American International University -Bangladesh

City Dust Monitoring System

Advance Operating Sytem

*Submitted By*

|  |  |
| --- | --- |
| 15-30781-3 | Al-Amin |
| 15-30823-3 | Jawad Al Hasan |
| 15-30762-3 | Karim, Masudul |

Department of Computer Science

Faculty of Science & Technology

American International University -Bangladesh

5th May, 2019

***Introduction:*** The project name is City Dust Monitoring System. This is a system that is used to measure the quantity of dust in surrounding area. It is an IOT based project where a programmable microcontroller used with a dust sensor.

**Project scheduling and timing:**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Name | Duration(hours) | Date |
| Requirement Specification & Resource gathering | Al-Amin  Jawad  Masud | 7  5  6 | 10-April-2019 to  12-April-2019 |
| Equipment’s collected | Jawad  Al-Amin  Masud | 6  7  6 | 15-April-2019 |
| Arduino IDE set up | Masud  Jawad  Al-Amin | 9  9  11 | 20-April-2019 |
| Connecting Dust sensor | Masud  Jawad  Al-Amin | 4  6  7 | 21-April-2019 |
| Development | Masud  Jawad  Al-Amin | 8  9  12 | 21-April-2019 to  28-April-2019 |

***Materials:***

1. ESP32 ( DOIT ESP32 DEVKIT V1)

2. GP2Y1010AU0F ( Dust Sensor)

3. 150 ohm resistor

4. 220 microfarad, 25 volt capacitor

5. Bread Board

6. Jumper wires

***Procedure:***

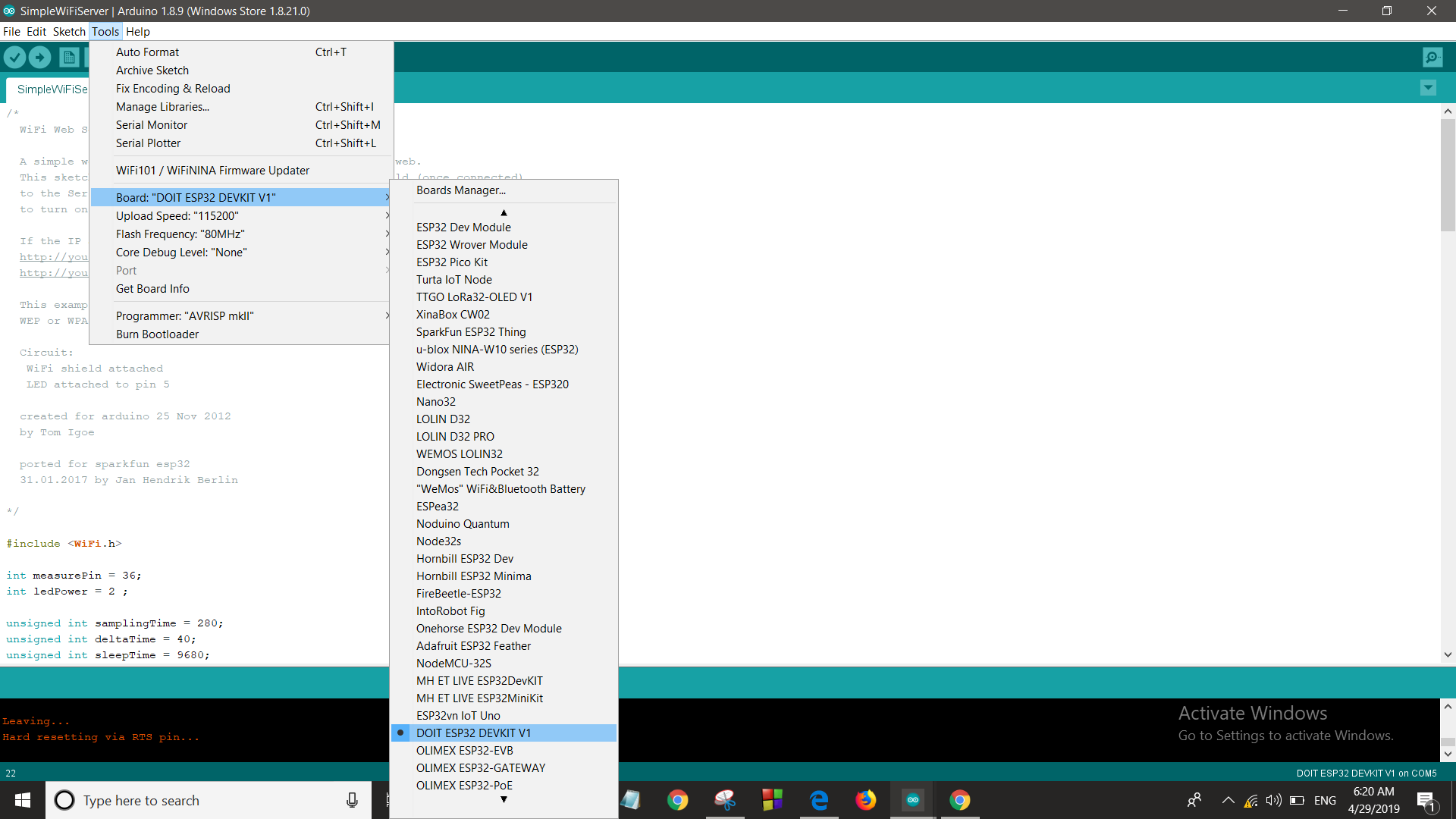
1. ESP-32 was connected to a computer via usb cable then we open Arduino IDE. In the IDE we navigated to tools > board and selected the board DOIT ESP32 DEVKIT V1 and Port>COM 10(as the board was connected to COM 10 port).
2. Now, The ESP-32 is ready for code. The code which was implemented to the ESP-32 Board was from the source of instructables.com and build-in Example code
3. ( File>Examples>WiFi>SimpleWiFiServer). The final code was the combination and modification of these two source code in order to implement the DUST SENSORE and to connect ESP-32 wirelessly to web server.
4. We compiled the code in IDE and uploaded the code to ESP-32.
5. Then we opened the serial monitor to check the connection of ESP-32 to the web server and to find the IP Address of ESP-32.
6. Unplugged ESP-32 from the computer.
7. Then we configured the system, at first we connected the VCC wire of dust sensor to the VIN pin of esp32 and 150 ohm resistor, the VO wire to the pin VP. The S-ground and LED\_ground pin was connect to the GND and a 220 microfarad capacitor , the cathode of the capacitor is connected to the cathode of resistor, the VLED wire is connected to the cathode of capacitor and resistor, and the LED wire is connected to the pin 22.
8. When connection is complete, if we open a browser which device is connected to same network of esp32 load the IP which we found in Serial monitor.
9. Then we loaded the IP which we found in Serial monitor.
10. We will see the realtime Voltage and the amount of dust in the webpage of browser.

***Processing:***

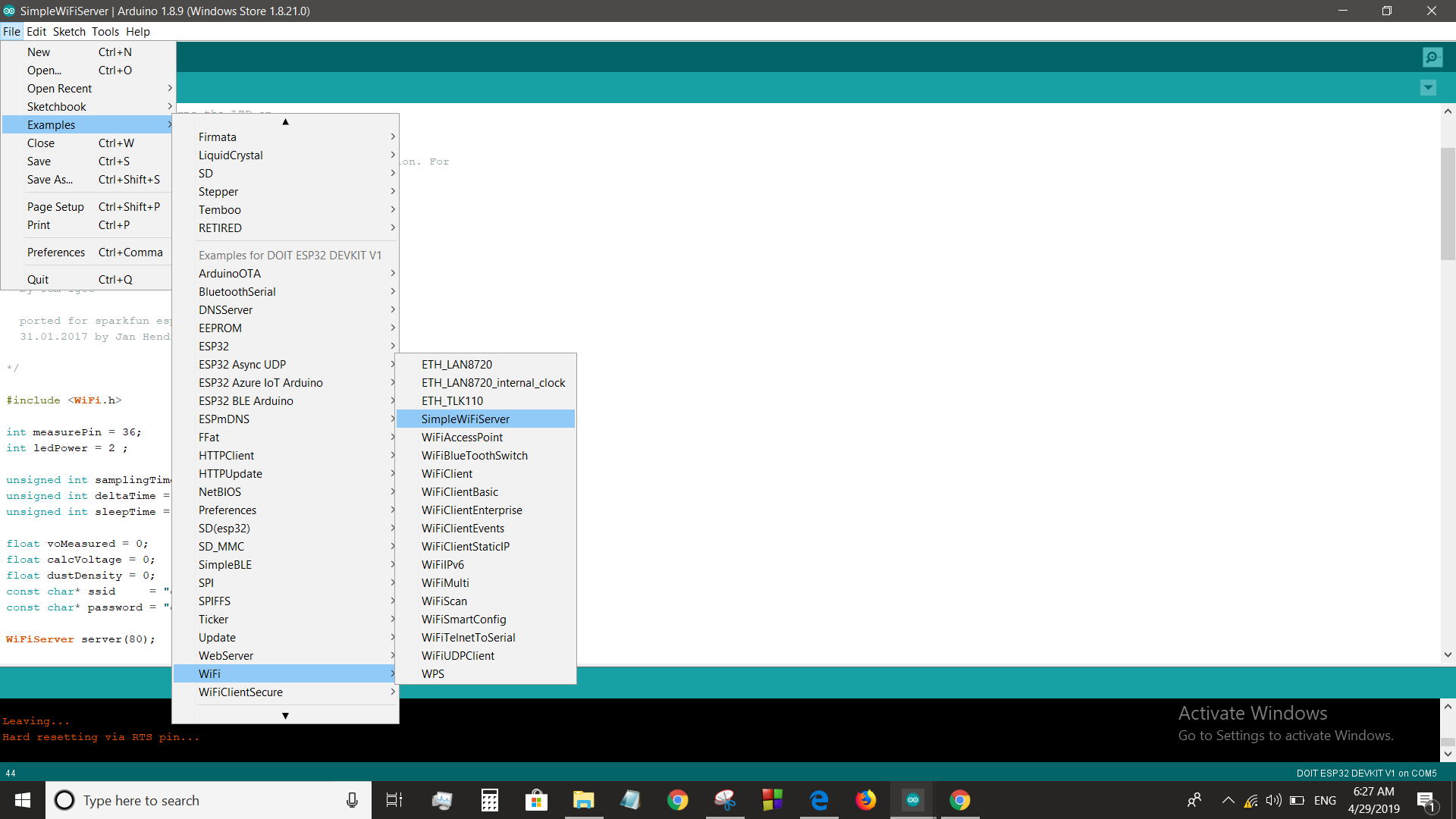
The voltage is passed to the dust sensor via VIN pins and VO pins passes some analogue output frequently which is taken as input to esp32 via GPIO pin A0. Led pin 22 was connected to the LED wire of the sensor to make sure data passing is full functional. Two GND wires are connected to the GND pins of esp32 to give ground connection. When browse the IP , the Dust quantity and voltages is shown to the webpage as the code implemented.

The Code and all other sources are included to this paper.

DOIT ESP-32 DEVKIT V1 setup Pictorial view :

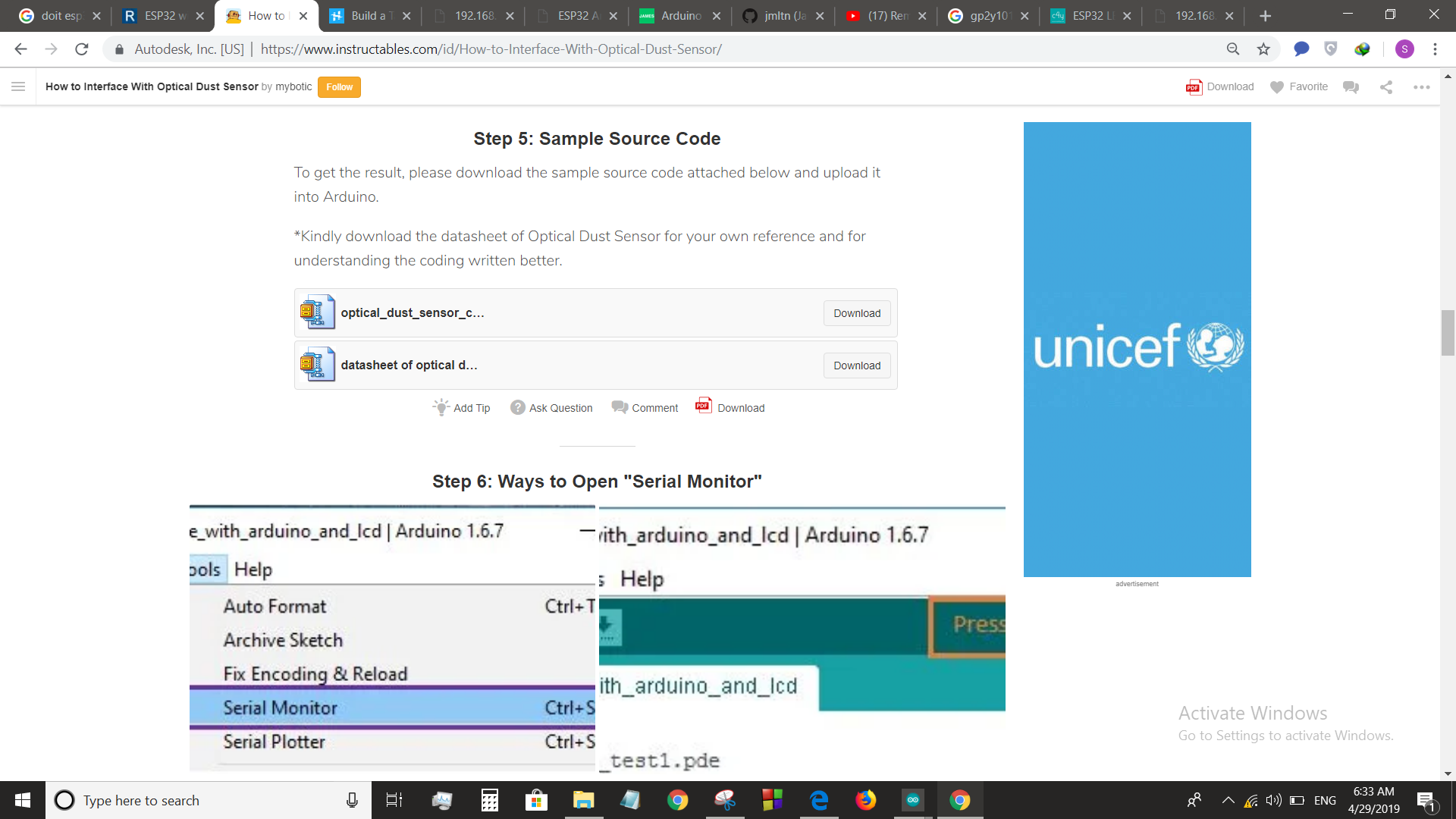


Files of SimpleWiFiServer Pictorial View :

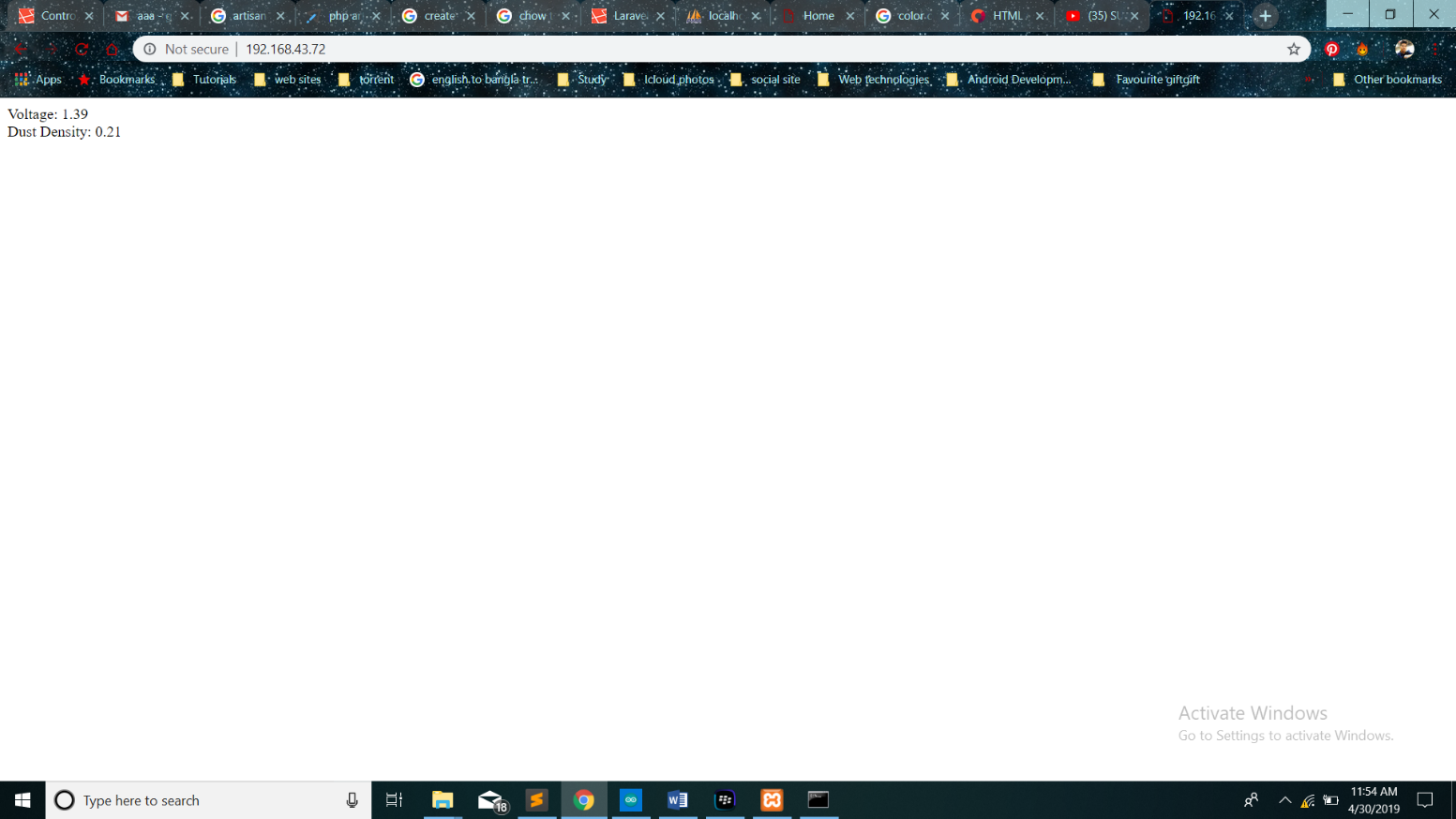


***Download code from the following this Source:***

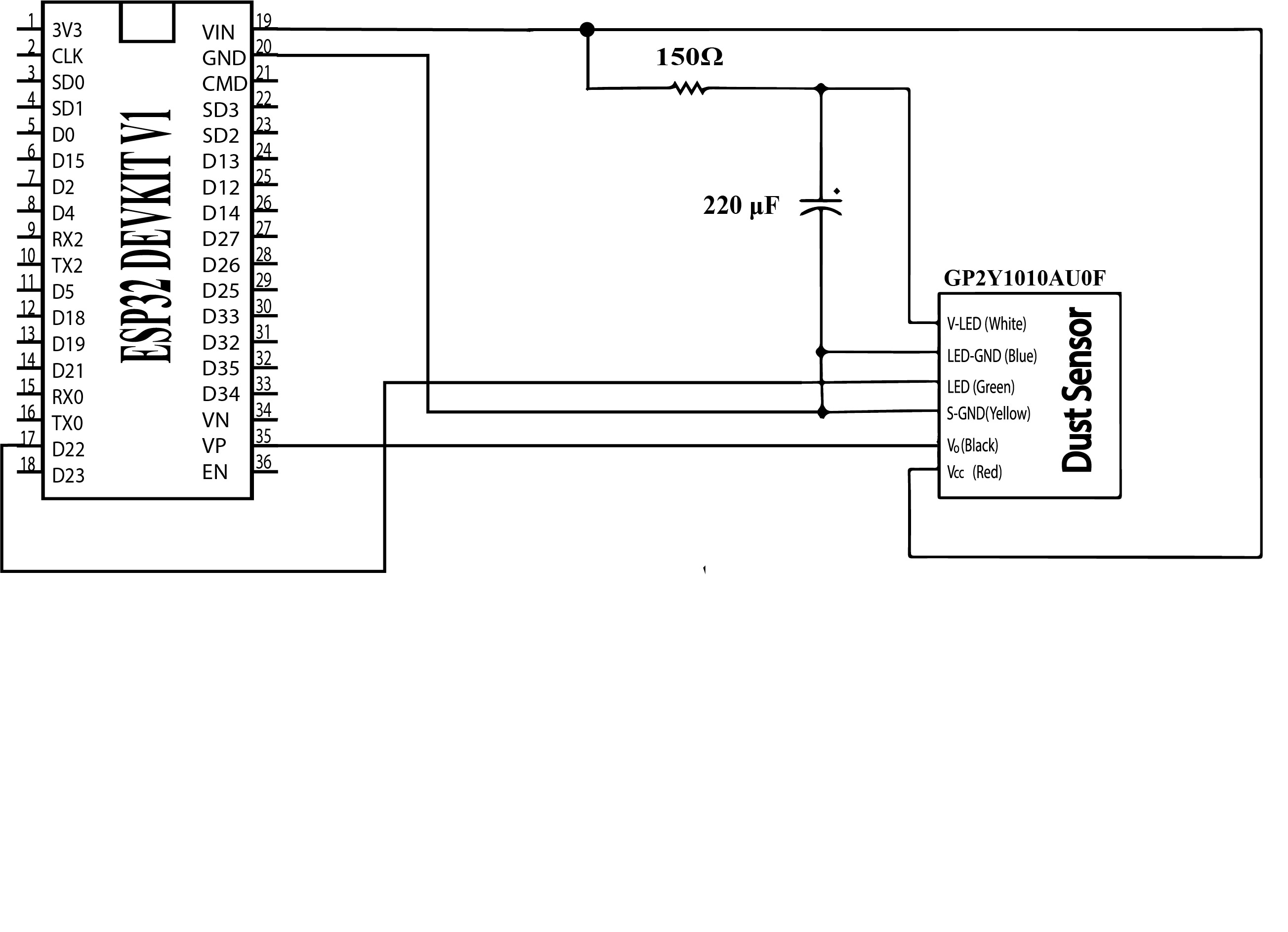
<https://www.instructables.com/id/How-to-Interface-With-Optical-Dust-Sensor/?fbclid=IwAR35Gehzg79LS4Ji4MrleP_qcEYMNlgwPH5QcG7HlIWbNXwer-3U_XKJhJQ>

gg

***Dust Density Sample Results :***



***Circuit Diagram of City Dust Monitoring System :***



***CODE of City Dust Monitoring System :***

#include <WiFi.h>

int measurePin = A0;

int ledPower = 22;

unsigned int samplingTime = 280;

unsigned int deltaTime = 40;

unsigned int sleepTime = 9680;

float voMeasured = 0;

float calcVoltage = 0;

float dustDensity = 0;

const char\* ssid = "gazi";

const char\* password = "gaziiiii";

WiFiServer server(80);

void setup()

{

Serial.begin(115200);

pinMode(ledPower, OUTPUT); // set the LED pin mode

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

int value = 0;

void loop()

{

WiFiClient client = server.available(); // listen for incoming clients

if (client)

{ // if you get a client,

Serial.println("New Client."); // print a message out the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected())

{ // loop while the client's connected

if (client.available())

{ // if there's bytes to read from the client,

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

if (c == '\n')

{ // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0)

{

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

digitalWrite(ledPower, LOW);

delayMicroseconds(samplingTime);

voMeasured = analogRead(measurePin);

delayMicroseconds(deltaTime);

digitalWrite(ledPower, HIGH);

delayMicroseconds(sleepTime);

calcVoltage = voMeasured \* (5.0 / 1024);

dustDensity = 0.17 \* calcVoltage -0.1;

if ( dustDensity < 0)

{

dustDensity = 0.00;

}

//##############

Serial.println(calcVoltage);

Serial.println(dustDensity);

//client.print("Raw Signal Value (0-1023):");

//client.print(voMeasured);

client.print("Voltage: ");

client.print(calcVoltage);

client.println();

client.println();

client.print("<br>");

client.print("Dust Density: ");

client.print(dustDensity);

// the content of the HTTP response follows the header:

//client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 5 on.<br>");

//client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 5 off.<br>");

// The HTTP response ends with another blank line:

client.println();

// break out of the while loop:

break;

} else { // if you got a newline, then clear currentLine:

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

}

// Check to see if the client request was "GET /H" or "GET /L":

if (currentLine.endsWith("GET /H")) {

digitalWrite(22, HIGH); // GET /H turns the LED on

}

if (currentLine.endsWith("GET /L")) {

digitalWrite(22, LOW); // GET /L turns the LED off

}

}

}

// close the connection:

//client.stop();

Serial.println("Client Disconnected.");

}

}