# 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).

|  |  |  |
| --- | --- | --- |
| 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.

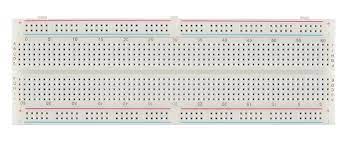


Figure 4.3 Bread Board

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 PM2.5 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 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.

Figure 4.5 SDS 011 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:**

|  |  |
| --- | --- |
| 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:**

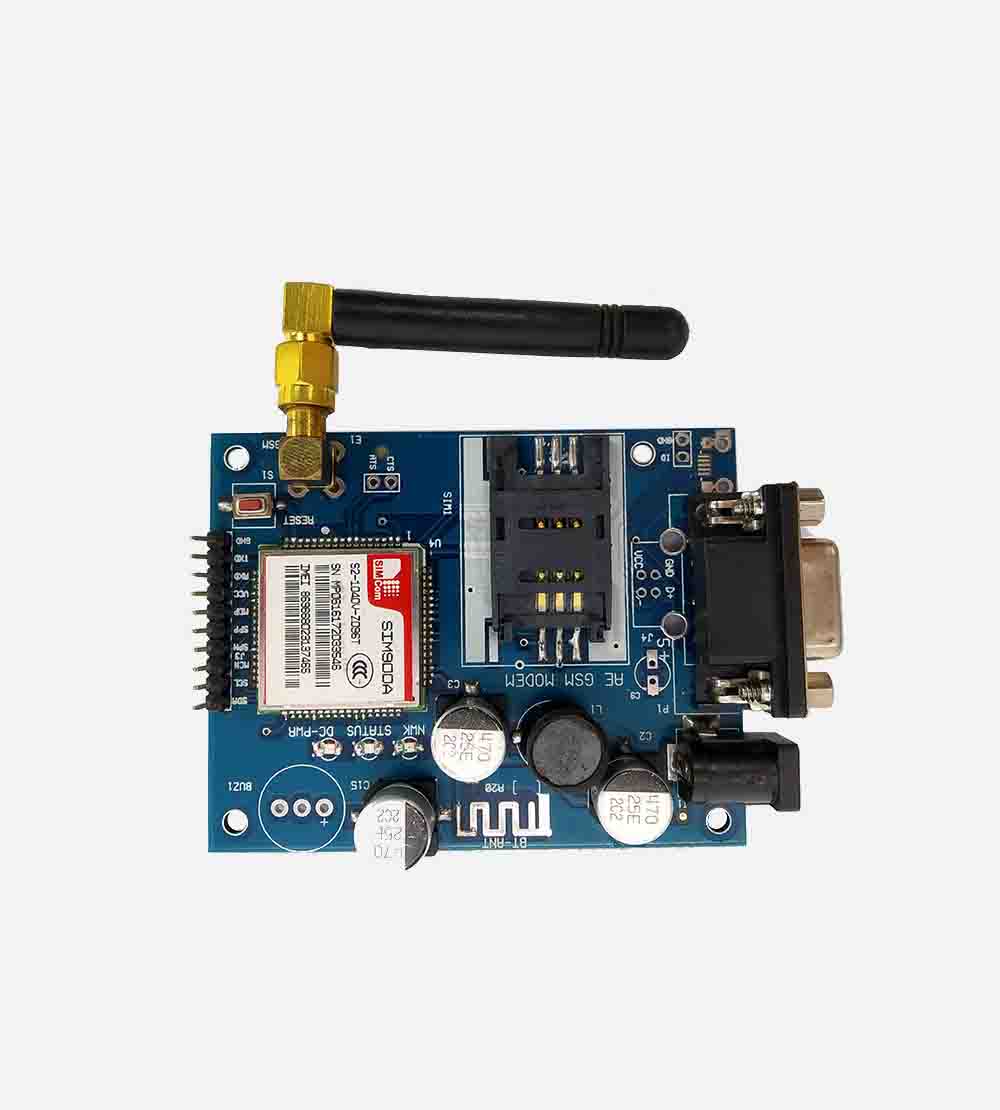
Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular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

Figure 4.6 GSM Module

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.

**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.

**BASIC STRUCTURE OF GPS:**

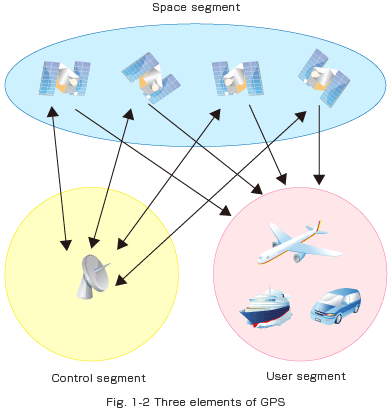
****

Figure 4.7.1 Three Elements of GPS

**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)

**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:**

|  |  |
| --- | --- |
| 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 |

* 1. **SOFTWARE SPECIFICATIONS**
     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.’

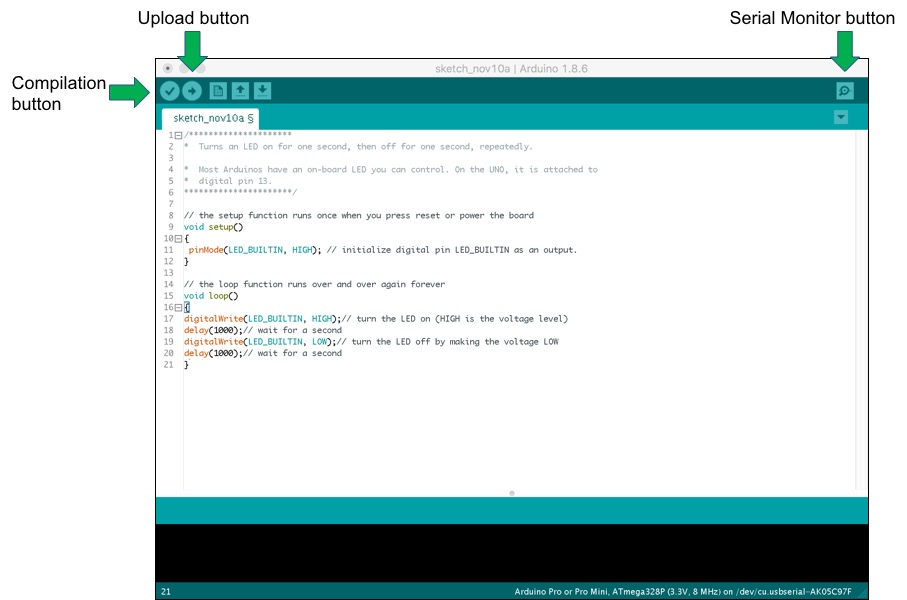
There are two required functions in an Arduino sketch, **setup () and loop ()**. Other functions must be created outside the brackets of those two functions. As an example, we will create a simple function to multiply two numbers. It has two elements, hardware and software that form a system to rapidly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

Figure 4.8 Arduino IDE main window

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.

**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.

**THE 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);

}

\*\*\*\*/