Applied Electrical and Electronic Engineering: Construction Project (EEEE1039 UNNC)

Project Session III and Project Session IV Individual Report

Student Name: Guan-Sheng Chen

Student ID: 20314386

Group: QY-19

Affiliation: Department of Electrical & Electronic Engineering,

University of Nottingham Ningbo China

Abstract

The purpose for these two experiments was to build a highly advanced vehicle enhanced with motion sensing, remote control and line following. As a result, this study showed the method of building the advanced vehicle, as well as some significant theory and possible analysis. Theory examples included motion sensing, bidirectional wireless communication, remote control, and PID control approaches. All the experiment's data and graphs were also covered. Circuits and codes were particularly important in this experiment because it was done in the background of many changes and tests of circuits and codes. The reconstructed and integrated vehicle, as well as the remote controller, were included in this report.

Introduction

The aims of session 3 and 4 were to enhance the vehicle with motion sensing and remote control to fulfill the tasks about line following.

In the session 3, five subjects were included. The understanding of the MPU-6050 accelerometer and gyroscope module were learned in topic 1. The MPU-6050 sensor was used to measure the inclination after the programs have been designed, in preparation for the final demo. In topic 2, the nRF24L01+ 2.4GHz RF wireless module was utilized to construct a bidirectional communication between two Arduino Nano boards. A Bluetooth wireless communication between an Arduino Nano board and a cellphone was also built using the HC-06. The understanding of logic gates and PWM (Pulse Width Modulation) was studied in topic 3, and the individual task was to construct a logic board (74 series IC based, one channel) as an interface to replace the baseboard CPLD. A handle design with a joystick and an HMI (Human-Machine Interface) was for topic 4. HMI hardware was designed with LCD, joystick, press buttons and any other components, and be attached to the remote controller. The nRF24L01 and MPU-6050 were used for Task B-2, which involved remote control with the

remote handle. The HC-06 was used for task B-3, which required remote control with cellphone.

Session 4 consisted of four themes. In topic 1, the knowledge of TCRT5000 component was learned and it was used to record output voltages (at A0) at various distances from a black and white surface, respectively. It was an individual task to record. In the topic 2, two TCRT5000 sensors were used to do a basic line following. The codes were designed to make the vehicle run two cycles continuously. The PID control was optional to be used with two TCRT5000 sensors in this topic. In the topic 3, the PID control codes had to be designed, and the parameters had to be tweaked with the HMI (LCD + buttons + encoder). The final demonstration was topic 4. The advanced line following task in this work required going through any color two times with SF-8CHIDTS and PID control. There was other three requirements: 1. HMI (LCD + encoder) used to tune PID parameters 2. Controlling RT at a specific frequency (>100Hz) 3. LCD display of system status (run/stop, L&R speeds, PID parameter, and error).

- **Method**

Session 3:

1. Wireless (2.4GHz & Bluetooth) communication

(a.) nRF24L01+wireless module

As shown in Figure 1.1, the nRF24L01+ wireless module was used for wireless system communication. The car controller was modified to allow its movements to be remotely controlled by an operator over a wireless radio frequency (RF) link. The RF modules given used the 2.4GHz band which was split into 128 channels. The unique channel used by group 19 was 91. The connection with Arduino board was shown in Figure 1.2 and Table 1.

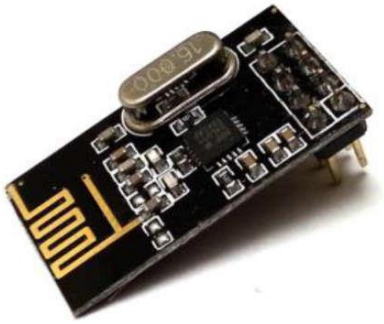


Figure 1.1 nRF24L01+ wireless module

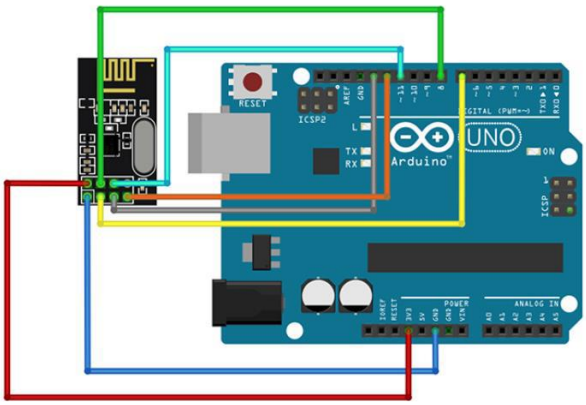


Figure 1.2 nRF24L01+wireless module with Arduino

Table1. nRF24L01 connection with Arduino board

nRF24L01	GND	CE	SCK	MISO	VCC	CSN	MOSI
Arduino	GND	7	13	12	3.3V	8	~11

(b.) HC-06 with Arduino

The HC-06 Bluetooth module was designed for short-range wireless data transfer (less than 100 meters). The UART interface was used in this module, it could be connected to practically any microcontroller or processor. This module used the Bluetooth 2.0 communication protocol to deliver files at a speed of up to 2.1Mbps. The frequency range was 2.402 GHz - 2.480 GHz, while the operating voltage was 3.3V - 6V. The connection was shown in Figure 1.3 and Table 2. AT commands were used to setup the module and were entered in the serial monitor.

The steps to connect to the cellphone were:

(1) Enter ‘AT’ to test the connection and the respond was ‘OK’.

(2) Enter ‘AT+NAME’ to set the name of the module

(3) Enter ‘AT+PIN’ to set the code for connection

(4) Enter ‘AT+BAUD4’ to modify the baud rate to 9600

(5) Finding the name of module in the app and entering the pin code to connect

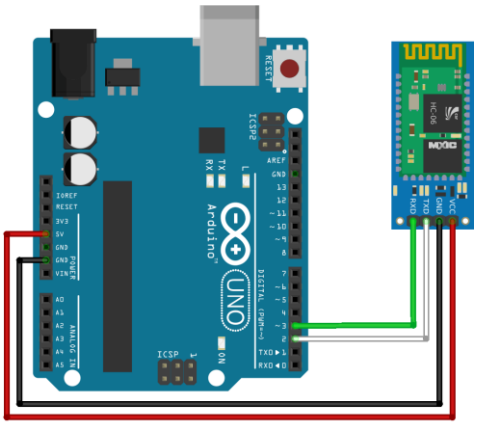


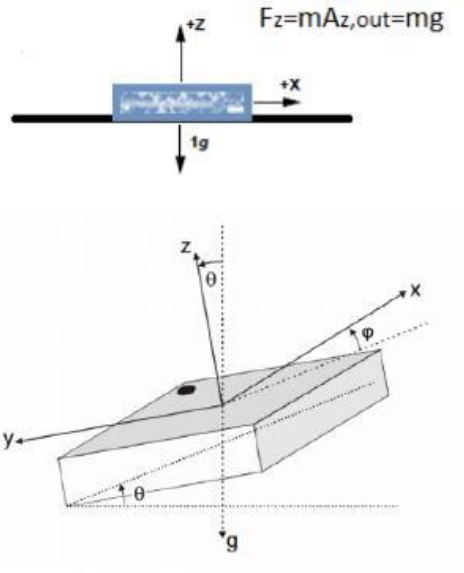
Figure 1.3 HC-06 with Arduino

Table 2 HC-06 connection with Arduino

HC-06	RXD	TXD	GND	VCC
Arduino	~3	2	GND	5V

2. Motion sensor MPU-6050 to measure inclination and other data

An accelerometer, as shown in Figure 1.4, was a device that measured acceleration, which was the rate of change of an object's velocity, measured in meters per second squared (ms2) or in G-forces (g). According to Newton's Second Law, when a physical body accelerates in one direction, it is subjected to a force equal to F=ma. The accelerometer's readings indicated that the acceleration was correct. Proper acceleration was defined as an object's physical acceleration in relative theory. It was thus inertial acceleration, or acceleration relative to a free-fall. Accelerometers could be utilized to detect tilt. The tilt angle of an accelerometer placed on a ramp could be computed using the equation presented in Figure 1.4.



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

Figure1.4 Working principle about accelerometer

Along the Roll, Pitch, and Yaw axes, the gyroscope sensor detected rotational velocity (angular velocity, degree/sec). Roll was the rotation around the X-axis, Pitch was the rotation around the Y-axis, and Yaw was the rotation around the Z-axis, as seen in Figure 1.5. Gyroscopes were used in inertial navigation systems, such as the Hubble Space Telescope.

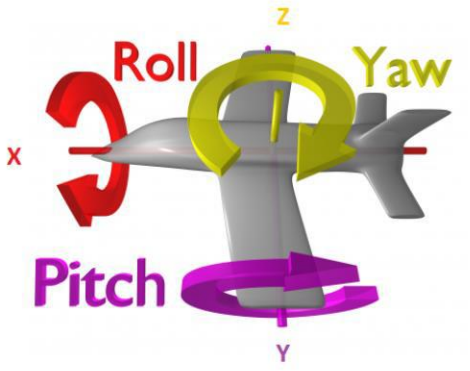


Figure 1.5 Working principle about gyroscope

The accelerometer and gyroscope of the MPU-6050 were investigated. The chip

in this module was MPU-6050, as indicated in Figures 1.6. With an on-board LDO regulator, this module's operational voltage ranged from 3 to 5 volts. I2C fast speed mode 400kHz was used for communication. It has a 16bit AD converter and 16bit data output built in. It came with four distinct gyroscope ranges: +/- 250, 500, 1000, and 2000 degrees per second. The acceleration ranges were as follows: +/- 2g, +/-4g, +/-8g, and +/-16g. The MPU6050 module also included an on-chip 1024-byte FIFO buffer as well as a DMP engine for data fusion and sensor synchronization. The construction of MPU-6050 with Arduino was shown in Figure 1.7 and the connection was shown in Table 3. The tilt was measured using the accelerometer output from the MPU6050. The equation was shown in Figure 1.4. The device and the information it gave were utilized to detect a ramp on a straight path.

Figure 1.6 MPU-6050

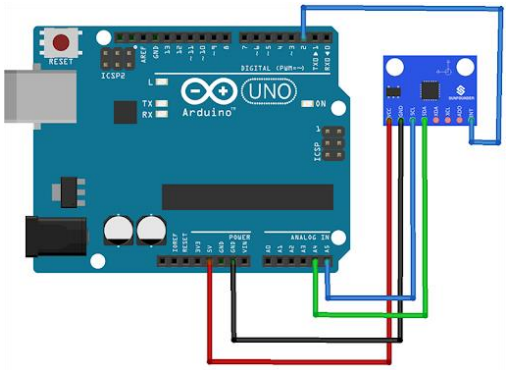


Figure 1.7 MPU-6050 with Arduino

Table 3. MPU-6050 connection with Arduino

MPU-6050	VCC	GND	SCL	SDA	INT
Arduino	5V	GND	A5	A4	2

3. Logic gate and PWM

The aim of this topic was to design and solder a logic board (74 series IC based, one channel) to replace the baseboard CPLD. The requirements of the tasks were in the upper left corner of Figure 1.8. The table was transferred into the truth table in lower left corner of Figure 1.8. Next, the truth table was transferred to Karnaugh maps in the lower right corner of Figure 1.8. Then, Karnaugh maps were transferred into the Mathematical expression. Therefore, the circuit of the logic gates was in Figure 1.9.

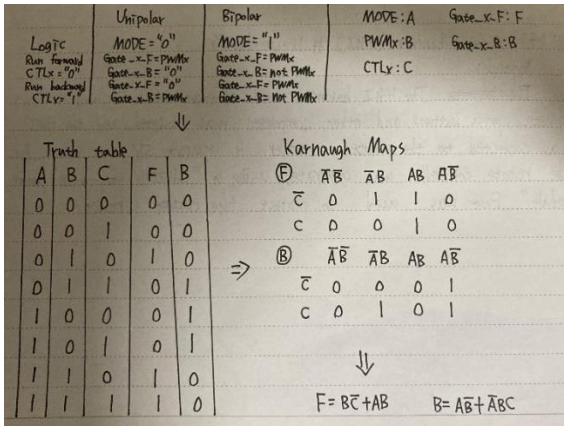


Figure 1.8 Truth table, Karnaugh map and table on Moodle

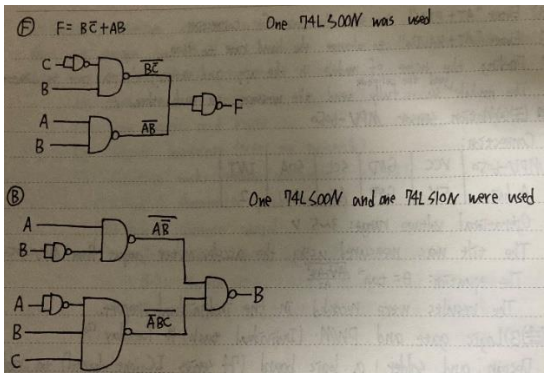


Figure 1.9 The schematic of the logic gate

4. System hardware (HMI e.g., joystick)

The joystick was shown in Figure 1.10. The HMI hardware and software, which included an LCD, a joystick, press buttons, and other components, was designed, and the HMI was connected to the remote controller. The remote HMI's soldering work was completed. A separate 5V power source for the remote controller was generated using a "LM2596 step-down power

module." The circuit was also protected by a fuse.

Figure 1.10 Joystick

5. Challenge tasks (remote / automatic control)

1) Task A: Automatic tracking 1 cycle (2 times with different ramp place)

   The vehicle had to be equipped with an MPU-6050 and motor encoder, and the vehicle status had to be displayed on an LCD (run/stop, inclination, distance from start) all the time. The vehicle had to pass through a simple path leaved to a ramp on any segment, then wait for 2 seconds at the top of the ramp before rotating 360 degrees and continuing running. The challenge had to be implemented twice, each time with a different ramp location.

2) Task B1: Remote control with nRF24L01 + joystick

   The vehicle in this task was controlled by nRF24L01 and the remote handle with the joystick to finish the track. The vehicle status (run/stop, speed, distance from start) on the LCD was optional to get the full mark.

3) Task B2: Remote control with nRF24L01 and MPU-6050

   The vehicle in this task was controlled by the remote handle with nRF24L01 and MPU-6050 to finish the track.

4) Task B-3: Remote control with HC-06 (with cellphone)

   The vehicle in this track was controlled by the cellphone and HC-06

was used to finish the track.

Only one of Tasks B2 and B3 had to be completed.

Session 4:

1. The basic knowledge of optical sensors

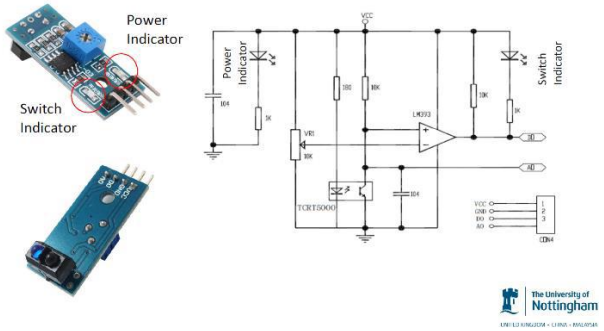


Figure 1.11 TCRT5000 sensor

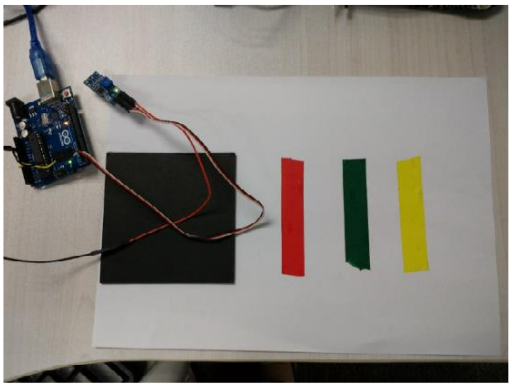


Figure 1.12. The schematic of the measurement

The TCRT5000 sensor module and its schematic were shown in Figure 1.11. The output voltages (at A0 and D0) with a particular distance and darkness/color of the reflecting surface were recorded, as shown in Figure 1.12.

2. Basic line following (two sensors)

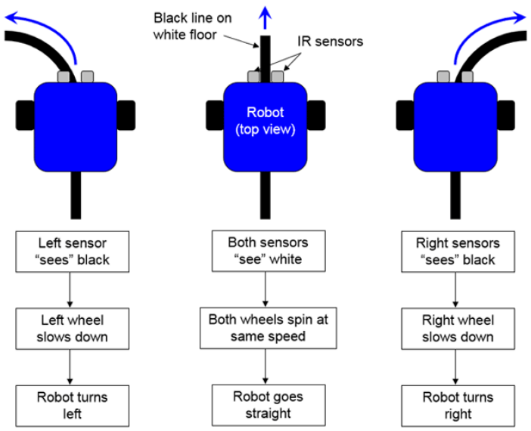


Figure 1.13 Bang-bang control

The line following with Bang-bang control codes were created, and basic line following with two of these optical sensors was implemented. The basic concept of Bang-bang control was depicted in Figure 1.13.

3. Advanced line following with PID (Proportional-Integral-Derivative) control and 8 sensors.

The SF-8CHIDTS and the I2C serial interface were investigated. For sophisticated line following, the Sunfounder Line Follower Module SF-8CHIDTS was employed, as illustrated in Figure 1.14. The I2C serial interface was used to link the module to the Arduino. The amount of reflected light seen by each sensor was represented by the output data. The sensor module handled the conversion of analog voltage to digital value, saving Arduino time and effort.

Figure 1.14 Sunfounder Line Follower Module SF-8CHIDTS

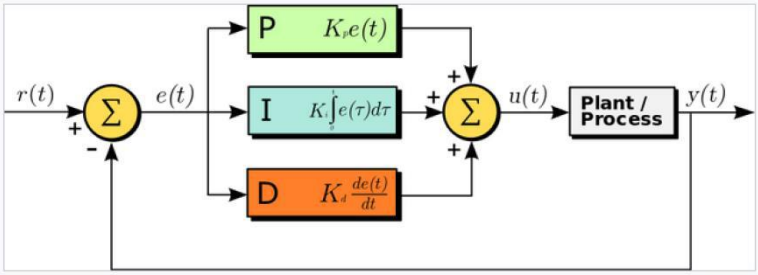
The use of the weighted average technique was comprehended. Based on the provided need, a weighted average algorithm and calibration technique for

the sensor were developed. Each sensor was calibrated to yield a minimum and maximum value, such as 0 for white and 1000 for black. And each sensor value was given a weighting, such as the distance from a reference point like the vehicle's center point. The equation for the distance from center point was shown in Figure 1.15.

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

Figure 1.15 Equation for weighted average calculation

The magnitude of the error (e.g., distance from the reference point) was accurately known, and therefore a control scheme could be implemented known as Proportional Integral Derivative (PID). The basic principle of PID control was depicted in Figure 1.16.



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).

Figure 1.16 Proportional Integral Derivative (PID)

Three steps for PID control:

(1) An error value e(t) was determined as the difference between a planned setpoint SP=r(t) and a continuously calculated measured process variable PV=y(t): e(t)=SP-PV.

(2) To reduce the error over time, a correction was made to a control variable u(t) based on proportional, integral, and derivative terms, as illustrated in Figure 1.17.

(3) With accurate and optimal control, the final measured process variable

was stable and with minimal error.

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,d\tau + K_{\mathrm{d}} \frac{de(t)}{dt}$$

K_p,K_i, K_d are non-negative the coefficients for the proportional, integral, and derivative terms

Figure 1.17 The equation for PID

The PID was separated into three parts. The current value of the SP-PV error e was proportional to term P. (t). When proportional control is used alone, an error between the setpoint and the actual process value will always occur. Term I accounted for previous SP-PV error e(t) values and integrated them across time to generate the I term. The integral term attempted to reduce residual error by adding a control effect based on the error's historical cumulative value. Term D was the best prediction of the SP-PV error e (t) trend based on its current rate of change. By exerting a control impact generated by the rate of error change, the derivative term attempted to reduce the effect of the SP-PV error e(t).

The PID tuning approach followed these:

(1) First set Kp, Ki and Kd to zero.

(2) Started tuning Kp only to try and get the robot to follow the line (this was a P controller).

(3) Now tuned Kd to tune the robot response to the rate of change of error (this was a PD controller).

(4) Then tuned Ki to improve performance as much as possible (this was now a PID controller).

If it all went wrong, started again.

4. Challenge task of line following (HMI, parameter tuning, competition)

In this session, two challenges were needed to complete.

1) Basic line following:

   The vehicle with digital output of TCRT5000 module was needed to run two cycles continuously, and the vehicle with analog output of TCRT5000 module + PID control to finish the task could get full mark.

2) Advanced line following:

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

- **Results and Discussions**

Session 3:

1. Wireless (2.4GHz & Bluetooth) communication

(a.) nRF24L01+wireless module

```
#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(83);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_2MBPS);
  rf24.openReadingPipe(pipe, addr);
  rf24.startListening();
  Serial.println("nRF24L01 ready!");
}

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

Figure 2.1 The codes for the receiver

```
#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(83);
  rf24.openWritingPipe(addr);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_2MBPS);
  rf24.stopListening();
}

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

Figure 2.2 The codes for the transmitter

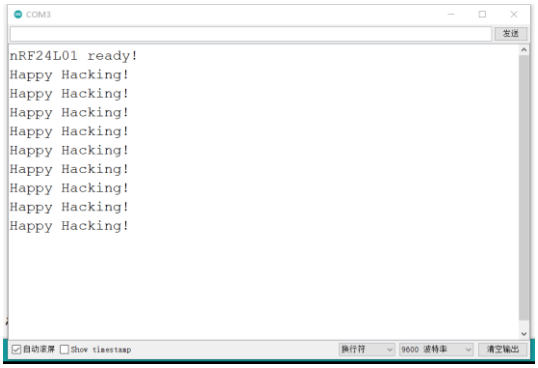


Figure 2.3 The output on the serial monitor

In Figure 2.1, the receiver delivered the message to the transmitter, and in Figure 2.2, the transmitter sent the other message to the receiver, and in Figure 2.3, the receiver sent the same message to the transmitter. Therefore, the bidirectional wireless communication was realized.

(b.) HC-06 with Arduino

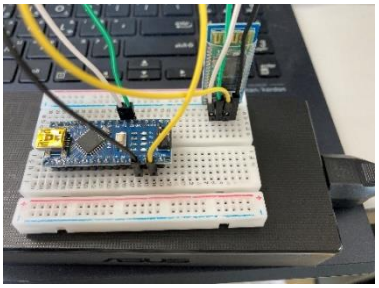


Figure 2.4 The connection diagram

Figure 2.5 The output on the serial monitor

The steps in the part of Method to connect to the cellphone was followed, and the communication between HC-06 and the cellphone was realized. The message which was entered on the serial monitor was displayed on the cellphone, and the message which was entered on the cellphone was displayed on the serial monitor.

2. Motion sensor MPU-6050 to measure inclination and other data

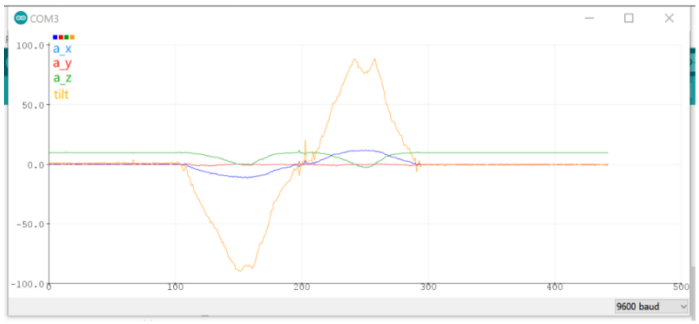


Figure 2.6 Verification with Serial Plotter

The design was verified with Serial Plotter as shown in Figure 2.6. Because there was no transverse rotation of the vehicle when the tilt angle was detected, the y-axis reaction was close to y=0. As a result, the acceleration of the y-axis remained 0. The acceleration of the x-axis increased as the absolute value of the tilt angle increased, while the acceleration of the z-axis decreased. As a result, there was a

positive nonlinear correlation between acceleration of the x-axis and tilt, and a negative nonlinear correlation between acceleration of the z-axis and tilt.

3. Logic gate and PWM

The result was comparable with the truth table in Figure 1.8 when using the connection method in Figure 1.8 and measuring using a power supply and a multimeter.

4. System hardware (HMI e.g., joystick)

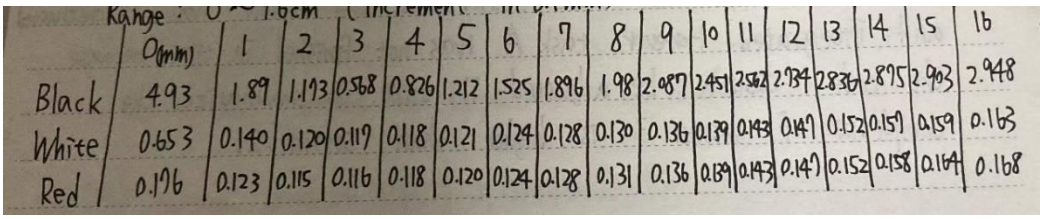
The narrow input range was revealed during testing for the joystick, which was a design flaw. The values obtained from the built-in encoder were all the maximum value (e.g. 1023) if the joystick was toggled to be near the four edges, making the joystick too sensitive to drive the vehicle easily.

5. Challenge tasks (remote / automatic control)

Task A: This task was not finished because it took too long to debug the code. For the Task B, only one of Tasks B2 and B3 had to be finished. Debugging both hardware and software was the most critical and hardest component. As a result, the lesson from the previous lab session that was applied to this lab session was that the hardware was debugged first to ensure that it worked properly before moving on to the software debugging. Thus, Task B1 and B3 were successfully completed.

Session 4:

1. The basic knowledge of optical sensors



Range : 0 ~ 1.6cm (increment)	0(min)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Black	4.93	1.89	1.193	0.568	0.826	1.212	1.525	1.896	1.98	2.087	2.451	2.562	2.734	2.836	2.895	2.903	2.948
White	0.653	0.140	0.120	0.119	0.118	0.121	0.124	0.128	0.130	0.136	0.139	0.143	0.149	0.152	0.159	0.159	0.163
Red	0.196	0.123	0.115	0.116	0.118	0.120	0.124	0.128	0.131	0.136	0.139	0.143	0.149	0.152	0.158	0.164	0.168

Figure 2.7 The output voltage of TCRT5000

The output voltages (at A0 and D0) with a certain distance and darkness/color of the reflection surface were recorded. The data obtained from a multimeter was shown in Figure2.7. In this experiment, TCRT5000 modules were used to obtain data via the multimeter. The sensors were sensitive to black but not sensitive to other colors. This was due to the fact that the sensor was sensing reflected light, which was related to not only the colors but also the materials and other conceivable circumstances. It was also discovered that the sensor was more sensitive near the colored lines than on the vehicle, implying that the sensor was less sensitive the further away it was from the colored lines. As a result, it was utilized to identify black lines, although it was nearly difficult to distinguish them from other colors.

2. Basic line following (two sensors)

In Figure 2.8, combining the principle of TCRT5000 and bang-bang control to design the codes for basic line following. The TCRT5000 is a reflecting sensor with an infrared emitter and phototransistor housed in a lead-free packaging that blocks visible light.

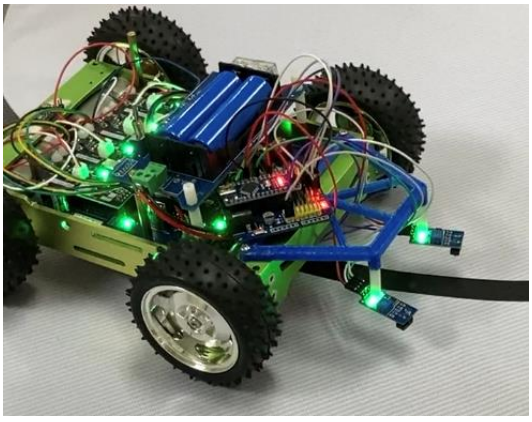


Figure 2.8 The vehicle with two TCRT5000 sensors

3. Advanced line following with PID (Proportional-Integral-Derivative) control and 8 sensors

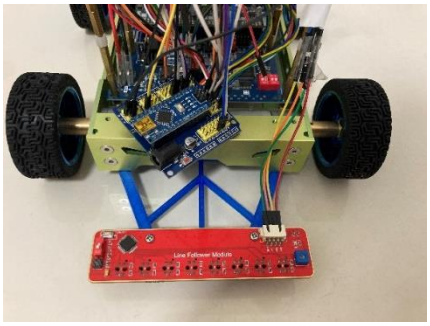


Figure 2.9 The vehicle with SF-8CHIDTS

To prevent vehicle shaking, the algorithm used PID control, which allowed the vehicle to follow the line smoothly and precisely without hitting the obstacles. The most challenging component of the PID control was finding the three variables, which needed a lot of trial and error. The vehicle was unable to run the track in a straight line when the parameters were originally checked because they were all set at the same time. Further investigation found that the Ki and Kd were solely used for line following perfection, with the Kp being the only parameter needed for line following.

4. Challenge task of line following (HMI, parameter tuning, competition)

Switches on the vehicle were required to modify PID variables. Because the Arduino was unable to remember any of the PID variables every time it was rebooted, requiring the PID to be set each time, which was time consuming. Using unidirectional wireless communication, the LCD was utilized to display the vehicle's status, several PID variables, and PID error remotely. This section was not completed because debugging the code took too long.

Conclusions

Session 3 of the experiment covered motion detection with the MPU-6050 and wireless communication with the nRF24L01+ module, as well as joysticks and the HC-06 for remote control. Basic understanding of MPU-6050, HC-06, and bidirectional wireless communication were learned in the first and second topics,

and the vehicle was reconstructed and integrated for the final tasks. Individual tasks in the third topic included designing and soldering a logic board (based on 74 series ICs, one channel) to replace the baseboard CPLD. To complete the last challenge, an extra remote controller was combined with HMI and a joystick to realize remote controlled line following. Session 4 covered a variety of techniques for implementing automatic line following. The vehicle was first augmented with basic line following using two optical TCRT5000 sensor modules and the bang-bang control algorithm during this session. To achieve rapid and accurate line following, the advanced line following was designed using the Sunfounder line follower module SF-8CHIDTS and the PID control algorithm. Individual tasks included recording output voltages (at A0) from black, white and red surfaces at various distances.

To summarize, these two lab sessions performed better than the second. Only task A was not completed in session 3 due to the time spent integrating and debugging the hardware. Only the work of basic line following was achieved in session 4 since the adaptation code was too time-consuming. To improve, time management and adequate task preparation were essential.

References

[1] H61AEE: Project Session III Introduction. Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618736/mod_resource/content/32/Session_3_Introduction_2122_V1.pdf [Accessed: 20 December 2021]

[2] H61AEE: Project Session IV Introduction. Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 2019 [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618778/mod_resource/content/22/Session_4_Introduction_2122_V1.pdf [Accessed: 21 December 2021]

- **Appendices**

Session 3:

(1) Wireless (2.4GHz & Bluetooth) communication

(a.) nRF24L01+wirless module

Receiver:

```
#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(83);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_2MBPS);
  rf24.openReadingPipe(pipe, addr);
  rf24.startListening();
  Serial.println("nRF24L01 ready!");
}

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

```
}
```

Transmitter:

```
#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(83);
  rf24.openWritingPipe(addr);
  rf24.setPALevel(RF24_PA_MAX);
  rf24.setDataRate(RF24_2MBPS);
  rf24.stopListening();
}

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

(b.) HC-06

```
#include <SoftwareSerial.h>
SoftwareSerial hc06(2,3); //RX, TX
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());
}
}
```

(2) MPU-6050

```
//#include <helper_3dmath.h>
//#include <MPU6050.h>
//#include <MPU6050_6Axis_MotionApps20.h>
//#include <MPU6050_9Axis_MotionApps41.h>
//verification 1
#include<Wire.h>
#include<Math.h>

const int MPU_addr = 0x68;//I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
double a_x = 0, a_y = 0, a_z = 0;
double d_xz = 0,d_xy = 0,d_yz = 0;

void setup(){
Wire.begin ();
Wire.beginTransmission (MPU_addr);
```

```
Wire.write(0x6B);//PWR MGMT 1 register
Wire.write(0);//set to zero (wakes up the MPU-6050)
Wire.endTransmission(true);
Serial.begin (9600);
}

void loop(){
Wire.beginTransmission (MPU_addr);
Wire.write(0x3B);//starting with register 0x3B(ACCEL XOUT H)
Wire.endTransmission (false);
Wire.requestFrom(MPU_addr,14,true);//request a total of 14reaisters
AcX = Wire.read()<< 8 | Wire.read();//0x3B (ACCEL XOUT H)&
0x3C(ACCEL XoUT L)
AcY = Wire.read()<< 8 | Wire.read();//0x3D(ACCEL YOUT H)&
0x3E(ACCEL YOUT L)
AcZ = Wire.read()<< 8 | Wire.read();//0x3F(ACCEL ZOUT H)& 0x40
(ACCEL ZOUT L)
//Tmp=Wire.read()<<8|Wire.read();//0x41(TEMP OUT H)& 0x42(TEMP
OUT L)
//GyX=Wire.read()<<8|Wire.read();//0x43(GYRO XoUT H)& 0x44
(GYRO XOUT L)
//GyY=Wire.read()<<8|Wire.read();//0x45(GYRO YoUT H) & 0x46(GYRO
YoUT L)
 //GyZ=Wire.read()<<8|Wire.read();//0x47 (GYRO ZOUT H)& 0x48
(GYRO ZOUT L)

 //rough calibration
 a_x =(AcX + 300)* 9.8 / 14368;
 a_y =(AcY - 104)* 9.8 / 14368;
 a_z =(AcZ)* 9.8 /14368;
```

```
 Serial.print("a_x = ");    Serial.print(a_x);
 Serial.print("| a_y = "); Serial.print(a_y);
 Serial.print("| a_z =");  Serial.print(a_z);

 //degrees
 int flag = 0;
 if (a_x> 0)flag = 1;
 else flag = -1;
 d_xz = acos(a_z/ 9.8)/ 3.14 * 180 * flag;
 Serial.print("| degree ="); Serial.println(d_xz);
 delay(333);
}
```

(3) Task B1

```
#include <SPI.h>
#include <Wire.h>
#include "RF24.h"
RF24 rf24(6,9);

int a,b,c;
const byte addr[] = "1Node";
const byte pipe =1;

void setup() {
Serial.begin(9600);
rf24.begin();
rf24.setChannel(101);
rf24.setPALevel(RF24_PA_MAX);
rf24.setDataRate(RF24_2MBPS);
rf24.openReadingPipe(pipe, addr);
rf24.startListening();
```

```
Serial.println("ready!");
}

void loop() {

   Wire.beginTransmission(42);
   Wire.write("ha");
   Wire.endTransmission();


rf24.read(&b,sizeof(b));
rf24.read(&c,sizeof(c));


Serial.print("b:");
Serial.println(b);
Serial.print("c:");
Serial.println(c);

if(c<50){
  Serial.println("Run forward");
  Wire.beginTransmission(42);
  Wire.write("b1f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b2f");
    Wire.write(60);
```

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

  Wire.beginTransmission(42);
  Wire.write("b3f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b4f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();
}



else if(c>1020){
  Serial.println("Run backward");

  Wire.beginTransmission(42);
  Wire.write("b1r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b2r");
    Wire.write(60);
```

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

  Wire.beginTransmission(42);
  Wire.write("b3r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b4r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();
 }
 else if(b<50){
  Serial.println("left");

  Wire.beginTransmission(42);
  Wire.write("b1f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b2r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();
```

```
  Wire.beginTransmission(42);
  Wire.write("b3f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b4r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();
 }
 else if(b>1020){
  Serial.println("right");

  Wire.beginTransmission(42);
  Wire.write("b1r");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b2f");
    Wire.write(60);
    Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b3r");
    Wire.write(60);
```

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

  Wire.beginTransmission(42);
  Wire.write("b4f");
   Wire.write(60);
   Wire.write(0);
  Wire.endTransmission();
}
delay(150);




}

Transmit:
#include <SPI.h>
#include <Wire.h>
#include "RF24.h"
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <LiquidCrystal.h>
#include <Wire.h>
#define LINE_TARGET  (900)
#define TURNING_TARGET  (300)
RF24 rf24(6,9); // CE, CSN
float Speed;
float Dis=0;
```

```
const byte addr[] = "1Node";
const int rs = 8, en = 5, d4 = 1, d5 = 2, d6 = 3, d7 = 4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
Serial.begin (9600);
rf24.begin();
rf24.setChannel(101);
rf24.openWritingPipe(addr);
rf24.setPALevel(RF24_PA_MIN);
rf24.setDataRate(RF24_2MBPS);
rf24.stopListening();
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("Distance:");
  lcd.print(Dis);
  lcd.setCursor(0,1);
  lcd.print("Speed:");
  lcd.print(Speed);
}

void loop () {

  int x,y,buttonVal;
  int Speed;
  x = analogRead (A0);
  y = analogRead (A1);
  Serial.print("X = ");
  Serial.println (x, DEC);
  rf24.write(&x, sizeof(x));
```

```
 Serial.print ("Y = ");
 Serial.println (y, DEC);
 rf24.write(&y, sizeof(y));


 lcd.setCursor(6,1);
 lcd.print("0m/s");
 lcd.setCursor(11,1);
 lcd.print("stop!");

if(x>=1020)
{


 lcd.setCursor(9,0);
 lcd.print(Dis);
 lcd.setCursor(6,1);
 lcd.print("2m/s");
 lcd.setCursor(11,1);
 lcd.print("Back!");


 }

 else if(x<=6)
{

 Dis = Dis + 0.12;
 lcd.setCursor(9,0);
 lcd.print(Dis);
```

```
  lcd.setCursor(6,1);
  lcd.print("1m/s");
  lcd.setCursor(11,1);
  lcd.print("Run!!");
 }
else if((y>=1020))
{

  lcd.setCursor(6,1);
  lcd.print("1m/s");
  lcd.setCursor(11,1);
  lcd.print("Left!");
 }
else if((y<=6))
{
  lcd.setCursor(6,1);
  lcd.print("1m/s");
  lcd.setCursor(11,1);
  lcd.print("Right");
}


 delay(100);
}
```

(4) Task B3

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


SoftwareSerial CAL(2,3);
```

```
char val;
int i;
void setup() {
Wire.begin();
Serial.begin(9600);
CAL.begin(9600);
 }


void Run(int a) //speed setting
{
Wire.beginTransmission(42);
Wire.write("sa");
for (i=0;i<=3;)
{
Wire.write(a);
Wire.write(0);
i++;
 }
 Wire.endTransmission();
 }


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

```
void loop()
 {
if (Serial.available()) //set AT command
{
 val = Serial.read();
 CAL.print(val);
 }
 if (CAL.available())
{
 val = CAL.read();
 switch(val) //switch the command from smatphone
 {
 case 'w': //moving forward
{
 Wire.beginTransmission(42);
 Wire.write("daffff");
 Wire.endTransmission();
 Run(50);

 break;
 }
 case 's': //moving backward
 {
 Wire.beginTransmission(42);
 Wire.write("darrrr");
 Wire.endTransmission();
 Run(40);
 delay(500);
 Motorstop();
```

```
 break;
 }
case 'a': //turn left
 {
 Wire.beginTransmission(42);
 Wire.write("darrff");
 Wire.endTransmission();
 Run(50);
 delay(300);
 Motorstop();

 break;
 }
 case 'd': //turn right
 {
 Wire.beginTransmission(42);
 Wire.write("daffrr");
 Wire.endTransmission();
 Run(50);
 delay(300);
 Motorstop();
 break;
 }
 case 't': //stop motor
 {
 Motorstop();
 break;
 }

 }
```

```
}
}
```

Session 4:

(1) Basic line following with two sensors

```
#include "Wire.h"
const int pinLED_RIGHT = 8;
const int pinLED_LEFT = 9;
const int pin_right_d = 7;
const int pin_right_a = A2;
const int pin_Left_d = 10;
const int pin_Left_a = A3;

int IRvalueA1_RIGHT = 0;
int IRvalueD1_RIGHT = 0;
int IRvalueA1_LEFT = 0;
int IRvalueD1_LEFT = 0;



void setup()
{
  Serial.begin(9600);
  pinMode(pin_right_d,INPUT);
  pinMode(pin_right_a,INPUT);
  pinMode(pin_Left_d,INPUT);
  pinMode(pin_Left_a,INPUT);
  pinMode(pinLED_RIGHT,OUTPUT);
  pinMode(pinLED_LEFT,OUTPUT);
  Wire.begin();
}
```

```
void loop()
{
  Serial.print("\t Digital Reading(RIGHT)=");
  Serial.println(IRvalueD1_RIGHT);
  Serial.print("\t Digital Reading(LEFT)=");
  Serial.println(IRvalueD1_LEFT);


  IRvalueA1_RIGHT = analogRead(pin_right_a);
  IRvalueD1_RIGHT = digitalRead(pin_right_d);
  IRvalueA1_LEFT = analogRead(pin_Left_a);
  IRvalueD1_LEFT = digitalRead(pin_Left_d);


  if((IRvalueD1_RIGHT == 0)&&(IRvalueD1_LEFT == 0))
  //RUN FORWARD
  {
  Wire.beginTransmission(42);
  Wire.write("b1f");
  Wire.write(22);
  Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(42);
  Wire.write("b2f");
  Wire.write(22);
  Wire.write(0);
  Wire.endTransmission();
```

```
Wire.beginTransmission(42);
Wire.write("b3f");
Wire.write(22);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
Wire.write("b4f");
Wire.write(22);
Wire.write(0);
Wire.endTransmission();
}


else if((IRvalueD1_RIGHT == 0)&&(IRvalueD1_LEFT == 1))
//TURN LEFT
{
Wire.beginTransmission(42);
Wire.write("b1r");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
Wire.write("b2f");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
```

```
Wire.write("b3r");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
Wire.write("b4f");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();
}

else if((IRvalueD1_RIGHT == 1)&&(IRvalueD1_LEFT == 0))
//TURN RIGHT
{
Wire.beginTransmission(42);
Wire.write("b1f");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
Wire.write("b2r");
Wire.write(85);
Wire.write(0);
Wire.endTransmission();

Wire.beginTransmission(42);
Wire.write("b3f");
Wire.write(85);
```

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

  Wire.beginTransmission(42);
  Wire.write("b4r");
  Wire.write(85);
  Wire.write(0);
  Wire.endTransmission();
  }
   else if((IRvalueD1_RIGHT == 1)&&(IRvalueD1_LEFT == 1))
  //STOP
  {
    Wire.beginTransmission(42);
    Wire.write("ha");
    Wire.endTransmission();
  }
}
```

(2) TCRT5000 optical sensor

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

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

```
}

void loop()
{
  Serial.print("Analog Reading=");
  Serial.print(IRvalueA);
  Serial.print("\t Digital Reading=");
  Serial.println(IRvalueD);
  delay(1000);

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

(3) Advance line following

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

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

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(500);
}

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
  }
}
```