PROJECT REPORT

18CSE309T-DESIGN OF SMART SPACE MANAGEMENT

TITLE: Temperature and Humidity Monitoring System

SUBMITTED BY:
Guna shekar daram (RA2011026010208)

Aim: Temperature and Humidity Monitoring System can help ensure optimal conditions for people, equipment, and processes, while also reducing energy costs and improving sustainability.

Abstract: A Temperature and Humidity Monitoring System is a technological solution designed to measure and track temperature and humidity levels in a given environment. The system includes sensors that capture data at regular intervals, which can be stored locally or in the cloud for easy access and analysis. The system also provides alerts when temperature or humidity levels exceed predefined thresholds, and visualizes the data in real-time using charts, graphs, or dashboards. Analytics can be performed on the data to identify trends and anomalies, and the system can be integrated with other building automation and management systems for a comprehensive solution.

Introduction:

Temperature and humidity are important environmental parameters that are widely used in various applications. To monitor these parameters, sensors are used to capture data and provide real-time readings. In this report, we will discuss the use of an ESP8266 microcontroller board along with a DHT11 sensor to create a temperature and humidity sensor. The ESP8266 is a low-cost Wi-Fi microcontroller board that is popular for IoT projects due to its integrated Wi-Fi module.

To create a temperature and humidity sensor using ESP, you will need the following:

1. ESP8266 or ESP32 microcontroller board
2. DHT11 temperature and humidity sensor
3. Breadboard and jumper wires
4. USB cable for power and programming
5. Arduino IDE or ESP-IDF software development kit (SDK)

Applications:

Temperature and humidity sensors are used in various industries and applications to monitor and control the environment. Here are some general real-world usages of temperature and humidity sensors:

- HVAC systems: Temperature and humidity sensors are used in heating, ventilation, and air conditioning (HVAC) systems to monitor and control the indoor environment. The data obtained from the sensors can be used to adjust the temperature and humidity levels to ensure optimal comfort for the occupants.

- Agriculture: Temperature and humidity sensors are used in agriculture to monitor the environment in greenhouses and control the temperature and humidity levels to ensure optimal conditions for plant growth.

- Food and beverage industry: Temperature and humidity sensors are used in the food and beverage industry to monitor the temperature and humidity levels during food processing and storage. The sensors can help ensure that the food and beverages are stored at the proper temperature and humidity levels to prevent spoilage.

- Medical industry: Temperature and humidity sensors are used in the medical industry to monitor the temperature and humidity levels in hospitals and laboratories. The sensors can help ensure that the environment is sterile and optimal for the storage of medications and medical supplies.

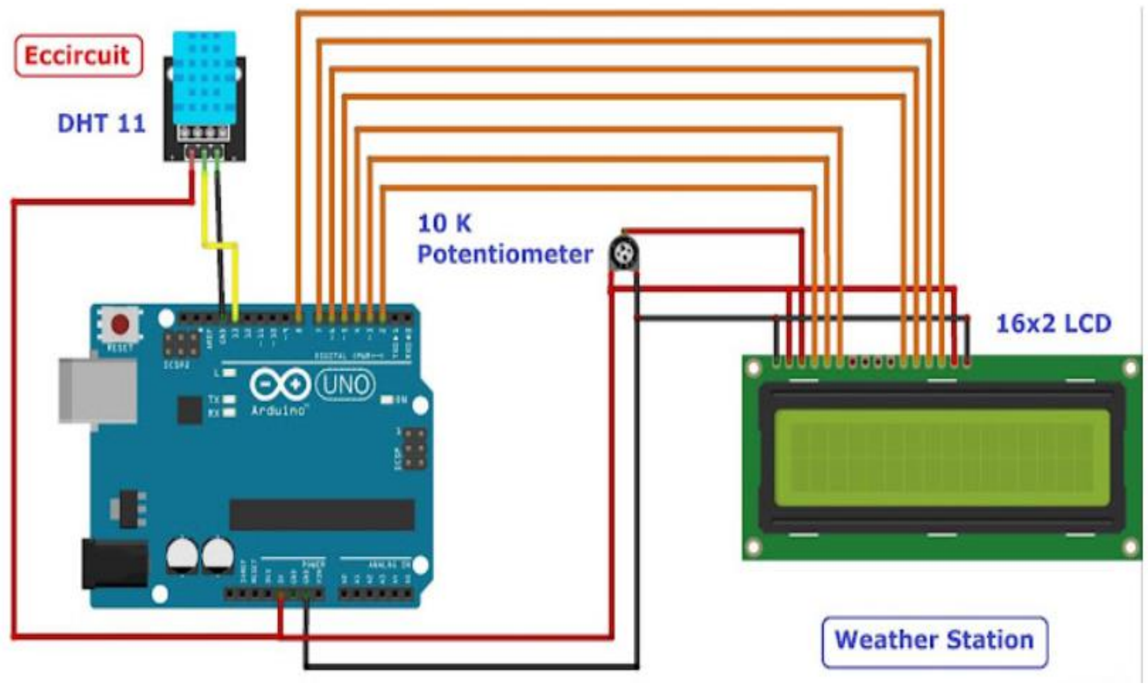
- Weather forecasting: Temperature and humidity sensors are used in weather forecasting to measure the atmospheric conditions and provide real-time data on weather patterns.

- Automotive industry: Temperature and humidity sensors are used in the automotive industry to monitor the temperature and humidity levels inside vehicles. The sensors can help ensure optimal comfort for the occupants.

In summary, temperature and humidity sensors are used in various industries and applications to monitor and control the environment. The sensors can help ensure optimal conditions for plant growth, food storage, medical supplies, and automotive comfort.

Here are the steps to follow:

1. Connect the DHT sensor to the ESP board. Connect the sensor's VCC pin to the 3.3V pin on the ESP board, GND to GND, and the data pin to a digital input pin (e.g. D2).
2. Open the Arduino IDE or ESP-IDF software development kit (SDK) and create a new sketch or project.
3. Add the DHT library to your project by going to "Sketch -> Include Library -> DHT sensor library". This library will provide functions to read the temperature and humidity from the DHT sensor.
4. In the setup() function, initialize the DHT sensor by calling dht.begin() and set the digital input pin used by the sensor by calling dht.attach(pin).
5. In the loop() function, call the dht.readTemperature() and dht.readHumidity() functions to read the temperature and humidity from the sensor.
6. Print the temperature and humidity values to the serial monitor using Serial.println().
7. Upload the sketch to the ESP board and open the serial monitor to view the temperature and humidity readings.



Working:

The temperature and humidity sensor using ESP works by using the ESP8266 or ESP32 microcontroller board along with the DHT11 or DHT22 temperature and humidity sensor.

The ESP8266 or ESP32 board is a microcontroller board that is equipped with WiFi capabilities, making it an ideal choice for IoT projects. The board communicates with the DHT11 or DHT22 sensor to obtain temperature and humidity readings.

The DHT11 or DHT22 sensor is a digital sensor that uses a thermistor and a capacitive humidity sensor to measure the temperature and humidity, respectively. The sensor outputs a digital signal that is read by the ESP board. The DHT11 sensor has a lower accuracy compared to the DHT22, but it is more affordable.

The ESP board is programmed to communicate with the DHT sensor using the DHT library, which provides functions to read the temperature and humidity values from the sensor. In the setup() function, the DHT sensor is initialized and the digital input pin used by the sensor is set. In the loop() function, the temperature and humidity values are read from the sensor using the dht.readTemperature() and dht.readHumidity() functions, respectively. The readings are then printed to the serial monitor using the Serial.println() function.

The temperature and humidity sensor can be powered using a USB cable and can be programmed using the Arduino IDE or the ESP-IDF software development kit (SDK). The sensor can be used in various applications such as greenhouse monitoring, weather stations, and indoor climate control. The accuracy of the readings may vary depending on the quality of the sensor and environmental factors.

Code:

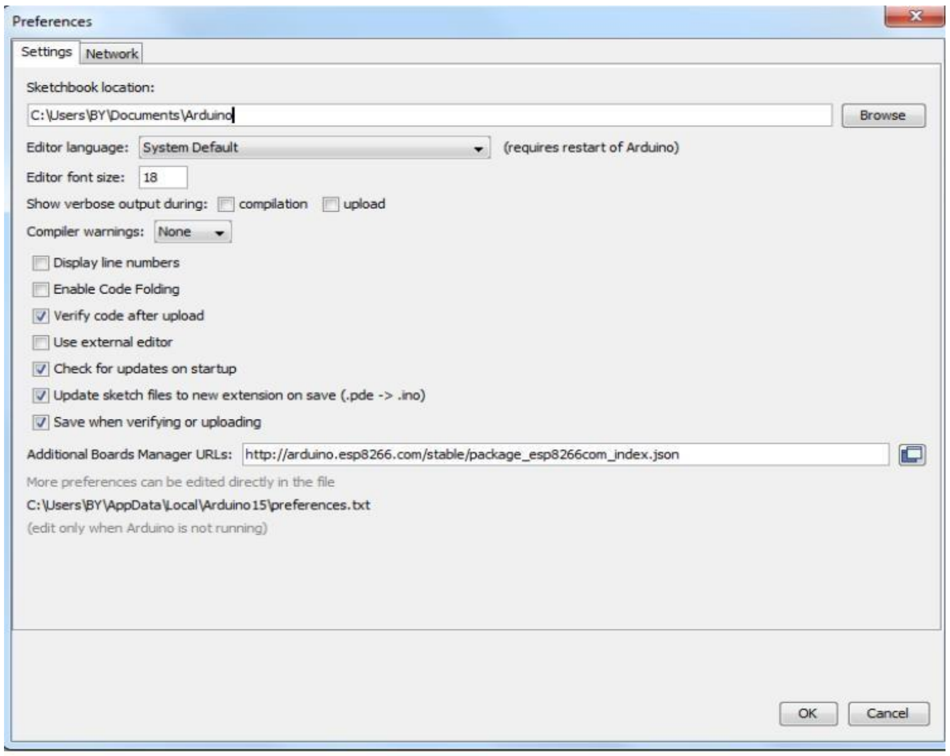
```
#define IO_USERNAME  "weather001"
#define IO_KEY       "aio_cYqW48UnKXvay8lzV5ZBocVMKZZX"

#define WIFI_SSID   "project1"
#define WIFI_PASS   "project1"

#include <DHT.h>                    //Including the DHT library
#include <Wire.h>                    //For using I2C connection of BMP180 in order to
connect it to the board

#define DHTPIN D5                              //Connect the DHT11 sensor's data pin to
GPIO2(D4) of Nodemcu
#define DHTTYPE DHT11                             //Mention the type of sensor we are
using, Here it it DHT11, for DHT22 just replace DHT11 with DHT22
DHT dht(DHTPIN, DHTTYPE); //Defining the pin and the dhttype

int mq;
int t; int
h;

AdafruitIO_Feed *s1 = io.feed("s1");
AdafruitIO_Feed *s2 = io.feed("s2");
AdafruitIO_Feed *s3 = io.feed("s3");

int cnt=0;

 void sendSensor()
{
        float h = dht.readHumidity();

        float t = dht.readTemperature(); // or dht.readTemperature(true) for
Fahrenheit
mq = analogRead(A0);

  s1->save(t);              s2-
>save(h);   s3->save(mq);
  Serial.println("updated");
 }


void setup()
 {
pinMode(2, OUTPUT);
digitalWrite(2, LOW);

          Serial.begin(9600); //Initializing the Serial Monitor with a Baud Rate of
9600
          while(! Serial);
          dht.begin();        //Initializing    the    DHT    sensor
delay(10);
          Serial.print("Connecting        to        Adafruit        IO");
io.connect();
          while(io.status() < AIO_CONNECTED)
          {
           Serial.print(".");
           delay(500);
         }
         Serial.println(io.statusText());
for(int i=0;i<5;i++)
{
digitalWrite(2,          LOW);
delay(200);   digitalWrite(2,
HIGH);
delay(200);
}
sendSensor();

}

void loop()
  {
io.run(); delay(15000);
sendSensor();
  }
```

Implementation:

The most basic way to use the ESP8266 module is to use serial commands, as the chip is basically a WiFi/Serial transceiver. However, this is not convenient. What we recommend is using the very cool Arduino ESP8266 project, which is a modified version of the Arduino IDE that you need to install on your computer. This makes it very convenient to use the ESP8266 chip as we will be using the well-known Arduino IDE. Following the below step to install ESP8266 library to work in Arduino IDE environment.

Install the Arduino IDE 1.6.4 or greater
Download Arduino IDE from Arduino.cc (1.6.4 or greater) - don't use 1.6.2 or lower version! You can use your existing IDE if you have already installed it. You can also try downloading the ready-to-go package from the ESP8266-Arduino project, if the proxy is giving you problems. 3.2

Install the ESP8266 Board Package Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences


Preferences

Settings | Network

Sketchbook location: C:\Users\BY\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 18

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

- [ ] Display line numbers
- [ ] Enable Code Folding
- [x] Verify code after upload
- [ ] Use external editor
- [x] Check for updates on startup
- [x] Update sketch files to new extension on save (.pde -> .ino)
- [x] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

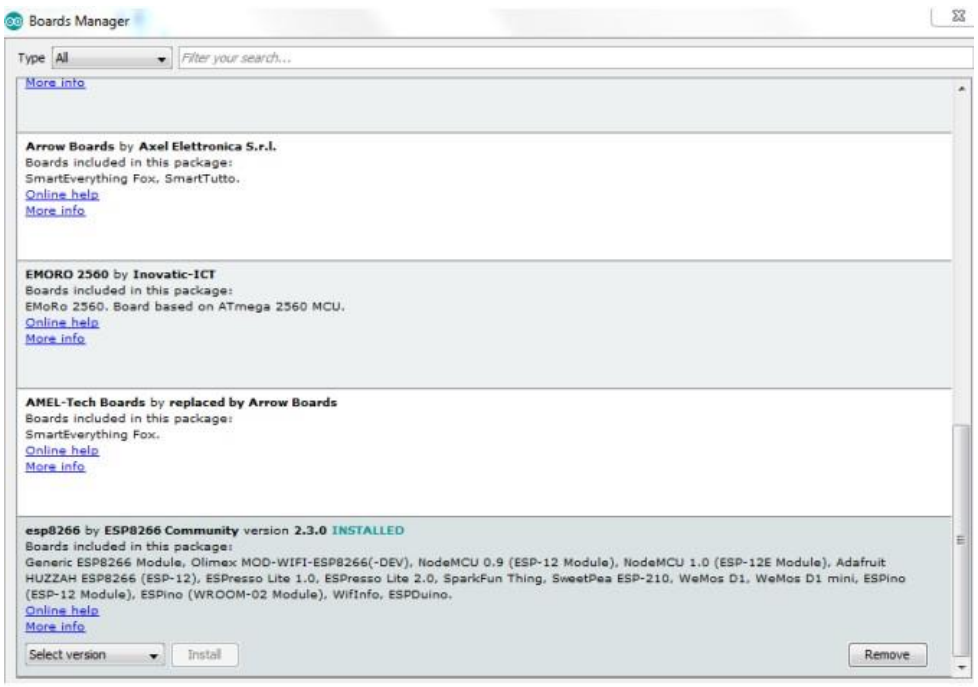
More preferences can be edited directly in the file

C:\Users\BY\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

Click 'File' -> 'Preferences' to access this panel.

Next, use the Board manager to install the ESP8266 package.


Boards Manager

Type All | Filter your search...

More info

Arrow Boards by **Axel Elettronica S.r.l.**
Boards included in this package:
SmartEverything Fox, SmartTutto.
Online help
More info

EMORO 2560 by **Inovatic-ICT**
Boards included in this package:
EMoRo 2560. Board based on ATmega 2560 MCU.
Online help
More info

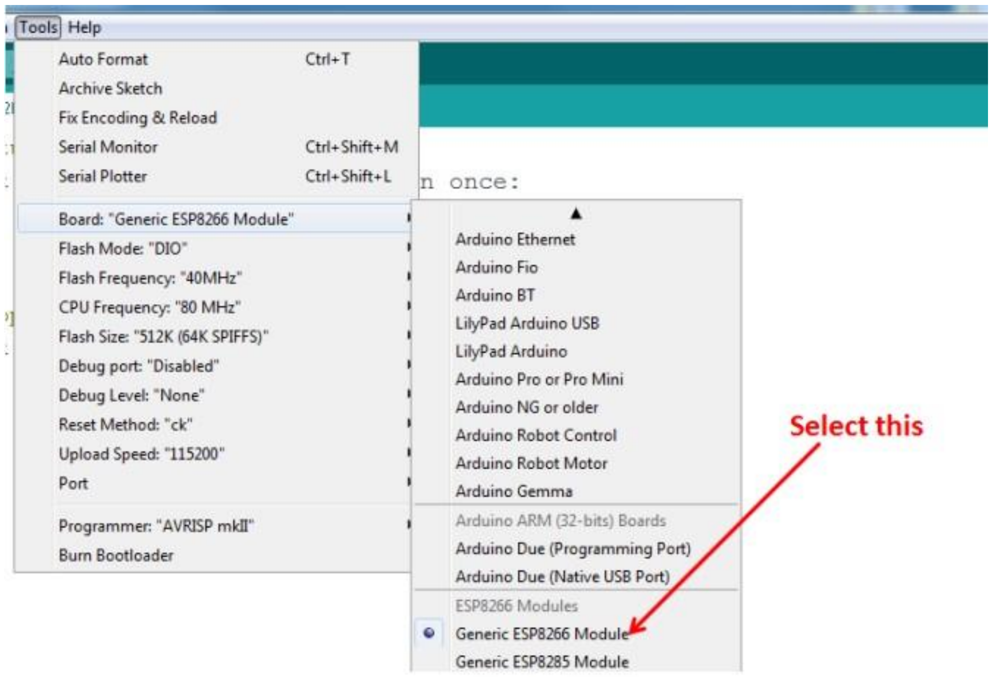
AMEL-Tech Boards by **replaced by Arrow Boards**
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by **ESP8266 Community** version **2.3.0** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

Click 'Tools' -> 'Board:' -> 'Board Manager...' to access this panel.

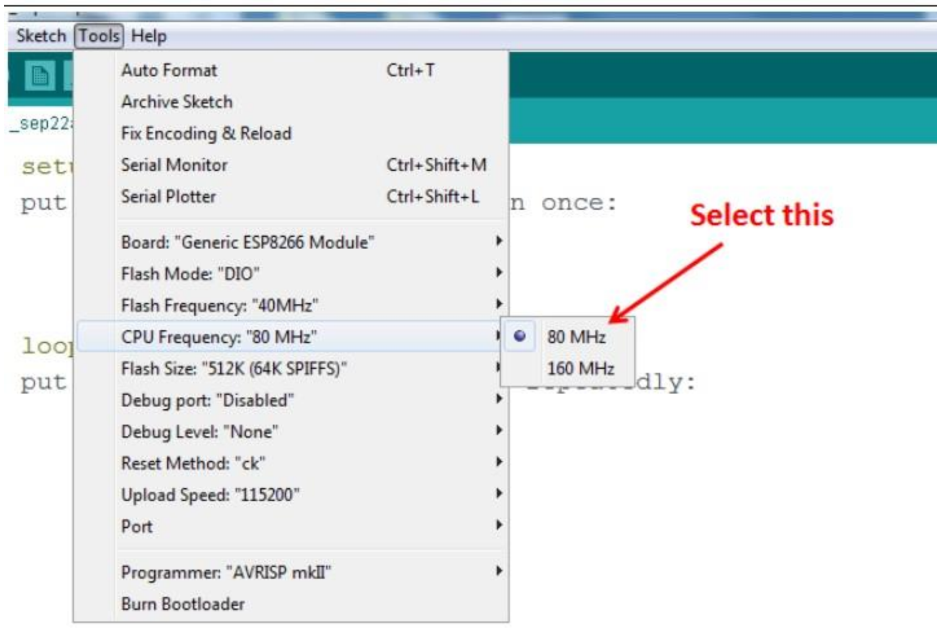
Scroll down to ' esp8266 by ESP8266 Community ' and click "Install" button to install the ESP8266 library package. Once installation completed, close and reopen Arduino IDE for ESP8266 library to take effect

3.3 Setup ESP8266 Support When you've restarted Arduino IDE, select ‘Generic ESP8266 Module’ from the ‘Tools’ -> ‘Board:’ dropdown menu.



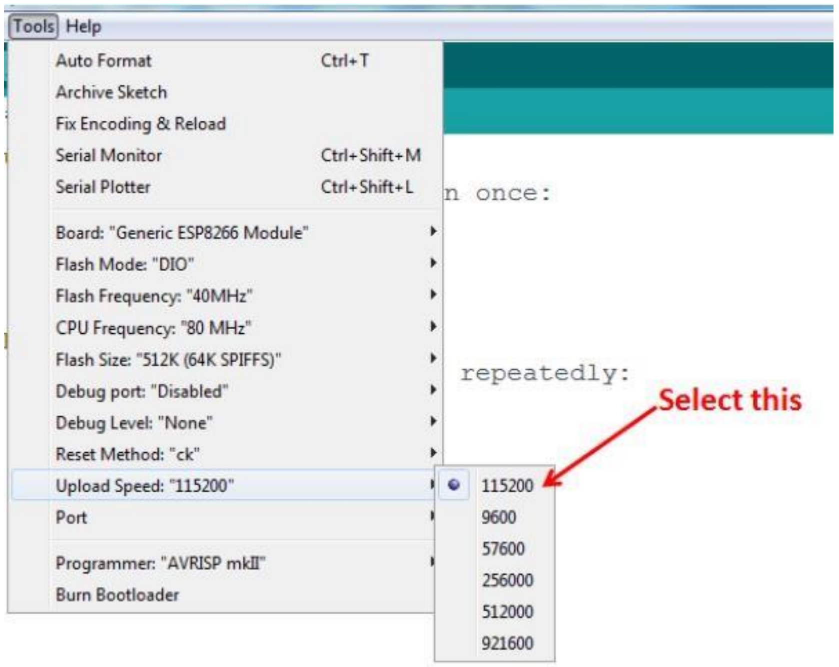
Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Programmer: "AVRISP mkII"

Select 80 MHz as the CPU frequency (you can try 160 MHz overclock later)

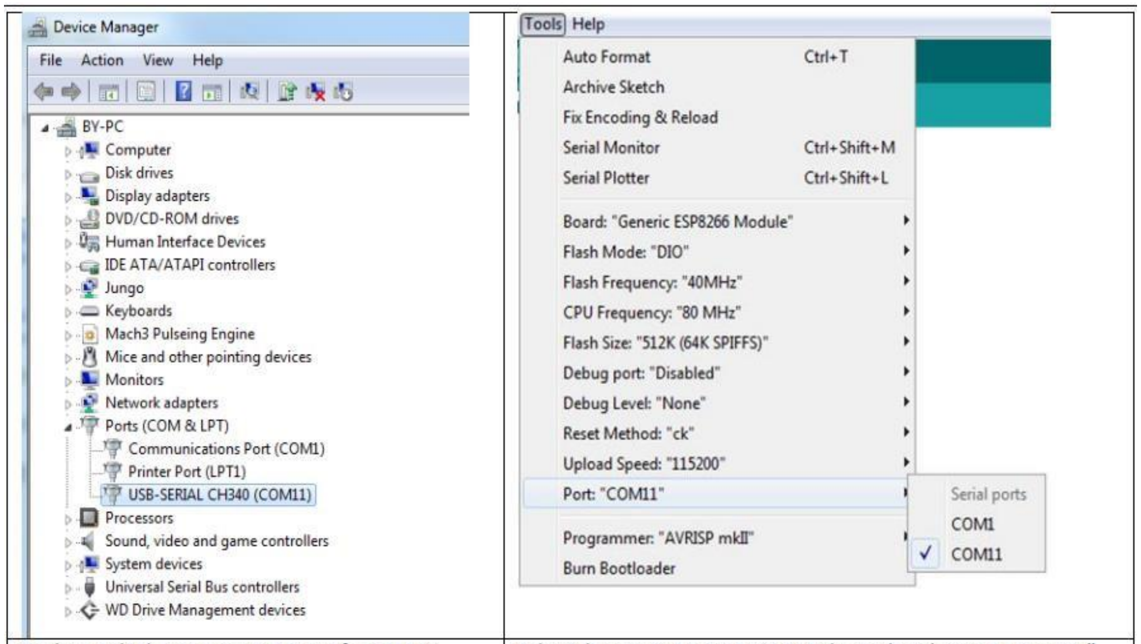


Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Programmer: "AVRISP mkII"

Select ‘115200’ baud upload speed is a good place to start - later on you can try higher speeds but 115200 is a good safe place to start.



Go to your Windows 'Device Manager' to find out which Com Port 'USB-Serial CH340' is assigned to. Select the matching COM/serial port for your CH340 USBSerial interface.



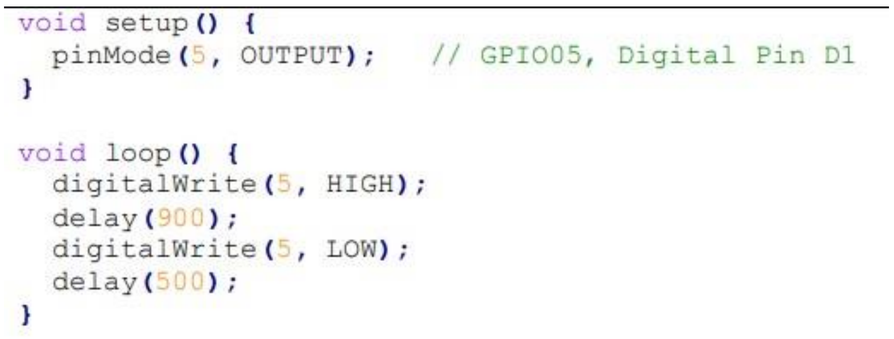
Find out which Com Port is assign for CH340	Select the correct Com Port as indicated on 'Device Manager"

Note: if this is your first time using CH340 " USB-to-Serial " interface, please install the driver first before proceed the above Com Port setting. The CH340 driver can be download from the below site:

https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers

Blink Test:

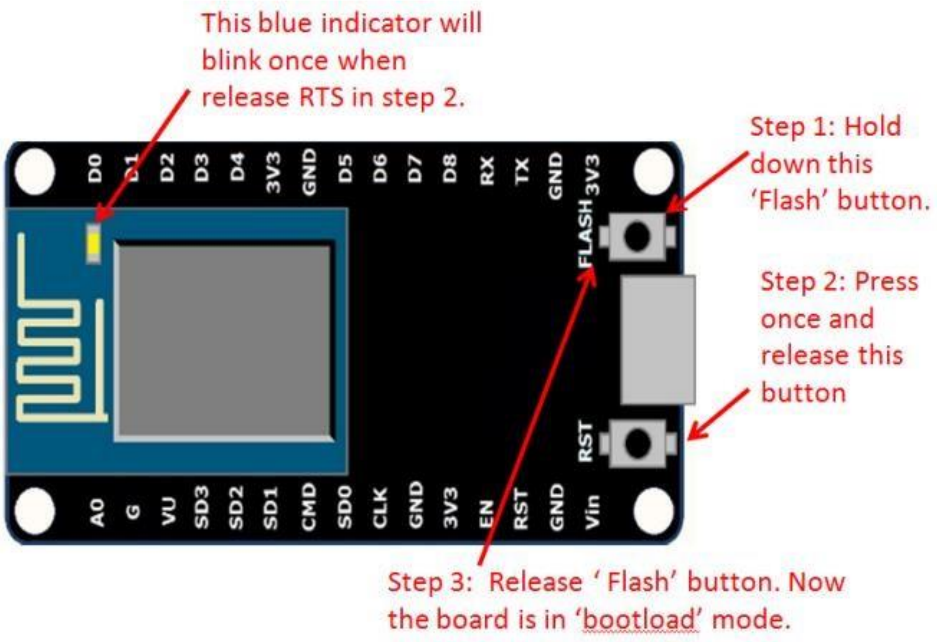
We'll begin with the simple blink test. Enter this into the sketch window (and save since you'll have to). Connect a LED as shown in Figure3-1

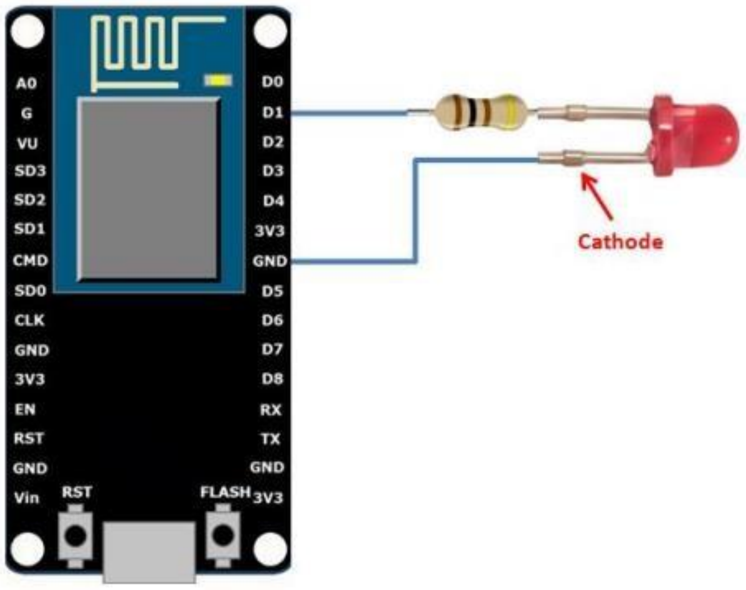


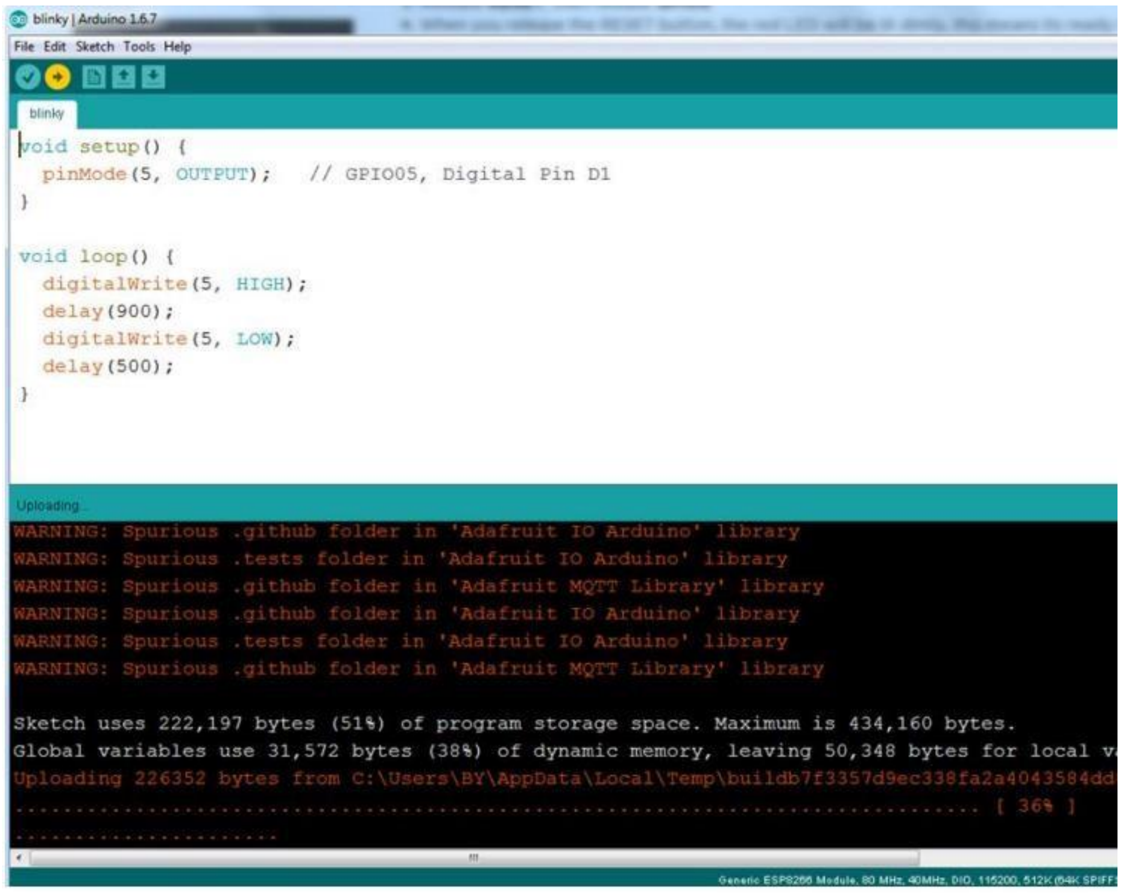
```
void setup() {
  pinMode(5, OUTPUT);   // GPIO05, Digital Pin D1
}

void loop() {
  digitalWrite(5, HIGH);
  delay(900);
  digitalWrite(5, LOW);
  delay(500);
}
```

Now you'll need to put the board into bootload mode. You'll have to do this before each upload. There is no timeout for bootload mode, so you don't have to rush!

- Hold down the ‘Flash’ button.
- While holding down ‘ Flash’, press the ‘RST’ button.
- Release ‘RST’, then release ‘Flash
- When you release the ‘RST’ button, the blue indication will blink once, this means its ready to bootload.


This blue indicator will blink once when release RTS in step 2.
Step 1: Hold down this 'Flash' button.
Step 2: Press once and release this button
Step 3: Release ' Flash' button. Now the board is in 'bootload' mode.
D0
D1
D2
D3
D4
3V3
GND
D5
D6
D7
D8
RX
TX
GND
3V3
FLASH
RST
A0
G
VU
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
Vin


A0
G
VU
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
Vin
D0
D1
D2
D3
D4
3V3
GND
D5
D6
D7
D8
RX
TX
GND
3V3
RST
FLASH
Cathode


blinky | Arduino 1.6.7
File Edit Sketch Tools Help
blinky
Uploading...
Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFF

```
void setup() {
  pinMode(5, OUTPUT);   // GPIO05, Digital Pin D1
}

void loop() {
  digitalWrite(5, HIGH);
  delay(900);
  digitalWrite(5, LOW);
  delay(500);
}
```

```
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library

Sketch uses 222,197 bytes (51%) of program storage space. Maximum is 434,160 bytes.
Global variables use 31,572 bytes (38%) of dynamic memory, leaving 50,348 bytes for local v
Uploading 226352 bytes from C:\Users\BY\AppData\Local\Temp\buildb7f3357d9ec338fa2a4043584dd
................................................................................ [ 36% ]
......................
```

The sketch will start immediately - you'll see the LED blinking. Hooray!

Analysis:

Temperature and humidity sensors are widely used in various applications such as climate control, weather forecasting, and agriculture. The sensors provide real-time data on temperature and humidity levels, which can be used to monitor and control the environment.

The accuracy of temperature and humidity sensors depends on several factors such as the quality of the sensor, the environment, and the calibration of the sensor. Some sensors may have a higher accuracy compared to others, and some may be affected by factors such as temperature drift, hysteresis, and aging.

To improve the accuracy of the readings, the sensors can be calibrated periodically to ensure that they are providing accurate data. Calibration involves comparing the readings from the sensor to a reference instrument and adjusting the sensor's output accordingly.

Temperature and humidity sensors can be used in various applications such as greenhouse monitoring, where they are used to monitor and control the environment to ensure optimal conditions for plant growth. In weather forecasting, temperature and humidity sensors are used to measure atmospheric conditions and provide real-time data on weather patterns.

In indoor climate control, temperature and humidity sensors are used to monitor and control the indoor environment to ensure optimal comfort levels for the occupants. The data obtained from the sensors can be used to adjust heating, ventilation, and air conditioning systems to maintain the desired temperature and humidity levels.

In summary, temperature and humidity sensors are important tools for monitoring and controlling the environment in various applications. The accuracy of the readings depends on several factors, and calibration is essential to ensure that the sensors are providing accurate data.

CASE STUDY:

Here is a case study for a temperature and humidity sensor using ESP8266:

Background:

A company that specializes in indoor climate control systems for commercial buildings wanted to develop a new product that would provide real-time data on temperature and humidity levels to their clients. They wanted to use a costeffective solution that would be easy to install and maintain.

Solution:

The company decided to use an ESP8266 microcontroller board along with a DHT22 temperature and humidity sensor to develop their new product. The ESP8266 board was selected because of its low cost and Wi-Fi capabilities, which would allow the sensor to be connected to the internet and provide real-time data to their clients. The DHT22 sensor was chosen because of its high accuracy and reliability.

The system was designed to be powered using a USB cable and programmed using the Arduino IDE. The sensor was housed in a small enclosure that could be mounted on a wall or placed on a desk.

The software for the system was developed using the Arduino IDE and the DHT library. The software was designed to read the temperature and humidity values from the DHT22 sensor and transmit the data to a server using the Wi-Fi connection.

The system was tested in a laboratory environment, and the temperature and humidity readings were compared to a reference instrument. The results showed that the system provided accurate readings with a maximum error of +/- 2 degrees Celsius and +/- 5% relative humidity.

The product was tested in several commercial buildings, and the data obtained from the sensor was used to optimize the indoor climate control systems. The data was also provided to the clients through a web-based dashboard, allowing them to monitor the temperature and humidity levels in real-time.

Results:

The temperature and humidity sensor using ESP8266 and DHT22 proved to be a cost-effective solution for the company, allowing them to develop a new product that provided real-time data on temperature and humidity levels to their clients. The sensor was accurate and reliable, and the data obtained from the sensor was used to optimize the indoor climate control systems in several commercial buildings.

Conclusion:

The temperature and humidity sensor using ESP8266 and DHT22 is a reliable and cost-effective solution for monitoring indoor environments. The system provides accurate readings, and the data obtained from the sensor can be used to optimize the indoor climate control systems.