Sharp GP2Y1010AU0F Optical Dust Sensor Hook-Up Guide

Introduction

The Sharp GP2Y1010AU0F optical dust sensor is a small (46.0 × 30.0 × 17.6 mm) lightweight (~15g) air quality sensor containing an infra-red LED and a phototransistor capable of detecting air-borne dust and smoke. The sensor is available from SparkFun and their distributors (Product ID: COM-09689) and has an approximate sensitivity of 0.5V/0.1mg/m^3.

Sharp have produced a detailed datasheet for the sensor:
http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y1010au_e.pdf
together with a rather useful application note:
http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y1010au_appl_e.pdf

Components

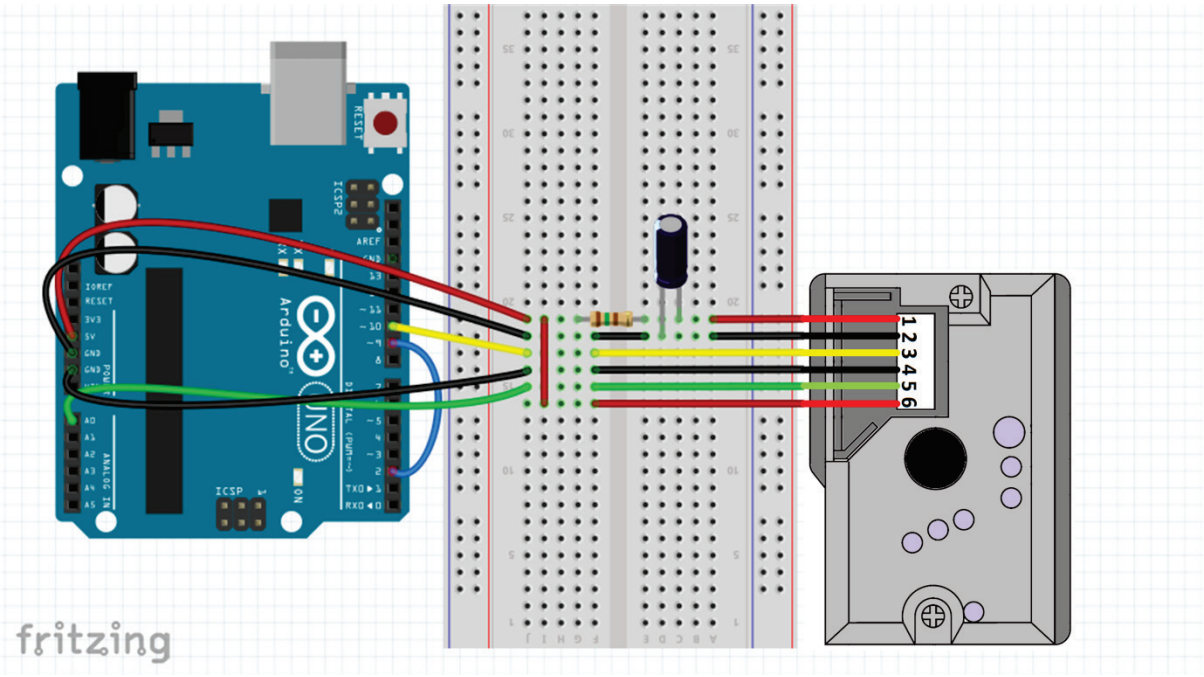
You will need:

- Arduino Uno or equivalent (e.g. the SparkFun RedBoard – product ID: DEV-12757)
- Sharp GP2Y1010AU0F optical dust sensor (SparkFun ID: COM-09689)
- 6-Way JST ZHR-6 connector housing (SparkFun ID: PRT-09690)
- >= 6 JST SZH-002T-P0.5 crimp pins (SparkFun ID: PRT-09728)
- 150R Resistor
- 220µF Electrolytic Capacitor
- Small Breadboard or Veroboard
- Hook-up wires

Wiring

The tricky bit is crimping the six SZH-002T-P0.5 crimp pins onto suitable wires to interface to the Arduino. JST do of course offer a dedicated crimp tool for the job but it is expensive. The crimp pins are very small but can be crimped using a regular crimp tool so long as care is taken. Wouldn't it be great if the nice people at SparkFun could offer a ready-crimped cable as a new product?!

Connect the Arduino and dust sensor as follows:



Connect Arduino 5V:

- directly to pin 6 of the dust sensor
- to pin 1 of the dust sensor via a 150R resistor

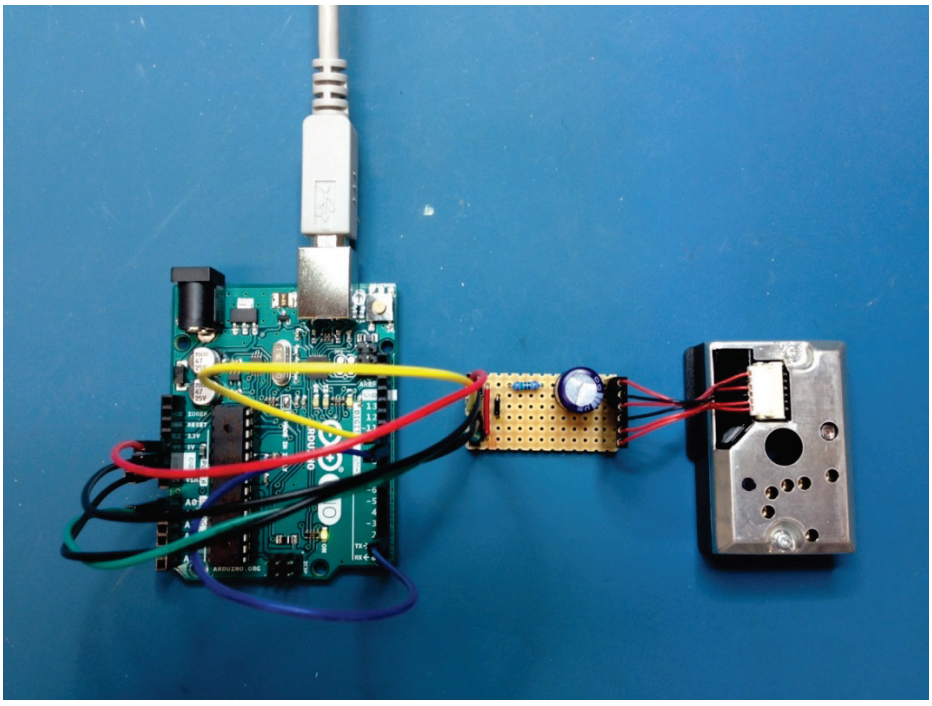
Connect Arduino GND to pins 2 and 4 of the dust sensor

Connect Arduino A0 to pin 5 of the dust sensor

Connect Arduino D10 to pin 3 of the dust sensor

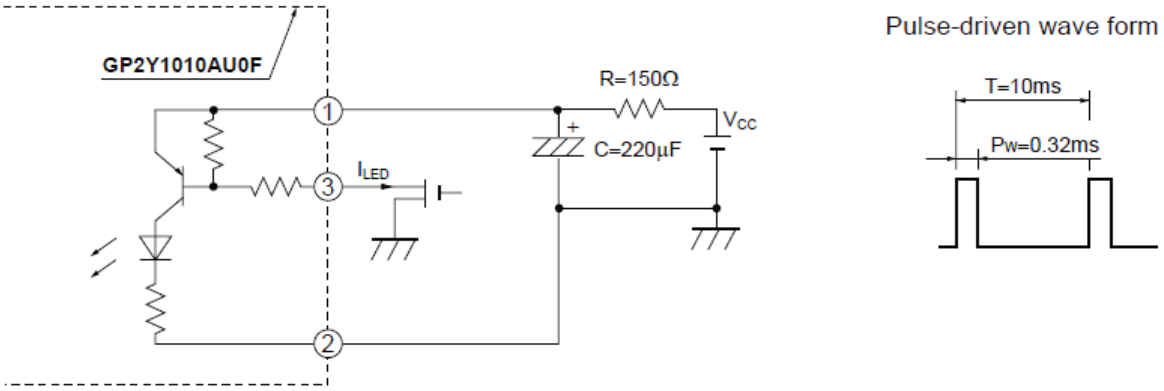
Connect a 220µF capacitor between pin 1 (+ve) and pin 2 (GND) of the dust sensor

Connect Arduino D9 to D2



LED Pulses

The GP2Y1010AU0F data sheet calls for the sensor LED (pin 3) to be pulsed on for 0.32msec every 10msec (100Hz). The datasheet schematic includes an additional transistor (FET) between the controller and pin 3, so in reality pin 3 needs to be pulled LOW by the Arduino to turn the LED on.



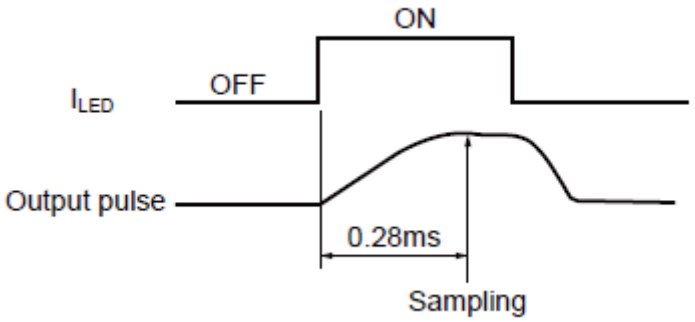
We can use the Arduino Timer 1 OC1B output (D10) to create the correct pulse-width modulated waveform for the LED: low for 0.32msec every 10msec.

Fortunately, Jesse Tane and others have written a rather nice library which lets us access Timer 1 functions with ease. You can download the library from:
http://playground.arduino.cc/Code/Timer1
https://code.google.com/p/arduino-timerone/downloads/list
and can find instructions on how to add this to the Arduino libraries here:
https://www.arduino.cc/en/Guide/Libraries

We need to set Timer 1 to have a pulse length of 10,000 microseconds (10 milliseconds or 100Hz). Pin 10 is put into Pulse Width Modulation mode with a HIGH period of 9.68 milliseconds.

Phototransistor Samples

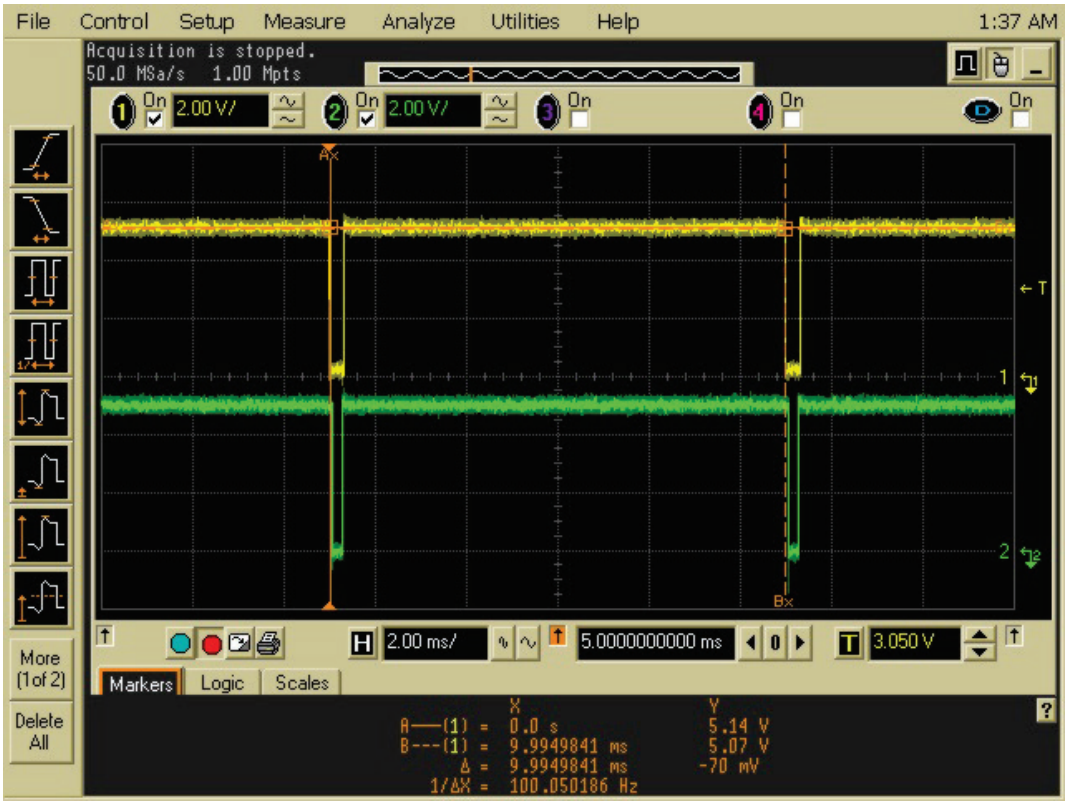
The GP2Y1010AU0F data sheet calls for the phototransistor output (pin 5) to be sampled 0.28msec into the LED pulse.



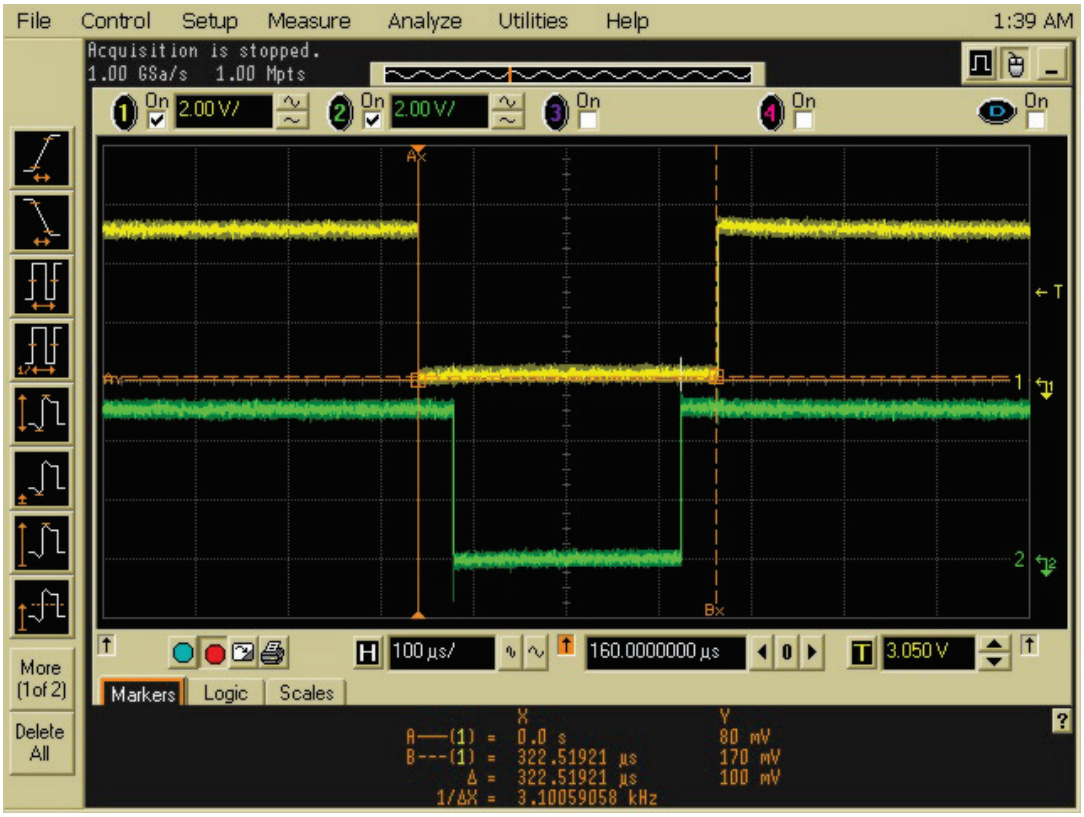
We can do this quite easily by programming the Timer 1 OC1A output (D9) to produce a PWM output with a LOW period of 0.28msec every 10msec (HIGH for 9.72msec). By connecting D9 to D2 we can generate an interrupt (INT0) at the correct point in the cycle to sample the sensor output voltage via A0. We generate the interrupt on the RISING edge of D2.

A quick check with an oscilloscope confirms that we are generating the correct waveforms on OC1B (D10, Yellow trace) and OC1A (D9, Green trace) :

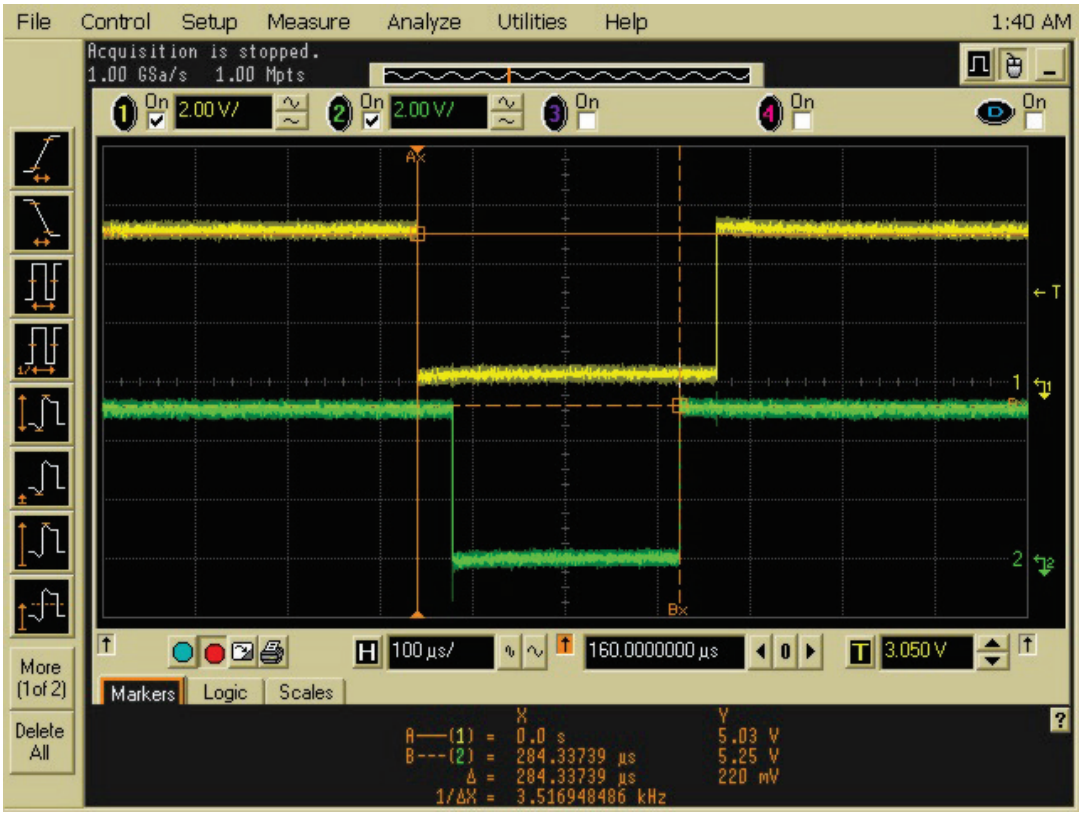
Correct timing of the OC1B pulses: low every 10msec:



Correct low pulse width of 0.32msec (320µsec) on OC1B (D10):



Correct OC1A (D9) delay of 0.28msec (280µsec) from LED switch on to INT0 interrupt (D2):



Here is the Arduino code:

```
// Paul's GP2Y1010AU0F Dust Sensor Sketch

/*
Arduino Uno (ATMEGA328) Port Allocations:

A0:    A/D:                Dust Sensor Analog Signal
D2:    INT0 I/P:           Dust Sensor Interrupt - Link to D9
D9:    Timer 1 OC1A PWM O/P:   Dust Sensor Samples - Link to D2
D10:   Timer 1 OC1B PWM O/P:   Dust Sensor LED Pulses
*/

#define VOLTAGE 5.0 // Arduino supply voltage

// Download the Timer 1 Library (r11) from:
// http://playground.arduino.cc/Code/Timer1
// https://code.google.com/p/arduino-timerone/downloads/list

#include <TimerOne.h>

/*
  Smoothing

  Reads repeatedly from an analog input, calculating a running average.
  Keeps readings in an array and continually averages them.

  Created 22 April 2007
  By David A. Mellis  <dam@mellis.org>
  modified 9 Apr 2012
  by Tom Igoe
  http://www.arduino.cc/en/Tutorial/Smoothing

  This example code is in the public domain.
*/

/* With thanks to Adafruit for the millis code - plagiarised from the Adafruit GPS Library */

const int numReadings = 100;    // samples are taken at 100Hz so calculate average over
1sec
```

```
int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
long int total = 0;             // the running total
int latest_reading = 0;         // the latest reading
int average_reading = 0;        // the average reading

int dustPin = A0;

// Initialisation routine

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

  // Configure PWM Dust Sensor Sample pin (OC1A)
  pinMode(9, OUTPUT);
  // Configure PWM Dust Sensor LED pin (OC1B)
  pinMode(10, OUTPUT);
  // Configure INT0 to receive Dust Sensor Samples
  pinMode(2, INPUT_PULLUP);

  // Put Timer 1 into 16-bit mode to generate the 0.32ms low LED pulses every 10ms
  // A0 needs to be sampled 0.28ms after the falling edge of the LED pulse - via INT0 driven
by OC1A (D9)
  Timer1.initialize(10000); // Set a timer of length 10000 microseconds (or 10ms - or 100Hz)
  Timer1.pwm(10, 991); // Set active high PWM of (10 - 0.32) * 1024 = 991
  Timer1.pwm(9, 999); // Set active high PWM of (10 - 0.28) * 1024 = 995 BUT requires a
fiddle factor making it 999

  // Attach the INT0 interrupt service routine
  attachInterrupt(0, takeReading, RISING); // Sample A0 on the rising edge of OC1A

  // Initialise sample buffer
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;
   }
}

// Dust sample interrupt service routine
void takeReading() {
   // subtract the last reading:
```

```
  total = total - readings[readIndex];
  // read from the sensor:
  latest_reading = analogRead(dustPin);
  readings[readIndex] = latest_reading;
  // add the reading to the total:
  total = total + latest_reading;
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...wrap around to the beginning:
  if (readIndex >= numReadings) readIndex = 0;

  // calculate the average:
  average_reading = total / numReadings; // Seems to work OK with integer maths - but
total does need to be long int
}

// Main loop

uint32_t timer = millis();

void loop()                    // run over and over again
{
 // if millis() or timer wraps around, we'll just reset it
 if (timer > millis())  timer = millis();

 // approximately every second or so, print out the dust reading
 if (millis() - timer > 1000) {
  timer = millis(); // reset the timer

  float latest_dust = latest_reading * (VOLTAGE / 1023.0);
  float average_dust = average_reading * (VOLTAGE/ 1023.0);

  Serial.print("Latest Dust Reading (V): ");
  Serial.print(latest_dust);
  Serial.print("\t\tAverage Dust Reading (V): ");
  Serial.println(average_dust);
 }
}
```