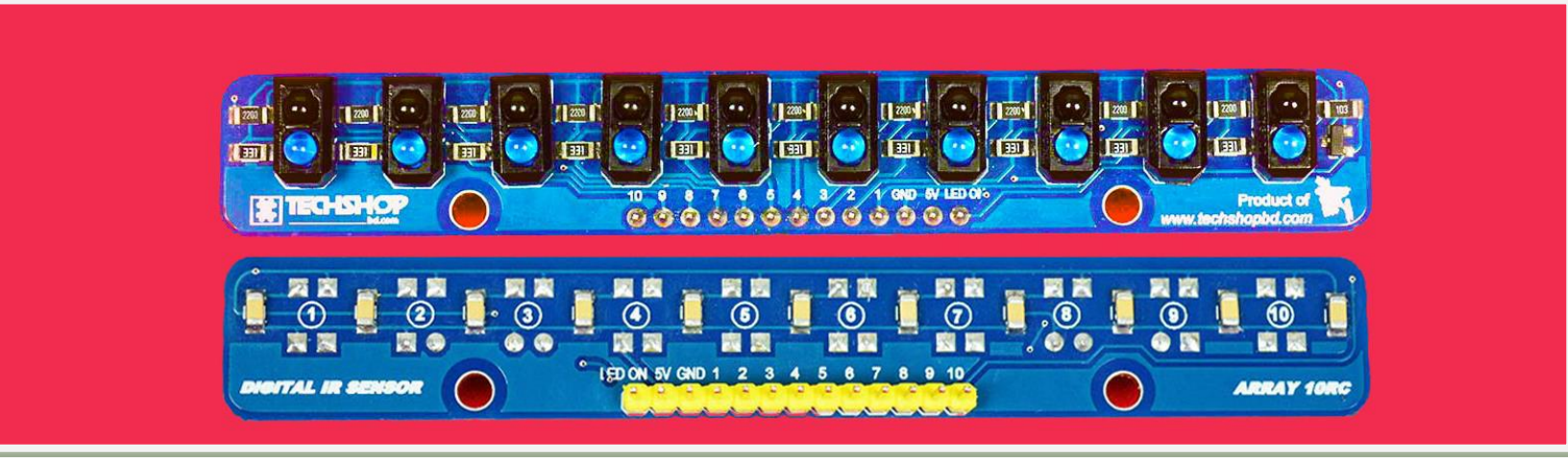
USER MANUAL

Digital IR Sensor Array TCRT-10RC

(Line follower robot sensor with 10 digital outputs)



If your Arduino Uno (that has only 6 ADC pins) is restricting you from buying a **better** 8 or 10 IR based **analog** line detection sensor, this product is for you. It's similar to the polulu QTR-8RC sensor, but contains more IR sensors. It has 1cm spacing between IR sensors and got a total of 10 sensors. The signal pins can be connected to any digital/analog pin of any Arduino board.

Unlike other digital line follower sensors that use op-amps as comparator and provide only **1 & 0** as signal, it provides 0-4096 by using capacitor discharge time. That means this digital sensor can provide outputs like an analog sensor. It can measure white & black & all the greyscale colors in-between.

Moreover, this sensor has got IR LED control option that lets you turn ON and OFF the IR LEDs of the sensor array using a digital pin of Arduino. That ensures unnecessary power consumption when the sensors are not in use. Moreover, the ability to read IR sensors while turning ON and OFF the IR LEDs helps to eliminate effect of flash lights and sunlight.

The sensor is fully compatible with Pololu QTR library.

Features:

1. Best digital line sensor for making any type of line follower robot.
2. Supports Pololu QTR library (QTR RC). You can use it just like a Pololu QTR RC sensor.
3. Sunlight and flash resistant. Ability to turn ON & OFF IR LEDs by arduino digital pin.
4. Small and compact design. Uniform 1cm distance between IR sensors for better line detection.

Specifications:

- Features TCRT500 that has effective plastic packaging and built-in sunlight effect compensation.
- 1 cm spacing between two adjacent sensors, covering 9cm distance by the entire sensor array.
- Recommended operational height from ground 3mm-50mm. (height of IR sensors tip, not the PCB).
- Generates high discharge time for black surface and low discharge time for white surface on signal pins.

Connecting to Arduino:

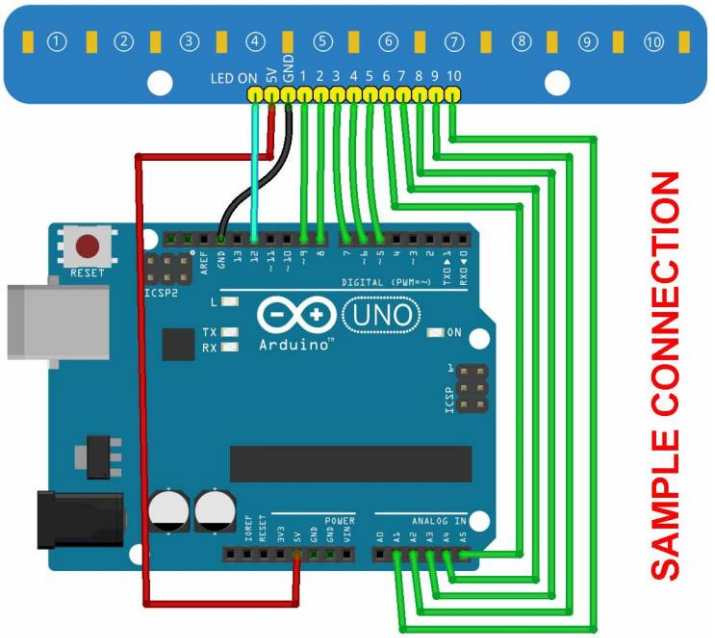
You can use this sensor with any 5V Arduino or any type of microcontroller that runs on 5V. As an example, here we'll connect this sensor to Arduino Uno.

FYI, Arduino analog pins can also be used as digital pins (Uno Analog pins 0-5 = Uno digital pin 14-19). This example shows how to get reading from this sensor using both analog and digital pins.

Connecting the "LED ON" pin is optional but will give you better results if you choose to connect the LED ON pin to digital pin 12 like this example.

Make sure that you never connect any signal pin to arduino pin 0 & 1 (RX & TX) while Serial0 is initiated. And don't connect to digital pin 13 as there's a LED connected to it.

Sensor Pin	Arduino Pin
5V	5V
GND	GND
LED ON	12
1	9
2	8
3	7
4	6
5	5
6	A5 (digital pin19)
7	A4 (digital pin18)
8	A3 (digital pin17)
9	A2 (digital pin16)
10	A1 (digital pin15)



Sample test Code using Pololu QTR Library:

You can also use the sensor with Pololu QTR library.

Download Pololu QTR library and install to Arduino IDE. Then run the following code.

```
#include <QTRSensors.h>
/* 10 sensor pins, Num of sensors, timeout or full black value, LED ON pin*/
QTRSensorsRC qtrrc((unsigned char[]){9,8,7,6,5,19,18,17,16,15}, 10, 4000, 12);
unsigned int sensorValues[10];

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  qtrrc.read(sensorValues);
  for (unsigned char i = 0; i < 10; i++)
  {
    Serial.print(sensorValues[i]);
    Serial.print('\t');
  }
  Serial.println();
  delay(50);
}
```

Sample test code (without library):

If you want to use the sensor with raw code, then use the following one.

```
unsigned int sensor[10];/*sensor readings are saved here*/
unsigned int sensorWhileLEDoff[10]; /* sensor reading while LED is off*/
boolean firstData[10];/*to eliminate effect of noise*/
byte sensorPin[10] = {9,8,7,6,5,19,18,17,16,15};/*arduino pins to read
sensors*/
byte NumOfSensor = 10;
byte i; /*just to run for loop!!*/
unsigned int MaxWaitTime = 4000; /*equivalent to 12 bit ADC*/



void setup()
{
  Serial.begin(9600);
  pinMode(12,OUTPUT);
}


void loop()
{
    digitalWrite(12, LOW); // turn ON IR LEDs
    delay(20); // give some time to turn ON
    readSensor();  // read line sensors while IR LEDs are OFF

    // store current sensor data to sensorWhileLEDoff
    for (i = 0; i <= 9; i++) sensorWhileLEDoff[i] = MaxWaitTime - sensor[i];

    digitalWrite(12, HIGH); // turn ON IR LEDs
    delay(20); // give some time to turn ON
    readSensor();  // read line sensors while IR LEDs are ON

    // calculate external light compensated readings
    for (i = 0; i <= 9; i++) sensor[i] = sensor[i] + sensorWhileLEDoff[i];

    showsensorData();
}
```

```
void readSensor()
{
  for (i = 0; i < NumOfSensor; i++)
  {
    digitalWrite(sensorPin[i], HIGH);
pinMode(sensorPin[i], OUTPUT);
  }
  delayMicroseconds(10);
  for (i = 0; i < NumOfSensor; i++)
  {
    pinMode(sensorPin[i], INPUT);
    digitalWrite(sensorPin[i], LOW);
    sensor[i] = MaxWaitTime;
    firstData[i] = false;
  }
  unsigned long startTime = micros();
  while (micros() - startTime < MaxWaitTime)
  {
    unsigned int time = micros() - startTime;
    for (i = 0; i < NumOfSensor; i++)
    {
      if ((digitalRead(sensorPin[i]) == LOW) && (firstData[i] == false))
      {
        sensor[i] = time;
        firstData[i] = true;
      }
    }
  }
}

void showsensorData()
{
  for (i = 0; i < NumOfSensor; i++)
  {
    Serial.print(sensor[i]);
    Serial.print(" ");
  }
  Serial.println();
}
```


Note:

1. Do not connect any of the signal pins to arduino **pin 13** or any other pin that you have connected to an LED.
2. Do not connect any of the signal pins to arduino **serial0 pins (digital pin 0 & digital pin 1)** if you are about to view data using Serial0.