|  |
| --- |
| #include <MeMCore.h> |
|  | #include <PID\_v1.h> |
|  | MeBuzzer buzzer; |
|  |  |
|  | #define NOTE\_E5 659 |
|  | #define NOTE\_F5 698 |
|  | #define NOTE\_FS5 740 |
|  | #define NOTE\_G5 784 |
|  | #define NOTE\_GS5 831 |
|  | #define NOTE\_A5 880 |
|  | #define NOTE\_AS5 932 |
|  | #define NOTE\_B5 988 |
|  | #define NOTE\_C6 1047 |
|  | #define NOTE\_CS6 1109 |
|  | #define NOTE\_D6 1175 |
|  | #define NOTE\_DS6 1245 |
|  | #define NOTE\_E6 1319 |
|  |  |
|  | #define LIGHT\_SENSOR A6 |
|  | #define MICROPHONE A0 |
|  | #define LEFT\_IR A2 |
|  | #define RIGHT\_IR A3 |
|  |  |
|  | // takes care of the PID |
|  | double SetpointLeft, InputLeft, OutputLeft; |
|  | double SetpointRight, InputRight, OutputRight; |
|  |  |
|  | // initialize PID objects |
|  | PID leftPID(&InputLeft, &OutputLeft, &SetpointLeft,0.5,0.2,0, DIRECT); |
|  | PID rightPID(&InputRight, &OutputRight, &SetpointRight,0.5,0.2,0, DIRECT); |
|  |  |
|  | // Initialize both motors |
|  | MeDCMotor motor1(M1); |
|  | MeDCMotor motor2(M2); |
|  |  |
|  | // Initialize sensors |
|  | MeLightSensor lightSensor(PORT\_6); |
|  | MeLineFollower lineFinder(2); |
|  |  |
|  | // Movement constants |
|  | uint8\_t motorOneSpeed = 225; // Left |
|  | uint8\_t motorTwoSpeed = 225; // Right |
|  |  |
|  | // function detects the amount of light on sensor |
|  | // then it proceeeds to solve the challenges based on the amount of light |
|  | void light\_challenge(){ |
|  | float voltage\_lux = 0; |
|  | // takes average amount of light |
|  | for (int i = 0; i < 5; i++) { |
|  | voltage\_lux += (analogRead(LIGHT\_SENSOR)/1023.0)\*5; |
|  | } |
|  | voltage\_lux /= 5; |
|  |  |
|  | if ((0.0 <= voltage\_lux) && (voltage\_lux < 1.3)){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | sound\_challenge(); |
|  | } |
|  | else if((1.3 <= voltage\_lux) && (voltage\_lux < 2.5)){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(500); |
|  | TurnRight(); |
|  | } |
|  | else if((2.5<= voltage\_lux)&&(voltage\_lux < 3.6)){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(500); |
|  | TurnLeft(); |
|  | } |
|  | else if(voltage\_lux >= 3.6){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(500); |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | } |
|  | } |
|  |  |
|  | // function detects the intensity of the sound on mic |
|  | // it then proceeds to solve challenges based on sound intensity |
|  | void sound\_challenge() { |
|  | float micVoltage = 0; |
|  | // takes average reading of intensity of sound |
|  | for (int i = 0; i < 5; i++) { |
|  | micVoltage += (analogRead(MICROPHONE)/1023.0 ) \* 5.0; |
|  | } |
|  | micVoltage /= 5; |
|  |  |
|  | if ( micVoltage < 0.90){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | victory\_tune(); |
|  | } else if ((0.90 <= micVoltage) && (micVoltage <= 2.7)){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(500); |
|  | TurnRight(); |
|  | } else if (micVoltage > 2.7 ){ |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(500); |
|  | TurnLeft(); |
|  | } |
|  | } |
|  |  |
|  | void victory\_tune(){ |
|  | int melody[] = { |
|  | NOTE\_C6, NOTE\_C6, NOTE\_C6, |
|  | NOTE\_C6, |
|  | NOTE\_GS5, NOTE\_AS5, |
|  | NOTE\_C6, 0, NOTE\_AS5, |
|  | NOTE\_C6, |
|  | 0, |
|  | }; |
|  | int noteDurations[] = { |
|  | 12, 12, 12, |
|  | 4, |
|  | 4, 4, |
|  | 12, 12, 12, |
|  | 1, |
|  | 2, |
|  | }; |
|  | for (int thisNote = 0; thisNote < 11; thisNote++) { |
|  |  |
|  | // to calculate the note duration, take one second |
|  | // divided by the note type. |
|  | // e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc. (Assuming 1 beat per sec) |
|  |  |
|  | int noteDuration = 1000/noteDurations[thisNote]; |
|  | buzzer.tone(8, melody[thisNote],noteDuration); |
|  |  |
|  | // to distinguish the notes, set a minimum time between them. |
|  | // the note's duration + 30% seems to work well: |
|  | int pauseBetweenNotes = noteDuration \* 1.30; |
|  | delay(pauseBetweenNotes); |
|  |  |
|  | // stop the tone playing: |
|  | buzzer.noTone(8); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | // moves the mBot forward |
|  | void moveForward(){ |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | } |
|  |  |
|  | // turns the mBot left |
|  | void TurnLeft(){ |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(motorTwoSpeed); |
|  | delay(675); |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | } |
|  |  |
|  | // turns the mBot right |
|  | void TurnRight(){ |
|  | motor1.run(-motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | delay(620); |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | } |
|  |  |
|  | // Detects a black line |
|  | // returns 1 if there is black line or 0 if there is no black line |
|  | int is\_black\_line(){ |
|  | int sensorState = lineFinder.readSensors(); |
|  | if (sensorState == S1\_IN\_S2\_IN ||sensorState == S1\_IN\_S2\_OUT||sensorState == S1\_OUT\_S2\_IN){ |
|  | return 1; |
|  | } |
|  | else{ |
|  | return 0; |
|  | } |
|  | } |
|  |  |
|  | void setup() { |
|  | Serial.begin(9600); |
|  | pinMode(A6, INPUT); // Light |
|  | pinMode(A1, INPUT); // Sound |
|  |  |
|  | double sum\_left = 0, sum\_right = 0; |
|  | int i; |
|  | // Calibrates the initial left and right distance of the mBot |
|  | for (i = 0; i < 10; i++) { |
|  | InputRight = analogRead(RIGHT\_IR); |
|  | InputLeft = analogRead(LEFT\_IR); |
|  | sum\_left += InputLeft; |
|  | sum\_right += InputRight; |
|  | delay(100); |
|  | } |
|  | SetpointLeft = sum\_left/10; |
|  | SetpointRight = sum\_right/10; |
|  | // turn PID on |
|  | leftPID.SetMode(AUTOMATIC); |
|  | rightPID.SetMode(AUTOMATIC); |
|  | } |
|  |  |
|  | void loop() { |
|  |  |
|  | Serial.println((analogRead(A6)/1023.0)\*5.0); |
|  | // take data from IR sensor |
|  | InputRight = analogRead(RIGHT\_IR); |
|  | InputLeft = analogRead(LEFT\_IR); |
|  | // run PID |
|  | leftPID.Compute(); |
|  | rightPID.Compute(); |
|  | // Detects for a black line |
|  | if (is\_black\_line() == 1) { |
|  | motor1.stop(); |
|  | motor2.stop(); |
|  | delay(1000); |
|  | light\_challenge(); |
|  | delay(300); |
|  | } |
|  | // adjust motor speed based on PID |
|  | motorOneSpeed = (OutputRight/2.2)+150; |
|  | motorTwoSpeed = (OutputLeft/2.2)+150; |
|  | motor1.run(motorOneSpeed); |
|  | motor2.run(-motorTwoSpeed); |
|  | } |