**Technocrats Task Report for Week 3 and Week 4**

**CYTRON LINE FOLLOWING SENSOR(LSA08):**

LSA08 is an advanced line following sensor. It has 8 pairs (transmitter and receiver)

of sensor. The transmitter LED transmits a light of specific wavelength of green colour. It can operate in coloured surfaces also. The system has different output modes(digital output line(8 parallel lines), UART output mode and analog output port) based on the requirement and also on the mode in which it is operated. It can be broken down into Digital mode, Serial mode with Digital output and analog mode.

Digital Mode:

This just gives you the output as logic level high or low.

The arduino UNO and LSA08 connection can be made as follows:

Vin – Vin

GND – GND

O0 – pin 0

O1 – pin 1

O2 – pin2

and so on, until…

O7 – pin 7

Vin is 12V

There are two buttons on the right side of the sensor. The bottom one is for switching the options between the menu. The top one is the 'Select button (SEL)'

The LINEMODE setting helps us to inform whether the dark line or the bright line is going to be detected. That is to give high when in dark line or the bright line.

The "CALB" mode on the setting is for auto calibration. After selecting this mode move the sensor across the line(dark/white for which it is going to be used) rapidly. Once the calibrating is done automatically the sensor starts detecting the line. The sensor which is HIGH is indicated on the LCD display.

Code:

const byte dir1 = 13; // Connect DIR1 of motor driver to pin 13

const byte dir2 = 12; // Connect DIR2 of motor driver to pin 12

const byte pwm1 = 11; // Connect PWM1 of motor driver to pin 11

const byte pwm2 = 10; // Connect PWM2 of motor driver to pin 10

void setup() {

// Setting pin 0 - 7 as digital input pin

for(byte i=0;i<=7;i++) {

pinMode(i,INPUT);

}

// Setting pin 10 - 13 as digital output pin

for(byte i=10;i<=13;i++) {

pinMode(i,OUTPUT);

}

// Setting the initial condition of motors

// make sure both PWM pins are LOW

digitalWrite(pwm1,LOW);

digitalWrite(pwm2,LOW);

// State of DIR pins are depending on your physical connection

// if your robot behaves strangely, try changing thses two values

digitalWrite(dir1,LOW);

digitalWrite(dir2,LOW);

}

void loop() {

// Checking for junction crossing, if juction detected,

// keep moving forward

if(digitalRead(2) && digitalRead(5))

moveForward();

// Checking for sensor number 1 and 2, if line detected, move left

else if(digitalRead(1) || digitalRead(2))

moveLeft();

// Checking for sensor number 5 and 6, if line detected, move right

else if(digitalRead(5) || digitalRead(6))

moveRight();

// Checking for sensors number 3 and 4,

// if line is detected by either of these sensor, move forward

else if(digitalRead(3) || digitalRead(4))

moveForward();

// If no line is detected, stay at the position

else

wait();

// Put some delay to avoid the robot jig while making a turn

delay(100);

}

// The values work good in my case, you could use other values set

// to archieve a performance that satisfy you

void moveLeft() {

// For robot to move left, right motor has to be faster than left motor

analogWrite(pwm1,90);

analogWrite(pwm2,10);

}

void moveRight() {

// For robot to move right, left motor has to be faster than right motor

analogWrite(pwm1,10);

analogWrite(pwm2,90);

}

void moveForward() {

// For robot to move forward, both motors have to be same speed

analogWrite(pwm1,70);

analogWrite(pwm2,70);

}

void wait() {

// Function to makes the robot stay

analogWrite(pwm1,0);

analogWrite(pwm2,0);

}

Serial Mode:

LSA08 pin Arduino Uno board

1 RX (pin 0)

2 TX (pin 1)

3 digital pin 2

5 digital pin 4

9 Vin

10 GND

The individual bars indicates the value of each sensor. The number indicates the current line position. Switching between the modes and using the select button are the same as in digital mode. J appears when there is a junction.

It has several modes.

LCD contrast: To adjust the contrast of LCD.

CALB: This mode is for enabling the auto-calibration of the sensor.

LINEMODE: Helps you to fix the type of line that you are going to trace. Either the dark one or the white line.

THRES: Helps the user to fix the number of sensors has to be triggered to sense a line.

JWIDTH: This mode is to define the juunction.

BAUDRATE: Set the baudrate in srial communication.

UART mode: Set the type of output.

0 no data output

1 8-bits digital output

2 1 byte analog output

3 8 continuous bytes of analog output package

LCD B/L: For enabling the backlight of the LCD screen.

const byte rx = 0; // Defining pin 0 as Rx

const byte tx = 1; // Defining pin 1 as Tx

const byte serialEn = 2; // Connect UART output enable of LSA08 to pin 2

const byte junctionPulse = 4; // Connect JPULSE of LSA08 to pin 4

const byte dir1 = 13; // Connect DIR1 of motor driver to pin 13

const byte dir2 = 12; // Connect DIR2 of motor driver to pin 12

const byte pwm1 = 11; // Connect PWM1 of motor driver to pin 11

const byte pwm2 = 10; // Connect PWM2 of motor driver to pin 10

unsigned int junctionCount = 0; // Variable to store junction count value

void setup() {

pinMode(serialEn,OUTPUT); // Setting serialEn as digital output pin

pinMode(junctionPulse,INPUT); // Setting junctionPulse as digital input pin

// Setting pin 10 - 13 as digital output pin

for(byte i=10;i<=13;i++) {

pinMode(i,OUTPUT);

}

// Setting initial condition of serialEn pin to HIGH

digitalWrite(serialEn,HIGH);

// Setting the initial condition of motors

// make sure both PWM pins are LOW

digitalWrite(pwm1,LOW);

digitalWrite(pwm2,LOW);

// State of DIR pins are depending on your physical connection

// if your robot behaves strangely, try changing thses two values

digitalWrite(dir1,LOW);

digitalWrite(dir2,LOW);

// Begin serial communication with baudrate 9600

Serial.begin(9600);

// Clear internal junction counter of LSA08

clearJunction();

}

void loop() {

byte dummy = 0; // Declare a dummy variable to store incoming data

// Checking for junction crossing, if juction detected,

// keep moving forward

if(digitalRead(junctionPulse)) {

while(digitalRead(junctionPulse)) {

moveForward();

}

// Retrieve the junction count from LSA08

// You can do whatever you want with the junction count

junctionCount = getJunction();

}

digitalWrite(serialEn,LOW); // Set serialEN to LOW to request UART data

while(Serial.available() <= 0); // Wait for data to be available

dummy = Serial.read(); // Read incoming data and store in dummy

digitalWrite(serialEn,HIGH); // Stop requesting for UART data

// Checking for sensor number 1 and 2, if line detected, move left

if(bitRead(dummy,1) || bitRead(dummy,2))

moveLeft();

// Checking for sensor number 5 and 6, if line detected, move right

else if(bitRead(dummy,5) || bitRead(dummy,6))

moveRight();

// Checking for sensors number 3 and 4,

// if line is detected by either of these sensor, move forward

else if(bitRead(dummy,3)||bitRead(dummy,4))

moveForward();

// If no line is detected, stay at the position

else

wait();

// Put some delay to avoid the robot jig while making a turn

delay(100);

}

// Function to clear internal junction counter of LSA08

void clearJunction() {

char address = 0x01;

char command = 'X';

char data = 0x00;

char checksum = address + command + data;

Serial.write(address);

Serial.write(command);

Serial.write(data);

Serial.write(checksum);

}

// Function to retrieve junction count from LSA08

int getJunction() {

char address = 0x01;

char command = 'X';

char data = 0x01;

char checksum = address + command + data;

Serial.write(address);

Serial.write(command);

Serial.write(data);

Serial.write(checksum);

while(Serial.available() <= 0);

return (int(Serial.read()));

}

// The values work good in my case, you could use other values set

// to archieve a performance that satisfy you

void moveLeft() {

// For robot to move left, right motor has to be faster than left motor

analogWrite(pwm1,90);

analogWrite(pwm2,10);

}

void moveRight() {

// For robot to move right, left motor has to be faster than right motor

analogWrite(pwm1,10);

analogWrite(pwm2,90);

}

void moveForward() {

// For robot to move forward, both motors have to be same speed

analogWrite(pwm1,70);

analogWrite(pwm2,70);

}

void wait() {

// Function to makes the robot stay

analogWrite(pwm1,0);

analogWrite(pwm2,0);

}

Analog mode:

In this analog mode the sensor return a value ranging from 0-255. 0-70 when there is a dark line and 255 when white space is detected. And this value is converted to a nominal voltage.ranging from 0-4.5V. This type of values are obtained from ADC(Analog to Digital Converters).

LSA08 in analog mode is interfaced with a 10 bit ADC convertor which shows its resolution is 5/2^10.

0 – 18, the line at left

19 – 52, the line at center

53 – 70, the line at right

>70 absence of line

const byte analogPin = 0; // Connect AN output of LSA08 to analog pin 0

const byte junctionPulse = 4; // Connect JPULSE of LSA08 to pin 4

const byte dir1 = 13; // Connect DIR1 of motor driver to pin 13

const byte dir2 = 12; // Connect DIR2 of motor driver to pin 12

const byte pwm1 = 11; // Connect PWM1 of motor driver to pin 11

const byte pwm2 = 10; // Connect PWM2 of motor driver to pin 10

int readVal,positionVal; // Variables to store analog and line position value

unsigned int junctionCount = 0; // Variable to store junction count value

void setup() {

pinMode(junctionPulse,INPUT);

// Setting pin 10 - 13 as digital output pin

for(byte i=10;i<=13;i++) {

pinMode(i,OUTPUT);

}

// Setting the initial condition of motors

// make sure both PWM pins are LOW

digitalWrite(pwm1,LOW);

digitalWrite(pwm2,LOW);

// State of DIR pins are depending on your physical connection

// if your robot behaves strangely, try changing thses two values

digitalWrite(dir1,LOW);

digitalWrite(dir2,LOW);

}

void loop() {

// Checking for junction crossing, if juction detected,

// keep moving forward

if(digitalRead(junctionPulse)) {

while(digitalRead(junctionPulse)) {

moveForward();

}

// Increment junction count by 1 after the junction

// You can do whatever you want with the junction count

junctionCount++;

}

readVal = analogRead(analogPin); // Read value from analog pin

// Convert voltage level into line position value

positionVal = ((float)readVal/921)\*70;

// Line at left when 0 - 18, move left

if(positionVal <= 18)

moveLeft();

// Line at middle when 19 - 52, move forward

else if(positionVal <= 52)

moveForward();

// Line at right when 53 - 70, move right

else if(positionVal <= 70)

moveRight();

// If no line is detected, stay at the position

else

wait();

// Put some delay to avoid the robot jig while making a turn

}

// The values work good in my case, you could use other values set

// to archieve a performance that satisfy you

void moveLeft() {

// For robot to move left, right motor has to be faster than left motor

analogWrite(pwm1,90);

analogWrite(pwm2,10);

}

void moveRight() {

// For robot to move right, left motor has to be faster than right motor

analogWrite(pwm1,10);

analogWrite(pwm2,90);

}

void moveForward() {

// For robot to move forward, both motors have to be same speed

analogWrite(pwm1,70);

analogWrite(pwm2,70);

}

void wait() {

// Function to makes the robot stay

analogWrite(pwm1,0);

analogWrite(pwm2,0);

}