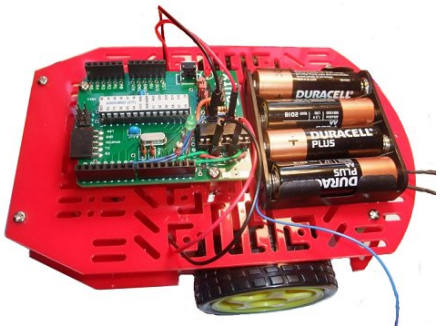
Magician Chassis Line Avoidance

Magician Chassis Line Avoidance

We are going to add a QTR-1RC Infra Red Reflectance Sensor to our Magician Chassis and program it to avoid high reflectance areas (i.e. white lines)



If you look at the specifications for the sensor, you will notice the Optimal sensing distance is only 0.125" (3 mm). It's no good mounting our sensor high up on the chassis as it wont detect very much. We need to get it as close to the ground as possible for accurate detection of our line. 3mm is perhaps too close, so we have mounted ours with the lowest part of the sensor about 6mm off the floor. This at least allows some ground clearance and protection of the sensor.

We mounted the sensor onto a small nylon standoff using a small screw and Araldite. The standoff is then screwed into one of the spare metal standoff left over from the Magician chassis.

The sensor is mounted at the front of the chassis

and wired to Arduino pins 12 (signal), 13 (power) and GND.

Here is the sketch used for this part of the project

```
/* Magician Chassis Test
   Hobbytronics ltd, 2011

   White Line avoidance (detector) example program
   This example program uses a single Pololu QTR-1RC sensor mounted at the front
   of the Magician Chassis approx 6mm above the floor.
   Sensor is connected to pins 12, 13 and GND
   Pin 12 is the sensor input, Pin 13 supplies Power.

   It uses the Pololu QTR Sensor Arduino Library, but doesn't use their line
   position functions. Instead it reads the sensor values directly and looks
   for a drop in value (increase in Reflectance) of an amount DETECT_LEVEL (150)
   A de-bounce method is used to remove false positives.

   Once a white line is detected, The LED will light, the robot will stop
   and turn right approx 60 degrees and then try and carry on

   On switch on, a baseline reading of the sensors is taken so the Chassis should
   be on the floor and not on the white line at this point.
*/

#include <PololuQTRSensors.h>

#define NUM_SENSORS   1     // number of sensors used
#define TIMEOUT       2500  // waits for 2500 us for sensor outputs to go low
#define DETECT_LEVEL  150   // Sensor needs to change this amount for detection
#define TRIGGER_COUNT 10    // Wait n successive readings valid detection

// Create instance of sensors. We are only using one sensor connected to pin 12
PololuQTRSensorsRC qtrrc((unsigned char[]) {12},
                         NUM_SENSORS, TIMEOUT, QTR_NO_EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS];
unsigned int sensorValuesBase[NUM_SENSORS];
unsigned int detect_count[NUM_SENSORS];    // Number of successive times sensor
                                           // has detected a drop in value. If
                                           // this is greater than trigger_count
                                           // then white line detected
// Setup pins for SN754410 Motor chip

int lf = 6;      // Left motor Forward
int lr = 5;      // Left motor Reverse
int rf = 11;     // Right motor Forward
int rr = 10;     // Right motor Reverse


int led = 3;    // LED and resistor across pins 2 and 3
int led_gnd = 2;

unsigned char line_detect=1;
unsigned char do_turn=0;

void setup()  {
  pinMode(led, OUTPUT);
  pinMode(led_gnd, OUTPUT);
  digitalWrite(led_gnd, LOW);    // LED ground

  pinMode(13, OUTPUT);
```

```
  digitalWrite(13, HIGH);         // turn on Sensor
  //Read and store the baseline values
  qtrrc.read(sensorValues);
  sensorValuesBase[0] = sensorValues[0];
}

void loop()  {

  if(line_detect==1){
    // Line Detect is enabled
    qtrrc.read(sensorValues);
    if(sensorValues[0] < (sensorValuesBase[0]- DETECT_LEVEL)) {
      // Detected drop on value - brighter object detected
      detect_count[0]++;   // incerment count
      if(detect_count[0] >= TRIGGER_COUNT) {
        // White Line detected
        detect_count[0]=0;           // Reset count
        do_turn=1;                   // Start the turn
        digitalWrite(led, HIGH);     // turn on LED
        line_detect=0;               // Turn off detection until turn complete

        // Come to s stop
        analogWrite(lf, 0);
        analogWrite(rf, 0);
        analogWrite(lr, 0);
        analogWrite(rr, 0);
        delay(200);
      }
    }
    else {
      detect_count[0]=0;
    }

  }

  if(do_turn==1){
    //Turn right a bit to avoid obstacle
    analogWrite(lr, 0);
    analogWrite(rf, 0);
    analogWrite(lf, 200);
    analogWrite(rr, 200);
    delay(500);                 // Turn for half a second
    line_detect=1;              // turn line_detect back on
    do_turn=0;                  // turn off
    digitalWrite(led, LOW);     // turn off Led
  }
  else {
    // go forward until we detect something
    // Forward
    analogWrite(lr, 0);
    analogWrite(rr, 0);
    analogWrite(lf, 250);
    analogWrite(rf, 250);
  }

}
```

And check out the following video to see the performance.