#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;

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); **//?**

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 (all motors rotate in left direction)

left();

}

else if(t == 'R'){ //move right (all motors rotate in right direction)

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);//4

digitalWrite(aphase1, HIGH);//5

digitalWrite(bphase, HIGH);//8

digitalWrite(bphase1, LOW);//7

analogWrite(aenbl, posa);//3

analogWrite(benbl, posb);//6

}

void forward() {

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

digitalWrite(aphase, LOW);//motor1,**make input pin high/low but how to find?**

digitalWrite(aphase1, HIGH);//m1

digitalWrite(bphase, HIGH);//m2

digitalWrite(bphase1, LOW);//m2

analogWrite(aenbl, 100); //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);

}

void back() {

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

digitalWrite(aphase, HIGH);

digitalWrite(aphase1, LOW);

digitalWrite(bphase, LOW);

digitalWrite(bphase1, HIGH);

analogWrite(aenbl, 100);

analogWrite(benbl, 100);

}

void left() {

//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, 0);

analogWrite(benbl, 100);

}

void right() {

//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, 100);

analogWrite(benbl, 0);

}

void stops() {

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

digitalWrite(aphase, HIGH);

digitalWrite(aphase1, LOW);

digitalWrite(bphase, LOW);

digitalWrite(bphase1, HIGH);

analogWrite(aenbl, 0);

analogWrite(benbl, 0);

}