|  |  |
| --- | --- |
|  | #include "MeMCore.h" |
|  | #define OPSPD 235 // Operating Speed |
|  | #define LOSPD 165 // Low Speed |
|  | #define TDURATION 270 // Standard time taken for 90 degree turn |
|  | //notes for victory music: |
|  | #define NOTE\_FS3 185 |
|  | #define NOTE\_G3 196 |
|  | #define NOTE\_A3 220 |
|  | #define NOTE\_B3 247 |
|  | #define NOTE\_C4 262 |
|  | #define NOTE\_CS4 277 |
|  | #define NOTE\_D4 294 |
|  | #define NOTE\_E4 330 |
|  |  |
|  |  |
|  | //Global variables |
|  | bool noAction = false; //META |
|  |  |
|  | MeDCMotor leftWheel(M1); // MOVEMENT |
|  | MeDCMotor rightWheel(M2); |
|  | MeLineFollower lineFinder(PORT\_2); |
|  | MePort IR(PORT\_3); |
|  | int8\_t LIR, RIR, a = 0, b = 0, us = 12; // IR + ULTRASONIC |
|  | float left, right, leftBound, rightBound; //IR WALL DETECTION |
|  |  |
|  | MePort SND(PORT\_4); // SOUND |
|  |  |
|  | MeRGBLed rgbled\_7(7); // COLOUR |
|  | MeLightSensor lightsensor\_6(6); |
|  | double red, blue, green, white; |
|  |  |
|  | //Victory music data: |
|  |  |
|  | int melody[] = {NOTE\_D4,NOTE\_CS4,NOTE\_B3,NOTE\_FS3,NOTE\_FS3,NOTE\_FS3,NOTE\_FS3,NOTE\_FS3,NOTE\_FS3,NOTE\_B3, |
|  | NOTE\_B3,NOTE\_B3,NOTE\_B3,NOTE\_A3,NOTE\_B3,NOTE\_G3,NOTE\_G3,NOTE\_G3,NOTE\_G3,NOTE\_G3, |
|  | NOTE\_B3,NOTE\_B3,NOTE\_B3,NOTE\_B3,NOTE\_B3,NOTE\_CS4,NOTE\_D4,NOTE\_A3,NOTE\_A3,NOTE\_A3, |
|  | NOTE\_A3,NOTE\_A3,NOTE\_A3,NOTE\_D4,NOTE\_D4,NOTE\_D4,NOTE\_D4,NOTE\_E4,NOTE\_E4,NOTE\_CS4}; |
|  |  |
|  | int noteDurations[] = {2,2,4,4,8,8,8,8,8,8,8,8,4,8,4,3,8,8,8,8,8,8,8,8,4,8,4,3,8,8,8,8,8,8,8,8,4,8,4,3}; |
|  |  |
|  |  |
|  | //Function prototypes for those with default arguments |
|  | void leftTurn(int16\_t turnSpeed = OPSPD, uint16\_t duration = TDURATION); |
|  | void rightTurn(int16\_t turnSpeed = OPSPD, uint16\_t duration = TDURATION); |
|  |  |
|  | void setup(){ |
|  | Serial.begin(9600); |
|  | left = IR.aRead1()/51.0 , right = IR.aRead2()/51.0; |
|  | leftBound = left - 1.0, rightBound = right - 1.0; //CALIBRATE BOUNDS |
|  | pinMode(LIR, INPUT); |
|  | pinMode(RIR, INPUT); |
|  | //delay(300); |
|  | stepForward(50); //START MOVING |
|  | lineFinder.readSensors(); //CALIBRATE SENSORS |
|  | } |
|  |  |
|  | void loop(){ |
|  | colourAction(); |
|  | // if(lineFinder.readSensors() != 3 && ultraSound() < 25){ |
|  | // stopWheels(); |
|  | // solveChallenge(); |
|  | // } |
|  | // moveForward(); |
|  | } |
|  |  |
|  | // Challenge Solver |
|  | void solveChallenge(void){ |
|  | colourAction(); |
|  | } |
|  |  |
|  | //General movement functions |
|  | void stopWheels(void){ |
|  | leftWheel.stop(); |
|  | rightWheel.stop(); |
|  | } |
|  |  |
|  | int16\_t ultraSound(void){ |
|  | pinMode(us, OUTPUT); |
|  | digitalWrite(us, LOW); |
|  | delayMicroseconds(2); |
|  | digitalWrite(us, HIGH); |
|  | delayMicroseconds(2); |
|  | digitalWrite(us, LOW); |
|  | pinMode(us, INPUT); |
|  | return pulseIn(us, HIGH); |
|  | } |
|  |  |
|  | void moveForward(void){ |
|  | float left = IR.aRead1()/51.0, right = IR.aRead2()/51.0; |
|  | if (right < rightBound){ //correct towards left |
|  | b = 0; |
|  | leftWheel.run(-LOSPD); |
|  | rightWheel.run(OPSPD); |
|  | a++; |
|  | } |
|  | else if (left < leftBound){ //correct towards right |
|  | a = 0; |
|  | leftWheel.run(-OPSPD); |
|  | rightWheel.run(LOSPD); |
|  | b++; |
|  | } |
|  | else if (a > 125){ //repay correction debt |
|  | leftWheel.run(-OPSPD); |
|  | rightWheel.run(LOSPD); |
|  | } |
|  | else if (b > 125){ //repay correction debt |
|  | leftWheel.run(-LOSPD); |
|  | rightWheel.run(OPSPD); |
|  | } |
|  | else{ |
|  | leftWheel.run(-OPSPD); |
|  | rightWheel.run(OPSPD); |
|  | } |
|  | } |
|  |  |
|  | void stepForward(uint16\_t duration){ |
|  | leftWheel.run(-LOSPD); |
|  | rightWheel.run(LOSPD); |
|  | delay(duration); |
|  | stopWheels(); |
|  | } |
|  |  |
|  | void leftTurn(int16\_t turnSpeed = OPSPD, uint16\_t duration = TDURATION){ |
|  | leftWheel.run(turnSpeed); |
|  | rightWheel.run(turnSpeed); |
|  | delay(duration); |
|  | stopWheels(); |
|  | long dist = ultraSound(); |
|  | while (dist > 2000 && dist < 4200){ //distance approx > 1 tile and approx < 2 tiles - ignored when mBot is straight |
|  | leftWheel.run(-OPSPD + 50); |
|  | rightWheel.run(-LOSPD + 50); |
|  | dist = ultraSound(); |
|  | } |
|  | } |
|  |  |
|  | void rightTurn(int16\_t turnSpeed = OPSPD, uint16\_t duration = TDURATION){ |
|  | leftWheel.run(-turnSpeed); |
|  | rightWheel.run(-turnSpeed); |
|  | delay(duration); |
|  | stopWheels(); |
|  | long dist = ultraSound(); |
|  | while (dist > 2000 && dist < 4200){ |
|  | leftWheel.run(LOSPD - 50); |
|  | rightWheel.run(LOSPD - 50); |
|  | dist = ultraSound(); |
|  | } |
|  | } |
|  |  |
|  | void uTurn(void){ |
|  | float newleft = IR.aRead1()/51.0, newright = IR.aRead2()/51.0; |
|  | if (right - newright < left - newleft){ |
|  | leftWheel.run(-OPSPD); |
|  | rightWheel.run(-OPSPD); |
|  | } |
|  | else{ |
|  | leftWheel.run(OPSPD); |
|  | rightWheel.run(OPSPD); |
|  | } |
|  | delay(TDURATION \* 2); |
|  | stopWheels(); |
|  | } |
|  |  |
|  | void LLTurn(void){ |
|  | leftTurn(); |
|  | long t = millis(); |
|  | while(ultraSound() > 550 || millis()-t < 700){ //within 900 ms it will cover 1 tile distance |
|  | stepForward(100); |
|  | } |
|  | leftTurn(); |
|  | } |
|  |  |
|  | void RRTurn(void){ |
|  | rightTurn(); |
|  | long t = millis(); |
|  | while(ultraSound() > 550 || millis()-t < 700){ |
|  | stepForward(100); |
|  | } |
|  | rightTurn(); |
|  | } |
|  |  |
|  | // Colour based decisionmaking |
|  |  |
|  | void colourAction() { |
|  | // scan red |
|  | rgbled\_7.setColor(0,255,0,0); |
|  | rgbled\_7.show(); |
|  | delay(300); |
|  | red = lightsensor\_6.read(); |
|  | Serial.print("Red: "); |
|  | Serial.println(red); |
|  |  |
|  | // scan green |
|  | rgbled\_7.setColor(0,0,255,0); |
|  | rgbled\_7.show(); |
|  | delay(300); |
|  | green = lightsensor\_6.read(); |
|  | Serial.print("Green: "); |
|  | Serial.println(green); |
|  |  |
|  | // scan blue |
|  | rgbled\_7.setColor(0,0,0,255); |
|  | rgbled\_7.show(); |
|  | delay(300); |
|  | blue = lightsensor\_6.read(); |
|  | Serial.print("Blue: "); |
|  | Serial.println(blue); |
|  |  |
|  | // scan white |
|  | rgbled\_7.setColor(0,0,0,0); |
|  | rgbled\_7.show(); |
|  | delay(300); |
|  | white = lightsensor\_6.read(); |
|  | Serial.print("White: "); |
|  | Serial.println(white); |
|  |  |
|  | if (red > 550 && green > 550 && blue > 550) { // WHITE |
|  | uTurn(); |
|  | } |
|  | else if (red < 350 && green < 350 && blue < 350) { //BLACK |
|  | soundSense(); |
|  | if (noAction == true){ |
|  | play\_music(); |
|  | exit(0); |
|  | } |
|  | } |
|  | else if (blue > red - 50 && blue > green) { //BLUE |
|  | RRTurn(); |
|  | } |
|  | else if (green > red - 50 && green > blue) { //GREEN |
|  | rightTurn(); |
|  | } |
|  | else if (white < 180){ // RED |
|  | rgbled\_7.setColor(0,255,0,0); |
|  | rgbled\_7.show(); |
|  | // leftTurn(); |
|  | Serial.println("RED"); |
|  | } |
|  | else { // ORANGE |
|  | rgbled\_7.setColor(0,255,255,0); |
|  | rgbled\_7.show(); |
|  | // LLTurn(); |
|  | Serial.println("ORANGE"); |
|  | } |
|  | } |
|  |  |
|  | void soundSense() { |
|  | int fL = SND.aRead2(); |
|  | int fH = SND.aRead1(); |
|  | delay(1000); |
|  |  |
|  | float r = (float) fL / ((float) fH + 0.01); |
|  | if (r > 50 && r <= 5000){ |
|  | noAction = true; |
|  | } |
|  | else if (r > 5000) { // {300Hz > 3kHz} |
|  | leftTurn(); |
|  | } |
|  | else if (r < 0.46) { // {3kHz > 300Hz} |
|  | rightTurn(); |
|  | } |
|  | else if (r >= 0.46 && r <= 50) { // {3kHz same as 300Hz} |
|  | uTurn(); |
|  | } |
|  | } |
|  |  |
|  | void play\_music() { |
|  | while (true){ |
|  | for (int thisNote = 0; thisNote <= 39; thisNote++) { |
|  | int noteDuration = 1100/noteDurations[thisNote]; |
|  | tone(8, melody[thisNote],noteDuration); |
|  | int pauseBetweenNotes = noteDuration \* 1.30; |
|  | delay(pauseBetweenNotes); |
|  | noTone(8); |
|  | } |
|  | } |
|  | } |