CODE

```
#include SoftwareSerial.h

 include the library code
#include LiquidCrystal.h

 initialize the library with the numbers of the interface pins
LiquidCrystal lcd(13, 12, 6, 5, 4, 3);  LCD connections

 D13 == RS
GND  ==   RW
D12  == Enable
D6   == DB4
D5   == DB5
D4   == DB6
D3   == DB7


float t=0;
char data = 0;
String apiKey = XBQDVORXXGAROWDW;   Write API key

 connect 8 to TX of ESP
 connect 9 to RX of ESP
SoftwareSerial ser(8,9);  RX, TX



void setup()
{
 enable debug serial
  Serial.begin(9600);   serial data transmission at Baudrate of 9600

 enable software serial

  ser.begin(9600);
  lcd.begin(16, 2);   to intialize LCD

  lcd.setCursor(0,0);

  lcd.print(    Welcome);

  lcd.setCursor(0,1);

  lcd.print(       To        );

  delay(3000);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print(     AIR);

  lcd.setCursor(0,1);
  lcd.print(QUALITY MONITOR);

  delay(3000);
  ser.println(AT);   Attenuation

  delay(1000);

  ser.println(AT+GMR);  To view version info for ESP-01 output 00160901
and ESP-12 output 0018000902-AI03

  delay(1000);

  ser.println(AT+CWMODE=3);  To determine WiFi mode

1 = Station mode (client)
2 = AP mode (host)
3 = AP + Station mode (ESP8266 has a dual mode)


  delay(1000);

  ser.println(AT+RST);  To restart the module

  delay(5000);

  ser.println(AT+CIPMUX=1);  Enable multiple connections


    0 Single connection
    1 Multiple connections (MAX 4)




  delay(1000);

  String cmd=AT+CWJAP=SSID,PASSWORD;  connect to Wi-Fi

  ser.println(cmd);

  delay(1000);

  ser.println(AT+CIFSR);  Return or get the local IP address

  delay(1000);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print(      WIFI);

  lcd.setCursor(0,1);

  lcd.print(   CONNECTED);

 }
```

```
void loop()
{
  delay(1000);

  t = analogRead(A0);   Read sensor value and stores in a variable t

  Serial.print(Airquality = );

  Serial.println(t);

  lcd.clear();
  lcd.setCursor (0, 0);
  lcd.print (Air Qual );
  lcd.print (t);
  lcd.print ( PPM );

  lcd.setCursor (0,1);
  if (t=500)
   {
   lcd.print(Fresh Air);
   Serial.print(Fresh Air );

   }
  else if( t=500 && t=1000 )
   {
   lcd.print(Poor Air);
   Serial.print(Poor Air);

   }
  else if (t=1000 )
  {
  lcd.print(Very Poor);
  Serial.print(Very Poor);

  }
  lcd.scrollDisplayLeft();
  delay(10000);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print(  SENDING DATA);

  lcd.setCursor(0,1);

  lcd.print(    TO CLOUD);

  esp_8266();
}


void esp_8266()
{

    TCP connection AT+CIPSTART=4,TCP,184.106.153.149,80

    String cmd = nAT+CIPSTART=4,TCP,;   Establish TCP connection
   AT+CIPSTART=id,type,addr,port

  id 0-4, id of connection
  type String, “TCP” or “UDP”
  addr String, remote IP
  port String, remote port


  cmd += 184.106.153.149;  api.thingspeak.com

  cmd += ,80;

  ser.println(cmd);

  Serial.println(cmd);

  if(ser.find(Error))

  {

    Serial.println(AT+CIPSTART error);

    return;

  }

String getStr = GET updateapi_key=;   API key

getStr += apiKey;

getStr +=&field1=;

getStr +=String(h);

getStr +=&field1=;

getStr +=String(t);

getStr += rnrn;

 send data length

cmd = AT+CIPSEND=;  Send data AT+CIPSEND=id,length

cmd += String(getStr.length());

ser.println(cmd);

Serial.println(cmd);

delay(1000);

ser.print(getStr);

Serial.println(getStr);

 thingspeak needs 16 sec delay between updates
```

```
  delay(17000);
}
```