#include <QTRSensors.h>

QTRSensors qtr;//class created in lib,instant of Qtr

const uint8\_t SensorCount = 8;

uint16\_t sensorValues[SensorCount];

//Set up the drive motor carrier pins

int bphase1=7; **//wheel1 or2?**

int bphase = 8;

int benbl = 6;

int aphase = 5;

int aphase1=4;

int aenbl = 3;

int mode=10; **//?**

char t;

boolean Ok = false;

void setup() {

Serial.begin(9600);

qtr.setTypeAnalog();

//Set up the sensor array pins, arduino 8 pins connected with qtr sensor's output pin

qtr.setSensorPins((const uint8\_t[]){A0, A1, A2, A3, A4, A5, A6, A7}, SensorCount);

qtr.setEmitterPin(2);//LEDON PIN

pinMode(aphase1, OUTPUT);//5

pinMode(aphase, OUTPUT);//4

pinMode(aenbl, OUTPUT);//3

pinMode(bphase, OUTPUT);//8

pinMode(bphase1, OUTPUT);//7

pinMode(benbl, OUTPUT);//6

pinMode(mode, INPUT); **//what is mode?**

delay(500);//input as millisec,wait for 500 ms

pinMode(LED\_BUILTIN, OUTPUT);//make this pin(13) as output

// forward\_brake(0, 0); **//why 0,0?**

}

void calibration() {

digitalWrite(LED\_BUILTIN, HIGH);//change state of pin13 as high, turn on Arduino's LED to indicate we are in calibration mode

// analogRead() takes about 0.1 ms on an AVR.

// 0.1 ms per sensor \* 4 samples per sensor read (default) \* 6 sensors

// \* 10 reads per calibrate() call = ~24 ms per calibrate() call.

// Call calibrate() 400 times to make calibration take about 10 seconds.

for (uint16\_t i = 0; i < 400; i++)

{

qtr.calibrate();

}

digitalWrite(LED\_BUILTIN, LOW);// turn off Arduino's LED to indicate we are through with calibration

}

void loop() {

int Mode=digitalRead(mode);

/\* if(Mode==1){

while (Ok == false) { //the loop won't start until the robot is calibrated

calibration(); //calibrate the robot for 10 seconds

Ok = true;

}

PID\_control();

Serial.println("Maze robot");

}

else{\*/

Serial.println("BLE robot");

if(Serial.available()){

t = Serial.read();

Serial.println(t);

if(t == 'F'){ //move forward(all motors rotate in forward direction)

forward();

}

else if(t == 'B'){ //move reverse (all motors rotate in reverse direction)

back();

}

else if(t == 'L'){ // move left

left();

}

else if(t == 'R'){ // move right right();

}

else if(t == 'S'){ //stop

stops();

}

}

}

/\*

else {

forward\_brake(0,0);

}\*/

void forward\_brake(int posa, int posb) {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, LOW);

digitalWrite(aphase1, HIGH);

digitalWrite(bphase, HIGH);

digitalWrite(bphase1, LOW);

analogWrite(aenbl, posa);//a

analogWrite(benbl, posb);

}

void forward() {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, LOW); **//why and how?**

digitalWrite(aphase1, HIGH); **//why and how?**

digitalWrite(bphase, HIGH); **//why and how?**

digitalWrite(bphase1, LOW); **//why and how?**

analogWrite(aenbl, 100); **//why and how?** //pwm funct,put 0 instead of 100,gives 0v,255 gives 5V,127 then 2.5V, **100 then 1.9V how why?**, 8bit system

analogWrite(benbl, 100); **//why and how?**

}

void back() {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, HIGH); **//why and how?**

digitalWrite(aphase1, LOW); **//why and how?**

digitalWrite(bphase, LOW); **//why and how?**

digitalWrite(bphase1, HIGH); **//why and how?**

analogWrite(aenbl, 100); **//why and how?**

analogWrite(benbl, 100); **//why and how?**

}

void left() {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, LOW); **//why and how?**

digitalWrite(aphase1, HIGH); **//why and how?**

digitalWrite(bphase, HIGH); **//why and how?**

digitalWrite(bphase1, LOW); **//why and how?**

analogWrite(aenbl, 0); **//why and how?**

analogWrite(benbl, 100); **//why and how?**

}

void right() {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, LOW); **//why and how?**

digitalWrite(aphase1, HIGH); **//why and how?**

digitalWrite(bphase, HIGH); **//why and how?**

digitalWrite(bphase1, LOW); **//why and how?**

analogWrite(aenbl, 100); **//why and how?**

analogWrite(benbl, 0); **//why and how?**

}

void stops() {

//set the appropriate values for aphase and bphase so that the robot goes straight

digitalWrite(aphase, HIGH); **//why and how?**

digitalWrite(aphase1, LOW); **//why and how?**

digitalWrite(bphase, LOW); **//why and how?**

digitalWrite(bphase1, HIGH); **//why and how?**

analogWrite(aenbl, 0); **//why and how?**

analogWrite(benbl, 0); **//why and how?**

}

**NOTES:**

**1)Serial.begin(9600):** passes the value 9600 to the ***speed***parameter. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second.  That’s 9600 binary ones or zeros per second, and is commonly called a baud rate.

2) The **pinMode**() function is used to configure a specific pin to behave either as an input or an output.

3) **Delay(500):** input as millisec, wait for 500 ms, Pauses the program for the amount of time (in milliseconds) specified as parameter.

4) **pinMode(LED\_BUILTIN, OUTPUT):** make this pin(13) as output

5**) digitalWrite:** Write a HIGH or a LOW value to a digital pin.

6)**digitalWrite(LED\_BUILTIN, HIGH):** change state of pin13 as high, turn on Arduino's LED to indicate we are in calibration mode

7) **digitalWrite(LED\_BUILTIN, LOW):** turn off Arduino's LED to indicate we are through with calibration

10) **digitalRead():** Reads the value from a specified digital pin, either HIGH or LOW.

11) **Serial.println():** Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print().

12**) Serial.available()** :Get the number of bytes (characters) available for reading from the serial port. This is data that’s already arrived and stored in the serial receive buffer (which holds 64 bytes). Serial.available() inherits from the Stream utility class.

13) **Serial.read()**: Reads incoming serial data. Serial.read() inherits from the Stream utility class.

14) **analogWrite():** Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

15)**PWM:** Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.