4thsem,c-sec

Group-C14

"IOT BASED PM2.5 AND PM10 AIR MONITORING SYSTEM"

SUBMITTED BY:

M.Jahnavi(1NH18EC729)

Padmavati S Narayanapur (1NH19EC405)

NayanKumar K.C (1NH19EC410)

Rajashekhar Reddy T.V(1NH19EC411)

ABSTRACT

Air pollution is the mixture of gases and solid particles in air. Car emissions, chemicals from factories, industries, dust, pollen and mold pores may be suspended as particles. Effect of air pollution have many bad particles and causes health problems like asthma, cough and lung disorders. In addition pollutants cause global warming, acid rain, and it will disturb the plant growth. Human cannot determine whether the air is good or bad. Hence it is necessary to have an instrument to determine the air quality. This research is purposed to designing an air quality monitoring system by utilizing Esp32 Node MCU which is a successor of esp8266 module.As a result we can measure air quality using smart phone or any other requirement IOT based connected through Esp32 Node MCU Wi-Fi. Therefore the air condition can be monitored every point of time.There is so much of air pollution cases that can actually be changed if we are aware. In other words we can contribute a part of solution instead of pollution.

CHAPTER 1

INTRODUCTION:

Air pollution is the biggest problem of every nation whether it is developed or developing. Health problems of humans or animals or birds are gradually increasing especially in urban areas of developing countries where industrialization and growing number of vehicles leads to release the lot of harmful gases into the environment. Harmful effects of pollution include mild allergic reactions such as irritation of throat, eyes and nose as well as some serious problems like pneumonia, asthma, bronchitis, heart diseases, lung diseases and sometimes it may affect brain. According to a survey due to pollution 50,000 to1,00,000 premature deaths occur in U.S. where in EU number reaches to 3,00,000 and over 30,00,000 worldwide. IOT Based Air Quality Monitoring System monitors the Air quality over a web server using internet.

The Internet of things is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The scope of Internet of Things include connected security systems, thermostats, cars, electronic appliances, lights in household and commercial environments, alarm clocks, speaker systems, vending machines and more.

This is an ESP32 Node MCU-based Air Quality monitoring system to check particulate matter (PM), humidity, temperature, altitude and pressure levels. Various environmental conditions of the place are tested and displayed on the TFT display as well as on Thing Speak IOT (Internet of Things) platform.

PARTICULATE MATTER

High concentrations of dust or PM is a serious health concern. PM2.5 is less than 2.5 microns in diameter, and PM10 is less than 10 microns in diameter. This means, a PM10 report includes PM2.5 as well. Both these particles are much smaller than a human hair, which is about 70 microns in width.

PM10

Operations such as stone crushing, coal grinding, rotary kilning in cement industry and dust on road stirred by moving vehicles can increase PM10 levels. PM10 limit for 24-hour average is 150µg/m3. PM2.5 also can be measured under PM10.

PM2.5

This is a result of fine particles produced from all types of the combustion, including motor vehicles, thermal power plants, residential wood burning, forest fires, agricultural burning and industrial processes. PM2.5 limit for 24-hour average is 35µg/m3. PM2.5 also can measure in PM10.

The Internet of things is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The scope of Internet of Things include connected security systems, thermostats, cars, electronic appliances, lights in household and commercial environments, alarm clocks, speaker systems, vending machines and more.

This is an ESP32 NodeMCU-based Air Quality monitoring system to check particulate matter (PM), humidity, temperature, altitude and pressure levels. Various environmental conditions of the place are tested and displayed on the TFT display as well as on ThingSpeak IoT (Internet of Things) platform.

CHAPTER 2

LITERATURE SURVEY

The drawbacks of the conventional monitoring system is very large instruments are used, heavy weight to carry everywhere and expensiveness. These leads to sparse deployment of monitoring stations. The locations of these stations should be carefully placed because the air pollution in urban areas are very high due to human activities like construction activites and traffic choke points have much worse air quality than average.

IOT Based air quality monitoring system moniters air quality over a web server using internet and will be shown in TFT display. There are many harmful gases present in air like CO2, NH3, smoke and many air particles.

The system will show the air quality on TFT display as well as in a web page so that it can be monitored easily. Temperature and Humidity is detected and monitored in a system through graphs. In this IOT project it can monitor the pollution from anywhere using computer or mobile. This system can be installed from anywhere and can also triggers ome device when pollution go beyond some level, like we can send message alert to the user.

CHAPTER 3

PROPOSED METHODOLOGY

This is ESP32 NODE MCU-based environmental monitoring system for checking particulate matter, humidity, temperature, altitude, pressure levels. This will be displayed through TFT display.

Principle:

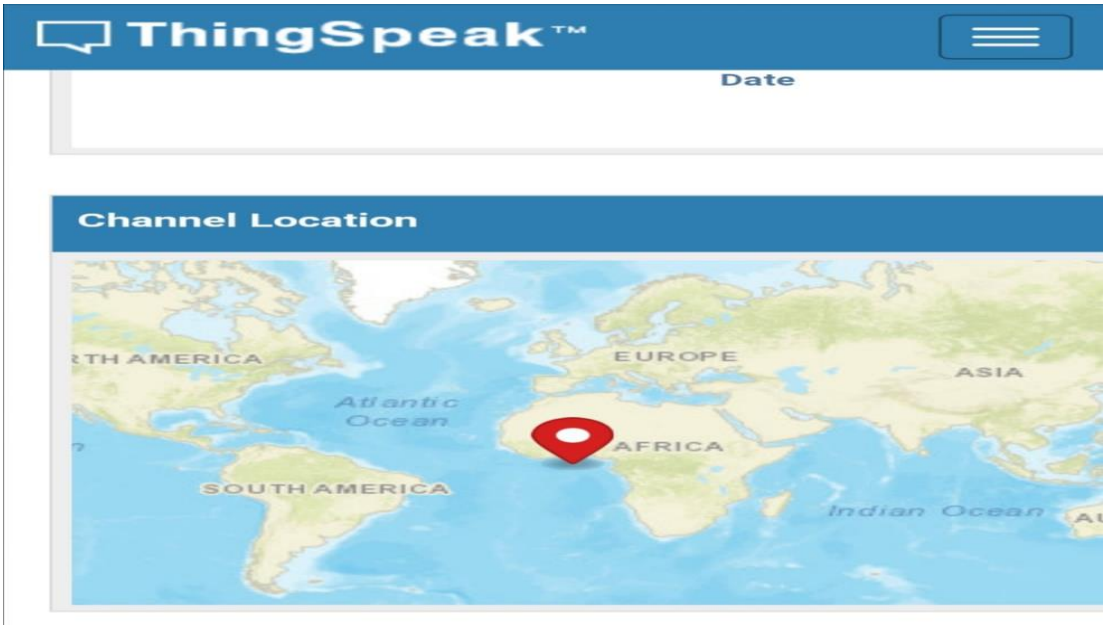
This project is based on the Thing Speak cloud computing. Thing speak means it is an open source IOT application and API to store and retrieve data from things using the HTTP protocol over internet via LAN. It also enables the creation of sensor logging information, Location Tracking applications and social network of things with status updates. This means if you are sending data from sensors to the Thing speak at regular intervals, it will create then store and display data in a trend automatically. It also has Mathematical modeling in the form of MATLAB documentation.

Account and channel set up. Then we can open an account and channel on www.thingspeak.com. For this we need to have valid email account. The site will send a conformation email. Click on the link that have sent in email to validate the email account and create a channel.

From here, we can create as many channels we want. The moment we create a new channels we get Thing speak identities: channel ID, write API key and Read API key. We can

even generate new API keys if we need. Note these down as we require it in source program later. We can Feed up to eight sensor data in a fields per channel, such as PM2.5, PM10, temperature, relative humidity, altitude and pressure.

After creating the channel we will get the ID and private access. Here we get a key and we have to note it down because it will be useful while programming source.

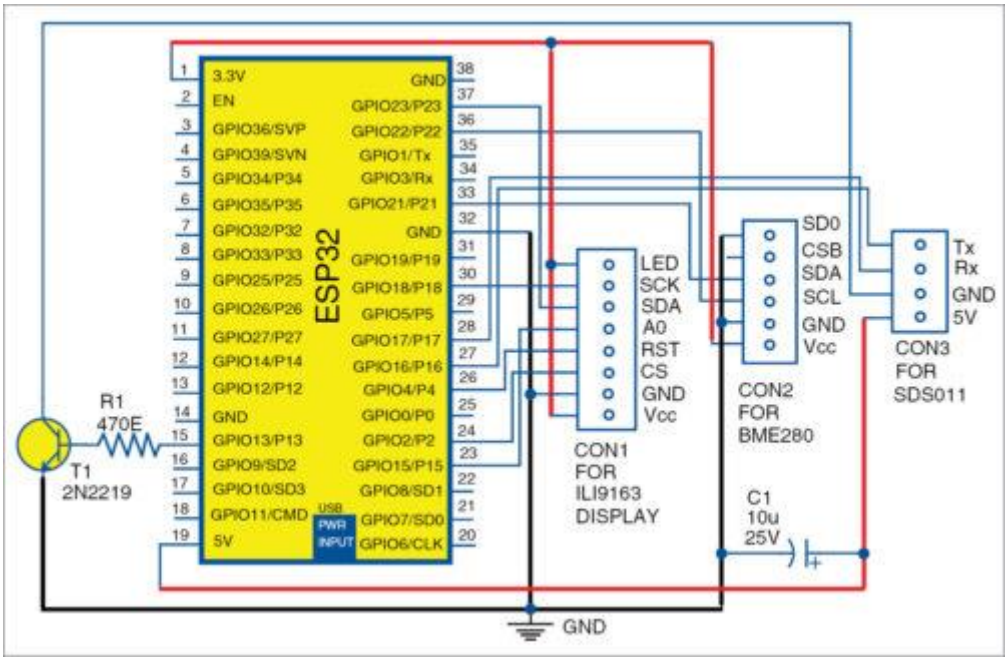


Here we get the channel location after the channel ID.

Eight data per channel. Each channel can take eight data signals from different devices. This means using the Thing Speak API we can upload eight data per channel, which were eventually gathered, logged and put into trend data by Thing Speak, like for eg: https://thingspeak.com/channels/279012.

CIRCUIT WORKING:

The circuit diagram of IOT-based air monitoring system consists of ESP32 NODE MCU, SDS011, BME280, ILI9163 TFT display, a resistor and a transistor 2N2219. ESP32 NODE MCU is the heart of the circuit



3.1 ESP32-based air quality monitoring system

Output will be shown in the form of graph in the system and the dirt particles will be shown in colored TFT display. We can also do this project through PCB .

Connections between the components will be shown in table.

Table 3.1: Connections between ILI9163 and ESP32 NODE MCU

ILI9163	**ESP32 NODE MCU**
LED, Vcc	3.3V
GND	GND
A0	GPIO15
CS	GPIO2
RST	GPIO4
SCK	GPIO18
SDA	GPIO23

ILI9163 is a TFT display which has 8 pins and these are connected to ESP32 NODE MCU which has 38pins as per the connections through Bread board.

Table3.2: Connections between SDS011 AND ESP32 NODE MCU

SDS011	**ESP32 NODE MCU**
5V	5V
GND	GND
TX	GPI016(p16)
RX	GPI017(p17)

SDS011 is a Nova dust sensor which has 4 pins.

Table 3.3: Connections between BME280 and ESP32 NODE MCU

BME280	**ESP32**
Vcc	3.3V
GND,SDO	GND
SCL	GPIO22
SDA	GPIO21
CSB	NC

BME280 has 6 pins are connected to ESP32

This can also be constructed using PCB. The actual PCB layout for air monitoring system is shown below.

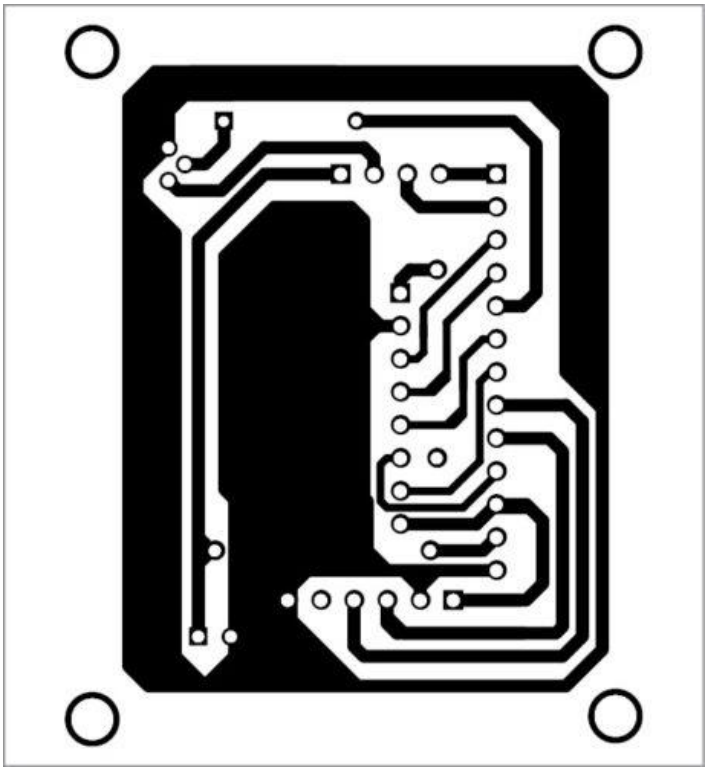


Fig 3.2: PCB layout of air quality monitoring system

A printed circuit boards supports mechanically and electrically connects electrical or electronic components using conductive tracks, pads and other features etched from one or more sheet layers or copper laminated onto the between sheet layers of the non-conductive substrate. Components are generally soldered onto the PCB to both electrically connected and then mechanically fasten them.

Mount various modules on PCB, we have to print the board and then to be etched using Ferric chloride, Hydrogen peroxide, sulfuric acid and hot water and then we have to etch it until the layer of the copper is removed. Copper allows the electricity to flow and then we have drill the holes and need to place the components on the board using the holes and then to be soldered then the printed circuit board is ready for testing. Components layout for the Printed Circuit board is shown below as per the figure we have to place the components.

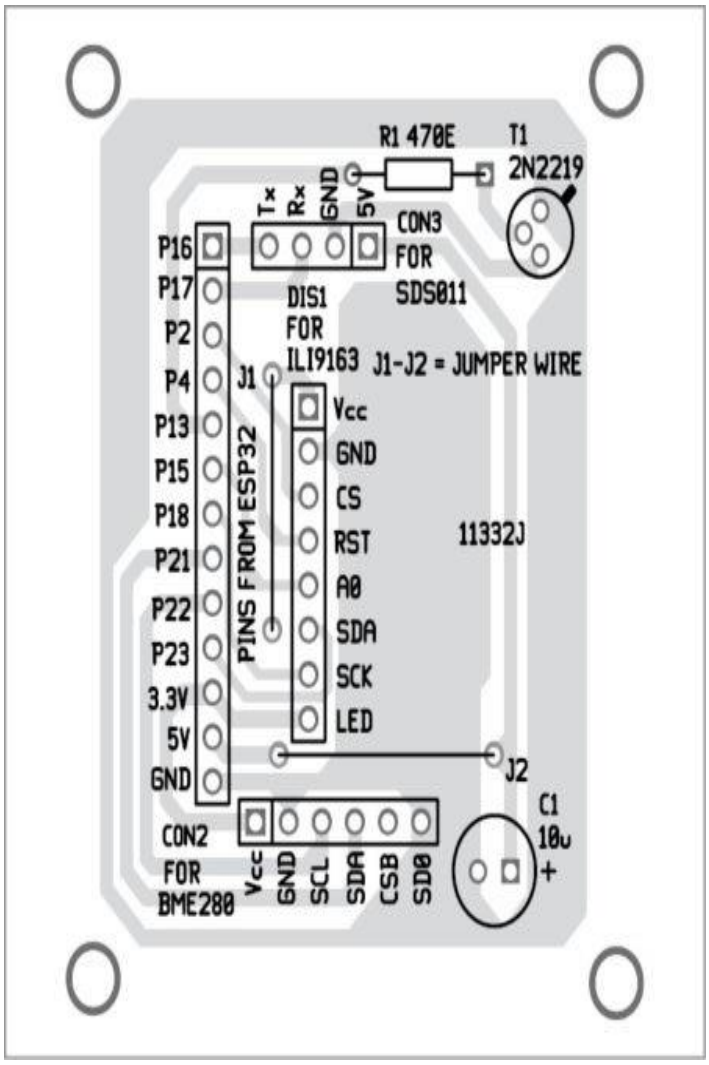


Fig 3.3: Components layout for the PCB

After soldering Switch on the power supply to ESP32 NODE MCU. If everything placement of components is correct, we will see the readings on TFT display and the same values will be appeared on that thing speak website on that which we have created a channel like graphs as shown below.

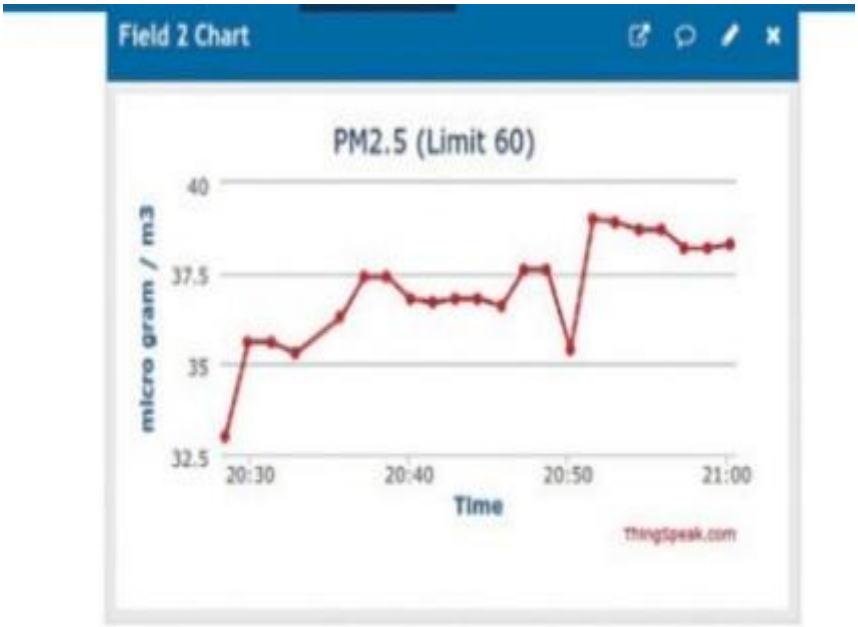

Field 2 Chart
PM2.5 (Limit 60)
micro gram / m3
40
37.5
35
32.5
20:30
20:40
20:50
21:00
Time
ThingSpeak.com

Fig3.4

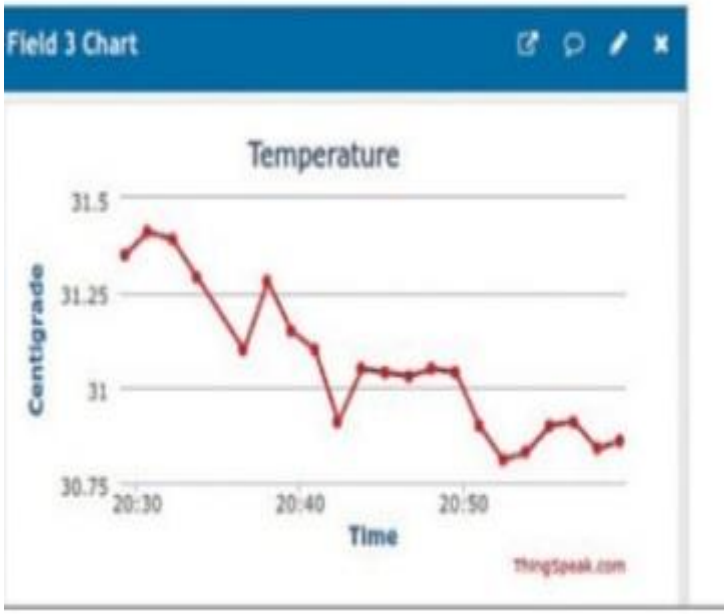

Field 3 Chart
Temperature
Centigrade
31.5
31.25
31
30.75
20:30
20:40
20:50
Time
ThingSpeak.com

Fig 3.5

Fig 3.6

Fig 3.7

We can also use it from our smart phones. This graph shows us the levels of PM2.5, PM10, temperature and pressure in our surroundings through Thing speak channel that we have previously created.

CHAPTER 4

PROJECT DESCRIPTION

4.1 HARDWARE DESCRIPTION

- 2N2219 NPN switching Transistor
- 470-ohm resistor
- 10µF, 25V electrolytic capacitor
- Jumper wires(8 pin connector,6pin connector,4pin connector)
- Bread board
- ESP32 NODE MCU
- ILI9163,3.6cm(1.44 inch colored TFT display)
- SDSO11 Nova dust sensor
- BME280 module

2N2219 NPN switching Transistor:

2N2219 transistor is very much similar to the 2N2222. But it comes in the metal can package and it can operate on voltage slightly higher than 2N2222 could handle. It is just another small signal transistor which is used commonly in switching and amplifying circuits.

This transistor used as the switch for the SDS011 Nova Dust sensor. Its connector is connected to the GND pin of the SDS011 and its emitter is connected to the circuit ground. When pin 15 of the ESP32 is logic high, T1 conducts and SDS011 gets ground connection through T1, otherwise SDS011 will be off.

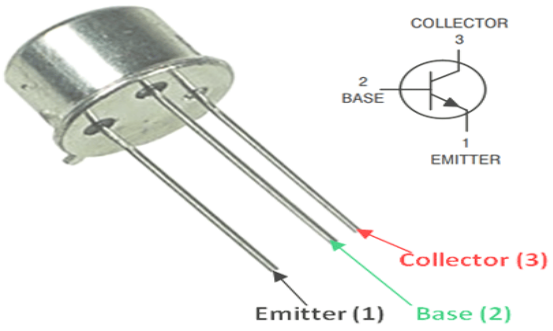


Fig 4.1

470-ohm resistor:

470-ohm resistor as per the color coding 1st digit is 4, then 1st color is yellow. The last bands in 4or 5 bands are the indicator of tolerance value of resistor. Gold indicates 5%tolerance.

The 1/4W resistance metal film resistors are good for DIY projects. With these resistors we can do it with Ardunio tested works perfectly. The resistor is very good for DIY projects.

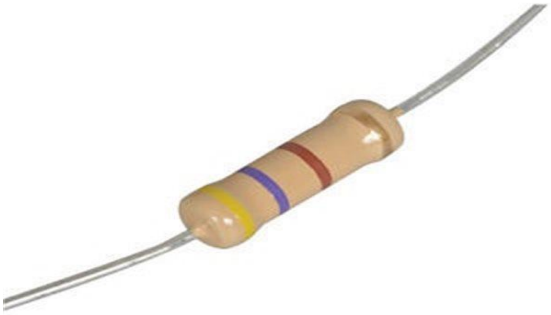


Fig 4.2

10µF, 25V electrolytic capacitor:

10µF is the capacitance and its working voltage is 25V. It is made up of the two metallic plates. With a dielectric material in between these plates when we apply a voltage over the two plates an electric field will be created.

The impact of a capacitor is known as capacitance. While some capacitance exists between any two electrical conductors in proximity in the circuit, a capacitor is a part designed to add capacitance to a circuit. The capacitor was originally called a condenser or condensator. This name and its cognates are nevertheless widely used in lots of languages, however rarely in English, one incredible exception being condenser microphones, also called capacitor microphones.

The bodily shape and production of sensible capacitors vary extensively and plenty of varieties of capacitor are in common use. Most capacitors incorporate as a minimum electric conductors regularly inside the form of metallic plates or surfaces separated by means of a dielectric medium. A conductor can be a foil, thin movie, sintered bead of steel, or an electrolyte.

The non conducting dielectric acts to boom the capacitor's charge ability. Materials usually used as dielectrics include glass, ceramic, plastic movie, paper, mica, air, and oxide layers. Capacitors are extensively used as components of electrical circuits in lots of commonplace electric devices. Unlike a resistor, a super capacitor does not expend electricity, despite the fact that actual-life capacitors do expend a small amount. When an electric ability, a voltage, is implemented across the terminals of a capacitor, for instance while a capacitor is hooked up across a battery, an electric discipline develops throughout the dielectric, causing a net high-quality price to collect on one plate and net terrible charge to gather on the alternative plate. No modern without a doubt flows via the dielectric. However, there's a drift of charge thru the source circuit. If the circumstance is maintained sufficiently long, the cutting-edge thru the source circuit ceases. If a time-varying voltage is carried out throughout the leads of the capacitor, the source reports an ongoing cutting-edge because of the charging and discharging cycles of the capacitor.

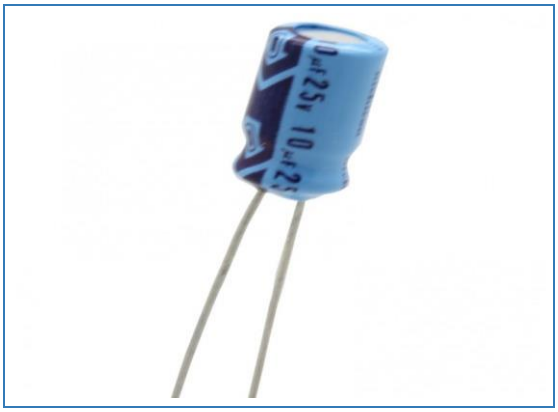


Fig 4.3

Jumper wires:

It refers to conducting wires that establish an electrical connection between two points. We use jumper wires to modify to diagnose problems in circuit.

Here we use 8 pin connector for TFT display, 6 pin connector for BME280, 4 pin connector for the SDS011 Nova dust sensor.

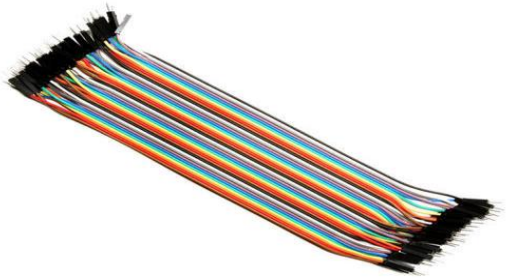


Fig 4.4

Bread board:

It is a construction base for prototyping of the electronics. It is a rectangular board with many mounting holes and it is used for creating electrical connections between electronic components and single board computers or microcontrollers such as arduino and raspberry pi.

Fig 4.5

ESP32 NODE MCU:

It is a low power micro controller with the integrated Wi-Fi and dual mode Bluetooth and it is successor of ESP8266 MCU. Arduino IDE is used to upload the code to ESP32 NODE MCU (ESP-Wroom-32version 1.1 is used in this project. It is the heart of the circuit.

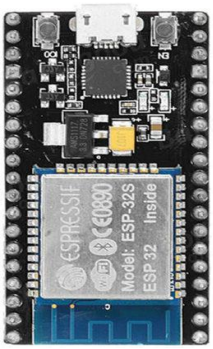


Fig. 4.6: ESP32 Node MCU

ILI9163 TFT display:

Thin film Transistor, liquid-crystal display is a variant of a liquid-crystal display which uses for the thin film transistor technology to improve image quality.

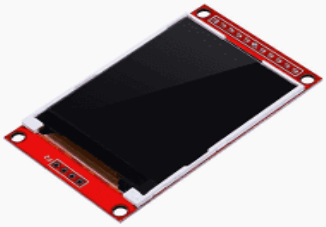


Fig 4.7

SDS011 Nova dust sensor:

It is one of the best particulate sensors in terms of size, accuracy. Operating voltage is 5V. We can also use two lithium ion batteries along with the small linear regulator like 7805 to step down to 5V. Alternatively we can also use tiny convertor that can boost the voltage from 1.5V to 5V.we can even use one lithium polymer battery along with converter making it a portable device.

Fig 4.8

BME280 module:

BME280 module is used for reading pressure and relative humidity. It is also used in mobile applications.

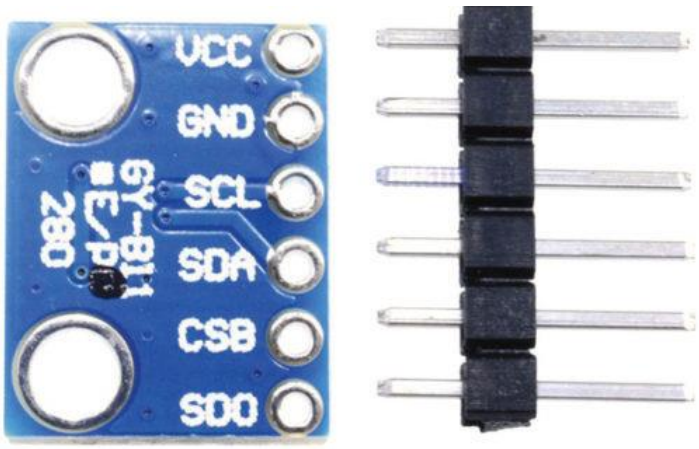


Fig 4.9

4.2 SOFTWARE DESCRIPTION:

Software description of this project is programmed in Arduino programming language. PM2.5, and PM10 have relation with the relative humidity(RH). So, RH correction is included in the program code to get the readings.

SDS011 on ESP32: ESP32 works on the Arduino IDE. Since Arduino is an advanced technology, there is a ready-made library for all the sensors. So for SDS011 we find a

readymade library as it is a dust sensor available that uses Software serial library and works out-of-the-box for Arduino Uno.

ESP32 doesn't work with software reserial. It has three UART ports and uses Hardware serial for connecting UART devices. Existing library and the command are used to get the PM data from the sensors. These commands were placed in processdata.h file, which should be kept in the same folder as the arduino sketch and compiled together with the main sketch.

Sleep mode of the SDS011: As per the specifications, SDS011 has a sleep current of 2mA. The sensors works on the sleep mode frequently. Sleep commands are also placed in processdata.h file for us to try. However , GPIO13(pin15) is used for switching on the npn Transistor(2N2219) to keep the SDS011 on during the active stage, before it goes into the power off mode.

Uploading the code into ESP32:

Now everything is set and then done by us in the circuit. Assemble the devices on the PCB or general-purpose Veroboard. Connect the USB power supply to ESP32

Now run Arduino IDE and add the available for ESP32 and other relevant libraries for sensors. Open the Arduino code and provide the Wi-fi ID and password towards beginning of the sketch. Replace the Thing speak channel number and with API key that we had already noted down.

Select the ESP32Dev module from the pull-down menu under the tools--→Board option. Next, select the correct USB COM port from the pull-down menu under the Tools-→port option. Press Upload, if everything is completed, the program will start communicating with ESP32 board and then upload the code into the ESP32 NODE MCU.

After the few seconds we can see PM2.5, PM10 and other required data shown in the TFT display. we can also use smart phone

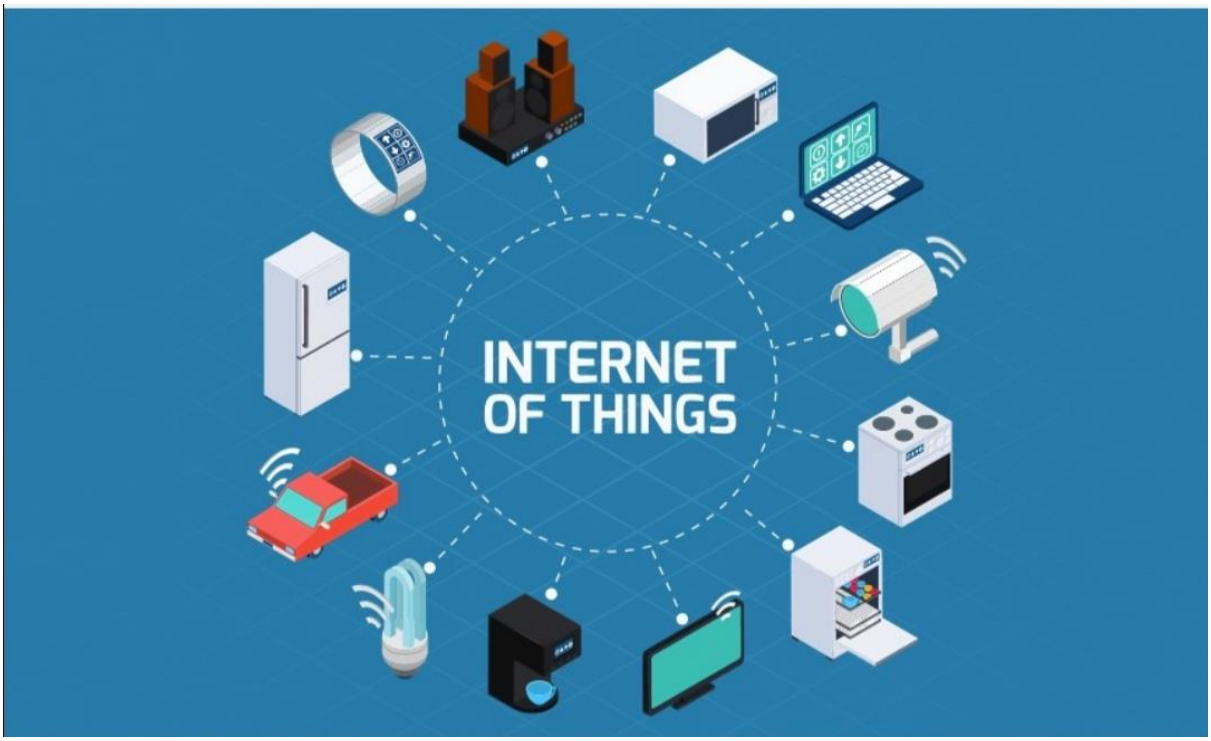


Fig 4.10

Internet of things is the system interrelated computing devices is which we are mainly using in this project.

CHAPTER 5

RESULTS AND DISCUSSION

The proposed system will monitor the air pollutants from different location and then uploads the data to data server for the further processing and the analysis.

The online application used to analyze the air quality data got from the sensors in this proposed system was ”Thing speak”. Thing speak is an open source internet of things application programming interface used to store the data and retrieve the data from interconnected things using hypertext protocol over the internet or via a local area network. It is shown through graphs in Thing speak cloud.

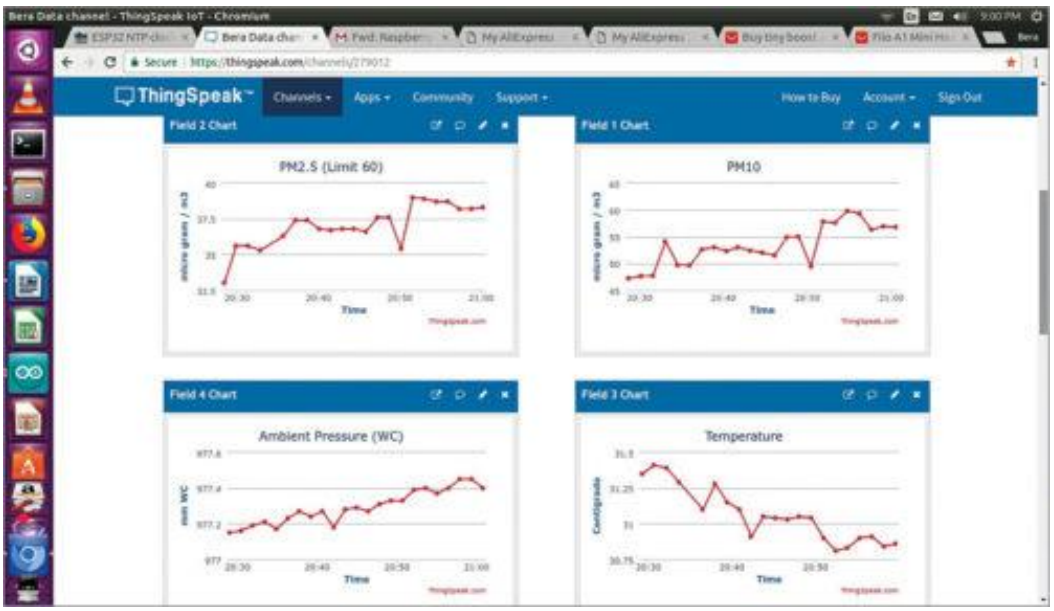


Fig 5.1

Graphs of the PM2.5, PM10, Ambient pressure, temperature, are shown in Thing speak. We get these graphs in the system.

These graphs are registered by the dust sensor in which the air particles are stored and shown through TFT display of PM2.5 and PM10 values are shown because of the dust sensor.

Pressure and temperature are shown due to BME280 module as it is used for showing the reading.

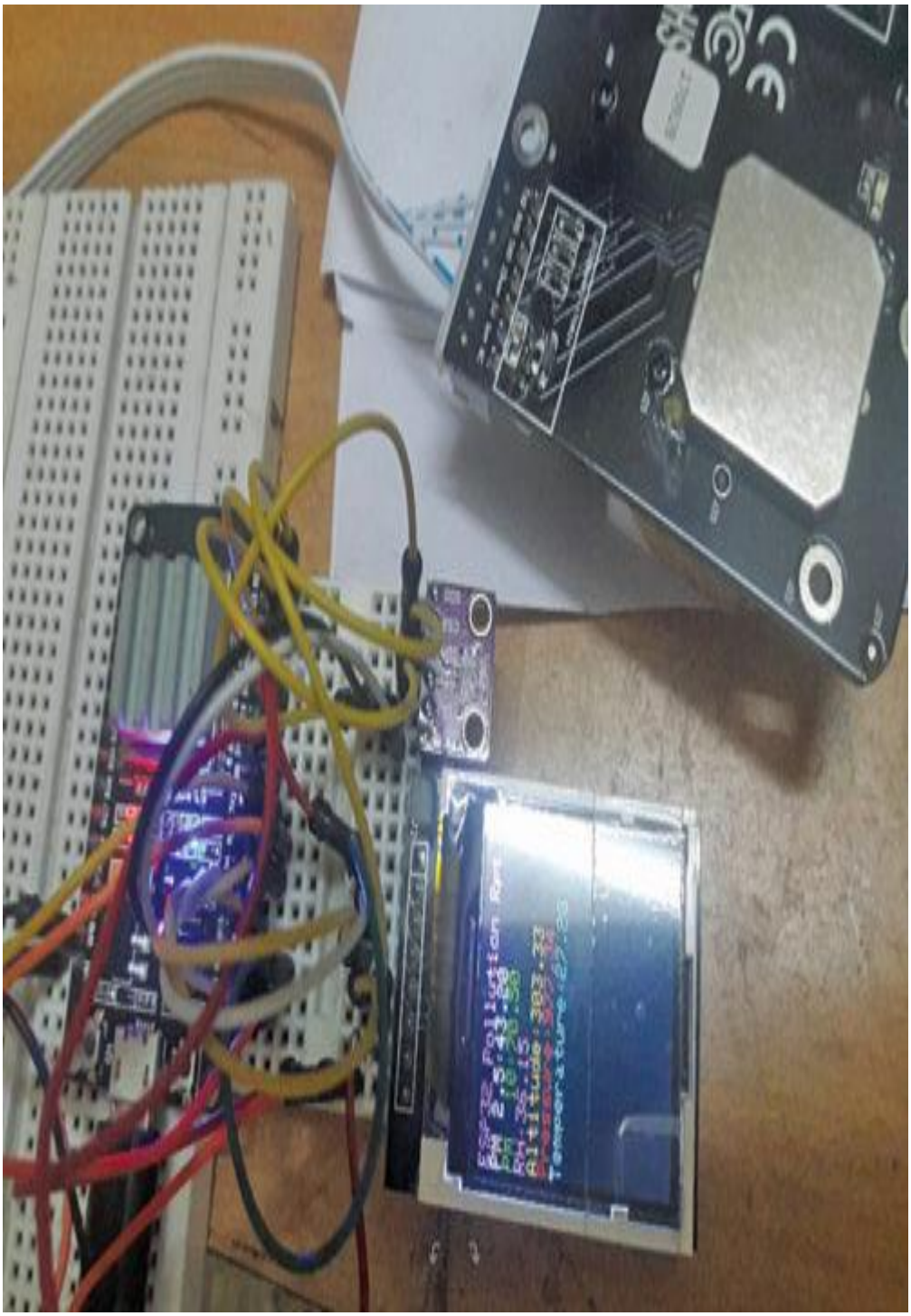


Fig5.2

Figure shows us the connection and reading of the surrounding atmosphere.

“Thing speak” is configured to receive the data from the remote system. The data was analyzed and published in the form of graphs and values in the TFT display.

The visual representation of the graphs on Thing speak was displayed in the colored TFT display. The Status of air quality can be displayed at any time and update in TFT display. The monitoring device will deliver real-time measurements of air quality.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

CONCLUSION:

IOT based Air pollution monitoring system for the smart cities. A few sensors are to monitor the temperature, absolute pressure, PM2.5, PM10 levels in the atmosphere. The research proposed an air monitoring system that constantly keeps track of air quality in an area and displays through the TFT display screen. It also sends the data measured to the "Thing speak" platform. The system helps to create the awareness of the quality of air that one breaths daily. This monitoring device deliver real time measurements of air quality.

Humans around the world are reason for the air pollution and this may lead to the death of all living beings. If we know the pollution values and particle present in the atmosphere so, that some of the people get the awareness of this monitoring system and due to readings they avoid to go to that area.

Air monitoring system shows us the reading of the air particles which affect the humans. We can modify the code and can use other sensors to know the other harmful gases. So that many people will be care and aware of these gases.

So, we conclude that Air quality improves in increase in temperature. Air quality deteriorates with increase in humidity. Air quality improves with increase in altitude. However pm values are significantly lower indoors even in lower altitudes.

FUTURE SCOPE:

- We can develop an app using these sensors in which people can know the particles contain in their surroundings.
- Users get most accurate reading through that app without carrying air quality sensors with them.
- We can also include calculating temperature and humidity.
- We can also develop it to calculate readings of organic and inorganic gases present in atmosphere**.**

REFERENCES

- Somnath Bera is an avid user of open source software. Professionally , he is a thermal power expert and work as additional general manger at NTPC Ltd.
- Arun Raj V., Priya R.M.P., and Meenakshi, V., "Air Pollution Monitoring In Urban
- Area," International Journal of Electronics and Communication Engineering , 2017 . Matthews V.O., Uzairue S.I., Noma-Osaghae E., and Nwukor F., Design and Simulation of a Smart Automated Traffic System in a Campus Community.", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5162, 5 (8), 2018, pp. 492-497, Available at :http://www.jetir.org/papers/JETIR1807794.pdf.
- Priyanka, V., "Review: Air Quality Monitoring System," International Journal of Advanced Research in Computer and Communication Engineering, 5(6), 2016.

APPENDIX

CODE

```
 #include <SoftwareSerial.h>
SoftwareSerial pmsSerial(2, 3);


void setup() {
  // our debugging output
  Serial.begin(115200);


  // sensor baud rate is 9600
  pmsSerial.begin(9600);
}


struct pms5003data {
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um, particles_50um, particles_100um;
  uint16_t unused;
  uint16_t checksum;
};
```

```
struct pms5003data data;
   void loop() {
  if (readPMSdata(&pmsSerial)) {
   // reading data was successful!
   Serial.println();
   Serial.println("---------------------------------------");
   Serial.println("Concentration Units (standard)");
   Serial.print("PM 1.0: "); Serial.p…
[0:19 pm, 06/05/2020] Jahnavi Reddy: Units (standard)");
   Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
   Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
   Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
   Serial.println("---------------------------------------");
   Serial.println("Concentration Units (environmental)");
   Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
   Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
   Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
   Serial.println("---------------------------------------");
   Serial.print("Particles > 0.3um / 0.1L air:"); Serial.println(data.particles_03um);
   Serial.print("Particles > 0.5um / 0.1L air:"); Serial.println(data.particles_05um)…
[0:19 pm, 06/05/2020] Jahnavi Reddy: if (! s->available()) {
```

```
  return false;
}

// Read a byte at a time until we get to the special '0x42' start-byte
if (s->peek() != 0x42) {
  s->read();
  return false;
}

// Now read all 32 bytes
if (s->available() < 32) {
  return false;
}
  uint8_t buffer[32];
uint16_t sum = 0;
s->readBytes(buffer, 32);
// get checksum ready
for (uint8_t i=0; i<30; i++) {
  sum += buffer[i];
}
/* debugging
for (uint8_t i=2; i<32; i++) {
```

```
   Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
  }
  Serial.println();
  */
   // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i=0; i<15; i++) {
   buffer_u16[i] = buffer[2 + i*2 + 1];
   buffer_u16[i] += (buffer[2 + i*
   buffer_u16[i] += (buffer[2 + i*2] << 8);
  }
  // put it into a nice struct :)
  memcpy((void *)&data, (void *)buffer_u16, 30);
if (sum != data.checksum) {
   Serial.println("Checksum failure");
   return false;
  }
  // success!
  return true;
}
```

Get reading from Nova PM Sensor SDS011

```
(dust sensor, air quality sensor, PM10, PM2,5)

Designed to run from cron and append CSV file.

Script tested using Python3.4 on Ubuntu 14.04.

TODO: choose by dev name using udev, add dev id info, python package pyudev

   udevadm info -q property --export /dev/ttyUSB0

import os

import csv

import io

import logging

import datetime

import argparse

try:

   import serial

except ImportError:

   print('Python serial library required, on Ubuntu/Debian:')

   print('    apt-get install python-serial python3-serial')

   raise


LOG_FORMAT = '%(asctime)-15s %(levelname)-8s %(message)s'

def append_csv(filename, field_names, row_dict):

   """

   Create or append one row of data to csv file.
```

```
    """
    file_exists =…os.path.isfile(filename)
    with io.open(filename, 'a', encoding='utf-8') as csvfile:
        writer = csv.DictWriter(csvfile,
                                delimiter=',',
                                lineterminator='\n',
                                fieldnames=field_names)
        if not file_exists:
            writer.writeheader()
        writer.writerow(row_dict)
def read_nova_dust_sensor(device='/dev/ttyUSB0'):
    dev = serial.Serial(device, 9600)
 if not dev.isOpen():
        dev.open()
 msg = dev.read(10)
    assert msg[0] == ord(b'\xaa')
    assert msg[1] == ord(b'\xc0')
    assert msg[9] == ord(b'\xab')
    pm2.5 = (msg[3] * 256 + msg[2]) / 10.0
    pm10 = (msg[5] * 256 + msg[4]) / 10.0
    checksum = sum(v for v in msg[2:8]) % 256
    assert checksum == msg[8]
```

```
    return {'PM10': pm10, 'PM2_5': pm25}
def main():
    logging.basicConfig(format=LOG_FORMAT, level=logging.INFO)

    parser = argparse.ArgumentParser(description='Read data from Nova PM sensor.')
    parser.add_argument('--device', default='/dev/ttyUSB0',
                        help='Device file of connected by USB RS232 Nova PM sensor')
    parser.add_argument('--csv', default=None,
                        help='Append results to csv, you can use year, month, day in format')
    args = parser.parse_args()
  data = read_nova_dust_sensor(args.device)
    logging.info('PM10=% 3.1f ug/m^3 PM2.5=% 3.1f ug/m^3', data['PM10'], data['PM2_5'])
  if args.csv:
        field_list = ['date', 'PM10', 'PM2_5']
        today = datetime.datetime.today()
        data['date'] = today.strftime('%Y-%m-%d %H:%M:%S')
        csv_file = args.csv % {'year': today.year,
                               'month': '%02d' % today.month,
                               'day': '%02d' % today.day,
                               }
        append_csv(csv_file, field_list, data)
```

```
if _name_ == '_main_':
    main()
```