CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Air pollution is the contamination of air due to the presence of substances in the atmosphere that are harmful to the health of humans and other living beings, or cause damage to the climate or to materials. There are many different types of air pollutants, such as gases (including ammonia, carbon monoxide, sulphur dioxide, nitrous oxides, methane, carbon dioxide and chlorofluorocarbons), particulates (both organic and inorganic), and biological molecules. Air pollution can cause diseases, allergies, and even death to humans; it can also cause harm to other living organisms such as animals and food crops, and may damage the natural environment (for example, climate change, ozone depletion or habitat degradation) or built environment (for example, acid rain). Air pollution can be caused by both human activities and natural phenomena.

Air pollution is a significant risk factor for a number of pollution-related diseases, including respiratory infections, heart disease, COPD, stroke and lung cancer. Growing evidence suggests that air pollution exposure may be associated with reduced IQ scores, impaired cognition, increased risk for psychiatric disorders such as depression and detrimental perinatal health. The human health effects of poor air quality are far reaching, but principally affect the body's respiratory system and the cardiovascular system. Individual reactions to air pollutants depend on the type of pollutant a person is exposed to, the degree of exposure, and the individual's health status and genetics.

Outdoor air pollution attributable to fossil fuel use alone causes ~3.61 million deaths annually, making it one of the top contributors to human death, with anthropogenic ozone and $PM_{2.5}$ causing ~2.1 million. Overall, air pollution causes the deaths of around 7 million people worldwide each year, or a global mean loss of life expectancy (LLE) of 2.9 years, and is the world's largest single environmental health risk, which has not shown significant progress since at least 2015. Indoor air pollution and poor urban air quality are listed as two of the world's worst toxic pollution problems in the 2008 Blacksmith Institute World's Worst Polluted Places report. The scope of the air pollution crisis is large: 90% of the world's population breathes dirty air to some degree. Although the health consequences are extensive, the way the problem is handled is considered largely haphazard or neglected.

Productivity losses and degraded quality of life caused by air pollution are estimated to

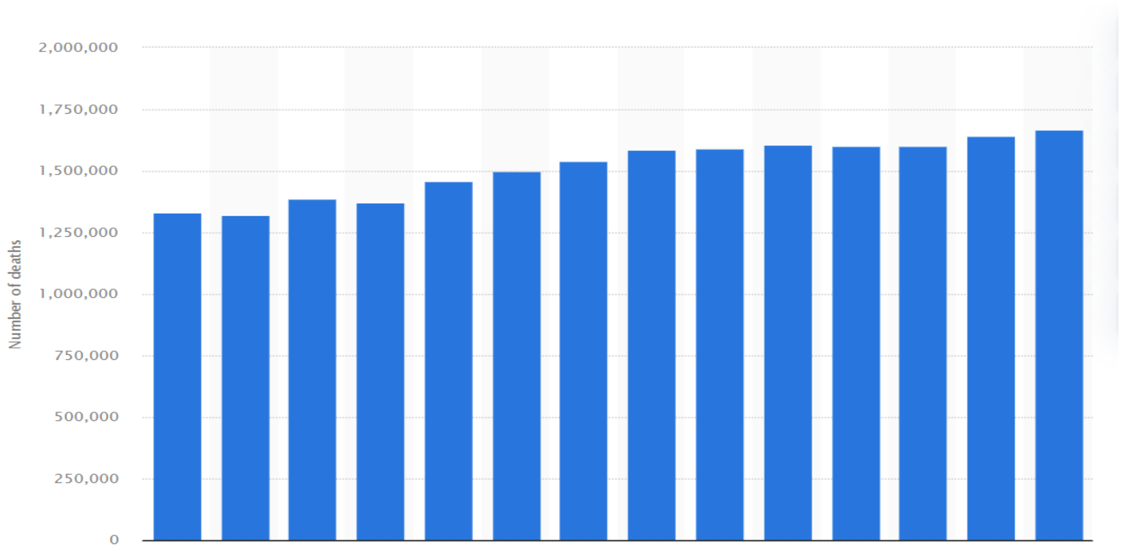


Figure 1.1 Number of deaths attributable to air pollution across India from 1990

cost the world economy $5 trillion per year but, along with health and mortality impacts, are an externality to the contemporary economic system and most human activity, albeit sometimes being moderately regulated and monitored. Various pollution control technologies and strategies are available to reduce air pollution. Several international and national legislation and regulation have been developed to limit the negative effects of air pollution. Local rules, when properly executed, have resulted in significant advances in public health. Some of these efforts have been successful at the international level, such as the Montreal Protocol, which reduced the release of harmful ozone depleting chemicals, and the 1985 Helsinki Protocol, which reduced sulphur emissions, while others, such as international action on climate change, have been less successful.

A **Vehicle Tracking System** combines the use of automatic vehicle location in individual vehicles with software that collects these fleet data for a comprehensive picture of vehicle locations. Modern vehicle tracking systems commonly use GPS or GLONASS technology for locating the vehicle, but other types of automatic vehicle location technology can also be used. Vehicle information can be viewed on electronic maps via the Internet or specialised software. Urban public transit authorities are an increasingly common user of vehicle tracking systems, particularly in large cities.

Several types of vehicles tracking devices exist. Typically, they are classified as "passive" and "active". "Passive" devices store GPS location, speed, heading and sometimes a trigger event such as key on/off, door open/closed. Once the vehicle returns to a predetermined point, the device is removed and the data downloaded to a computer for evaluation. Passive systems include auto download type that transfer data via wireless download. "Active" devices also collect the same information but usually transmit the data in near-real-time via cellular or satellite networks to a computer or data centre for evaluation.

Many modern vehicle tracking devices combine both active and passive tracking abilities: when a cellular network is available and a tracking device is connected it transmits data to a server; when a network is not available the device stores data in internal memory and will transmit stored data to the server later when the network becomes available again.

Historically, vehicle tracking has been accomplished by installing a box into the vehicle, either self-powered with a battery or wired into the vehicle's power system. For detailed vehicle locating and tracking this is still the predominant method; however, many companies are increasingly interested in the emerging cell phone technologies that provide tracking of multiple entities, such as both a salesperson and their vehicle. These systems also offer tracking of calls, texts, web use and generally provide a wider range of options.

1.2 INTRODUCTION TO EMBEDDED SYSTEMS

The microprocessor-based system is built for controlling a function or range of functions and is not designed to be programmed by the end user in the same way a PC is defined as an embedded system. An embedded system is designed to perform one particular task albeit with different choices and options.

Embedded systems contain processing cores that are either microcontrollers or digital signal processors. Microcontrollers are generally known as "chip", which may itself be packaged with other microcontrollers in a hybrid system of Application Specific Integrated Circuit (ASIC). In general, input always comes from a detector or sensors in more specific word and meanwhile the output goes to the activator which may start or stop the operation of the machine or the operating system.

An embedded system is a combination of both hardware and software, each embedded system is unique and the hardware is highly specialized in the application domain. Hardware consists of processors, microcontroller, IR sensors etc. On the other hand, Software is just like a brain of the whole embedded system as this consists of the programming languages used which makes hardware work. As a result, embedded systems programming can be a widely varying experience.

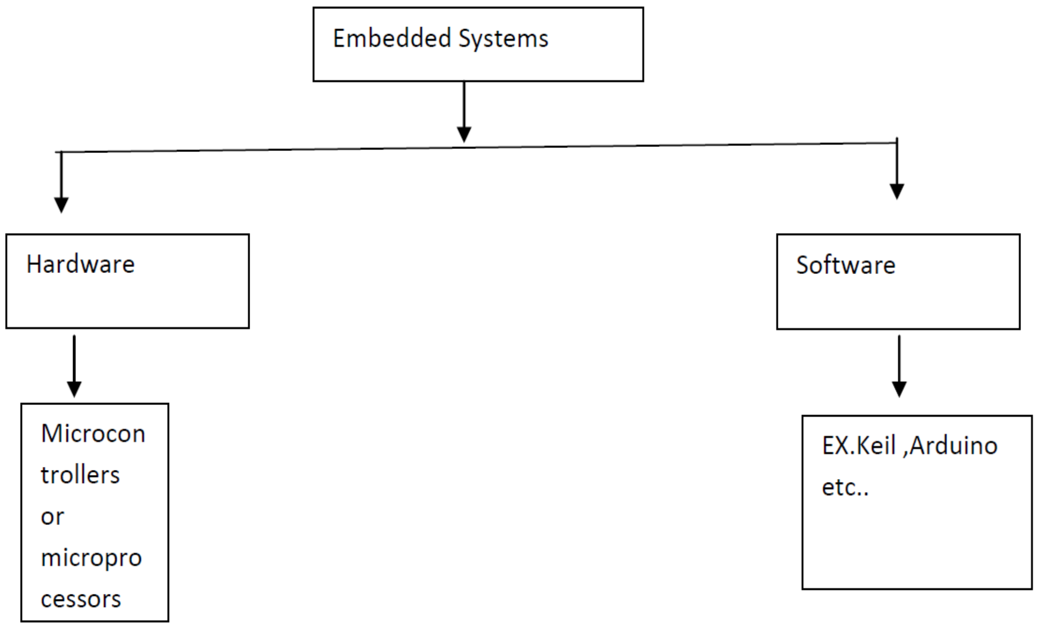


Figure 1.2 Block diagram of embedded system

An embedded system is combination of computer hardware and software, either fixed incapability or programmable, that is specifically designed for particular kind of application device. Industrial machines, automobiles, medical equipment, vending machines and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface, and embedded systems programming id specialized occupation.

Figure1.2 illustrate the Block diagram of Embedded System (ES consists of hardware and software part which again consists of programming language and physical peripherals respectively). On the other hand, the microcontroller is a single silicon chip consisting of all input, output and peripherals on it. A single microcontroller has the following features:

1. Arithmetic and logic unit
2. Memory for storing program
3. EEPROM for non-volatile and special function registers
4. Input/output ports
5. Analog to digital converter
6. Circuits
7. Serial communication ports

1.3 APPLICATIONS OF EMBEDDED SYSTEM

We are living in the embedded world. You are surrounded with many embedded products and your daily life largely depend on the proper functioning's of these gadgets, television, radio, CD layer of your living room, washing machines or microwave oven in your kitchen, card readers, access controllers, palm devices of your work space enable to do many of your tasks very effectively. Apart from all these, many controllers embedded in your car take care of your car operation between the bumper and most of the times tend to ignore all these controllers.

In recent days you are showered with variety of information about these embedded controllers in many places. All kind of magazines and journals regularly dish out details about latest technologies, new devices: fast applications which make you believe that your basic survival is controlled by these embedded products. Now you can agree to that fact these embedded products have successfully invaded into our world. You must be wandering about these embedded controllers or systems.

The computer you use to compose your mails, or create a document or analyze the database is known as standard desktop computer. These desktop computers are manufactured to serve many purposes and applications.

1.4 OBJECTIVE

The difficulties faced by people when a bus fails or one fails to board the bus on time lead to confusion among the people. Also, air pollution is one of the major environmental issues that cannot be ignored. Inhaling pollutants for a long time causes damage to human health.

A bus-tracking app forms part of the entire system of a real time bus management and serves as a convenient access point when downloaded onto the phone of the users. The bus-tracking app provides parents and authorities a map view of the current position of the bus with information of where the bus is currently located.

To begin with, a bus-tracking app uses modern GPS technology. Thus, a GPS device installed within a bus can transmit its location in real time, which can be displayed on a map.

As we could imagine, a situation where students travel on a daily basis forms an important segment of bus travel. When there is a sudden repair in the bus or the bus has diverged from the current route, it creates a dilemma for the passengers on the bus. They get confused whether to board another bus or to wait until the current bus gets repaired.

As mentioned above, the buses are installed with a GPS sensor that is connected to an app. The College management/College authorities (Transport In-charge) can track it from their respective places, meanwhile parents can track it on their mobile phones.

If there is any unplanned delay or a repair in the bus, this system also conveniently facilitates the driver in communicating directly with the college management and with the parents. Within the bus tracking application, we are adding another feature to monitor air pollution.

Nowadays, air pollution is one of the major risk factors which harms the environment. Several studies have shown the detrimental effects of air pollution on human health and wellbeing among all the air pollutants, particulate matter (PM) pollution is one of the particular concerns to worry about.

Particulate matter is classified into PM 2.5 and PM10, based upon their particle diameter. Existing networks of PM 2.5 and PM 10 monitors have shown that Particulate Matter concentrations have been increased. Due to their tiny size, they can penetrate deep into the lungs and mix with the blood stream. Which leads to cardiovascular and pulmonary diseases.

a. The primary aim of this project is to track the bus location and monitor air pollution levels within the environment through a mobile application.

b. To provide the information and location of the bus and amount of pollution during the journey to various places to the stakeholders for taking care and prior precautions.
c. To increase time efficiency, safety and with air pollution monitoring we are trying to create awareness on personal health safety.
d. The positional information or the coordinates of each visiting points are stored in a database, which later can be viewed on a display screen using digital maps. However, the users have to connect themselves to the web server with the respective vehicle ID stored in the database and only then she/he can view the location of vehicle travelled.
e. Analyse particulate matter in the surrounding by interfacing SDS sensor with ESP8266.
f. Understand the behaviour of particulate matter with respect to variation of temperature and humidity.

CHAPTER 2

LITERATURE STUDY

Payali Das, [1] proposed system is a low-cost, innovative Air Pollution Monitoring Device (APMD) with advanced features evaluation. A Particulate Matter (PM) sensor on-board is intended to measure PM 2.5 and PM 10. APMD also includes electrochemical sensors for measuring carbon monoxide, Sulphur dioxide, nitrogen dioxide, ozone, and temperature and humidity. To power the module, the node has a solar energy harvesting unit and a rechargeable battery as a backup. APMD uses an on-board GPS subsystem to package all of the collected air quality data into a frame that includes the physical location, time, and date, and sends it to a cloud server. Wi-Fi and NB-IoT connectivity are available to the node. The developed APMD was co-located with an accurate reference sensor node to validate sensing quality, and a series of field data were collected over a seven-day period. The on-board PM sensor saves up to 94% energy when fully turned on while maintaining a root mean square error (RMSE) of 0.58 for PM 2.5 and 2.5 for PM 10. A power control mechanism is also used on the PM sensor to control the fan speed by sending a pulse width modulated (PWM) signal to a switch connected to the fan's power supply. The on-board PM sensor is 97% more energy efficient than a commercial sensor at 100 ms switching period and 30% duty cycle, while maintaining sensing error (RMSE) as low as 0.7 for PM 2.5 and 2.7 for PM 10. Our outdoor deployment studies show that the designed APMD uses 90.8% less power than the reference setup while providing a significantly higher coverage range with an acceptable range of sensing error.

Muhammad Fareez Mohd Ainul Kakeem, [2] proposed system is a simple Internet of Things (IoT) prototype that allows users to view or authorities to monitor bus activity through a mobile application that displays available bus seats, bus schedules, and bus activities. The NodeMCU ESP32 controller, which communicates via Wi-Fi, is used in the design prototype. The input sensors are an infrared sensor and a GPS module. The data analysis is presented on mobile apps using Blynk and cloud applications. The mobile application was created so that users could view the number of passengers on the bus as well as its location. The online database is intended to collect all records of bus passengers entering and exiting the vehicle. As a result, the GPS module is able to determine the exact location of the bus as well as its latitude and longitude. Every 5 seconds, the activities of passengers entering and exiting the bus are recorded. In 3 minutes, the number of passengers at one bus stop increased to 20. The number of passengers who get off the bus is also recorded and analyzed. These activities can be monitored by authorities, which aids in providing good services, managing time, and managing bus transportation services.

Yuhan Huang, [3] proposed system employs on-road remote sensing (RS) technology for fast, accurate, and cost-effective identification of high-emitting vehicles as part of an enforcement programme to improve urban air quality. We discovered that a significant percentage of in-use petrol and LPG vehicles failed emission standards, particularly in high-mileage fleets, using large emission datasets from chassis dynamometer testing, RS, and air quality monitoring. The RS enforcement programme significantly cleaned these fleets in terms of high-emitter percentages, fleet average emissions, roadside and ambient pollutant concentrations, and emission inventory. The current enforcement program's challenges include conservative cut point setting, single-lane measurement sites, and a lack of application experience in diesel vehicles. Developing more accurate and vertical RS systems will improve and expand their applications.

Baichoo Bibi Humaira, [4] proposed system is to have an efficient Android Bus Tracking App, it needs to Our implementation involves tracking the real-time location of the bus using Global Positioning System (GPS) and Global System for Mobile Communication / General Packet Radio Service (GSM/GPRS) technology. Google Map API is used in the developed application to display the bus on the map. As a result, using the Application, users will be able to keep track of buses at all times. The distance and time required to reach the destination, as well as the time required for the bus to arrive at the user, will be displayed on the screen. An Internet connection, whether Wi-Fi or mobile data, is required for the user to receive this information as well as the map markers. SMS will be used as a backup communication method if there is no Internet access.

Temesegan Walelign Ayele, [5] proposed system is an Internet of Things (IoT)-based air pollution monitoring and prediction system. This system can be used to monitor air pollutants in a specific area, perform air quality analysis, and forecast air quality. The proposed system will monitor air pollutants by combining IoT with a machine learning algorithm known as Recurrent Neural Network, more specifically Long Short-Term Memory.
Vladimir Shakhov, [6] proposed system is a monitoring system, which sensors installed on vehicles. They offer the corresponding approach. The results help to optimize, rationalize, and manage efficient systems for monitoring of air pollution.

R. Santhana Krishnan, [7] proposed system is a new Android Application-based Smart Bus Transportation System that assists passengers in booking bus tickets using the Android Application and also keeps them updated on bus location based on their request. This system also sends alert messages to passengers a few minutes before the bus arrives at the passengers' boarding point. This system also sends out precautionary instructions to passengers ahead of time, which must be followed while riding in the bus. To provide additional safety to the passengers, the temperature of the passengers is monitored and communicated to the bus before they are allowed to board.

Siti Asma, [8] Two applications are proposed for the convenience of those who want to plan their journey with shuttle buses. One application will track the bus's location, while the other will be used by the students. Both proposed applications will be used in conjunction with an Android phone, which is commonly used by students. The primary goals of creating this application are to inform users of the current bus location and estimated arrival time. This application also offers users a real-time forum where they can initiate conversations with other users of the same application. Additionally, the driver's profile is included for the user's future reference.

Hina Gull, [9] proposed IoT-based Bus Tracking System will include a tracking website as well as an Android application for school administrators, bus drivers, and parents. The proposed system will charge the administrator with adding new bus drivers and students to the driver list. Furthermore, the application will generate a unique QR code for each student, which will be printed on a card with the student's personal information. In addition, the proposed system will track the bus's location via the driver's mobile device. Following that, the parents' application will display a map displaying the current bus position, which will be updated after each period, and the intervals between each update will be as short as possible to ensure everyone's safety. Furthermore, different types of notifications will be received by the school administration, parents, or both. For example, if the bus is late, the school administration and parents will be notified, and if there is a change in the daily bus schedule, the parents will be notified again.

S. Vigneshwaran, [10] prepared for the imaginative methodology of "Plan of Bus Tracking and Fuel Monitoring System" by combining current innovation with the requirement of data transmission. To overcome the drawbacks of previous paper-based techniques, we acquaint a task with tracking a vehicle using GPS and GSM. This Vehicle Tracking System can also be used for Accident Detection Alert System, Soldier Tracking System, and other applications by simply making a few changes in equipment and programming and broadly in following Cabs/Taxis, taken vehicles, school/university transports, and so on. The transport following framework is a useful and fruitful framework. This framework will be used to create four applications. The first application is to establish correspondence between the school server and the transport framework, which is equipped to provide constant information about the current state of transportation. The second application is sending a gathering message, for example, ready messages to understudies waiting at the next stop, changes in current course, transport number, and so on, saving understudies time. Period is the third application. There is no need for a plastic vehicle timer. The final application is developing a crisis management framework that will send ready messages to the school, police, and rescue vehicles in the event of an incident.

Rezowana Akter, [11] proposed system is designed to provide convenient transportation by minimizing problems encountered by passengers, drivers, and relevant authorities through the use of a handy Android application. For a satisfying bus fare calculation, our system employs Radio Frequency Identification [RFID], Global Positioning System [GPS], and an Android application for passenger management and real-time tracking features.

Hongjie Liu, [12] proposed system is a long short-term memory (LSTM) and Artificial neural networks (ANN) comprehensive prediction model based on spatial-temporal features vectors to combine the advantages of the two prediction models. The bus-to-station prediction is realized from the dimension of time feature, and the long-distance arrival-to-station prediction is realized from the dimension of spatial feature. Furthermore, experiments were carried out and tested on the basis of the entity dataset, and the results show that the proposed method has a high accuracy in bus arrival prediction problems.

Pau Ferrer-Cid, [13] proposed system compares two structured data-based techniques, one based on statistical methods and the other on signal smoothness, with a baseline technique based on node distance and not on measured signal data. To compare these techniques, the sensor signal is reconstructed using a supervised linear regression method and a semi-supervised Laplacian interpolation method, which allows reconstruction even when data is missing. The results on data sets measuring O 3, NO 2, and PM 10 show that the signal smoothness-based technique performs better than the other two and, when combined with Laplacian interpolation, is nearly optimal in comparison to the linear regression method. Furthermore, in the case of heterogeneous networks, the results show reconstruction accuracy comparable to in-situ calibrated sensors. Thus, using network data increases the network's robustness against possible sensor failures.

Ditsuhi Iskandaryan, [14] proposed system revises studies on air pollution prediction based on sensor data using machine learning algorithms in the context of smart cities. The most relevant papers were chosen using the most popular databases and the corresponding filtration. After thoroughly reviewing those papers, the main features were extracted and used as a foundation to link and compare them. As a result, we can conclude that: 1. instead of using simple machine learning techniques, authors now use advanced and sophisticated techniques; 2. China was the leading country in terms of a case study; 3.

Particulate matter with a diameter of 2.5 micrometres was the main prediction target; 4. in 41% of the publications, authors predicted the next day; 5. 66% of the studies used data with an hourly rate; and 6. 49% of the paper.

Zena A. Aziz, [15] proposed system is based on Wireless Sensor Networks, a revolutionary system that can detect, calculate, and gather information from the real world and relay data to the consumer based on the gathered data. One of the most important aspects of modern networks is the ability to assess the world at high resolutions. This report contains research on pollution sensors and monitoring systems.

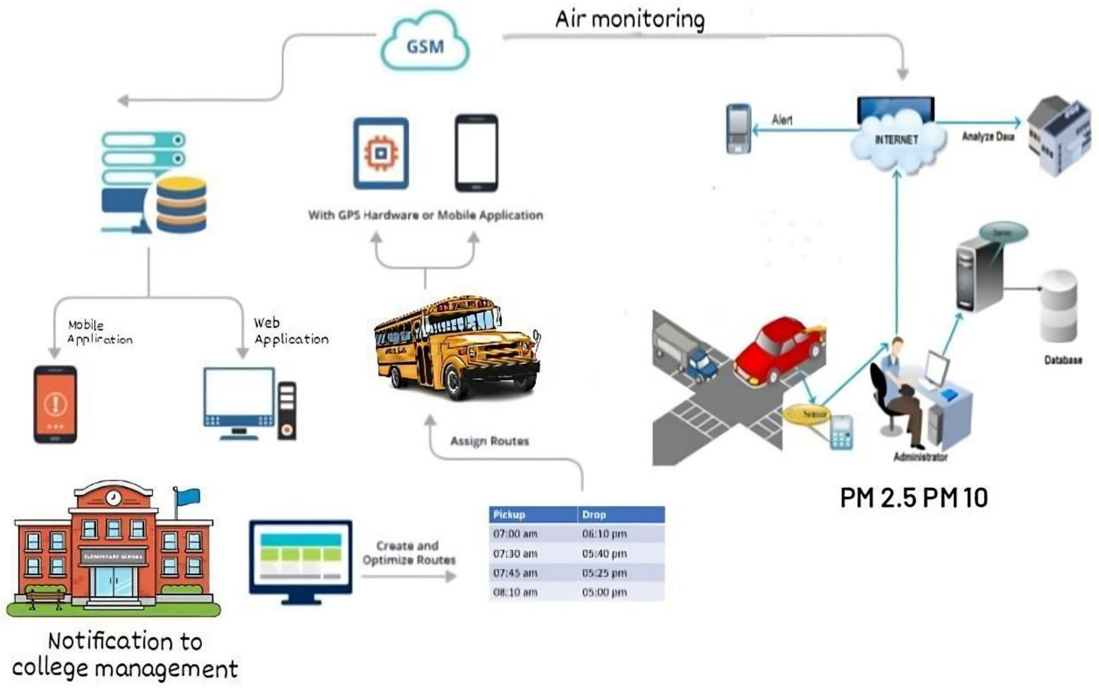
B. Perumal. [16] proposed system is a framework for tracking air pollution. An Arduino mini is used to control the proposed model. The air pollution observance system is designed to track and evaluate air quality in real time using data retrieved from an overseas server and saved online. The components of air quality are measured in Million Metrics (PPM) and analyzed in Microsoft Excel. The established system's air quality measurements were also accurate.

CHAPTER 3

METHODOLOGY

The proposed system consists of various sensors i.e., GPS sensor, Particulate Matter sensor (SDS 011), DHT11 sensor, used for humidity and temperature monitoring. The PM (Particulate Matter) sensor takes in data on the present air-borne particles under 2.5 and 10 µm. It updates at rapid time intervals.

The GPS Sensor that is connected to the bus sends in data to the server through GSM. Based on the proposed system these sensors are interfaced to an ESP8266 Microcontroller. Both the air pollution and GPS information can be accessed through a Mobile Application.



Pickup	Drop
07:00 am	06:10 pm
07:30 am	05:40 pm
07:45 am	05:25 pm
08:10 am	05:00 pm

Figure 3.1 An Overview of the complete working process

Above fig. 3.1 illustrates the overall working of the air pollution and bus tracking system. Starting from the host to the receiver.

3.1 BLOCK DIAGRAM

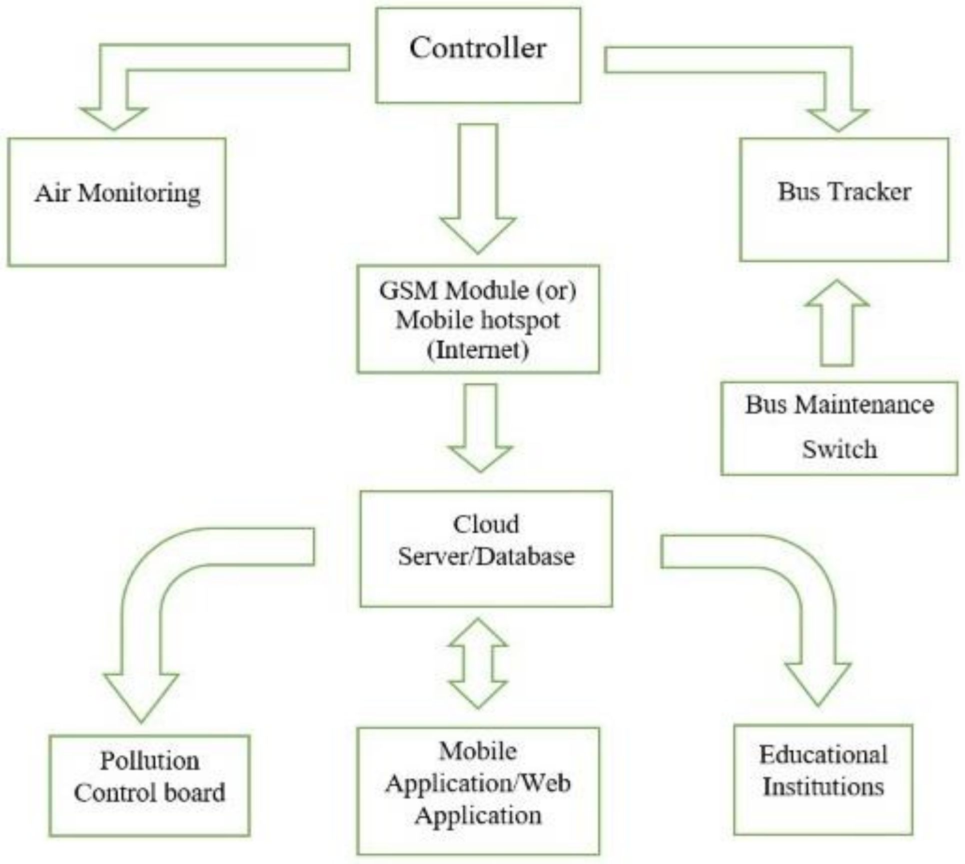


Figure 3.2 Block diagram of Mobile Air Pollution Monitoring and Bus Tracking System

3.2 FUNCTIONS OF EACH BLOCK

POWER SUPPLY:

The main purpose of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load. It supplies electric power to an electric load.

MICROCONTROLLER:

The microcontroller is used to manipulate the serial operation based on the program present in the output taken from one of the four ports.

GSM MODULE:

Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardisation group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 MHz It is to define the procedures for second-generation digital mobile networks that are used by the devices such as mobile phones.

GPS RECEIVER:

GPS, in full Global Positioning System, space-based radio-navigation system that broadcasts highly accurate navigation pulses to users on or near the Earth. In the United States' Navstar GPS, 24 main satellites in 6 orbits circle the Earth every 12 hours. In addition, Russia maintains a constellation called GLONASS (Global Navigation Satellite System).

CLOUD SERVER:

A cloud server is a pooled, centralised server resource that is hosted and delivered over a network—typically the Internet—and accessed on demand by multiple users. Cloud servers can perform all the same functions of a traditional physical server, delivering processing power, storage and applications. It acts as a database.

MOBILE APPLICATION:

The mobile application is used to access the data from the server wirelessly and control the functions of the received data. It also interprets data into the user viewable format.

DHT11 SENSOR:

DHT11is a digital temperature and humidity sensor embedded for reflow soldering. The dual-row flat leadless SMD package has a 4x5mm bottom and a height of 1.6mm. The minuscule sensor has a power supply range of 1.8-3.6V, but 3.3V is the recommended operating voltage.

SDS 011 SENSOR:

SDS 011 is an air quality measurement sensor which can be used to get dust particles and smoke concentration in the air. More precisely, it can measure particulate matter (PM) concentrations in the air. It can detect the dust particles concentration between 0.3 to 10um. Most importantly, Nova PM dust sensor provides an interrupt-based response when the concentration of dust particles changes in the air and the response time is less than 10 seconds. The operating voltage range is 4.7-5.3V which makes it suitable to use with standard voltage of 5 volts. Furthermore, it has a UART module and PWM outputs which can be used to get output from the SDS011 sensor.

ESP8266:

The ESP8266 is a low-cost Wi-Fi microchip. With built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems in Shanghai, China. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

CHAPTER 4

HARDWARE AND SOFTWARE DESCRIPTION

4.1 HARDWARE SPECIFICATIONS

4.1.1 NODEMCU:

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module.

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the "computer" on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

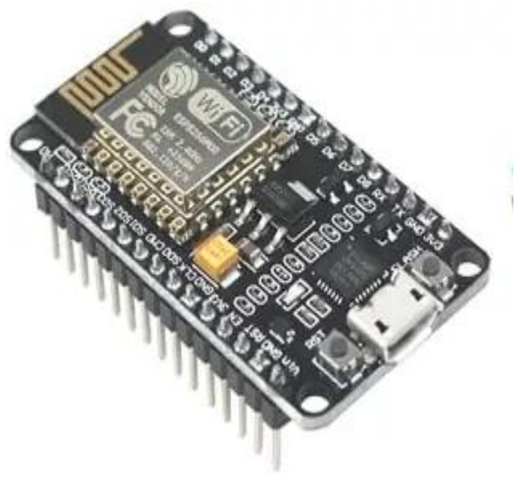


Figure 4.1 NodeMCU ESP8266

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

NodeMCU ESP8266 Specifications & Features:

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- SRAM: 64 KB
- Flash Memory: 4 MB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects
- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

The NodeMCU ESP8266 development board comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

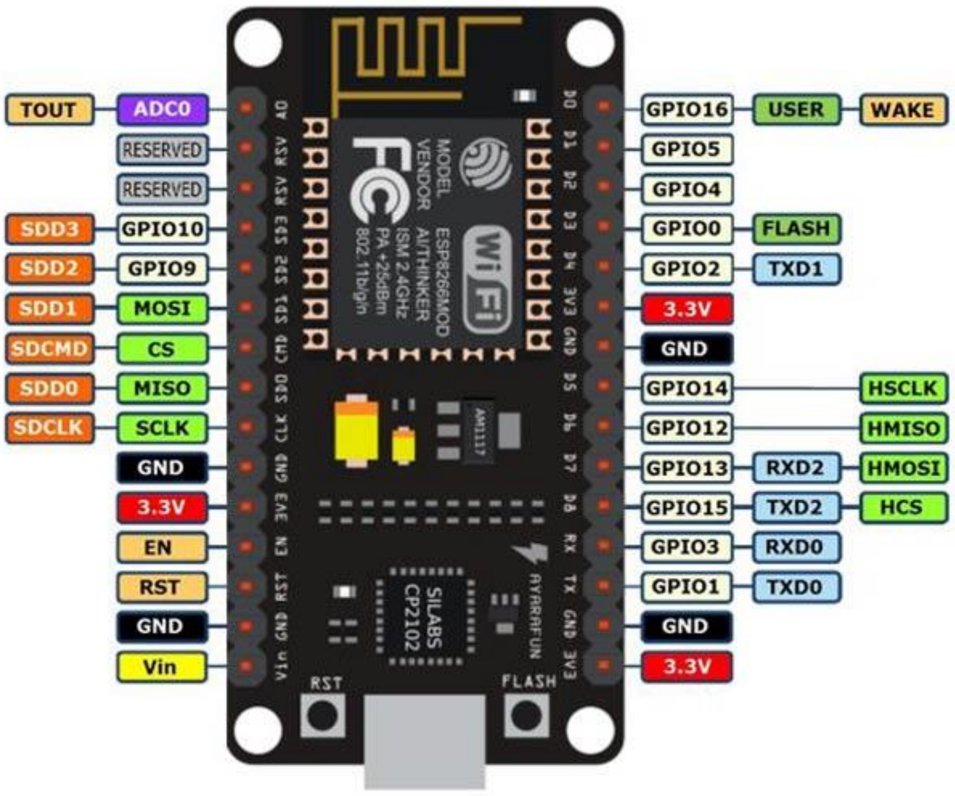


Figure 4.2 NodeMCU ESP8266 Pinout diagram:

For practical purposes ESP8266 NodeMCU V2 and V3 boards present identical pinouts. While working on the NodeMCU based projects we are interested in the following pins.

- Power pins (3.3 V).
- Ground pins (GND).
- Analog pins (A0).

- Digital pins (D0 – D8, SD2, SD3, RX, and TX – GPIO XX)

Most ESP8266 NodeMCU boards have one input voltage pin (Vin), three power pins (3.3v), four ground pins (GND), one analogue pin (A0), and several digital pins (GPIO XX).

Table 4.1 Interfacing pins of ESP8266 and Arduino Alias

PIN	CODE	ARDUINO ALIAS
A0	A0	A0
D0	GPIO 16	16
D1	GPIO 5	5
D2	GPIO 4	4
D3	GPIO 0	0
D4	GPIO 2	2
D5	GPIO 14	14
D6	GPIO 12	12
D7	GPIO 13	13
D8	GPIO 15	15
SD2	GPIO 9	9

SD3	GPIO 10	10
RX	GPIO 3	3
TX	GPIO 1	1

4.1.2 BREAD BOARD:

A breadboard allows for easy and quick creation of temporary electronic circuits or to carry out experiments with circuit design. Breadboards enable developers to easily connect components or wires thanks to the rows and columns of internally connected spring clips underneath the perforated plastic enclosure.

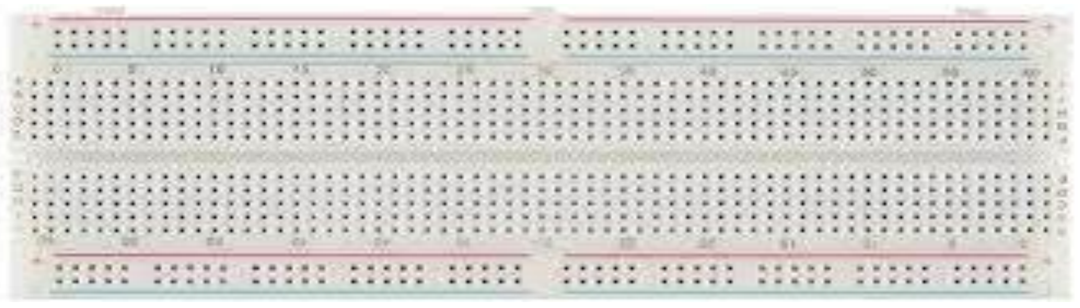


Figure 4.3 Bread Board

As the name suggests, the term breadboard can be derived from two terms namely bread & board. Initially, this was used to cut the bread into pieces. Further, it was called a breadboard & it was used in electronics projects and electronic devices in the year 1970. A breadboard is also known as a solderless board because the component used on the breadboard does not need any soldering to connect to the board, so it can be reused.

Compared to more permanent circuit connection methods, modern breadboards have high parasitic capacitance, relatively high resistance, and less reliable connections, which are subject to jostle and physical degradation. Signaling is limited to about 10 MHz, and not everything works properly even well below that frequency.

Breadboards have evolved over time, with the term now being used for all kinds of prototype electronic devices. For example, US Patent 3,145,483, was filed in 1961 and describes a wooden plate breadboard with mounted springs and other facilities. US Patent was filed in 1967 and refers to a particular layout as a Printed Circuit Breadboard. Both examples refer to and describe other types of breadboards.

The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

The specifications & features of a breadboard:

- Distribution Strips are two.
- Wire Size is 21 to 26 AWG wire.
- Tie Points are two hundred.
- Withstanding Voltage is 1,000V AC.
- Tie points within IC are 630.
- Insulation Resistance is DC500V or 500MΩ
- Dimension is 6.5*4.4*0.3 inch.
- Rating is 5Amps.

4.1.3 DHT-11 SENSOR:

The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

This sensor is used in various applications such as measuring humidity and temperature values in heating, ventilation and air conditioning systems. Weather stations also use these sensors to predict weather conditions. The humidity sensor is used as a preventive measure in homes where people are affected by humidity.

Working Principle of DHT11 Sensor:

`DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them.

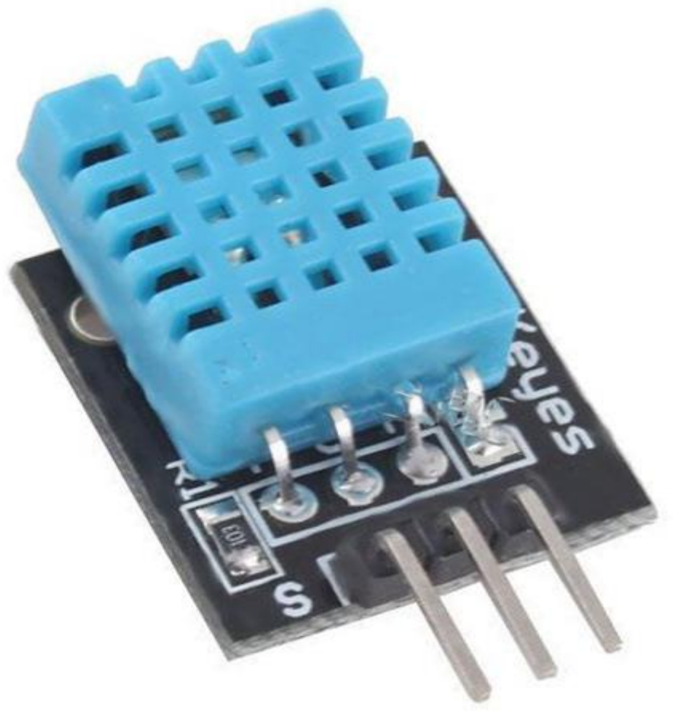


Figure 4.4 DHT-11 Sensor

Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form. For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and micro-controller.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz.i.e., it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

Communication Process:

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU.

Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low power-consumption mode until it receives a start signal from MCU again.

Applications:

- Measure temperature and humidity
- Local Weather station
- Automatic climate control
- Environment monitoring

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

4.1.4 SDS 011 SENSOR:

SDS 011 is an air quality measurement sensor which can be used to get dust particles and smoke concentration in the air. More precisely, it can measure particulate matter (PM) concentrations in the air. It can detect the dust particles concentration between 0.3 to 10um.

Various types of soot sensors, also known as particulate matter or PM sensors, are used for the control and diagnostics of emission systems utilizing diesel particulate filters (DPF).

A beam of light inside the sensor is passed through a sample of air, and particles in the sample scatter the light beam. The scattered light is measured and used to calculate the concentration of particles in the air sample.

Particulate matter (PM) includes microscopic matter suspended in air or water. Airborne particles are called aerosols. PM10 includes particles less than 10 µm in diameter, PM2. 5 those less than 2.5 µm. The toxicity of suspended particles is mainly due to particles with a diameter of less than 10µm.

The toxicity of suspended particles is mainly due to particles with a diameter of less than 10µm. They can be emitted directly into the air from anthropogenic activities (industry, residential, agriculture, transport) and natural sources (forest fires, volcanic eruptions, etc.).

Particles can also be formed directly in the atmosphere by physio-chemical reactions between pollutants already present in the atmosphere. So, PM10 refers to particles with an aerodynamic diameter smaller than 10 µm, and PM2.5 µm.

PM10 (particles with a diameter of 10 micrometers or less): these particles are small enough to pass through the throat and nose and enter the lungs. Once inhaled, these particles can affect the heart and lungs and cause serious health effects.

Particles in the $PM_{2.5}$ size range are able to travel deeply into the respiratory tract, reaching the lungs. Exposure to fine particles can cause short-term health effects such as eye, nose, throat and lung irritation, coughing, sneezing, runny nose and shortness of breath.

Particulate matter has been shown in many scientific studies to reduce, visibility and also to adversely affect climate, ecosystems and materials. PM, primarily PM2.5, affects visibility by altering the way light is absorbed and scattered in the atmosphere. With reference to climate change, some constituents of the ambient PM mixture promote climate warming (e.g., black carbon), while others have a cooling influence (e.g., nitrate and sulfate), and so ambient PM has both climate warming and cooling properties.PM can adversely affect ecosystems, including plants, soil and water through deposition of PM and its subsequent uptake by plants or its deposition into water where it can affect water quality and clarity. The

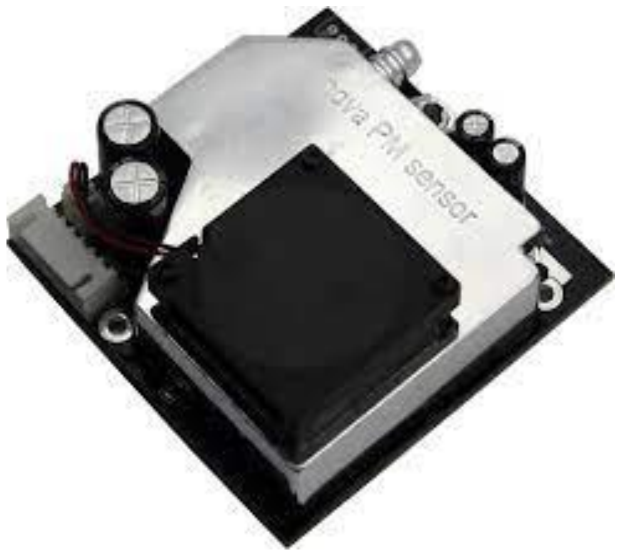


Figure 4.5 SDS 011 Sensor

metal and organic compounds in PM have the greatest potential to alter plant growth and yield. PM deposition on surfaces leads to soiling of materials.

Most importantly, Nova PM dust sensor provides an interrupt-based response when the concentration of dust particles changes in the air and the response time is less than 10 seconds. The operating voltage range is 4.7-5.3V which makes it suitable to use with standard voltage of 5 volts. Furthermore, it has a UART module and PWM outputs which can be used to get output from the SDS011 sensor.

Features:

- Accurate and Reliable: laser detection, stable, good consistency.
- Quick response: response time is less than 10 seconds when the scene changes.
- Easy integration: UART output (or IO output can be customized), fan built-in.
- High resolution: the resolution of 0.3ug/m3.

SDS 011 SENSOR Specifications:

Table 4.2 SDS 011 Sensor Specifications

Range	0.001 to 1.000 mg /m3
Minimum Detection Limit	0.001 mg/m3
Accuracy of Factory Calibration	± (0.005 mg/m3 + 15 % of reading)
Response Time	5 Seconds
Measurement Parameters	PM2.5 and PM10

Applications:

- PM2.5 Detector
- Purifier
- Air Exchangers
- Filtering system

4.1.5 GSM:

Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 MHz.

Global System for Mobile communications, is a digital cellular communications system, which has rapidly gained acceptance and market share worldwide, although it was initially developed in a European context. In addition to digital transmission, GSM incorporates many advanced services and features, including ISDN compatibility and worldwide roaming in other GSM networks.

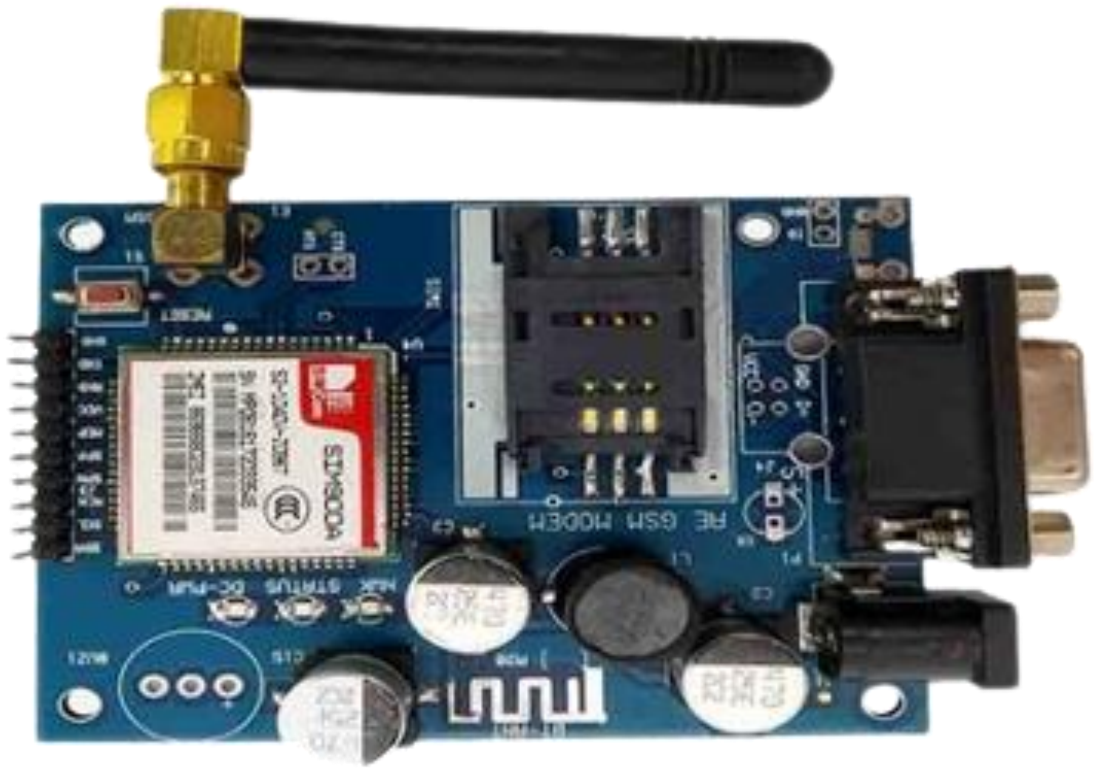


Figure 4.6 GSM Module

GSM SERVICES:

1.Tele-services

2. Bearer or Data Services

3. Supplementary services

Tele-services: Telecommunication services that enable voice communication via mobile phones Offered services, Mobile telephony, Emergency calling

Bearer or Data Services: Include various data services for information transfer between GSM and other networks like PSTN, ISDN etc. at rates from 300 to 9600 bps, Short Message Service (SMS) up to 160-character alphanumeric data transmission to/from the mobile terminal unified.

Supplementary services: Call related services like Call Waiting- Notification of an incoming call while on the handset, Call Hold- Put a caller on hold to take another call,

Call Barring- All calls, outgoing calls, or incoming calls, Call Forwarding- Calls can be sent to various numbers defined by the user, Multi Party Call Conferencing - Link multiple calls together.

1. CLIP – Caller line identification presentation
2. CLIR – Caller line identification restriction

Characteristics of GSM Standard:

Fully digital system using 900,1800 MHz frequency band.

1. TDMA over radio carriers (200 KHz carrier spacing)
2. 8 full rate or 16 half rate TDMA channels per carrier.
3. User/terminal authentication for fraud control.
4. Encryption of speech and data transmission over the radio path
5. Full international roaming capability.
6. Low speed data services (up to 9.6 Kb/s).
7. Compatibility with ISDN

Security in GSM:

On air interface, GSM uses encryption and TMSI instead of IMSI.

λ SIM is provided 4–8-digit PIN to validate the ownership of SIM

λ 3 algorithms are specified: -

1. A3 algorithm for authentication
2. A5 algorithm for encryption
3. A8 algorithm for key generation

4.1.6 GPS MODULE:

The Global Positioning System (GPS) is a satellite-based navigation system that provides location and time information. The system is freely accessible to anyone with a GPS receiver and unobstructed line of sight to at least four of GPS satellites.

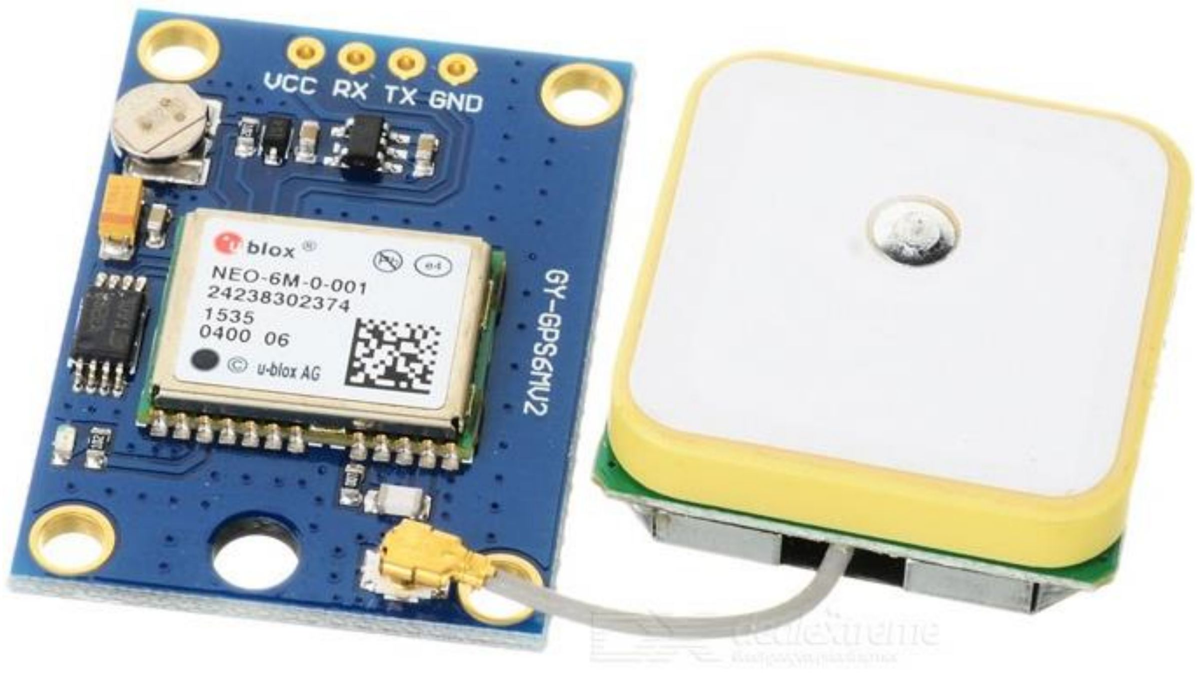


Figure 4.7 GPS Module

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites. GPS is nowadays widely used and also has become an integral part of smart phones. The GTPA010 module is easy to use, having RS232 as well as USB interface. It operates over 3.2 to 5V supply range thus enabling interfacing with microcontrollers with 3.3V as well as 5V. The module outputs GPS data in NMEA0183 format.

Each of message string starts with ‘$’ and then the message identifier. Each parameter is separated using a comma so that the message can be parse with the help of the commas. There is a numerous utilization of GPS Modules. Particularly, plenty of social activities are able to be developed by applications of these GPS Modules. Therefore, GPS Modules play important roles in various sectors, which are including Environmental Measurement, Transportation, Emergency Rescue, Agriculture, Entertainment and etc.

Firstly, the signal of time is sent from a GPS satellite at a given point. Subsequently, the time difference between GPS time and the point of time clock which GPS receiver receives the time signal will be calculated to generate the distance from the receiver to the satellite.

The same process will be done with three other available satellites. It is possible to calculate the position of the GPS receiver from distance from the GPS receiver to three satellites. However, the position generated by means of this method is not accurate, for there is an error in calculated distance between satellites and a GPS receiver, which arises from a time error on the clock incorporated into a GPS receiver.

SPACE SEGMENT (GPS satellites)

A number of GPS satellites are deployed on six orbits around the earth at the altitude of approximately 20,000 km (four GPS satellites per one orbit), and move around the earth at 12-hour-intervals.

CONTROL SEGMENT (Ground control stations)

Ground control stations play roles of monitoring, controlling and maintaining satellite orbit to make sure that the deviation of the satellites from the orbit as well as GPS timing are within the tolerance level.

USER SEGMENT (GPS receivers)

User segment (GPS receivers)

BASIC STRUCTURE OF GPS:

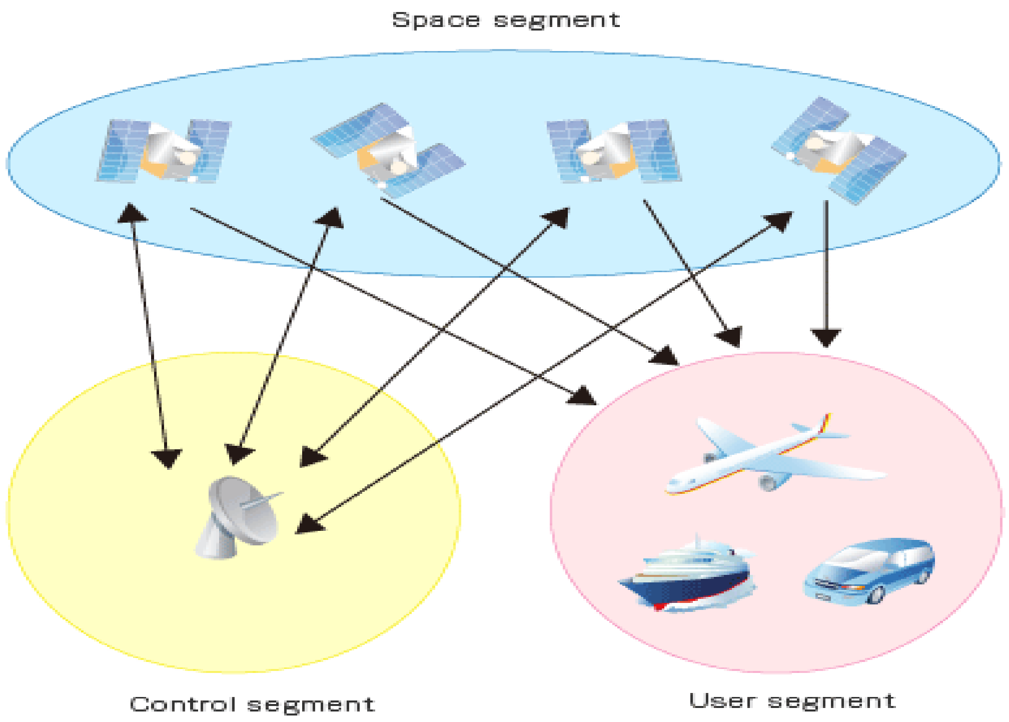


Figure 4.8 Three Elements of GPS

GPS SIGNALS:

GPS satellites transmit multiple frequencies, such as L1 (1575.42MHz), L2 (1227.60MHz) and L5 (1176.45MHz).

The typical signal sent out is the C/A code, which can be used for commercial purposes; the C/A code consists of a recognition code for each satellite, and information called a navigation message is sent at the same time. The data of the orbit of each satellite is called the ephemeris, and the data of orbit of all satellite is called the almanac. The navigation messages are broadcast at a rate of 50 bits per second. For a satellite, an atomic clock is incorporated to generate on-the-spot time information, but the time generated by clocks incorporated into GPS receivers is not as precise as the time generated by atomic clocks on satellites.

Here, the fourth satellite comes to play its role: the distance from the fourth satellite to the receiver can be used to compute the position in relations to the position data generated by distance between three satellites and the receiver, hence reducing the margin of error in position accuracy.

State of reception of GPS depends upon the number of satellites tracked for positioning. If the number of the tracked satellites is great, GPS positioning becomes greater, but if there were fewer satellites tracked for positioning, it would be difficult to generate GPS position.

FEATURES:

- MediaTek MT3329 Chipset, L1 Frequency, C/A code, 66 Channels
- 3m position accuracy
- Jammer detection and reduction
- Data output Baud rate: 9600 bps (Default)
- Low Power Consumption: 55mA @ acquisition, 40mA @ tracking
- High Sensitivity, -165 dBm, TCXO Design, superior urban performances
- Patch antenna
- High sensitivity
- DGPS(WAAS/EGNOS/MSAS/GAGAN) support

GPS MODULE Specifications:

Table 4.2 GPS Module Specifications

Model	Ublox NEO-6M
Receiver Type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS

Input Supply Voltage (VDC)	2.7 ~ 6 v
Tracking Sensitivity (dBm)	-161 dBm

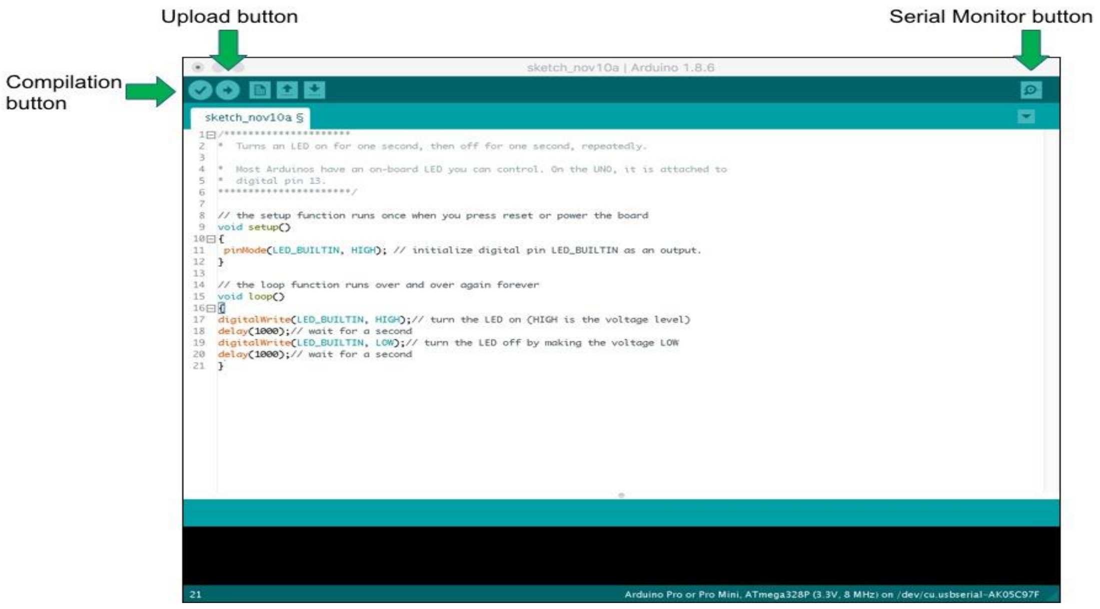
4.2 SOFTWARE SPECIFICATIONS

4.2.1 Arduino Software

The Arduino IDE is **an open-source software, which is used to write and upload code to the Arduino boards**. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++.

Here, IDE stands for Integrated Development Environment. The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '. ino.'

There are two required functions in an Arduino sketch, **setup () and loop ()**. Other functions must be created outside the brackets of those two functions. As an example, we will create a simple function to multiply two numbers.


Upload button
Serial Monitor button
Compilation button
sketch_nov10a | Arduino 1.8.6
sketch_nov10a §
21
Arduino Pro or Pro Mini, ATmega328P (3.3V, 8 MHz) on /dev/cu.usbserial-AK05C97F

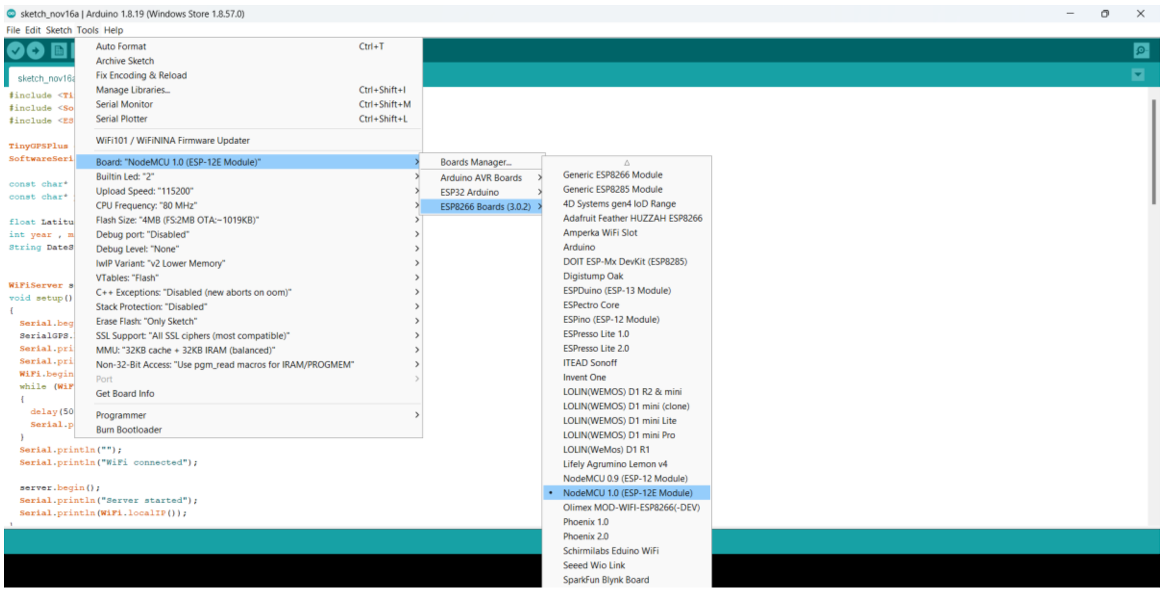
```
/**********************
*  Turns an LED on for one second, then off for one second, repeatedly.

*  Most Arduinos have an on-board LED you can control. On the UNO, it is attached to
*  digital pin 13.
**********************/

// the setup function runs once when you press reset or power the board
void setup()
{
 pinMode(LED_BUILTIN, HIGH); // initialize digital pin LED_BUILTIN as an output.
}

// the loop function runs over and over again forever
void loop()
{
digitalWrite(LED_BUILTIN, HIGH);// turn the LED on (HIGH is the voltage level)
delay(1000);// wait for a second
digitalWrite(LED_BUILTIN, LOW);// turn the LED off by making the voltage LOW
delay(1000);// wait for a second
}
```

Figure 4.9 Arduino IDE main window

It has two elements, hardware and software that form a system to rapidly develop microcontroller projects. It is based on the Atmel AVR microcontroller but you do not need to know this and it is hidden beneath the surface, which is one of the disadvantages of Arduino. The main window of the Arduino IDE is shown below, with the simple Blink example.The software is an open-source development environment, written in Java that can run under Linux, MAC or Windows. It runs a simple programming language called **Wiring**, which makes it Faily easy to write scripts to make the microcontroller carry out tasks. These scripts are called **Sketches** by Arduino. Most shields come with sketches already written that can be loaded in to the software, compiled and downloaded to the base board.Arduino board designs use a variety of microprocessor and controllers. The boards are equipped with sets of digital and analog (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages.

The Initial Setup

We need to setup the environment to Tools menu and select Board.Then select the type of nodeMCU



Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"

```
#include <Ti
#include <So
#include <ES

TinyGPSPlus
SoftwareSeri

const char*
const char*

float Latitu
int year , m
String DateS


WiFiServer s
void setup()
{
  Serial.beg
  SerialGPS.
  Serial.pri
  Serial.pri
  WiFi.begin
  while (WiF
  {
    delay(50
    Serial.p
  }
  Serial.println("");
  Serial.println("WiFi connected");

  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
```

Figure 4.10 Select the type of nodeMCU

Every sketch needs two void type functions, setup() and loop(). A void type function doesn’t return any value.

The setup() method is run once at the just after the Arduino is powered up and the loop() method is run continuously afterwards. The setup() is where you want to do any initialization steps, and in loop() you want to run the code you want to run over and over again.

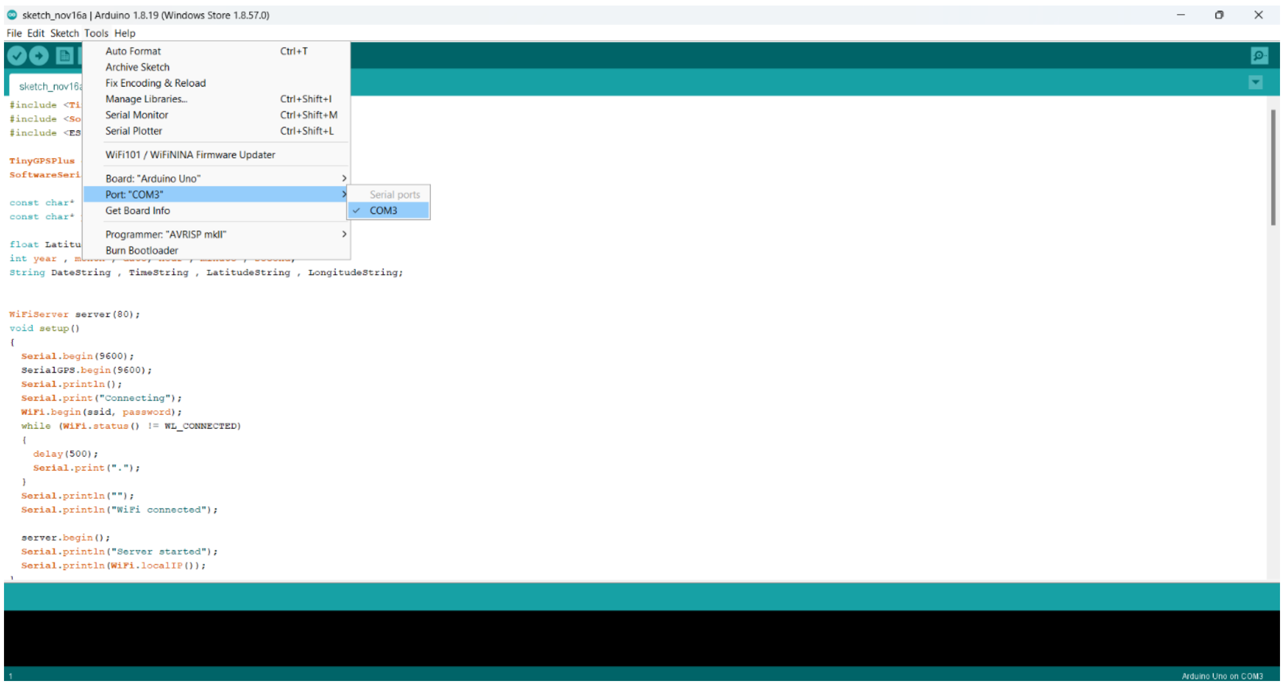
So, your basic sketch or program should look like this:

void setup()

{

void loop()

}


sketch_nov16a | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
sketch_nov16a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ COM3
Arduino Uno on COM3

```
#include <Ti
#include <So
#include <ES

TinyGPSPlus
SoftwareSeri

const char*
const char*

float Latitu
int year ,
String DateString , TimeString , LatitudeString , LongitudeString;


WiFiServer server(80);
void setup()
{
  Serial.begin(9600);
  SerialGPS.begin(9600);
  Serial.println();
  Serial.print("Connecting");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
```

Figure 4.11 Arduino IDE main window

Advantages and Disadvantages of Arduino:

Advantages:

- Not much knowledge required to get started
- Fairly low cost, depending on shields you need
- Lots of sketches and shields available
- No external programmer or power supply needed

DISADVANTAGES:

- No understanding of the AVR microcontroller
- Sketches and shields can be difficult to modify
- No debugger included for checking scripts
- You get no experience of C or professional development tools.

4.2.2 CLOUD SERVER

The cloud server we used in this system is **Thingspeak.** ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web services, social network and other APIs. We will consider each of these features in detail below.

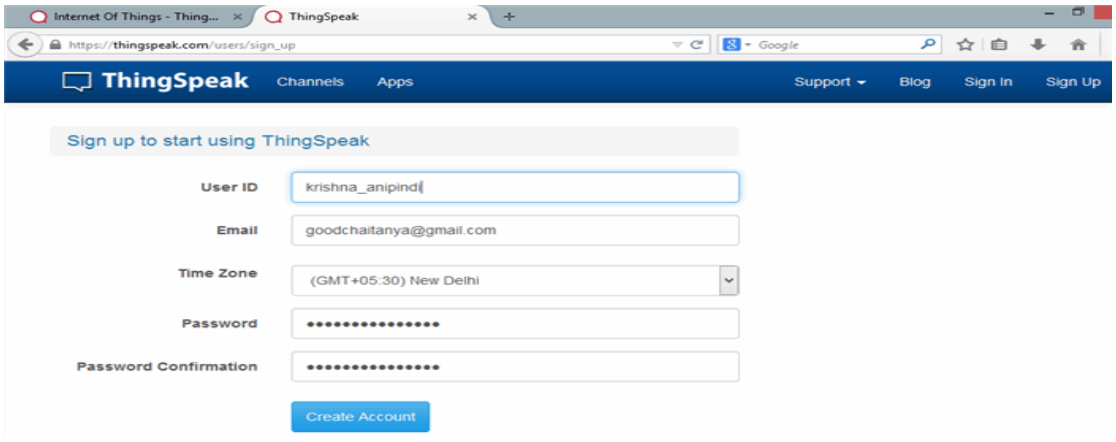
The core element of ThingSpeak is a 'ThingSpeak Channel'. A channel stores the data that we send to ThingSpeak and comprises of the below elements:

- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
- 1 status field - A short message to describe the data stored in the channel.

To use ThingSpeak, we need to sign up and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

How to start with Thingspeak?

Open https://thingspeak.com/ and click on the 'Get Started Now' button on the centre of the page and you will be redirected to the sign-up page(you will reach the same page when you click the 'Sign Up' button on the extreme right). Fill out the required details and click on the 'Create Account' button.

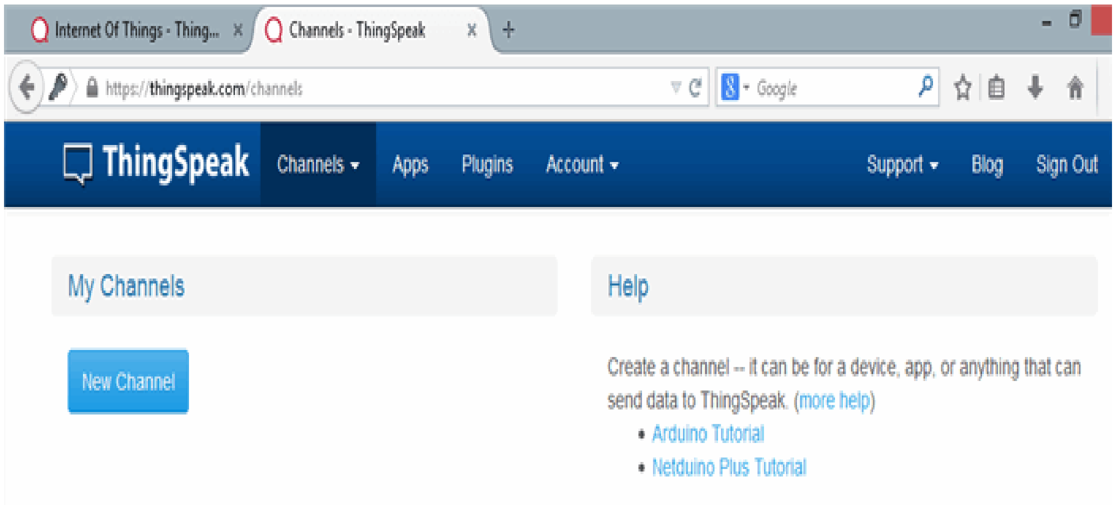


Sign up to start using ThingSpeak

User ID	krishna_anipindi
Email	goodchaitanya@gmail.com
Time Zone	(GMT+05:30) New Delhi
Password	•••••••••••••••
Password Confirmation	•••••••••••••••

Figure 4.12: Sign up process of Thingspeak

Now you should see a page with a confirmation that the account was successfully created. The confirmation message disappears after a few seconds and the final page should look as in the below screen:


Internet Of Things - Thing...

Channels - ThingSpeak

https://thingspeak.com/channels

ThingSpeak Channels Apps Plugins Account Support Blog Sign Out

My Channels

New Channel

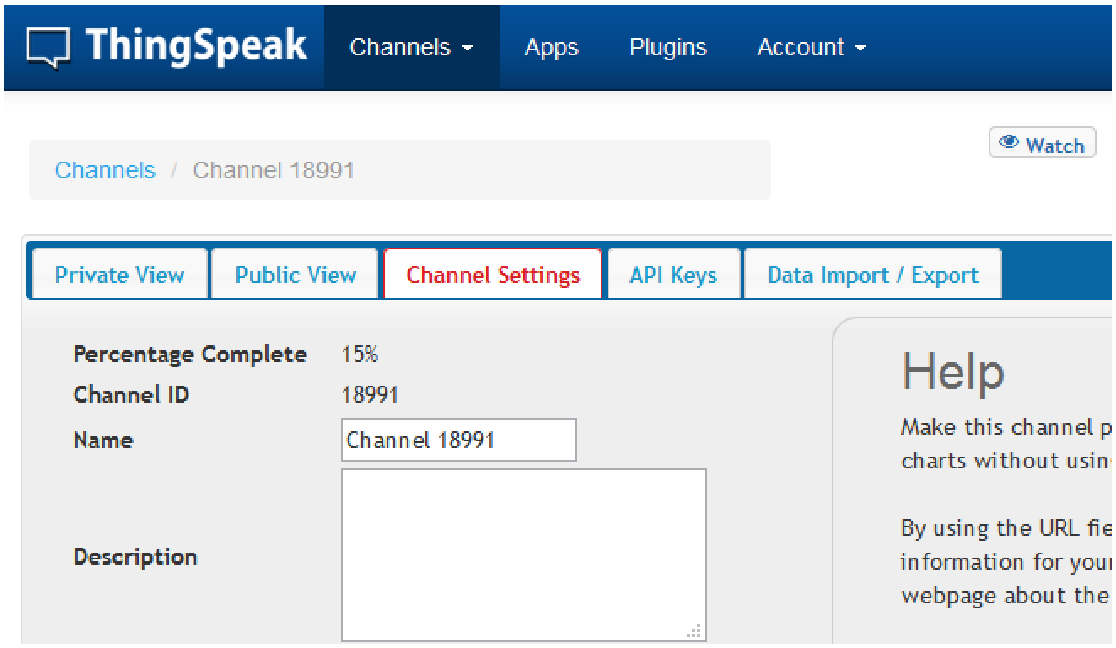
Help

Create a channel -- it can be for a device, app, or anything that can send data to ThingSpeak. (more help)

- Arduino Tutorial
- Netduino Plus Tutorial

Figure 4.13: Dashboard of Thingspeak

Go ahead and click on ‘New Channel’. You should see a page like the below:


ThingSpeak Channels Apps Plugins Account

Channels / Channel 18991

Watch

Private View | Public View | Channel Settings | API Keys | Data Import / Export

Percentage Complete 15%

Channel ID 18991

Name Channel 18991

Description

Help

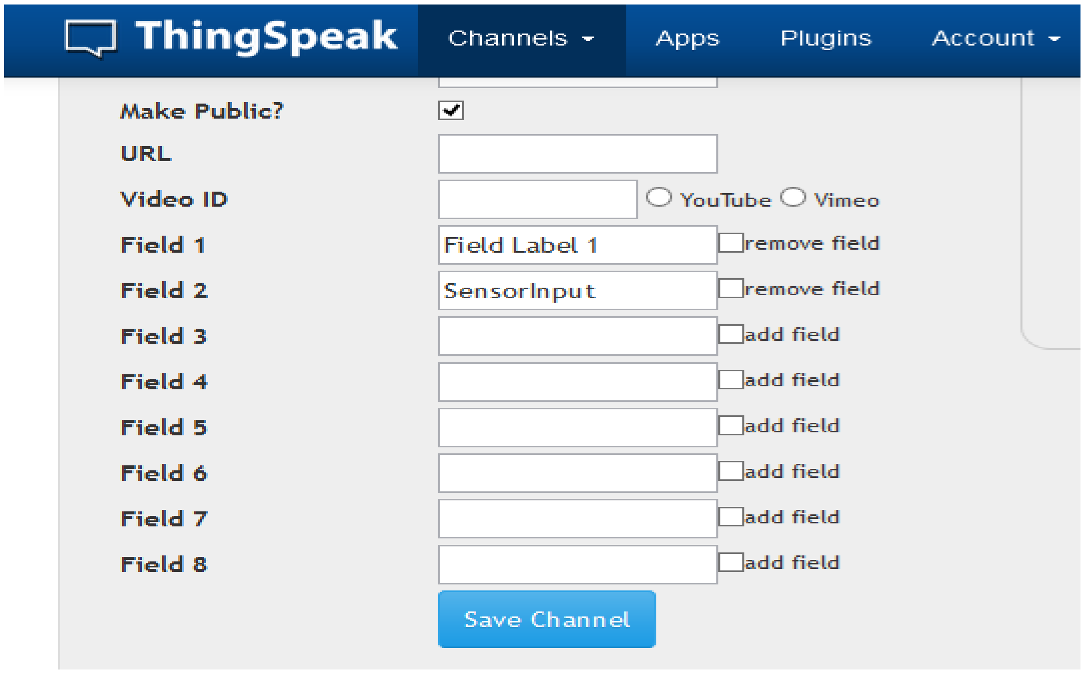
Make this channel p
charts without usin

By using the URL fie
information for you
webpage about the

Figure 4.14: Channel creation in Thingspeak

You can change the name to fit your need and you can add a description corresponding to the channel. You can add any other useful description into the metadata field. In the same page, you should see the fields for Latitude, Longitude and Elevation. Also, when you scroll down you should see a check box that says ‘Make Public?’. Let us consider the significance of the various fields and the tabs:

- Latitude, longitude and elevation - These fields correspond to the location of a ‘thing’ and are especially significant for moving things.
- Make Public? - If the channel is made public, anyone can view the channel's data feed and the corresponding charts. If this check box is not checked, the channel is private, which means for every read or write operation, the user has to pass a corresponding API key.
- URL - This can be the URL of your blog or website and if specified, will appear on the public view of the channel.
- Video ID - This is the ID corresponding to your YouTube or Vimeo ID. If specified, the video appears on the public view of the channel.
- Fields 1 to 8 - These are the fields which correspond to the data sent by a sensor or a ‘thing’. A field has to be added before it can be used to store data. By default, Field 1 is added. In case you try posting to fields that you have not added, your request will still be successful, but you will not be able to see the field in the charts and the corresponding data. You can click on the small box before the ‘add field’ text corresponding to each field to add it. Once you click the ‘add field’ box, a default label name appears in the text box corresponding to each field and the ‘add field’ text changes to ‘remove field’. You can edit the field text that appears by default when a field is added to make more sense. For example, in the below screen, I have modified the text for Field 2 to ‘Sensor Input’. To remove a field which is added, just check on the ‘remove field’ box. Once you click this, the text ‘remove field’ changes back to ‘add field’ and the corresponding field text is cleared.



Make Public?	✓	
URL		
Video ID		○ YouTube ○ Vimeo
Field 1	Field Label 1	☐ remove field
Field 2	SensorInput	☐ remove field
Field 3		☐ add field
Field 4		☐ add field
Field 5		☐ add field
Field 6		☐ add field
Field 7		☐ add field
Field 8		☐ add field

Figure 4.15: Field creation in Thingspeak

How to now the API Keys?

Now click on the ‘API Keys’ tab. You should see a screen similar to the below. The write API key is used for sending data to the channel and the read API key(s) is used to read the channel data. When we create a channel, by default, a write API key is generated. We generate read API keys by clicking the ‘Generate New Read API Key’ button under this tab. You can also add a note corresponding to each of the read API keys. Please note that clicking on the ‘Generate New Write API Key’ will over-write the previous key. You will only have one Write API key at any point of time. Also, in case your channel is private, others can only view the channel’s feed and charts by using a Read API key. Please share the Read API keys with people who are approved and authorized to view your channel.

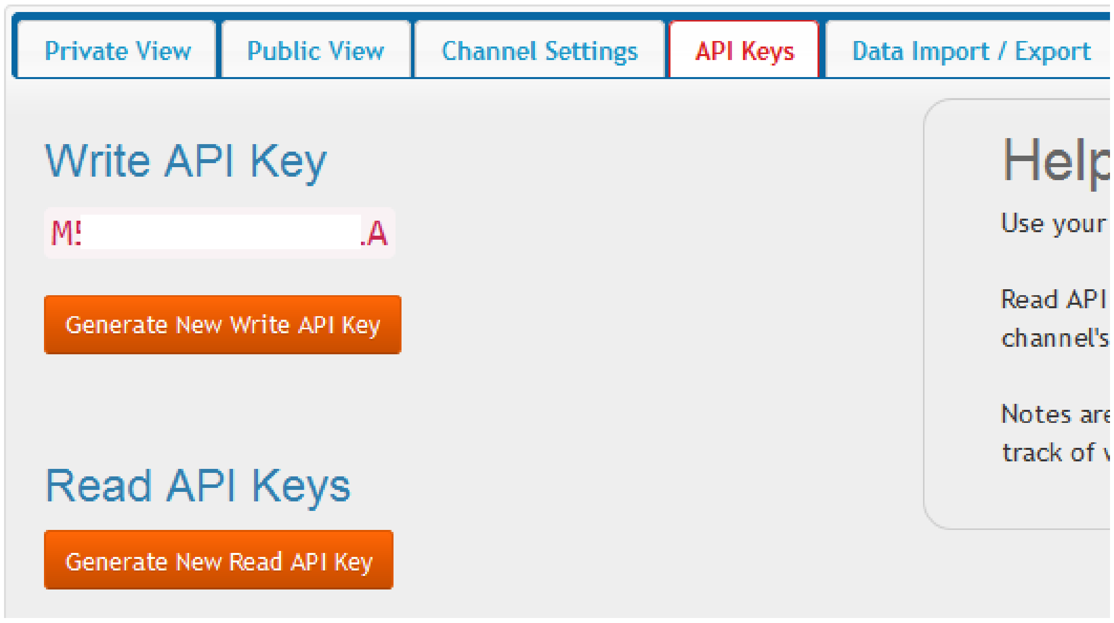


Figure 4.16: API keys of Thingspeak

ThingSpeak Apps

ThingSpeak provides apps that allow us for an easier integration with the web services, social networks and other APIs. Below are some of the apps provided by ThingSpeak:

- ThingTweet - This allows you to post messages to twitter via ThingSpeak. In essence, this is a TwitterProxy which re-directs your posts to twitter.
- ThingHTTP - This allows you to connect to web services and supports GET, PUT, POST and DELETE methods of HTTP.
- TweetControl - Using this, you can monitor your Twitter feeds for a specific key word and then process the request. Once the specific keyword is found in the twitter feed, you can then use ThingHTTP to connect to a different web service or execute a specific action.
- React - Send a tweet or trigger a ThingHTTP request when the Channel meets a certain condition.
- TalkBack - Use this app to queue up commands and then allow a device to act upon these queued commands.

- Timecontrol - Using this app, we can do a ThingTweet, ThingHTTP or a TalkBack at a specified time in the future. We can also use this to allow these actions to happen at a specified time throughout the week.

In addition to the above, ThingSpeak allows us to create the ThingSpeak applications as plugins using HTML, CSS and JavaScript which we can embed inside a website or inside our ThingSpeak channel.

One of the key elements of an IoT system is an IoT service. ThingSpeak is one such application platform offering a wide variety of features. At the heart of ThingSpeak is a channel which can be used for storing and processing data collected from the things. ThingSpeak also provides various apps for integration with web services, other APIs and social networks and provides the capability to create the applications as plugins. It is a great platform with extensive possibilities to explore the integration of the Internet of Things.

4.2.3 DESIGN & CODE

DESIGN

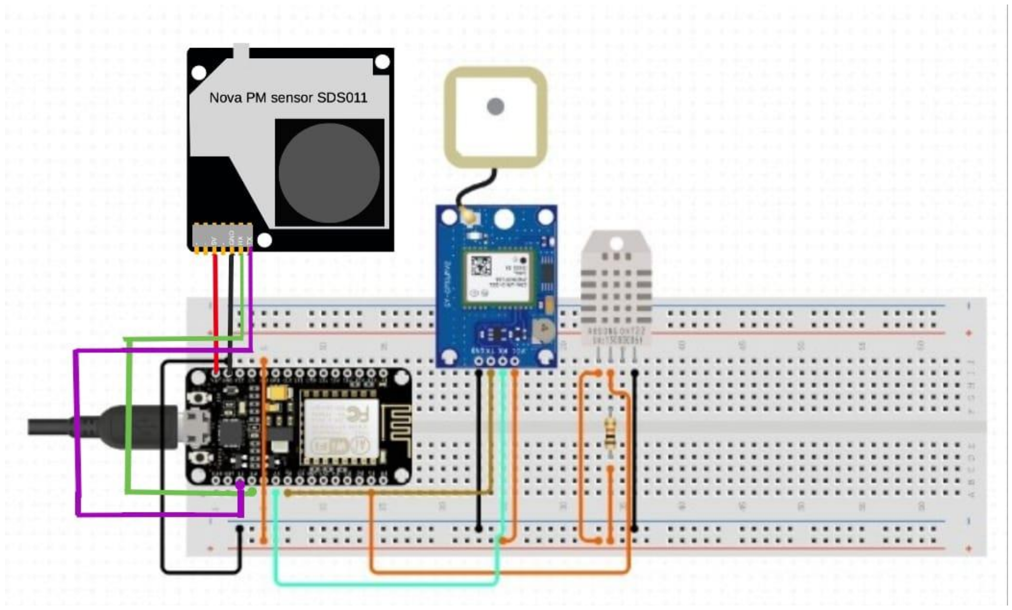


Figure 4.17: Hardware connections

Powering your ESP8266

There are three ways by which you can power your ESP8266 board

Micro USB Jack: Connect the mini-USB jack to a phone charger or computer through a cable and it will draw power required for the board to function

5V Pin: The 5V pin can be supplied with a Regulated 5V, this voltage will again be regulated to 3.3V through the on-board voltage regulator. Remember ESP8266 operated with 3.3V only.

3.3V Pin: If you have a regulated 3.3V supply, then you can directly provide this to the 3.3V pin of the ESP8266.

Input/output

The ESP8266 has 17 GPIO pins (0-16), however, you can only use 11 of them, because 6 pins (GPIO 6 - 11) are used to connect the flash memory chip.

- **UART interface:** The ESP8266 has two hardware UARTS (Serial ports): UART0 on pins 1 and 3 (TX0 and RX0 resp.), and UART1 on pins 2 and 8 (TX1 and RX1 resp.), however, GPIO8 is used to connect the flash chip. This means that UART1 can only transmit data.
- ISRs in ESP8266 are special kinds of functions that have some unique rules that most other functions do not have. An ISR cannot have any parameters, and they should not return anything. ISRs should be as short and fast as possible as they block normal program execution.
- CHANGE: Triggers the interrupt whenever the pin changes value, from HIGH to LOW ...
- FALLING: Triggers the interrupt when the pin goes from HIGH to LOW
- RISING: Triggers the interrupt when the pin goes from LOW to HIGH
- HIGH: Triggers the interrupt whenever the pin is HIGH

- **Reset Pin:** RST Pin. When the RST pin is pulled LOW, the ESP8266 resets. This is the same as pressing the on-board RESET button.

SOURCE CODE

```
#define SW_VERSION " ThinkSpeak.com" // SW version will appears at innitial LCD
Display

#include "SdsDustSensor.h"

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <SoftwareSerial.h>

#include <TinyGPS.h>

#include <DHT.h>

#define DHTPIN D5    // what digital pin we're connected to dht11

#define DHTTYPE DHT11   // DHT 22  (AM2302), AM2321)

DHT dht(DHTPIN, DHTTYPE);

float lat = 17.335492, lon = 78.289328; // create variable for latitude and longitude object

// used when we do not get the values from the gps module because of signal

SoftwareSerial gpsSerial(D3,D4);//rx,tx  for gps module

TinyGPS gps; // create gps object

uint32_t tsLastReport = 0;

void onBeatDetected(){

}

const char* MY_SSID = "miniproject";    // our wifi name

const char* MY_PWD = "12345678";     // our wifi password
```

```
 WiFiClient client;

const char* TS_SERVER = "api.thingspeak.com";  // data base

String TS_API_KEY = "6NBC1LPSPSATKZYR"; // api key should be kept ours

int rxPin = D7;                                   // defining data inputs for pm sensor

int txPin = D8;

SdsDustSensor sds(rxPin, txPin);

void connectWifi()

{

  Serial.print("Connecting to " + *MY_SSID);

  WiFi.begin(MY_SSID, MY_PWD);

  while (WiFi.status() != WL_CONNECTED)

  {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi Connected");

  Serial.println("");

}

/*

   Sending Data to Thinkspeak Channel
 **/
void sendDataTS(void)
{
  while(gpsSerial.available()){ // check for gps data
  if(gps.encode(gpsSerial.read()))// encode gps data
  {
  gps.f_get_position(&lat,&lon); // get latitude and longitude
  Serial.print("Position: ");
  Serial.print("Latitude:");
  Serial.print(lat,6);
  Serial.print(";");
  Serial.print("Longitude:");
  Serial.println(lon,6);
  Serial.print(lat);
  Serial.print(" ");
 }
}
String latitude = String(lat,6);
  String longitude = String(lon,6);
Serial.println(latitude+";"+longitude);
delay(1000);
```

```
PmResult pm = sds.readPm();
  if (pm.isOk()) {
    Serial.print("PM2.5 = ");
    Serial.print(pm.pm25);
    Serial.print(", PM10 = ");
    Serial.println(pm.pm10);
    // if you want to just print the measured values, you can use toString() method as well
    Serial.println(pm.toString());
  } else {
    // notice that loop delay is set to 0.5s and some reads are not available
    Serial.print("Could not read values from sensor, reason: ");
    Serial.println(pm.statusToString());
  }
  delay(500);
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
```

```
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
}
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);
Serial.print(F("Humidity: "));
Serial.println(h);
delay(1000);
 Serial.print(F("Temperature: "));
Serial.println(t);
delay(1000);
if (client.connect(TS_SERVER, 80))
{
  String postStr = TS_API_KEY;
  postStr += "&field1=";
  postStr += String(h);
   postStr += "&field2=";
  postStr += String(t);
   postStr += "&field3=";
```

```
    postStr += String(pm.pm25);

    postStr += "&field4=";

    postStr += String(pm.pm10);

    postStr += "&field5=";

    postStr += String(lat);

    postStr += "&field6=";

    postStr += String(lon);

    postStr += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");

    client.print("Host: api.thingspeak.com\n");

    client.print("Connection: close\n");

    client.print("X-THINGSPEAKAPIKEY: " + TS_API_KEY + "\n");

    client.print("Content-Type: application/x-www-form-urlencoded\n");

    client.print("Content-Length: ");

    client.print(postStr.length());

    client.print("\n\n");

    client.print(postStr);

    delay(1000);

  }

  client.stop();

}
```

```
void setup()
{
  Serial.begin(9600);
  delay(10);
  connectWifi();
  sds.begin();
  Serial.println("The GPS Received Signal:");
  gpsSerial.begin(9600); // connect gps sensor
  Serial.println(sds.queryFirmwareVersion().toString()); // prints firmware version
  Serial.println(sds.setActiveReportingMode().toString()); // ensures sensor is in 'active'
reporting mode
  Serial.println(sds.setContinuousWorkingPeriod().toString()); // ensures sensor has
continuous working period - default but not recommended
  dht.begin();
}
void loop()
{
  sendDataTS();
  delay(10);
}
/****
#include <TinyGPS++.h>
```

```
#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

TinyGPSPlus gps;

SoftwareSerial SerialGPS(D3, D4);

const char* ssid = "miniproject";

const char* password = "12345678";

float Latitude , Longitude;

int year , month , date, hour , minute , second;

String DateString , TimeString , LatitudeString , LongitudeString;

WiFiServer server(80);

void setup()

{

  Serial.begin(9600);

  SerialGPS.begin(9600);

  Serial.println();

  Serial.print("Connecting");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)

  {

    delay(500);

    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
}
void loop()
{
  while (SerialGPS.available() > 0)
    if (gps.encode(SerialGPS.read()))
    {
      if (gps.location.isValid())
      {
        Latitude = gps.location.lat();
        LatitudeString = String(Latitude , 6);
        Longitude = gps.location.lng();
        LongitudeString = String(Longitude , 6);
      }
      if (gps.date.isValid())
      {
```

```
  DateString = "";
  date = gps.date.day();
  month = gps.date.month();
  year = gps.date.year();
  if (date < 10)
  DateString = '0';
  DateString += String(date);
  DateString += " / ";
  if (month < 10)
  DateString += '0';
  DateString += String(month);
  DateString += " / ";
  if (year < 10)
  DateString += '0';
  DateString += String(year);
}
if (gps.time.isValid())
{
  TimeString = "";
  hour = gps.time.hour()+ 5; //adjust UTC
  minute = gps.time.minute();
```

```
      second = gps.time.second();

      if (hour < 10)

      TimeString = '0';

      TimeString += String(hour);

      TimeString += " : ";

      if (minute < 10)

      TimeString += '0';

      TimeString += String(minute);

      TimeString += " : ";

      if (second < 10)

      TimeString += '0';

      TimeString += String(second);

     }

   }

  WiFiClient client = server.available();

  if (!client)

  {

    return;

  }

  //Response

  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html>
 <html> <head> <title>NEO-6M GPS Readings</title> <style>";
```

```
  s += "table, th, td {border: 1px solid blue;} </style> </head> <body> <h1  style=";

  s += "font-size:300%;";

  s += " ALIGN=CENTER>NEO-6M GPS Readings</h1>";

  s += "<p ALIGN=CENTER style=""font-size:150%;""";

  s += "> <b>Location Details</b></p> <table ALIGN=CENTER style=";

  s += "width:50%";

  s += "> <tr> <th>Latitude</th>";

  s += "<td ALIGN=CENTER >";

  s += LatitudeString;

  s += "</td> </tr> <tr> <th>Longitude</th> <td ALIGN=CENTER >";

  s += LongitudeString;

  s += "</td> </tr> <tr>  <th>Date</th> <td ALIGN=CENTER >";

  s += DateString;

  s += "</td></tr> <tr> <th>Time</th> <td ALIGN=CENTER >";

  s += TimeString;

  s += "</td>  </tr> </table> ";

  if (gps.location.isValid())

  {

    s += "<p align=center><a style=""color:RED;font-size:125%;""
href=""http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=";

    s += LatitudeString;

    s += "+";
    s += LongitudeString;

    s += """ target=""_top"">Click here</a> to open the location in Google Maps.</p>";

  }


  s += "</body> </html> \n";

  client.print(s);

  delay(100);

}

****/
```

CHAPTER 5

RESULTS AND DISCUSSION

After the completion of the project kit assembling and all the connections made, the results of how each sensor is working and uploading the data to the cloud server are as followed below:

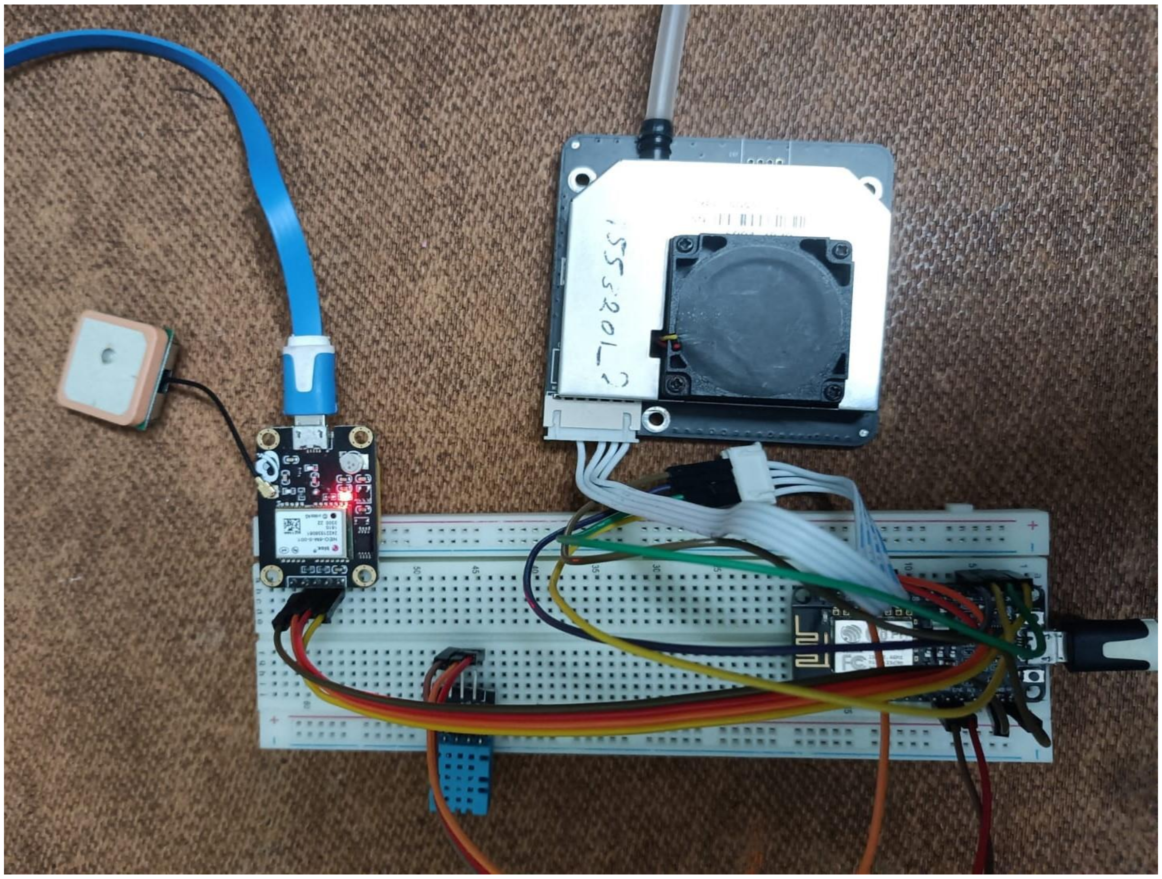


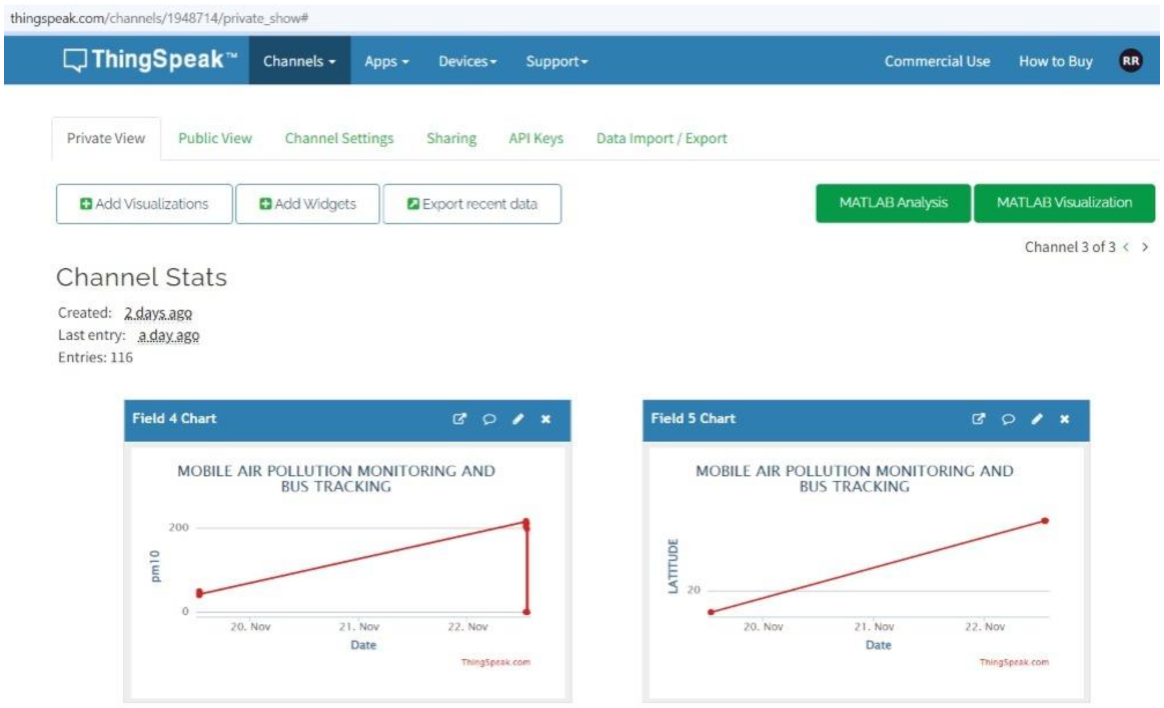
Figure 5.1 Circuit Connections

5.1 ESP8266

It is interfaced with all the three available sensors (GPS, DHT-11, SDS011). It acts as a medium to send and receive data from the connected sensors. The inputs are received from each sensor and combined and sent to the cloud using its in built Wi-Fi or using an additional GSM Modem.

5.2 SDS011

Firstly, sensor readings are in the form of analog data is sent to ESP8266, then it converts the analog data into digital data and fed to ThingSpeak server, then readings will be displayed on ThingSpeak server console.



Channel Stats

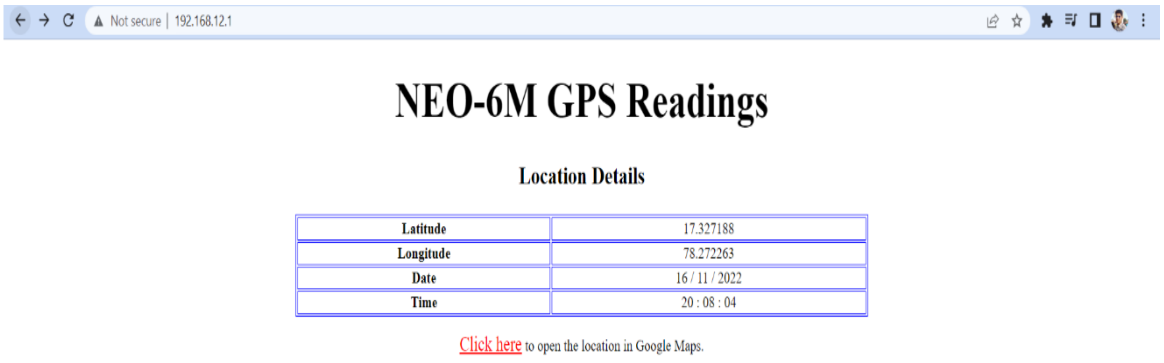
Created: 2 days ago
Last entry: a day ago
Entries: 116

MOBILE AIR POLLUTION MONITORING AND BUS TRACKING
pm10
200
0
20. Nov
21. Nov
22. Nov
Date
MOBILE AIR POLLUTION MONITORING AND BUS TRACKING
LATITUDE
20
20. Nov
21. Nov
22. Nov
Date

Figure 5.2 ThingSpeak Graph PM sensor readings

5.3 GPS Sensor

Firstly, sensor readings are in the form of analog data is sent to ESP8266, then it converts the analog data into digital data and fed to ThingSpeak server, then readings will be displayed on ThingSpeak server console.



NEO-6M GPS Readings

Location Details

Latitude	17.327188
Longitude	78.272263
Date	16 / 11 / 2022
Time	20 : 08 : 04

Click here to open the location in Google Maps.

Figure 5.3 GPS Readings in Server

When the server IP address is searched in a browser it opens up the information tab, which consists of latitude, longitude data along with date and time. Here we can also observe a hypertext link. Clicking on it will redirect us to the location coordinates mentioned above and it will display on the google maps as shown below.

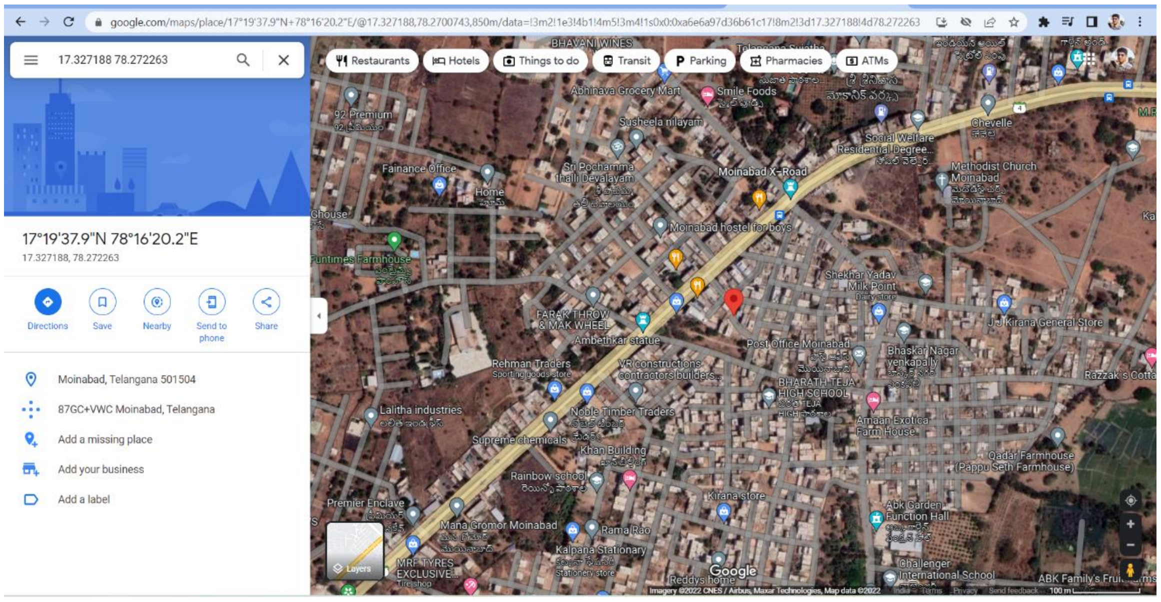


Fig 5.4 Redirect page of location from server

The server data will also be updated to the cloud through ThingSpeak and latitude, longitude and a map will be displayed in each of the specified field.

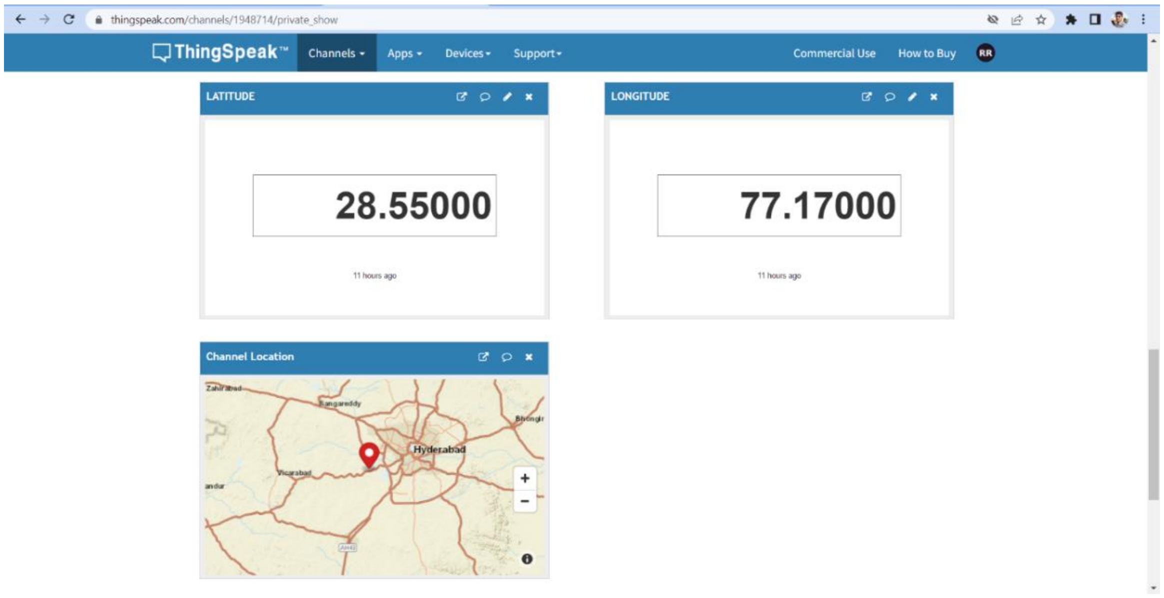


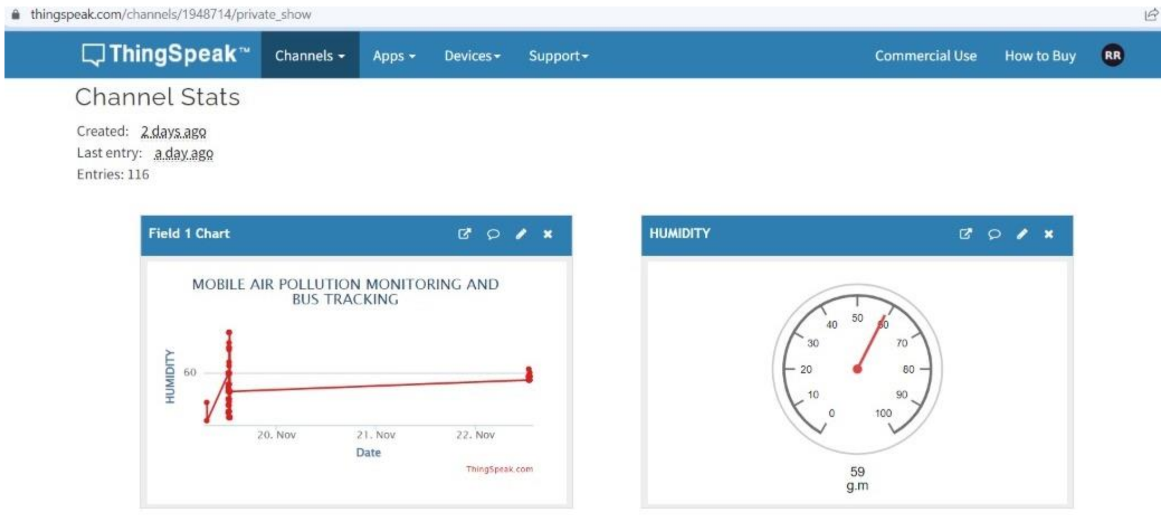
Figure 5.5 GPS module ThingSpeak readings and map

5.4 DHT11 Sensor

Firstly, sensor readings are in the form of analog data is sent to ESP32, then it converts the analog data into digital data and fed to ThingSpeak server, then readings will be displayed on ThingSpeak server console.

DHT11 Sensor measures the temperature and humidity of the surroundings.

The output in shown in ThingSpeak Fig 5.6 and Fig 5.7 below.


thingspeak.com/channels/1948714/private_show

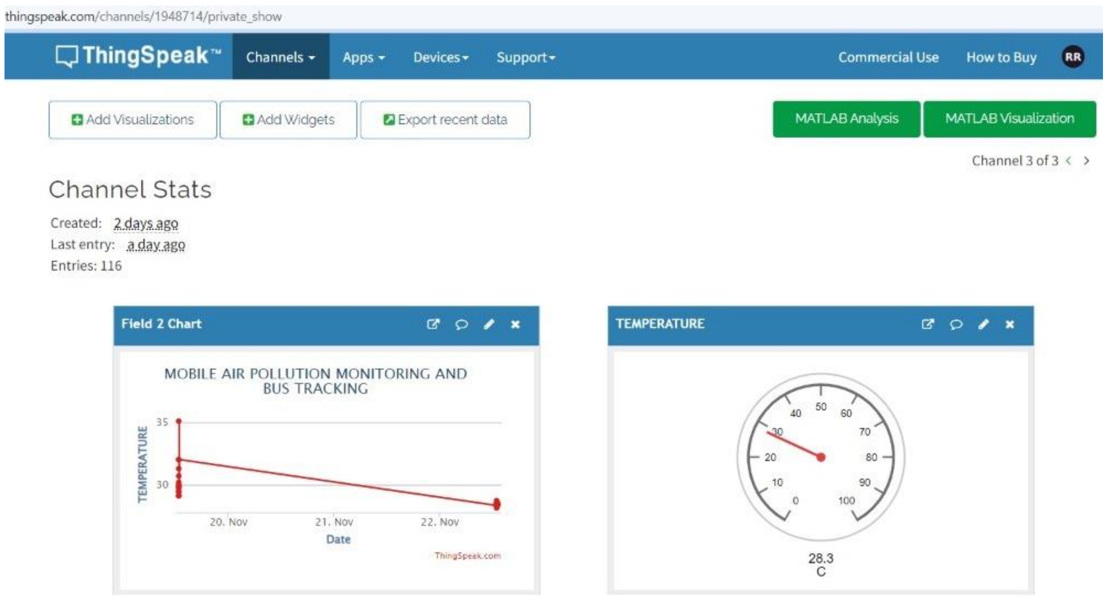
Channel Stats

Created: 2 days ago

Last entry: a day ago

Entries: 116

Figure 5.6 ThingSpeak Graph of Humidity


thingspeak.com/channels/1948714/private_show

Channel Stats

Created: 2 days ago

Last entry: a day ago

Entries: 116

Figure 5.7 ThingSpeak Graph of Temperature

The readings are displayed on graph and in gauge as shown in Fig 5.6 and Fig5.7. It also includes date in it.

The overall advantage with cloud is that, to analyze and visualize data.

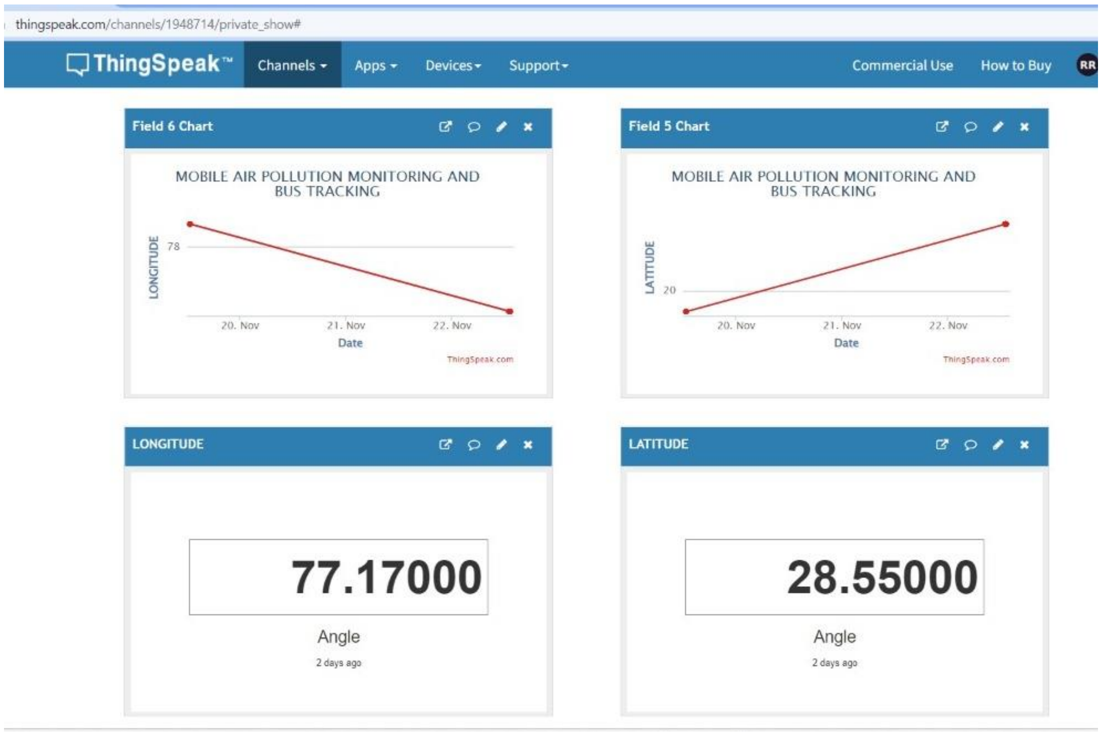
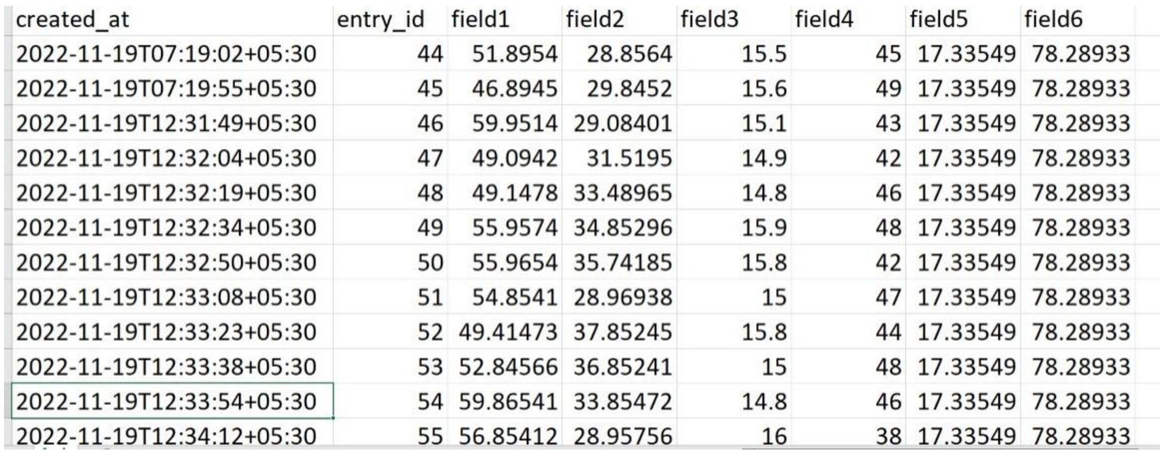


Figure 5.8 ThingSpeak Graph of Temperature and Humidity and location Readings

The past data can also be stored and accessed at any given time as shown in Fig 5.9 below.



created_at	entry_id	field1	field2	field3	field4	field5	field6
2022-11-19T07:19:02+05:30	44	51.8954	28.8564	15.5	45	17.33549	78.28933
2022-11-19T07:19:55+05:30	45	46.8945	29.8452	15.6	49	17.33549	78.28933
2022-11-19T12:31:49+05:30	46	59.9514	29.08401	15.1	43	17.33549	78.28933
2022-11-19T12:32:04+05:30	47	49.0942	31.5195	14.9	42	17.33549	78.28933
2022-11-19T12:32:19+05:30	48	49.1478	33.48965	14.8	46	17.33549	78.28933
2022-11-19T12:32:34+05:30	49	55.9574	34.85296	15.9	48	17.33549	78.28933
2022-11-19T12:32:50+05:30	50	55.9654	35.74185	15.8	42	17.33549	78.28933
2022-11-19T12:33:08+05:30	51	54.8541	28.96938	15	47	17.33549	78.28933
2022-11-19T12:33:23+05:30	52	49.41473	37.85245	15.8	44	17.33549	78.28933
2022-11-19T12:33:38+05:30	53	52.84566	36.85241	15	48	17.33549	78.28933
2022-11-19T12:33:54+05:30	54	59.86541	33.85472	14.8	46	17.33549	78.28933
2022-11-19T12:34:12+05:30	55	56.85412	28.95756	16	38	17.33549	78.28933

Figure 5.9 Past collected Readings in an Excel Sheet

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The project titled "**MOBILE AIR POLLUTION MONITORING AND BUS TRACKING SYSTEM FOR METROPOLITAN AREAS**" is a model for bus tracking unit with the help of gps receivers and GSM modem and it is better scheduling or route planning can enable you handle larger jobs loads within a particular time. Bus tracking both in case of personal as well as business purpose improves safety and security, communication medium, performance monitoring and increases productivity. As well as the air pollution monitoring will be done parallelly, the different areas have different levels of air quality at different times it is important for us to monitor what is happening. In this way we can identify trouble spots and ensure that we are taking the right steps. So, in the coming year, it is going to play a major role in our day-to-day living. We have completed the project as per the requirements of our project.

6.2 FUTURE SCOPE

1. The proposed system can be connected to a GSM module and when the air pollution levels cross a certain threshold, it can immediately alert the local pollution board and other concerned parties.

2. We can reduce the size of the kit and increase accuracy up to 3m by using more standard GPS receivers.

3.The proposed system can be modified and can also be upgraded to make more effective and real time we can add air purifier to the existing model which might also be bit complex.

4.The proposed system can be upgraded to collection of data like carbon dioxide, sulfur oxide, nitrogen oxide and so on so that we can accurately judge the air pollutant levels in the air at the specific location.

REFERENCES

[1] P. Das, S. Ghosh, S. Chatterjee and S. De, "A Low Cost Outdoor Air Pollution Monitoring Device With Power Controlled Built-In PM Sensor," in *IEEE Sensors Journal*, vol. 22, no. 13, pp. 13682-13695, 1 July1, 2022, doi: 10.1109/JSEN.2022.3175821.

[2] M. F. M. A. Hakeem, N. A. Sulaiman, M. Kassim and N. M. Isa, "IoT Bus Monitoring System via Mobile Application," *2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2022, pp. 125-130, doi: 10.1109/I2CACIS54679.2022.9815268.

[3] B. B. Humaïra and A. Chiniah, "An Adaptive Communication Model for Android Bus Tracking App," *2021 2nd Global Conference for Advancement in Technology (GCAT)*, 2021, pp. 1-6, doi: 10.1109/GCAT52182.2021.9587657.

[4] T. W. Ayele and R. Mehta, "Air pollution monitoring and prediction using IoT," *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 1741-1745, doi: 10.1109/ICICCT.2018.8473272.

[5] V. Shakhov and O. Sokolova, "On Modeling Air Pollution Detection With Internet of Vehicles," *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2021, pp. 1-3, doi: 10.1109/IMCOM51814.2021.9377350.

[6] R. S. Krishnan, S. Manikandan, J. R. F. Raj, K. L. Narayanan and Y. H. Robinson, "Android Application based Smart Bus Transportation System for Pandemic Situations," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021, pp. 938-942, doi: 10.1109/ICICV50876.2021.9388625.

[7] H. Gull, D. Aljohar, R. Alutaibi, D. Alqahtani, M. Alarfaj and R. Alqahtani, "Smart School Bus Tracking: Requirements and Design of an IoT based School Bus Tracking

System," *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2021, pp. 388-394, doi: 10.1109/ICOEI51242.2021.9452818.

[8] S. A. E. Yosif, M. M. Abdelwahab, M. Abd Elrahman ALagab and F. Muhammad, "Design of bus tracking and fuel monitoring system," *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, 2017, pp. 1-5, doi: 10.1109/ICCCCEE.2017.7867679.

[9] R. Akter, M. J. H. Khandaker, S. Ahmed, M. M. Mugdho and A. K. M. B. Haque, "RFID based Smart Transportation System with Android Application," *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 614-619, doi: 10.1109/ICIMIA48430.2020.9074869.

[10] H. Liu, H. Xu, Y. Yan, Z. Cai, T. Sun and W. Li, "Bus Arrival Time Prediction Based on LSTM and Spatial-Temporal Feature Vector," in *IEEE Access*, vol. 8, pp. 11917-11929, 2020, doi: 10.1109/ACCESS.2020.2965094.

[11] P. Ferrer-Cid, J. M. Barcelo-Ordinas and J. Garcia-Vidal, "Graph Learning Techniques Using Structured Data for IoT Air Pollution Monitoring Platforms," in *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13652-13663, 1 Sept.1, 2021, doi: 10.1109/JIOT.2021.3067717.

[12] B. Perumal, J. Deny, K. Alekhya, V. Maneesha and M. Vaishnavi, "Air Pollution Monitoring System by using Arduino IDE," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 797-802, doi: 10.1109/ICESC51422.2021.9533007.

[13] K. Ammar, M. Jalmoud, A. Boushehri and K. Fakhro, "A Real-time School Bus Tracking and Monitoring System," *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 0654-0660, doi: 10.1109/IEMCON.2019.8936199.

[14] S. Akter, T. Islam, R. F. Olanrewaju and A. A. Binyamin, "A Cloud-Based Bus Tracking System Based on Internet-of-Things Technology," *2019 7th International*

Conference on Mechatronics Engineering (ICOM), 2019, pp. 1-5, doi: 10.1109/ICOM47790.2019.8952037.

[15] M. A. Hafiizh Nur, S. Hadiyoso, F. B. Belladina, D. N. Ramadan and I. Wijayanto, "Tracking, Arrival Time Estimator, and Passenger Information System on Bus Rapid Transit (BRT)," *2020 8th International Conference on Information and Communication Technology (ICoICT)*, 2020, pp. 1-4, doi: 10.1109/ICoICT49345.2020.9166375.

[16] K. Premkumar, P. K., P. J., P. D. and P. P., "College Bus Tracking and Notification System," *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2020, pp. 1-4, doi: 10.1109/ICSCAN49426.2020.9262303.