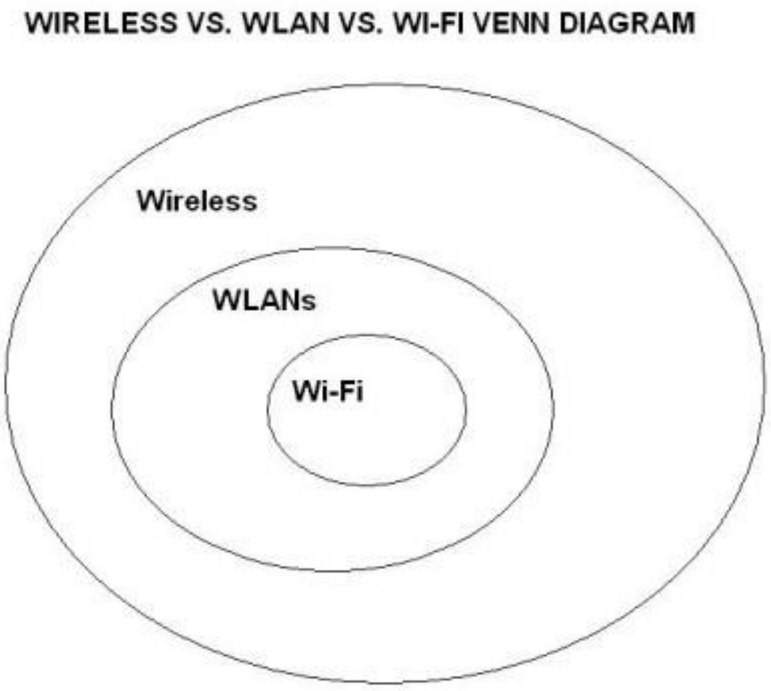
# WiFi

Wireless Communication

**Wireless communications** is a type of data **communication** that is performed and delivered wirelessly. This is a broad term that incorporates all procedures and forms of connecting and **communicating** between two or more devices using a **wireless** signal through **wireless communication** technologies and devices.

While wireless LANs refer to any local area network (LAN) that a mobile user can connect to through a wireless (radio) connection; Wi-Fi (short for "wireless fidelity") is a term for certain types of WLANs that use specifications in the 802.11 wireless protocol family



## WiFi – Introduction

- ☐ WiFi (Wireless Fidelity) is a technology that allows electronic devices to connect to a **wireless LAN** (**WLAN**) network, mainly using the 2.4 gigahertz (12 cm) UHF (Ultra High Frequency) and 5 gigahertz (6 cm) SHF (Super High Frequency) ISM radio bands
- ☐ A wireless network uses radio waves, just like cell phones, televisions and radios do

They transmit at frequencies of 2.4 GHz or 5 GHz

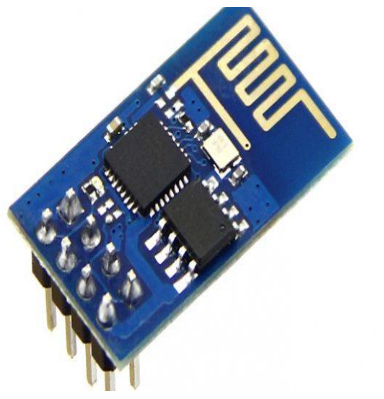
## WiFi’s benefits for enterprise IoT networking

- ☐ Many IoT devices rely on such wireless technologies as Bluetooth Low Energy, ZigBee and Z-Wave, but there's no single radio technology that dominates the market. Expect Wi-Fi to assume that role going forward, for a number of very good reasons:
- ☐ **Taking advantage of existing infrastructure**
- ☐ **Scalability**
- ☐ **Security**

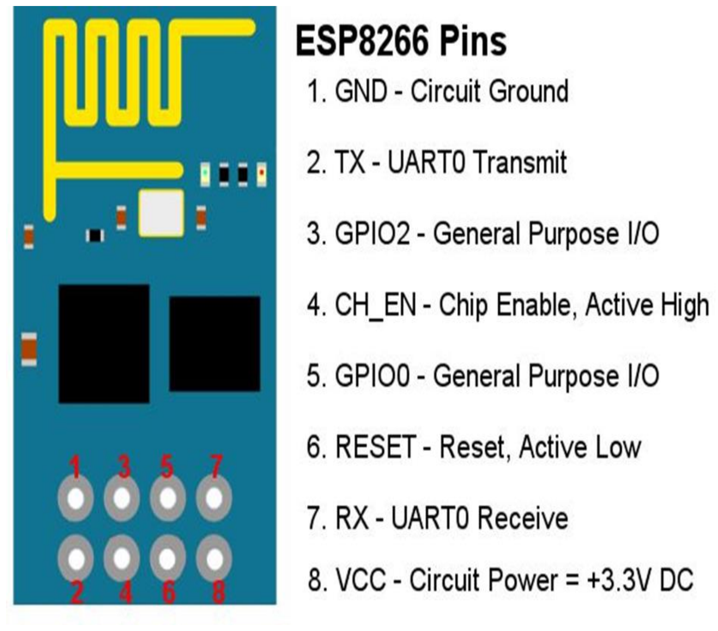
## Advantages

- Mobility
- Ease of Installation
- Flexibility
- Cost
- Reliability
- Security
- Use unlicensed part of the radio spectrum
- Roaming
- Speed

## ESP8266

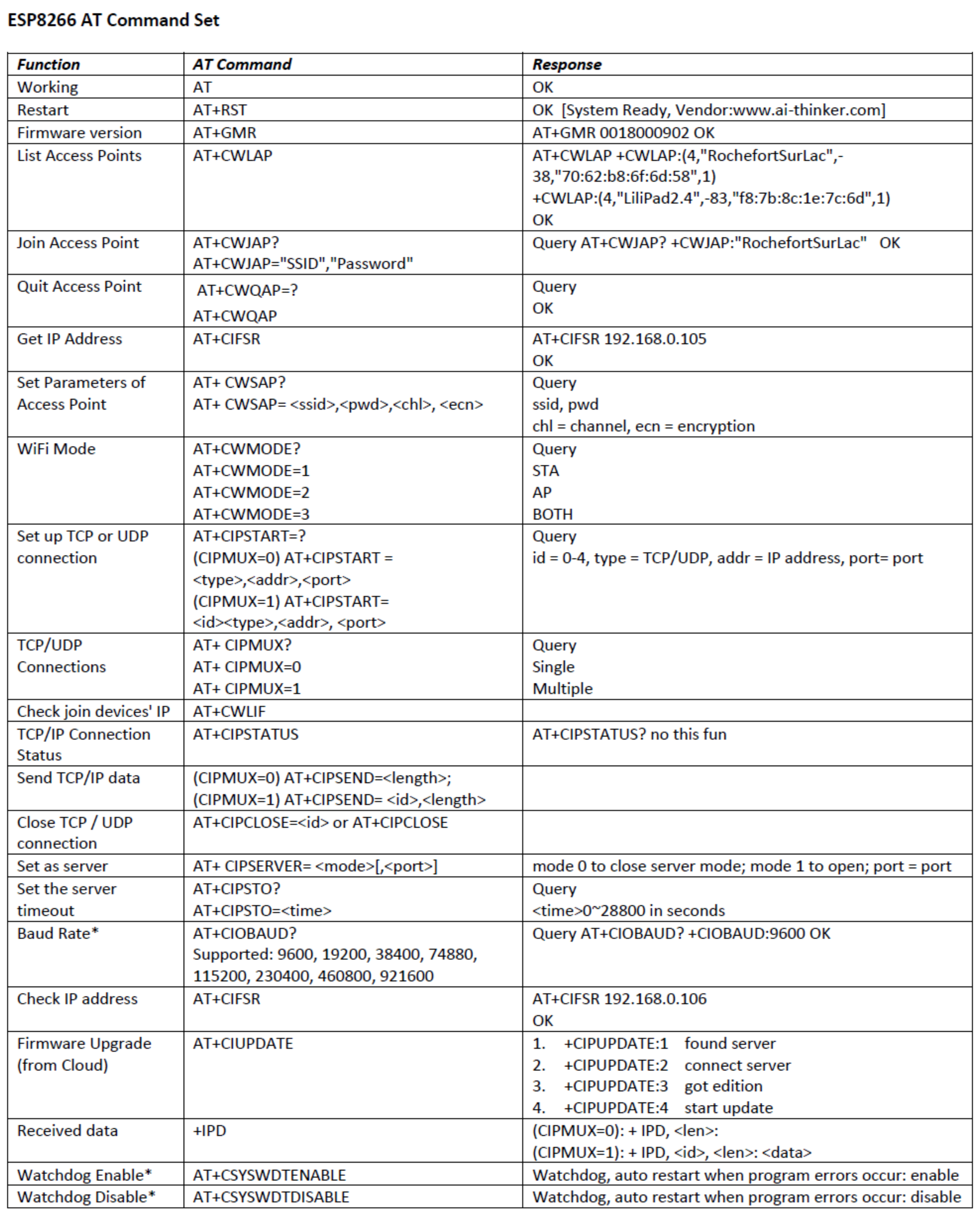


## Pin configuration


ESP8266 Pins
1. GND - Circuit Ground
2. TX - UART0 Transmit
3. GPIO2 - General Purpose I/O
4. CH_EN - Chip Enable, Active High
5. GPIO0 - General Purpose I/O
6. RESET - Reset, Active Low
7. RX - UART0 Receive
8. VCC - Circuit Power = +3.3V DC
1
3
5
7
2
4
6
8

ESP8266 –Arduino configuration

| **ESP8266** | **Arduino** |
|---|---|
| TX | RX |
| RX | TX |
| VCC | 3.5 V |
| GND | GND |
| CH_PD | 3.5 V |
| RESET | 3.5 V |



**ESP8266 AT Command Set**

| ***Function*** | ***AT Command*** | ***Response*** |
|---|---|---|
| Working | AT | OK |
| Restart | AT+RST | OK [System Ready, Vendor:www.ai-thinker.com] |
| Firmware version | AT+GMR | AT+GMR 0018000902 OK |
| List Access Points | AT+CWLAP | AT+CWLAP +CWLAP:(4,"RochefortSurLac",-38,"70:62:b8:6f:6d:58",1)<br>+CWLAP:(4,"LiliPad2.4",-83,"f8:7b:8c:1e:7c:6d",1)<br>OK |
| Join Access Point | AT+CWJAP?<br>AT+CWJAP="SSID","Password" | Query AT+CWJAP? +CWJAP:"RochefortSurLac" OK |
| Quit Access Point | AT+CWQAP=?<br>AT+CWQAP | Query<br>OK |
| Get IP Address | AT+CIFSR | AT+CIFSR 192.168.0.105<br>OK |
| Set Parameters of Access Point | AT+ CWSAP?<br>AT+ CWSAP= <ssid>,<pwd>,<chl>, <ecn> | Query<br>ssid, pwd<br>chl = channel, ecn = encryption |
| WiFi Mode | AT+CWMODE?<br>AT+CWMODE=1<br>AT+CWMODE=2<br>AT+CWMODE=3 | Query<br>STA<br>AP<br>BOTH |
| Set up TCP or UDP connection | AT+CIPSTART=?<br>(CIPMUX=0) AT+CIPSTART = <type>,<addr>,<port><br>(CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port> | Query<br>id = 0-4, type = TCP/UDP, addr = IP address, port= port |
| TCP/UDP Connections | AT+ CIPMUX?<br>AT+ CIPMUX=0<br>AT+ CIPMUX=1 | Query<br>Single<br>Multiple |
| Check join devices' IP | AT+CWLIF | |
| TCP/IP Connection Status | AT+CIPSTATUS | AT+CIPSTATUS? no this fun |
| Send TCP/IP data | (CIPMUX=0) AT+CIPSEND=<length>;<br>(CIPMUX=1) AT+CIPSEND= <id>,<length> | |
| Close TCP / UDP connection | AT+CIPCLOSE=<id> or AT+CIPCLOSE | |
| Set as server | AT+ CIPSERVER= <mode>[,<port>] | mode 0 to close server mode; mode 1 to open; port = port |
| Set the server timeout | AT+CIPSTO?<br>AT+CIPSTO=<time> | Query<br><time>0~28800 in seconds |
| Baud Rate* | AT+CIOBAUD?<br>Supported: 9600, 19200, 38400, 74880, 115200, 230400, 460800, 921600 | Query AT+CIOBAUD? +CIOBAUD:9600 OK |
| Check IP address | AT+CIFSR | AT+CIFSR 192.168.0.106<br>OK |
| Firmware Upgrade (from Cloud) | AT+CIUPDATE | 1. +CIPUPDATE:1 found server<br>2. +CIPUPDATE:2 connect server<br>3. +CIPUPDATE:3 got edition<br>4. +CIPUPDATE:4 start update |
| Received data | +IPD | (CIPMUX=0): + IPD, <len>:<br>(CIPMUX=1): + IPD, <id>, <len>: <data> |
| Watchdog Enable* | AT+CSYSWDTENABLE | Watchdog, auto restart when program errors occur: enable |
| Watchdog Disable* | AT+CSYSWDTDISABLE | Watchdog, auto restart when program errors occur: disable |

## Commands to Configure as access point

**1. AT+CWMODE=2**

**Configure as AP**

**2. AT+CWSAP="ESP8266","123", 3, 0**

Set SSID, password, channel and encryption.

To allow connections without a password set encryption parameter to 0 which is OPEN mode. To allow only users with a valid password, set encryption to 1, 2, 3 or 4. These encryption modes correspond to WEP, WPA-PSK, WPA2-PSK and WPA/WPA2-PSK with WPA2-PSK being the most secure and common for general use.

- **AT+CWSAP?**

**Verify that the AP settings have been set correctly.**

**3. AT+CIPMUX=1**

Mode of TCP connection (single/multiple)

Single connection mode is set with a 0 value while multiple connection mode is set with a value of 1. This mode can only be changed after all connections are disconnected. If server is started, reboot is required.

**4. AT+CIPSERVER=1**

Configure TCP Server.

A value of 1 creates the the server while 0 deletes the server in which case the module needs to restarted. The port can be optionally specified as a second argument otherwise it the default port of 333 is used.

## Commands to Configure as station

**1. AT+CWMODE=2**

Configure as Station

**2. AT+CWJAP="SSID","PASS“**

Connect to AP.

Command takes 2 arguments, SSID which is the name of the network and the PASSwhich is the password of the same network.

**3. AT+CIPMUX=0**

Mode of TCP connection(single/multiple)

**4. AT+CIPSTART="TCP","192.168.0.65","333"**

Start TCP or UDP connection in single connection mode**.**

The connection type can be either TCP or UDP. 192.168.0.65 is the IP address of the remove server the connection is being made to. 333 is the port of the same remote server.

- **AT+CIPSTART=1,"TCP","192.168.0.65","333“**

(Multiple Connection mode) where first argument is the id of the connection

## <u>Commands to send data</u>

**1. Send data in single connection mode.**

**AT+CIPSEND=15**

**When in single connection mode only the length of the data in bytes is required. Maximum length is 2048 bytes. Send command should be immediately followed by the actual data that matches the length specified.**

**2. Previous command should be immediately followed by the actual data.**

## <u>Interfacing ESP8266 with Arduino</u>

### Program to communicate with ESP866 serially and execute AT commands [This program accepts command from the serial monitor and executes in ESP8266 module. After execution the response is printed in the serial monitor]

```
#include<SoftwareSerial.h>

 SoftwareSerial wifi(2,3);

void setup() {

   pinMode(9, OUTPUT);

  Serial.begin(9600);

  wifi.begin(9600);

}

void loop() {

  if (Serial.available()) {
    while (Serial.available()) {
      wifi.write(Serial.read());
    }
  }
if (wifi.available()) {
    while (wifi.available()) {
      Serial.write(wifi.read());
    }
  }
}
```

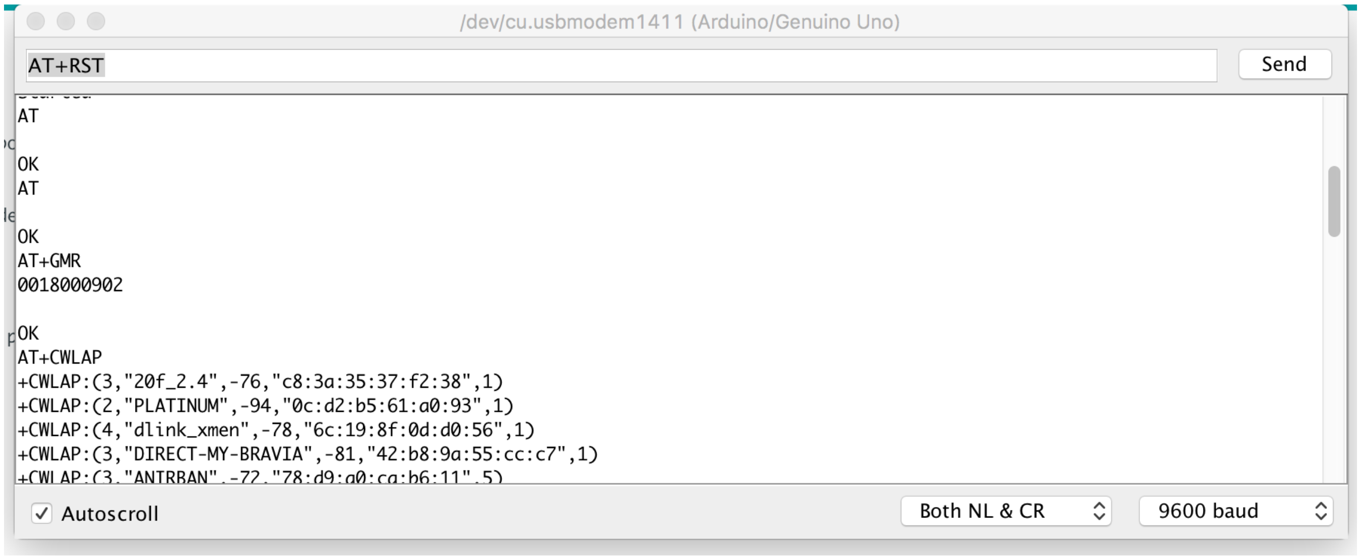
## Serial Commands and its output

```
AT
OK
AT+GMR
0018000902
OK
AT+CWLAP
+CWLAP:(3,"20f_2.4",-76,"c8:3a:35:37:f2:38",1)
+CWLAP:(2,"PLATINUM",-94,"0c:d2:b5:61:a0:93",1)
+CWLAP:(4,"dlink_xmen",-78,"6c:19:8f:0d:d0:56",1)
+CWLAP:(3,"DIRECT-MY-BRAVIA",-81,"42:b8:9a:55:cc:c7",1)
+CWLAP:(3,"ANIRBAN",-72,"78:d9:a0:ca:b6:11",5)
+CWLAP:(3,"ACDC",-86,"74:44:01:34:cd:d8",6)
+CWLAP:(2,"TIGER",-85,"c8:3a:35:15:e9:a8",6)
+CWLAP:(3,"GetyourownWIFI",-88,"00:25:5e:bb:21:a2",6)
+CWLAP:(1,"MGMNT",-86,"00:25:5e:bb:21:a3",6)
+CWLAP:(3,"Airtel-E5573-7F78",-89,"24:1f:a0:41:7f:78",6)
```

```
+CWLAP:(3,"biswas_network",-59,"e8:94:f6:2a:75:7c",9)
+CWLAP:(3,"Shiva",-64,"14:cc:20:e8:39:4c",10)
+CWLAP:(4,"GYOWF",-64,"6c:19:8f:ba:1a:8e",11)
+CWLAP:(1,"MGMNT",-81,"b8:c1:a2:12:b4:3d",11)
+CWLAP:(4,"Kliffhangr",-93,"78:e8:b6:41:b0:5f",11)
OK
AT+CWJAP="ANIRBAN","radharani1234"
OK

AT+CWJAP?
+CWJAP:"ANIRBAN"
OK

AT+CIFSR
192.168.4.1
OK
```



```
AT

OK
AT

OK
AT+GMR
0018000902

OK
AT+CWLAP
+CWLAP:(3,"20f_2.4",-76,"c8:3a:35:37:f2:38",1)
+CWLAP:(2,"PLATINUM",-94,"0c:d2:b5:61:a0:93",1)
+CWLAP:(4,"dlink_xmen",-78,"6c:19:8f:0d:d0:56",1)
+CWLAP:(3,"DIRECT-MY-BRAVIA",-81,"42:b8:9a:55:cc:c7",1)
+CWLAP:(3,"ANIRBAN",-72,"78:d9:a0:ca:b6:11",5)
```

**Program2: Wifi Server**

**This program will setup wifi server and wait for incoming commands from wifi client. An LED attached to pin 12 will ON/OFF based on the commands received from the client.**

```
#include "SoftwareSerial.h"


#define WIFISSID "KAYARIOT"      // WIFI Username

#define WIFIPASS ""         // WIFI Password


SoftwareSerial Serial1(2,3);

void sendToESP8266AndWaitForResponse (const char *cmd, const char *resp, bool waitForResponse, int duration) {

  String bytes;

  Serial.print ("CMD: "); Serial.println(cmd);

  do {

    Serial.print(".");

    Serial1.println (cmd);

    delay(duration);

    bytes = Serial1.readString();

  } while ( (waitForResponse) && (bytes.indexOf(resp)<0));

  Serial.print ("RESPONSE: ");

  Serial.print (bytes.c_str());

  Serial.println("\n------------------------------");

}


void setupAP() {

  String cwsapCmd = "AT+CWSAP=\"";

  cwsapCmd+=WIFISSID;  cwsapCmd+="\",\""; cwsapCmd+=WIFIPASS; cwsapCmd+="\",8,0";

  sendToESP8266AndWaitForResponse (cwsapCmd.c_str(), "OK", true, 50);

}
```

```
// the setup function runs once when you press reset or power the board
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  pinMode(11,OUTPUT);

  // Setup Wifi as AP
  sendToESP8266AndWaitForResponse ("AT+CWMODE=2", "", false, 5);
  setupAP();
  sendToESP8266AndWaitForResponse ("AT+CIPMUX=1", "OK", false, 50);
  sendToESP8266AndWaitForResponse ("AT+CIPSERVER=1", "OK", false, 50);
}

// the loop function runs over and over again forever
void loop() {
  if(Serial1.available())
  {
   String message = Serial1.readString();
   Serial.println (message);
    if(message.indexOf("LED ON")>0)
    {
      digitalWrite(11,HIGH);
    }
    else if(message.indexOf("LED OFF")>0)
    {
      digitalWrite(11,LOW);
    }
    else
    {
      Serial.println ("Nothing to do...");
    }

  }

}
```

**Output**

```
CMD: AT+CWSAP="IOT5A","",8,0
.RESPONSE: AT+CWSAP="KAYARIOT","",8,0
OK
CMD: AT+CIPMUX=1
.RESPONSE: AT+CIPMUX=1
OK
CMD: AT+CIPSERVER=1
.RESPONSE: AT+CIPSERVER=1
OK
Nothing to do...
```

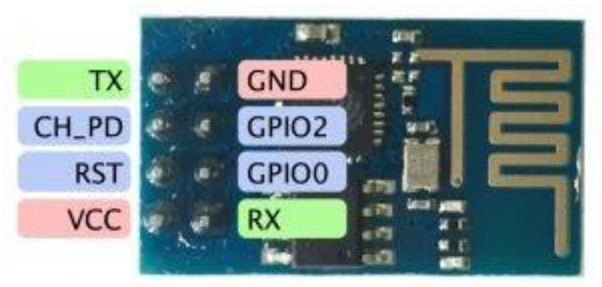
**Program to upload the temperature from Arduino client to Data.sparkfun.com cloud server**

**CLOUD (SPARKFUN)**

**Cloud Data:**
**https://data.sparkfun.com/streams/QGyLQKYb71F2Q1qMqQER**

**Pushing data to cloud (Only for reference)**
Pusing data to cloud is done by the program in Arduino. However sometime we might want to push manually from browser just for testing. Use below url for that
**https://data.sparkfun.com/input/QGyLQKYb71F2Q1qMqQER?private_key=JqyBXeWg9ViqeBy9yevR&temperature=33.33**

**Pin Diagram of ESP8266**



## Connections

| **Arduino** | **ESP8266** | **DHT Temperature sensor** |
|---|---|---|
| 5V | CH_PD, RST, VCC | |
| 3.3V | | VCC |
| GND | GND | GND |
| 2 (Software Serial RX) | TX | |
| 3 (Software Serial TX) | RX | |
| 5 | | DATA |

```
#define WIFISSID "SKS Cottage" // WIFI Username
#define WIFIPASS "kayar123" // WIFI Password
#define SERVERIP "api.thingspeak.com"// Server to post the update. "data.sparkfun.com"
#define POSTURL “POST
/input/QGyLQKYb71F2Q1qMqQER?private_key=JqyBXeWg9ViqeBy9yevR&temperature="
SoftwareSerial Serial1(2,3);
void sendToESP8266AndWaitForResponse (const char *cmd, const char *resp, bool waitForResponse, int duration) {
String bytes;
Serial.print ("CMD: "); Serial.println(cmd);
do {
Serial.print(".");
Serial1.println (cmd);
delay(duration);
bytes = Serial1.readString();
} while ( (waitForResponse) && (bytes.indexOf(resp)<0));
```

```
Serial.print ("RESPONSE: ");
Serial.print (bytes.c_str());
Serial.println("\n------------------------------");
}

void connectToWIFI() {
String cwjapCmd = "AT+CWJAP=\"";
cwjapCmd+=WIFISSID; cwjapCmd+="\",\""; cwjapCmd+=WIFIPASS; cwjapCmd+="\"";
sendToESP8266AndWaitForResponse (cwjapCmd.c_str(), "OK", true, 50);
}

void connectToServer() {
String cipstartCmd = "AT+CIPSTART=\"TCP\",\"";
cipstartCmd += SERVERIP; cipstartCmd += "\",80";
sendToESP8266AndWaitForResponse (cipstartCmd.c_str(), "Linked", true, 10);
}

// the setup function runs once when you press reset or power the board
void setup() {
Serial.begin(9600);
Serial1.begin(9600);

// Setup Wifi as STA and connect to AP
sendToESP8266AndWaitForResponse ("AT+CWMODE=1", "", false, 5);
connectToWIFI();
sendToESP8266AndWaitForResponse ("AT+CIPMUX=0", "OK", false, 50);
}

// the loop function runs over and over again forever
void loop() {
int rawvoltage= analogRead(A0);
float millivolts= (rawvoltage/1024.0) * 5000;
float temp= millivolts/10; //celcius

connectToServer();
char cmd[200],cipsend[100];
```

```
sprintf (cmd, "%s%d.%04d HTTP/1.0\r\n\r\n Host: %s\r\n\r\n",POSTURL, (int)temp,(int)trunc((temp-(int)temp)*10000),SERVERIP);
sprintf (cipsend, "AT+CIPSEND=%d",strlen(cmd));

sendToESP8266AndWaitForResponse (cipsend, ">", true, 10);
sendToESP8266AndWaitForResponse (cmd, "", false, 10);
delay(10000);
}
```