.write(0);
   Wire.write(50);
   Wire.write(0);
   Wire.write(56);
   Wire.write(0);
   Wire.write(58);
   Wire.write(0);
  Wire.endTransmission();
}


void back(){
  Wire.beginTransmission(42);//backward 0.5m
  Wire.write("barrrr");
   Wire.write(42);
   Wire.write(0);
   Wire.write(42);
   Wire.write(0);
   Wire.write(43);
   Wire.write(0);
   Wire.write(43);
   Wire.write(0);
  Wire.endTransmission();
}

void stopmotor(){
   Wire.beginTransmission(42);
   Wire.write("ha");
   Wire.endTransmission();  //stop motor
}

void upmode(){
   Wire.beginTransmission(42);
   Wire.write("sa");
     Wire.write(54);
     Wire.write(0);
     Wire.write(54);
     Wire.write(0);
     Wire.write(53);
     Wire.write(0);
     Wire.write(53);
     Wire.write(0);
   Wire.endTransmission(); //upmode init
}

void downmode(){
   Wire.beginTransmission(42);
   Wire.write("sa");
     Wire.write(20);
     Wire.write(0);
     Wire.write(20);
     Wire.write(0);
```

```
    Wire.write(23);
    Wire.write(0);
    Wire.write(23);
    Wire.write(0);
   Wire.endTransmission();
}
```

individual colour test code:

```
const int pinIRd = 8;
const int pinIRa = A0;
const int pinLED = 9;
float IRvalueA = 0;
int IRvalueD = 0;
float voltage = 0;

void setup()
{
 Serial.begin(9600);
 pinMode(pinIRd,INPUT);
 pinMode(pinIRa,INPUT);
 pinMode(pinLED,OUTPUT);

}

void loop()
{
 voltage = IRvalueA * (0.00488);

 Serial.print("Analog Reading=");
 Serial.print(IRvalueA);
 Serial.print("\t Digital Reading=");
 Serial.println(IRvalueD);
 Serial.print("Voltage=");
 Serial.println(voltage, 4);
 delay(100);

 IRvalueA = analogRead(pinIRa);
 IRvalueD = digitalRead(pinIRd);
}
```

Bang-bang control basic line following code:

```
#include <MsTimer2.h>
#include <Wire.h>

void straight();
void left();
void right();
void onTimer();
void right_enhance();
void left_enhance();
```

```
#define pin1a A0;
#define pin2a A1;

int a1 = 0;
int a2 = 0;


void setup() {
  Serial.begin(9600);
  Wire.begin();

  pinMode(A0, INPUT);
  pinMode(A1, INPUT);

  a1 = analogRead(A0);
  a2 = analogRead(A1);

  Serial.println(a1);
  Serial.println("&&&&&&&&&&&&");
  Serial.println(a2);
  Serial.println("%%%%%%%%%%%%");
}

void loop() {
  a1 = analogRead(A0);
  a2 = analogRead(A1);

  if(a1 < 50 && a2 < 50){
    straight();
    Serial.println(a1);
    Serial.println(a2);
  }

  if(a2 > 100  && a1 < 50){
    right();
    Serial.println(a1);
    Serial.println(a2);
    delay(100);
  }

  if(a1 > 100  && a2 < 50){
    left();
    Serial.println(a1);
    Serial.println(a2);
    delay(180);
  }
}

void straight(){
  Wire.beginTransmission(42);
  Wire.write("baffff");
    Wire.write(13);
```

```
    Wire.write(0);
    Wire.write(13);
    Wire.write(0);
    Wire.write(16);
    Wire.write(0);
    Wire.write(16 );
    Wire.write(0);
  Wire.endTransmission();
}

void left(){
  Wire.beginTransmission(42);
  Wire.write("barrff");
    Wire.write(65);
    Wire.write(0);
    Wire.write(65);
    Wire.write(0);
    Wire.write(61);
    Wire.write(0);
    Wire.write(61);
    Wire.write(0);
  Wire.endTransmission();
}

void right(){
  Wire.beginTransmission(42);
  Wire.write("baffrr");
    Wire.write(99);
    Wire.write(0);
    Wire.write(99);
    Wire.write(0);
    Wire.write(60);
    Wire.write(0);
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();
}

void right_enhance(){
  Wire.beginTransmission(42);
  Wire.write("baffrr");
    Wire.write(75);
    Wire.write(0);
    Wire.write(70);
    Wire.write(0);
    Wire.write(73);
    Wire.write(0);
    Wire.write(73);
    Wire.write(0);
  Wire.endTransmission();
}

void left_enhance(){
```

```
  Wire.beginTransmission(42);
  Wire.write("barrff");
    Wire.write(75);
    Wire.write(0);
    Wire.write(70);
    Wire.write(0);
    Wire.write(73);
    Wire.write(0);
    Wire.write(73);
    Wire.write(0);
  Wire.endTransmission();
}
```

Weighted Average Calculation code:

```
#include <Wire.h>          // Include the Wire library for I2C communication

unsigned char dataRaw[16];
unsigned int sensorData[8];

double distance = 0;
double sensorValue = 0;

void setup()
{
  Wire.begin();       // Join the I2C bus as a master (address optional for master)
  Serial.begin(57600);  // Start serial for output to PC
}

void loop()
{
  readSensorData();
  Serial.println("---------------");
  Serial.println(sensorData[0]);
  Serial.println(sensorData[1]);
  Serial.println(sensorData[2]);
  Serial.println(sensorData[3]);
  Serial.println(sensorData[4]);
  Serial.println(sensorData[5]);
  Serial.println(sensorData[6]);
  Serial.println(sensorData[7]);
  delay(100);

//  int i = 0;
//  for(i = 0; i < 8; i++){
//    sensorValue += sensorData[i];
//  }
//  delay(100);

  sensorValue = sensorData[0] +sensorData[1] +sensorData[2] +sensorData[3] +
  sensorData[4] +sensorData[5] +sensorData[6] +sensorData[7];
```

```
    distance = ((-45.5)*sensorData[0] + (-32.5)*sensorData[1] + (-19.5)*sensorData[2] + (-6.5)*sensorData[3]
    + 6.5*sensorData[4] + 19.5*sensorData[5] + 32.5*sensorData[6] + 45.5*sensorData[7]) / (sensorValue/8);

    Serial.print("Distance = ");
    Serial.println(distance);
    delay(200);
  }

  void readSensorData(void)
  {
    unsigned char n;          // Variable for counter value
    unsigned char dataRaw[16];  // Array for raw data from module

    // Request data from the module and store into an array
    n = 0;  // Reset loop variable
    Wire.requestFrom(9, 16);  // Request 16 bytes from slave device #9 (IR Sensor)
    while(Wire.available())  // Loop until all the data has been read
    {
      if (n < 16)
      {
        dataRaw[n] = Wire.read();  // Read a byte and store in raw data array
        n++;
      }
      else
      {
        Wire.read();  // Discard any bytes over the 16 we need
        //n = 0;
      }
    }

    // Loop through and covert two 8 bit values to one 16 bit value
    // Raw data formatted as "MSBs 10 9 8 7 6 5 4 3", "x x x x x 2 1 LSBs"
    for(n=0;n<8;n++)
    {
      sensorData[n] = dataRaw[n*2]<< 2;   // Shift the 8 MSBs up two places and store in array
      sensorData[n] += dataRaw[(n*2)+1];  // Add the remaining bottom 2 LSBs
    }
  }
```

set encoder and LCD code:

```
#include <PinChangeInterrupt.h>
#include <LiquidCrystal.h>
#include <MsTimer2.h>
#include <Wire.h>

#define encoder0PinA 2
#define encoder0PinB 3    //encoder pins
int encoder0Pos = 0;
int encoder1Pos = 0;    //encoder counter
```

```
float PID[3];
float kp, ki, kd;    //encoder output
int i;
char* k;

void doEncoder();

const int rs = 17, en = 16, d4 = 4, d5 = 6, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);      //LCD pins

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);

 // encoder and lcd control
 lcd.begin(16, 2);
 lcd.setCursor(0, 0);

 pinMode(encoder0PinA, INPUT_PULLUP);
 pinMode(encoder0PinB, INPUT_PULLUP);    //encoder init
 pinMode( 1, INPUT_PULLUP);

 attachInterrupt( 0, doEncoder, CHANGE);

 while (1)
 {
  k = "Kp = ";
  i = 0;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID[0]);
 }

 digitalWrite(1, HIGH);
 encoder0Pos = 0;
 delay(200);

 while (1)
 {
  k = "Ki = ";
  i = 1;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID[1]);
 }

 digitalWrite(1, HIGH);
 delay(200);
 encoder0Pos = 0;

 while (1)
```

```
 {
   k = "Kd = ";
   i = 2;
   attachInterrupt( 0, doEncoder, CHANGE);
   delay (15);
   if ( digitalRead(1) == LOW) break;
   Serial.println(PID[2]);
 }
 //end of encoder control

 Serial.print("PID[0] =");
 Serial.println(PID[0]);
 Serial.print("PID[1] =");
 Serial.println(PID[1]);
 Serial.print("PID[2] =");
 Serial.println(PID[2]);
}

void loop() {
 // put your main code here, to run repeatedly:

}

void doEncoder()
 {
   if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB))
   {
     encoder0Pos++;
     PID[i] = (float)encoder0Pos/20;
   }
   else
   {
     encoder0Pos--;
     PID[i] = (float)encoder0Pos/20;
   }

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print( k );
   lcd.setCursor(5, 0);
   lcd.print(PID[i]);

 }
```

PID control with encoder code:

```
#include <Wire.h>          // Include the Wire library for I2C communication
#include <LiquidCrystal.h>  // Include the Liquid Crystal library for driving the LCD
#include <PinChangeInterrupt.h>

void readSensorData(void);
float PID(float lineDist);
void updatePID(void);
```

```
void switchbutton();
void doEncoder();

#define encoder0PinA 2
#define encoder0PinB 3    //encoder pins

int encoder0Pos = 0;
int encoder1Pos = 0;    //encoder counter

float PID1[3];
float Kp, Ki, Kd;    //encoder output
int i;
char* k;
//encoder values

#define leftMotorBaseSpeed 40
#define rightMotorBaseSpeed 40

#define min_speed -50
#define max_speed 50

// Resolution of the normaised outputs from the sensor board
//(10-bit results are constrained to the calibration limits then remapped to this range)
#define outputRange 255

const int rs = 17, en = 16, d4 = 4, d5 = 6, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);      //LCD pins

unsigned int sensorData[8];     // Array for formatted data from module
int sensorWCal[8] = {1023,1023,1023,1023,1023,1023,1023,1023};    // Calibration values for
sensor (initalised to full range [10-bit results])
int sensorBCal[8] = {0,0,0,0,0,0,0,0};


float error, errorSum, errorOld;  // Variables for the PID loop
float leftMotorSpeed = 0; // Variables to hold the current motor speed (+-100%)
float rightMotorSpeed = 0;

int cnt = 0;
double distance = 0;
int sensorValue = 0;

void setup()
{
  Wire.begin();       // Join the I2C bus as a master (address optional for master)
  Serial.begin(115200);  // Start serial for output to PC
  lcd.begin(16, 2);    // Initialise the LCD

  leftMotorSpeed = 0;  // Initialise Speed variables
  rightMotorSpeed = 0;

  error = 0;    // Initialise error variables
  errorSum = 0;
```

```
errorOld = 0;

Serial.println("Line Follower Using Weighted Average");
delay(10);

// encoder and lcd control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
lcd.setCursor(0, 0);

pinMode(encoder0PinA, INPUT_PULLUP);
pinMode(encoder0PinB, INPUT_PULLUP);    //encoder init
pinMode( 1, INPUT_PULLUP);

attachInterrupt( 0, doEncoder, CHANGE);

while (1)
{
  k = "Kp = ";
  i = 0;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[0]);
}

digitalWrite(1, HIGH);
encoder0Pos = 0;
delay(200);

while (1)
{
  k = "Ki = ";
  i = 1;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[1]);
}

digitalWrite(1, HIGH);
delay(200);
encoder0Pos = 0;

while (1)
{
  k = "Kd = ";
  i = 2;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[2]);
}
```

```
    //end of encoder control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

     Kp = PID1[0];
     Ki = PID1[1];
     Kd = PID1[2];

    }

    void loop()
    {
     readSensorData();   // Get the data from the sensor board
     Serial.println("---------------");
     Serial.println(sensorData[0]);
     Serial.println(sensorData[1]);
     Serial.println(sensorData[2]);
     Serial.println(sensorData[3]);
     Serial.println(sensorData[4]);
     Serial.println(sensorData[5]);
     Serial.println(sensorData[6]);
     Serial.println(sensorData[7]);

     Serial.println("######################");


     sensorValue = sensorData[0] +sensorData[1] +sensorData[2] +sensorData[3] +
     sensorData[4] +sensorData[5] +sensorData[6] +sensorData[7];

     distance = ((-45.5)*sensorData[0] + (-32.5)*sensorData[1] + (-19.5)*sensorData[2] + (-6.5)*sensorData[3]
     + 6.5*sensorData[4] + 19.5*sensorData[5] + 32.5*sensorData[6] + 45.5*sensorData[7]) / (sensorValue/10);

     Serial.print("Distance = ");
     Serial.println(distance);

     float output = PID(distance); // Calculate the PID output.
     Serial.print("Output");
     Serial.println(output);

     leftMotorSpeed = leftMotorBaseSpeed + output;     // Calculate the modified motor speed
     rightMotorSpeed = rightMotorBaseSpeed - output;

     // Apply new speed and direction to each motor

     if(rightMotorSpeed > 0 && leftMotorSpeed >0){
     rightMotorSpeed = constrain(rightMotorSpeed, 0, 50);
     leftMotorSpeed = constrain(leftMotorSpeed, 0, 50);
      Serial.print("left");
     Serial.println(leftMotorSpeed);
     Serial.print("right");
     Serial.println(rightMotorSpeed);
     Wire.beginTransmission(42);
```

```
  Wire.write("baffff");
   Wire.write((int)leftMotorSpeed);
   Wire.write(0);
   Wire.write((int)leftMotorSpeed);
   Wire.write(0);
   Wire.write((int)rightMotorSpeed);
   Wire.write(0);
   Wire.write((int)rightMotorSpeed);
   Wire.write(0);
  Wire.endTransmission();
  }

  if(leftMotorSpeed < 0)
  {
  rightMotorSpeed = constrain(rightMotorSpeed, 0, max_speed);
  Wire.beginTransmission(42);
  Wire.write("barrff");
   Wire.write(-(int)leftMotorSpeed);
   Wire.write(0);
   Wire.write(-(int)leftMotorSpeed);
   Wire.write(0);
   Wire.write((int)rightMotorSpeed);
   Wire.write(0);
   Wire.write((int)rightMotorSpeed);
   Wire.write(0);
  Wire.endTransmission();
  }

  if(rightMotorSpeed < 0)
  {
  leftMotorSpeed = constrain(leftMotorSpeed, 0, max_speed);
  Wire.beginTransmission(42);
  Wire.write("baffrr");
   Wire.write((int)leftMotorSpeed);
   Wire.write(0);
   Wire.write((int)leftMotorSpeed);
   Wire.write(0);
   Wire.write(-(int)rightMotorSpeed);
   Wire.write(0);
   Wire.write(-(int)rightMotorSpeed);
   Wire.write(0);
  Wire.endTransmission();
  }
 }

 // Function to read the data from the IR sensor board
 void readSensorData(void)
 {
 //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
  unsigned char n;          // Variable for counter value
  unsigned char dataRaw[16];  // Array for raw data from module
```

```
 // Request data from the module and store into an array
 n = 0;  // Reset loop variable
 Wire.requestFrom(9, 16);  // Request 16 bytes from slave device #9 (IR Sensor)
 while (Wire.available())  // Loop until all the data has been read
 {
  if (n < 16)
  {
   dataRaw[n] = Wire.read();  // Read a byte and store in raw data array
   n++;
  }
  else
  {
   Wire.read();  // Discard any bytes over the 16 we need
   n = 0;
  }
 }

 // Loop through and covert two 8 bit values to one 16 bit value
 // Raw data formatted as "MSBs 10 9 8 7 6 5 4 3", "x x x x x 2 1 LSBs"
 for(n=0;n<8;n++)
 {
  sensorData[n] = dataRaw[n*2]<< 2;   // Shift the 8 MSBs up two places and store in array
  sensorData[n] += dataRaw[(n*2)+1];  // Add the remaining bottom 2 LSBs

  map(sensorData[n], 0, 1023, sensorBCal[n], sensorWCal[n]); //
 }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

}

// Function to calculate the PID output.
float PID(float lineDist)
{
 // PID loop
 errorOld = error;       // Save the old error for differential component
 error = 0 - lineDist;  // Calculate the error in position
 errorSum += error;

 float proportional = error * Kp;  // Calculate the components of the PID

 float integral = errorSum * Ki;

 float differential = (error - errorOld) * Kd;

 float output = proportional + integral + differential;  // Calculate the result

 return output;
}


void doEncoder()
 {
   if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB))
```

```
  {
    encoder0Pos++;
    PID1[i] = (float)encoder0Pos/20;
  }
  else
  {
    encoder0Pos--;
    PID1[i] = (float)encoder0Pos/20;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print( k );
  lcd.setCursor(5, 0);
  lcd.print(PID1[i]);
 }
```

PID control with LCD code:

```
#include <Wire.h>          // Include the Wire library for I2C communication
#include <LiquidCrystal.h>  // Include the Liquid Crystal library for driving the LCD
#include <PinChangeInterrupt.h>
#include <MsTimer2.h>

void readSensorData(void);
float PID(float lineDist);
void updatePID(void);
void Running();
void doEncoder();

#define encoder0PinA 2
#define encoder0PinB 3    //encoder pins

int encoder0Pos = 0;
int encoder1Pos = 0;    //encoder counter

float PID1[3];
float Kp, Ki, Kd;    //encoder output
int i, t = 1;
int tim = (millis())/1000;
char* k;
//encoder values

#define leftMotorBaseSpeed 40
#define rightMotorBaseSpeed 40

#define min_speed -60
#define max_speed 60

// Resolution of the normaised outputs from the sensor board
//(10-bit results are constrained to the calibration limits then remapped to this range)
#define outputRange 255
```

```
const int rs = 17, en = 16, d4 = 4, d5 = 6, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);      //LCD pins

unsigned int sensorData[8];     // Array for formatted data from module
int sensorWCal[8] = {1023,1023,1023,1023,1023,1023,1023,1023};   // Calibration values for sensor (initalised to full range [10-bit results])
int sensorBCal[8] = {0,0,0,0,0,0,0,0};


float error, errorSum, errorOld;  // Variables for the PID loop
float leftMotorSpeed = 0; // Variables to hold the current motor speed (+-100%)
float rightMotorSpeed = 0;

int cnt = 0;
double distance = 0;
int sensorValue = 0;

void setup()
{
 Wire.begin();       // Join the I2C bus as a master (address optional for master)
 Serial.begin(115200);  // Start serial for output to PC
 lcd.begin(16, 2);    // Initialise the LCD

 leftMotorSpeed = 0;  // Initialise Speed variables
 rightMotorSpeed = 0;

 error = 0;    // Initialise error variables
 errorSum = 0;
 errorOld = 0;

 Serial.println("Line Follower Using Weighted Average");
 delay(10);

 //                                encoder                                and                                lcd control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 lcd.setCursor(0, 0);

 pinMode(encoder0PinA, INPUT_PULLUP);
 pinMode(encoder0PinB, INPUT_PULLUP);    //encoder init
 pinMode( 1, INPUT_PULLUP);

 attachInterrupt( 0, doEncoder, CHANGE);

 while (1)
 {
  k = "Kp = ";
  i = 0;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[0]);
 }
```

```
  digitalWrite(1, HIGH);
  encoder0Pos = 0;
  delay(200);

  while (1)
  {
   k = "Ki = ";
   i = 1;
   attachInterrupt( 0, doEncoder, CHANGE);
   delay (15);
   if ( digitalRead(1) == LOW) break;
   Serial.println(PID1[1]);
  }

  digitalWrite(1, HIGH);
  delay(200);
  encoder0Pos = 0;

  while (1)
  {
   k = "Kd = ";
   i = 2;
   attachInterrupt( 0, doEncoder, CHANGE);
   delay (15);
   if ( digitalRead(1) == LOW) break;
   Serial.println(PID1[2]);
  }
  //end of encoder control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

  Kp = PID1[0];
  Ki = PID1[1];
  Kd = PID1[2];

  //lcd control &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
  if ( tim == t ){

   t++;
  }

  MsTimer2::set(5, Running);
  MsTimer2::start();

 }//end setup

 void loop()
 {
   //lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print( "Run" );
   lcd.setCursor(4, 0);
   lcd.print( "L:" );
   lcd.setCursor(3, 0);
```

```
  lcd.print( " " );
  lcd.setCursor(6, 0);
  lcd.print(leftMotorSpeed);
  lcd.setCursor(10, 0);
  lcd.print( "R:" );
  lcd.setCursor(12, 0);
  lcd.print(rightMotorSpeed);
  lcd.setCursor(0, 1);
  lcd.print(distance, 1);
  lcd.setCursor(5, 1);
  lcd.print(Kp, 1);
  lcd.setCursor(9, 1);
  lcd.print(Ki, 1);
  lcd.setCursor(13, 1);
  lcd.print(Kd, 1);
}

void Running ()
{
  interrupts();

  readSensorData();   // Get the data from the sensor board
  Serial.println("---------------");
  Serial.println(sensorData[0]);
  Serial.println(sensorData[1]);
  Serial.println(sensorData[2]);
  Serial.println(sensorData[3]);
  Serial.println(sensorData[4]);
  Serial.println(sensorData[5]);
  Serial.println(sensorData[6]);
  Serial.println(sensorData[7]);

  Serial.println("######################");


  sensorValue = sensorData[0] +sensorData[1] +sensorData[2] +sensorData[3] +
  sensorData[4] +sensorData[5] +sensorData[6] +sensorData[7];

  distance = ((-45.5)*sensorData[0] + (-32.5)*sensorData[1] + (-19.5)*sensorData[2] + (-
6.5)*sensorData[3]
  + 6.5*sensorData[4] + 19.5*sensorData[5] + 32.5*sensorData[6] + 45.5*sensorData[7]) /
(sensorValue/8);

  if(distance < -110){
    cnt++;
  if(cnt%2 == 1){
  Wire.beginTransmission(42);
  Wire.write("baffrr");
    Wire.write(60);
    Wire.write(0);
    Wire.write(60);
    Wire.write(0);
    Wire.write(64);
```

```
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
Wire.endTransmission();
delay(490);
}
else{
Wire.beginTransmission(42);
Wire.write("baffrr");
  Wire.write(60);
  Wire.write(0);
  Wire.write(60);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
Wire.endTransmission();
delay(700);
}
}  //right

if(distance > 110){
Wire.beginTransmission(42);
Wire.write("barrff");
  Wire.write(60);
  Wire.write(0);
  Wire.write(60);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
Wire.endTransmission();
delay(400);
}   //left

Serial.print("Distance = ");
Serial.println(distance);

float output = PID(distance); // Calculate the PID output.

Serial.print("Output");
Serial.println(output);

leftMotorSpeed = leftMotorBaseSpeed + output;     // Calculate the modified motor speed
rightMotorSpeed = rightMotorBaseSpeed - output;

// Apply new speed and direction to each motor

if(rightMotorSpeed > 0 && leftMotorSpeed >0){
rightMotorSpeed = constrain(rightMotorSpeed, 0, 50);
leftMotorSpeed = constrain(leftMotorSpeed, 0, 50);
```

```
  Serial.print("left");
 Serial.println(leftMotorSpeed);
 Serial.print("right");
 Serial.println(rightMotorSpeed);
 Wire.beginTransmission(42);
 Wire.write("baffff");
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }

 if(leftMotorSpeed < 0)
 {
 rightMotorSpeed = constrain(rightMotorSpeed, 0, max_speed);
 Wire.beginTransmission(42);
 Wire.write("barrff");
  Wire.write(-(int)leftMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }

 if(rightMotorSpeed < 0)
 {
 leftMotorSpeed = constrain(leftMotorSpeed, 0, max_speed);
 Wire.beginTransmission(42);
 Wire.write("baffrr");
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)rightMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }
}

// Function to read the data from the IR sensor board
void readSensorData(void)
{
```

```
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
  unsigned char n;          // Variable for counter value
  unsigned char dataRaw[16];  // Array for raw data from module

  // Request data from the module and store into an array
  n = 0;  // Reset loop variable
  Wire.requestFrom(9, 16);  // Request 16 bytes from slave device #9 (IR Sensor)
  while (Wire.available())  // Loop until all the data has been read
  {
   if (n < 16)
   {
    dataRaw[n] = Wire.read();  // Read a byte and store in raw data array
    n++;
   }
   else
   {
    Wire.read();  // Discard any bytes over the 16 we need
    n = 0;
   }
  }

  // Loop through and covert two 8 bit values to one 16 bit value
  // Raw data formatted as "MSBs 10 9 8 7 6 5 4 3", "x x x x x x 2 1 LSBs"
  for(n=0;n<8;n++)
  {
   sensorData[n] = dataRaw[n*2]<< 2;   // Shift the 8 MSBs up two places and store in array
   sensorData[n] += dataRaw[(n*2)+1];  // Add the remaining bottom 2 LSBs

   map(sensorData[n], 0, 1023, sensorBCal[n], sensorWCal[n]); //
  }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

}

// Function to calculate the PID output.
float PID(float lineDist)
{
  // PID loop
  errorOld = error;       // Save the old error for differential component
  error = 0 - lineDist;  // Calculate the error in position
  errorSum += error;

  float proportional = error * Kp;  // Calculate the components of the PID

  float integral = errorSum * Ki;

  float differential = (error - errorOld) * Kd;

  float output = proportional + integral + differential;  // Calculate the result

  return output;
}
```

```
void doEncoder()
 {
   if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB))
   {
    encoder0Pos++;
    PID1[i] = (float)encoder0Pos/20;
   }
   else
   {
    encoder0Pos--;
    PID1[i] = (float)encoder0Pos/20;
   }

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print( k );
   lcd.setCursor(5, 0);
   lcd.print(PID1[i]);
   lcd.setCursor(2, 1);
   lcd.print("stop");
}
```

Advanced line following task code：

```
#include <Wire.h>          // Include the Wire library for I2C communication
#include <LiquidCrystal.h>  // Include the Liquid Crystal library for driving the LCD
#include <PinChangeInterrupt.h>
#include <MsTimer2.h>

void readSensorData(void);
float PID(float lineDist);
void updatePID(void);
void Running();
void doEncoder();
void Clear();

#define encoder0PinA 2
#define encoder0PinB 3    //encoder pins

int encoder0Pos = 0;
int encoder1Pos = 0;    //encoder counter

float PID1[3];
float Kp, Ki, Kd;    //encoder output
int i, t = 1;
int tim = (millis())/1000;
char* k;
//encoder values

#define leftMotorBaseSpeed 40
#define rightMotorBaseSpeed 40
```

```
#define min_speed -60
#define max_speed 90

// Resolution of the normaised outputs from the sensor board
//(10-bit results are constrained to the calibration limits then remapped to this range)
#define outputRange 255

const int rs = 17, en = 16, d4 = 4, d5 = 6, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);      //LCD pins

unsigned int sensorData[8];     // Array for formatted data from module
int sensorWCal[8] = {1023,1023,1023,1023,1023,1023,1023,1023};   // Calibration values for sensor (initalised to full range [10-bit results])
int sensorBCal[8] = {0,0,0,0,0,0,0,0};

float error, errorSum, errorOld;  // Variables for the PID loop
float leftMotorSpeed = 0; // Variables to hold the current motor speed (+-100%)
float rightMotorSpeed = 0;

int cnt = 0;
double distance = 0;
int sensorValue = 0;

void setup()
{
  Wire.begin();       // Join the I2C bus as a master (address optional for master)
  Serial.begin(115200);  // Start serial for output to PC
  lcd.begin(16, 2);    // Initialise the LCD

  leftMotorSpeed = 0;  // Initialise Speed variables
  rightMotorSpeed = 0;

  error = 0;    // Initialise error variables
  errorSum = 0;
  errorOld = 0;

  Serial.println("Line Follower Using Weighted Average");
  delay(10);

  // encoder and lcd control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
  lcd.setCursor(0, 0);

  pinMode(encoder0PinA, INPUT_PULLUP);
  pinMode(encoder0PinB, INPUT_PULLUP);    //encoder init
  pinMode( 1, INPUT_PULLUP);

  attachInterrupt( 0, doEncoder, CHANGE);

  while (1)
  {
    k = "Kp = ";
```

```
  i = 0;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[0]);
 }

 digitalWrite(1, HIGH);
 encoder0Pos = 0;
 delay(200);

 while (1)
 {
  k = "Ki = ";
  i = 1;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[1]);
 }

 digitalWrite(1, HIGH);
 delay(200);
 encoder0Pos = 0;

 while (1)
 {
  k = "Kd = ";
  i = 2;
  attachInterrupt( 0, doEncoder, CHANGE);
  delay (15);
  if ( digitalRead(1) == LOW) break;
  Serial.println(PID1[2]);
 }
 //end of encoder control&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 Kp = PID1[0];
 Ki = PID1[1];
 Kd = PID1[2];

 //lcd control &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 if ( tim == t ){

  t++;
 }

 MsTimer2::set(5, Running);  //200Hz
 MsTimer2::start();

}//end setup

void loop()
```

```
{

  lcd.setCursor(0, 0);
  lcd.print( "Run" );
  lcd.setCursor(4, 0);
  lcd.print( "L:" );
  lcd.setCursor(3, 0);
  lcd.print( " " );
  lcd.setCursor(6, 0);
  lcd.print(leftMotorSpeed);
  lcd.setCursor(10, 0);
  lcd.print( "R:" );
  lcd.setCursor(12, 0);
  lcd.print(rightMotorSpeed);
  lcd.setCursor(0, 1);
  lcd.print(-distance, 0);
  if( -distance > 0) {
  lcd.setCursor(2, 1);
  lcd.print(" ");
  }
  lcd.setCursor(3, 1);
  lcd.print("  ");
  lcd.setCursor(5, 1);
  lcd.print(Kp, 1);
  lcd.setCursor(9, 1);
  lcd.print(Ki, 1);
  lcd.setCursor(13, 1);
  lcd.print(Kd, 1);
}

void Running ()
{
  interrupts();

  readSensorData();   // Get the data from the sensor board

  int i;
  for(i = 0; i < 8; i++){
    sensorValue += sensorData[i];
  }

  distance = ((-45.5)*sensorData[0] + (-32.5)*sensorData[1] + (-19.5)*sensorData[2] + (-
6.5)*sensorData[3]
  + 6.5*sensorData[4] + 19.5*sensorData[5] + 32.5*sensorData[6] + 45.5*sensorData[7]) /
(sensorValue/8);

  if(distance < -110){
    cnt++;
    if(cnt%2 == 1){
      Wire.beginTransmission(42);
      Wire.write("baffrr");
        Wire.write(70);
        Wire.write(0);
```

```
    Wire.write(70);
    Wire.write(0);
    Wire.write(64);
    Wire.write(0);
    Wire.write(64);
    Wire.write(0);
   Wire.endTransmission();
   delay(450);
  }
  else{
   Wire.beginTransmission(42);
   Wire.write("baffrr");
    Wire.write(70);
    Wire.write(0);
    Wire.write(70);
    Wire.write(0);
    Wire.write(64);
    Wire.write(0);
    Wire.write(64);
    Wire.write(0);
   Wire.endTransmission();
   delay(650);
  }
} // sharp right

if(distance > 110){
Wire.beginTransmission(42);
Wire.write("barrff");
  Wire.write(60);
  Wire.write(0);
  Wire.write(60);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
  Wire.write(64);
  Wire.write(0);
Wire.endTransmission();
delay(400);
}  //sharp left

float output = PID(distance); // Calculate the PID output.

leftMotorSpeed = leftMotorBaseSpeed + output;     // Calculate the modified motor speed
rightMotorSpeed = rightMotorBaseSpeed - output;

// Apply new speed and direction to each motor

if(rightMotorSpeed > 0 && leftMotorSpeed >0){
rightMotorSpeed = constrain(rightMotorSpeed, 0, 90);
leftMotorSpeed = constrain(leftMotorSpeed, 0, 90);
Wire.beginTransmission(42);
Wire.write("baffff");
  Wire.write((int)leftMotorSpeed);
```

```
  Wire.write(0);
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }

 if(leftMotorSpeed < 0)
 {
 rightMotorSpeed = constrain(rightMotorSpeed, 0, max_speed);
 Wire.beginTransmission(42);
 Wire.write("barrff");
  Wire.write(-(int)leftMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
  Wire.write((int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }

 if(rightMotorSpeed < 0)
 {
 leftMotorSpeed = constrain(leftMotorSpeed, 0, max_speed);
 Wire.beginTransmission(42);
 Wire.write("baffrr");
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write((int)leftMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)rightMotorSpeed);
  Wire.write(0);
  Wire.write(-(int)rightMotorSpeed);
  Wire.write(0);
 Wire.endTransmission();
 }

  constrain(leftMotorSpeed, -100, 90);
  constrain(rightMotorSpeed, -100, 90);
}

// Function to read the data from the IR sensor board
void readSensorData(void)
{
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
 unsigned char n;          // Variable for counter value
 unsigned char dataRaw[16];  // Array for raw data from module
```

```
 // Request data from the module and store into an array
 n = 0;  // Reset loop variable
 Wire.requestFrom(9, 16);  // Request 16 bytes from slave device #9 (IR Sensor)
 while (Wire.available())  // Loop until all the data has been read
 {
   if (n < 16)
   {
     dataRaw[n] = Wire.read();  // Read a byte and store in raw data array
     n++;
   }
   else
   {
     Wire.read();  // Discard any bytes over the 16 we need
     n = 0;
   }
 }

 // Loop through and covert two 8 bit values to one 16 bit value
 // Raw data formatted as "MSBs 10 9 8 7 6 5 4 3", "x x x x x x 2 1 LSBs"
 for(n=0;n<8;n++)
 {
   sensorData[n] = dataRaw[n*2]<< 2;   // Shift the 8 MSBs up two places and store in array
   sensorData[n] += dataRaw[(n*2)+1];  // Add the remaining bottom 2 LSBs

   map(sensorData[n], 0, 1023, sensorBCal[n], sensorWCal[n]); //
  }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

}

// Function to calculate the PID output.
float PID(float lineDist)
{
 // PID loop
 errorOld = error;       // Save the old error for differential component
 error = 0 - lineDist;  // Calculate the error in position
 errorSum += error;

 float proportional = error * Kp;  // Calculate the components of the PID

 float integral = errorSum * Ki;

 float differential = (error - errorOld) * Kd;

 float output = proportional + integral + differential;  // Calculate the result

 return output;
}


void doEncoder()
 {
```

```
   if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB))
   {
     encoder0Pos++;
     PID1[i] = (float)encoder0Pos/20;
   }
   else
   {
     encoder0Pos--;
     PID1[i] = (float)encoder0Pos/20;
   }

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print( k );
   lcd.setCursor(5, 0);
   lcd.print(PID1[i]);
   lcd.setCursor(0, 1);
   lcd.print("stop");
}

void Clear(){
  lcd.clear();
}
```