|  |
| --- |
| /\* |
|  | \*\* Line Follower Basic |
|  | \*\* Version 0.6 |
|  | \*\* Last Update: 2013-05-24 |
|  | \*\* By Stan, http://42bots.com |
|  | \*/ |
|  |  |
|  | /\* Define motor controll inputs \*/ |
|  | const int motorRPin1 = 2; // signal pin 1 for the right motor, connect to IN1 |
|  | const int motorRPin2 = 3; // signal pin 2 for the right motor, connect to IN2 |
|  | const int motorREnable = 5; // enable pin for the right motor (PWM enabled) |
|  |  |
|  | const int motorLPin1 = 4; // signal pin 1 for the left motor, connect to IN3 |
|  | const int motorLPin2 = 7; // signal pin 2 for the left motor, connect to IN4 |
|  | const int motorLEnable = 6; // enable pin for the left motor (PWM enabled) |
|  |  |
|  | /\* Define the pins for the IR receivers \*/ |
|  | const int irPins[6] = {A0, A1, A2, A3, A4, A5}; |
|  |  |
|  | /\* Define values for the IR Sensor readings \*/ |
|  |  |
|  | // an array to hold values from analogRead on the ir sensor (0-1023) |
|  | int irSensorAnalog[6] = {0,0,0,0,0,0}; |
|  |  |
|  | // an array to hold boolean values (1/0) for the ir sensors |
|  | int irSensorDigital[6] = {0,0,0,0,0,0}; |
|  |  |
|  | int treashold = 700; // IR sensor treashold value for line detection |
|  |  |
|  | // binary representation of the sensor reading |
|  | //from left to right when facing the same direction as the robot |
|  | int irSensors = B000000; |
|  |  |
|  | int count = 0; // number of sensors detecting the line |
|  |  |
|  | // A score to determine deviation from the line [-180 ; +180]. |
|  | // Negative means the robot is left of the line. |
|  | int error = 0; |
|  |  |
|  | int errorLast = 0; // store the last value of error |
|  |  |
|  | // A correction value, based on the error from target. |
|  | // It is used to change the relative motor speed with PWM. |
|  | int correction = 0; |
|  |  |
|  | int lap = 0; // keep track of the laps |
|  |  |
|  | /\* Set up maximum speed and speed for turning (to be used with PWM) \*/ |
|  | int maxSpeed = 255; // used for PWM to control motor speed [0 - 255] |
|  |  |
|  | /\* variables to keep track of current speed of motors \*/ |
|  | int motorLSpeed = 0; |
|  | int motorRSpeed = 0; |
|  |  |
|  | void setup() { |
|  |  |
|  | /\* Set up motor controll pins as output \*/ |
|  | pinMode(motorLPin1,OUTPUT); |
|  | pinMode(motorLPin2,OUTPUT); |
|  | pinMode(motorLEnable,OUTPUT); |
|  |  |
|  | pinMode(motorRPin1,OUTPUT); |
|  | pinMode(motorRPin2,OUTPUT); |
|  | pinMode(motorREnable,OUTPUT); |
|  |  |
|  | /\* Set-up IR sensor pins as input \*/ |
|  | for (int i = 0; i < 6; i++) { |
|  | pinMode(irPins[i], INPUT); |
|  | } |
|  |  |
|  | /\* |
|  | Optional change for better PWM control of the DC motors below. |
|  |  |
|  | Change the PWM frequency of digital pins 5 and 6 (timer0) to Phase-correct |
|  | PWM of 31.250 kHz from the default of ~500Hz. Using code from "Adjusting PWM |
|  | Frequencies http://playground.arduino.cc/Main/TimerPWMCheatsheet". |
|  |  |
|  | This requires a separate change in the wiring.c function in the Arduino |
|  | program file hardware\arduino\cores\arduino\wiring.c from: |
|  | #define MICROSECONDS\_PER\_TIMER0\_OVERFLOW (clockCyclesToMicroseconds(64 \* 256)) |
|  |  |
|  | to: |
|  | #define MICROSECONDS\_PER\_TIMER0\_OVERFLOW (clockCyclesToMicroseconds(1 \* 510)) |
|  |  |
|  | Without the change to wiring.c time functions (millis, delay, as well as |
|  | libraries using them will not work corectly. |
|  | \*/ |
|  | TCCR0A = \_BV(COM0A1) | \_BV(COM0B1) | \_BV(WGM01) | \_BV(WGM00); |
|  | TCCR0B = \_BV(CS00); |
|  |  |
|  | /\* Set-up console output for debugging. \*/ |
|  | //Serial.begin(115200); |
|  | } |
|  |  |
|  | void loop() { |
|  | Scan(); |
|  | UpdateError(); |
|  | UpdateCorrection(); |
|  | Drive(); |
|  | //Serial.println(); |
|  | } |
|  |  |
|  | void Scan() { |
|  | // Initialize the sensor counter and binary value |
|  | count = 0; |
|  | irSensors = B000000; |
|  |  |
|  | for (int i = 0; i < 6; i++) { |
|  | irSensorAnalog[i] = analogRead(irPins[i]); |
|  |  |
|  | if (irSensorAnalog[i] >= treashold) { |
|  | irSensorDigital[i] = 1; |
|  | } |
|  | else {irSensorDigital[i] = 0;} |
|  | //Serial.print(irSensorAnalog[i]); |
|  | //Serial.print("|"); |
|  | count = count + irSensorDigital[i]; |
|  | int b = 5-i; |
|  | irSensors = irSensors + (irSensorDigital[i]<<b); |
|  | } |
|  | } |
|  |  |
|  | void UpdateError() { |
|  |  |
|  | errorLast = error; |
|  |  |
|  | switch (irSensors) { |
|  |  |
|  | case B000000: |
|  | if (errorLast < 0) { error = -180;} |
|  | else if (errorLast > 0) {error = 180;} |
|  | break; |
|  |  |
|  | case B100000: // leftmost sensor on the line |
|  | error = -150; |
|  | break; |
|  |  |
|  | case B010000: |
|  | error = -90; |
|  | break; |
|  |  |
|  | case B001000: |
|  | error = -30; |
|  | break; |
|  |  |
|  | case B000100: |
|  | error = 30; |
|  | break; |
|  |  |
|  | case B000010: |
|  | error = 90; |
|  | break; |
|  |  |
|  | case B000001: // rightmost sensor on the line |
|  | error = 150; |
|  | break; |
|  |  |
|  | /\* 2 Sensors on the line \*/ |
|  |  |
|  | case B110000: |
|  | error = -120; |
|  | break; |
|  |  |
|  | case B011000: |
|  | error = -60; |
|  | break; |
|  |  |
|  | case B001100: |
|  | error = 0; |
|  | break; |
|  |  |
|  | case B000110: |
|  | error = 60; |
|  | break; |
|  |  |
|  | case B000011: |
|  | error = 120; |
|  | break; |
|  |  |
|  | /\* 3 Sensors on the line \*/ |
|  |  |
|  | case B111000: |
|  | case B011100: |
|  | error = -150; |
|  | break; |
|  |  |
|  | case B000111: |
|  | case B001110: |
|  | error = 150; |
|  | break; |
|  |  |
|  | /\* 4 Sensors on the line \*/ |
|  | case B111100: |
|  | error = -150; |
|  | break; |
|  |  |
|  | case B111010: |
|  | error = -150; |
|  | break; |
|  |  |
|  | case B001111: |
|  | error = 150; |
|  | break; |
|  |  |
|  | case B010111: |
|  | error = 150; |
|  | break; |
|  |  |
|  | /\* 5 Sensors on the line \*/ |
|  | case B111110: |
|  | error = -150; |
|  | break; |
|  |  |
|  | case B011111: |
|  | error = +150; |
|  | break; |
|  |  |
|  | case B111111: |
|  | lap = lap + 1; |
|  | error = 0; |
|  | break; |
|  |  |
|  | default: |
|  | error = errorLast; |
|  | // Serial.print("Unhandled case: "); |
|  | // Serial.println(count); |
|  | // Serial.print("| "); |
|  | // Serial.println(irSensors); |
|  | } |
|  | } |
|  |  |
|  | void UpdateCorrection() { |
|  |  |
|  | if (error >= 0 && error < 30) { |
|  | correction = 0; |
|  | } |
|  |  |
|  | else if (error >=30 && error < 60) { |
|  | correction = 15; |
|  | } |
|  |  |
|  | else if (error >=60 && error < 90) { |
|  | correction = 40; |
|  | } |
|  |  |
|  | else if (error >=90 && error < 120) { |
|  | correction = 55; |
|  | } |
|  |  |
|  | else if (error >=120 && error < 150) { |
|  | correction = 75; |
|  | } |
|  |  |
|  | else if (error >=150 && error < 180) { |
|  | correction = 255; |
|  | } |
|  |  |
|  | else if (error >=180) { |
|  | correction = 305; |
|  | } |
|  |  |
|  | if (error <= 0 && error > -30) { |
|  | correction = 0; |
|  | } |
|  |  |
|  | else if (error <= -30 && error > -60) { |
|  | correction = -15; |
|  | } |
|  |  |
|  | else if (error <= -60 && error > -90) { |
|  | correction = -40; |
|  | } |
|  |  |
|  | else if (error <= -90 && error > -120) { |
|  | correction = -55; |
|  | } |
|  |  |
|  | else if (error <= -120 && error > -150) { |
|  | correction = -75; |
|  | } |
|  |  |
|  | else if (error <= -150 && error > -180) { |
|  | correction = -255; |
|  | } |
|  |  |
|  | else if (error <= -180) { |
|  | correction = -305; |
|  | } |
|  |  |
|  | /\* Adjust the correction value if maxSpeed is less than 255 \*/ |
|  | correction = (int) (correction \* maxSpeed / 255 + 0.5); |
|  |  |
|  | if (correction >= 0) { |
|  | motorRSpeed = maxSpeed - correction; |
|  | motorLSpeed = maxSpeed; |
|  | } |
|  |  |
|  | else if (correction < 0) { |
|  | motorRSpeed = maxSpeed; |
|  | motorLSpeed = maxSpeed + correction; |
|  | } |
|  | } |
|  |  |
|  | void Drive() { |
|  | if (motorRSpeed > 255) {motorRSpeed = 255;} |
|  | else if (motorRSpeed < -255) {motorRSpeed = -255;} |
|  |  |
|  | if (motorLSpeed > 255) {motorLSpeed = 255;} |
|  | else if (motorLSpeed < -255) {motorLSpeed = -255;} |
|  |  |
|  | if (motorRSpeed > 0) { // right motor forward (using PWM) |
|  | analogWrite(motorREnable, motorRSpeed); |
|  | digitalWrite(motorRPin1, HIGH); |
|  | digitalWrite(motorRPin2, LOW); |
|  | } |
|  |  |
|  | else if (motorRSpeed < 0) {// right motor reverse (using PWM) |
|  | analogWrite(motorREnable, abs(motorRSpeed)); |
|  | digitalWrite(motorRPin1, LOW); |
|  | digitalWrite(motorRPin2, HIGH); |
|  | } |
|  |  |
|  | else if (motorRSpeed == 0) { // right motor fast stop |
|  | digitalWrite(motorREnable, HIGH); |
|  | digitalWrite(motorRPin1, LOW); |
|  | digitalWrite(motorRPin2, LOW); |
|  | } |
|  |  |
|  | if (motorLSpeed > 0) { // left motor forward (using PWM) |
|  | analogWrite(motorLEnable, motorLSpeed); |
|  | digitalWrite(motorLPin1, HIGH); |
|  | digitalWrite(motorLPin2, LOW); |
|  | } |
|  |  |
|  | else if (motorLSpeed < 0) { // right motor reverse (using PWM) |
|  | analogWrite(motorLEnable, abs(motorLSpeed)); |
|  | digitalWrite(motorLPin1, LOW); |
|  | digitalWrite(motorLPin2, HIGH); |
|  | } |
|  |  |
|  | else if (motorLSpeed == 0) { // left motor fast stop |
|  | digitalWrite(motorLEnable, HIGH); |
|  | digitalWrite(motorLPin1, LOW); |
|  | digitalWrite(motorLPin2, LOW); |
|  | } |
|  | } |