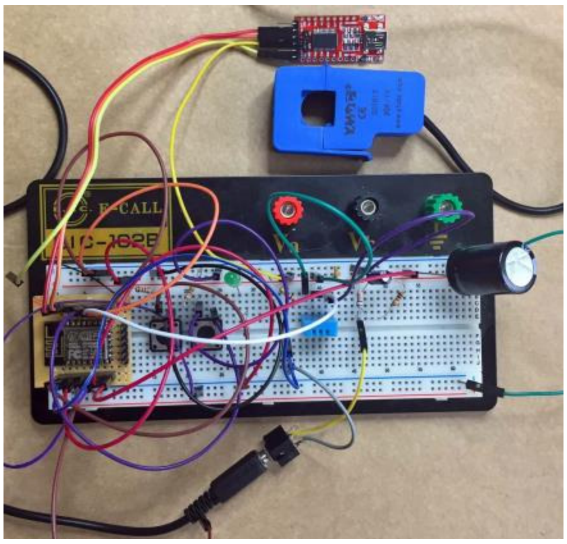
Standalone ESP8266 Toolbox: Integrated multiple sensor measurements – temperature, humidity and AC power monitoring plus alarm messaging, push data to the cloud inc NTP time stamping of all data plus Twitter and email messaging and STILL room for your application!

Step 1. Introduction

The arrival of the ESP8266 has significantly boosted the design possibilities open to us due to it's super low cost and built-in features especially in the Internet of Things (IoT) arena which immediately leads you to think of features such as automatic emails, tweeting, cloud storage and wireless remote control. All readily available in the Arduino world but not always so easy to implement especially for the novice.

As a former professional design engineer, one of my design approaches was always to "look at the bigger picture" and try to create whenever possible re-usable "modules" both within the hardware and software worlds. My other desire is not to have to learn new tools for each project and in this case, the Arduino IDE long ago became my preferred design environment. Thanks to Arduino we now have the ESP8266 integrated into its IDE so now we have the best of both worlds.

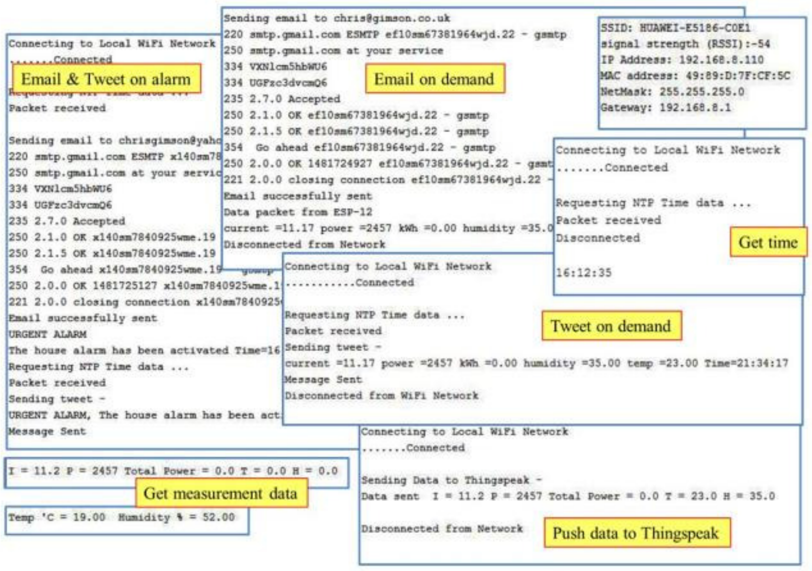


This instructable is a first step towards my goal of creating a range of re-usable ESP8266 application modules using the Arduino IDE but not requiring the use of an actual Arduino board (I.e. they fit solely within a single ESP8266 module.

There are several projects that cover many of the functions included here BUT I am yet to find one that successfully integrates everything seamlessly into one project. With this in mind, I present a "one shop" project that includes several popular and useful functions integrated into a simple hardware configuration based around a single ESP-12 module (however equally usable with the other available ESP8266 modules subject to the IO limits of each module). It is able to connect to the house WiFi network and thus gain access to the internet and all of the described functions work seamlessly and reliably together and with each other or can be simply called on their own within your own sketch.

As well as the standalone functions, I have included a very simple demonstration application in the void() loop to allow you to test and see how each feature operates. You can replace this with your own application including other measurements and functions to suit your own requirements simply by calling the relevant functions when required and/or modifying them when necessary. Within this sketch we have ….

1. Temperature and humidity measurement using the DHT11 or DHT22 sensor
2. AC mains current and power consumption measurement using a current transformer.
3. Simple alarm detection and messaging using any alarm sensor
4. Transmission of up to 8 separate measured data values to the Cloud using a ThingSpeak account .
5. Emailing of this measured data and/or alarm messages to any specified email address
6. Tweeting of the measured data or alarm message to a Twitter account



```
Connecting to Local WiFi Network
.......Connected
Requesting NTP Time data ...
Packet received

Sending email to chrisgimson@yaho
220 smtp.gmail.com ESMTP x140sm78
250 smtp.gmail.com at your servic
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 2.7.0 Accepted
250 2.1.0 OK x140sm7840925wme.19
250 2.1.5 OK x140sm7840925wme.19
354  Go ahead x140sm7840925wme.19
250 2.0.0 OK 1481725127 x140sm7840925wme.1
221 2.0.0 closing connection x140sm7840925
Email successfully sent
URGENT ALARM
The house alarm has been activated Time=16
Requesting NTP Time data ...
Packet received
Sending tweet -
URGENT ALARM, The house alarm has been act
Message Sent
```

```
Sending email to chris@gimson.co.uk
220 smtp.gmail.com ESMTP ef10sm67381964wjd.22 - gsmtp
250 smtp.gmail.com at your service
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 2.7.0 Accepted
250 2.1.0 OK ef10sm67381964wjd.22 - gsmtp
250 2.1.5 OK ef10sm67381964wjd.22 - gsmtp
354  Go ahead ef10sm67381964wjd.22 - gsmtp
250 2.0.0 OK 1481724927 ef10sm67381964wjd.22 - gsmt
221 2.0.0 closing connection ef10sm67381964wjd.22 -
Email successfully sent
Data packet from ESP-12
current =11.17 power =2457 kWh =0.00 humidity =35.0
Disconnected from Network
```

```
SSID: HUAWEI-E5186-C0E1
signal strength (RSSI):-54
IP Address: 192.168.8.110
MAC address: 49:89:D:7F:CF:5C
NetMask: 255.255.255.0
Gateway: 192.168.8.1
```

```
Connecting to Local WiFi Network
.......Connected

Requesting NTP Time data ...
Packet received
Disconnected

16:12:35
```

```
Connecting to Local WiFi Network
...........Connected

Requesting NTP Time data ...
Packet received
Sending tweet -
current =11.17 power =2457 kWh =0.00 humidity =35.00 temp =23.00 Time=21:34:17
Message Sent
Disconnected from WiFi Network
```

```
I = 11.2 P = 2457 Total Power = 0.0 T = 0.0 H = 0.0
```

```
Temp 'C = 19.00  Humidity % = 52.00
```

```
Connecting to Local WiFi Network
.......Connected

Sending Data to Thingspeak -
Data sent  I = 11.2 P = 2457 Total Power = 0.0 T = 23.0 H = 35.0

Disconnected from Network
```

The project is presented in breadboard format without a final PCB layout or power supply solution since the final implementation will depend on your own application needs but the complete design as shown as measured on my multi-meter runs on a 3.3 V supply at a maximum current of 90 mA mA.

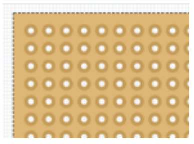
Full downloadable instructions and where relevant component data sheets are supplied for each element of the project.

Step 2. Parts List

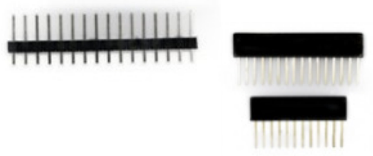
Most if not all of the components can be purchased from the usual online retails including Amazon, ebay, bidorbuy and of course a host of Chinese online retails such as Bangood and AliExpress.

Basic components

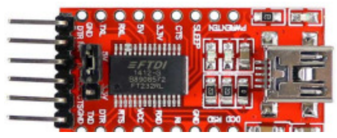
- ESP-12 or ESP12E board. Other types (E.g. ESP-01, ESP 03 etc) can easily be used taking care that the board used has the necessary inputs and outputs that your project requires (I.e. digital and analog IO).

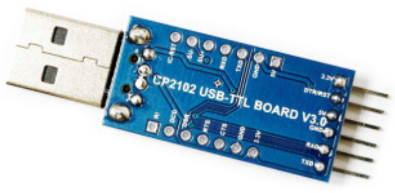


- Perforated Stripboard for preparing the ESP-12 module for breadboard use. (size approx. 30 x 35 mm)
- 2 x 8 way male or female connectors for use with the stripboard. I used the so called "stacking header". They have extended pins that when soldered to the board, they allow you to plug the board into a solderless breadboard and allow you to plug wires or components to the top of the board. If you are using the ESP-12E, you will also need a 6 way female or male header but without the extended stacking pins – just use a normal header connector.



- 2 x SPST pushbutton switches used for manual reset and the programming of the ESP8266. The type is not critical and for this project a simple PCB mounting type was used.





- USB to TTL serial interface board to Arduino IDE. Used to allow us to program the ESP-12 board from the Arduino IDE and also for the keyboard input and displayed output via the serial monitor. There are two general types readily available - the one used here is a FT232RL but equally usable is the other commonly used CP2102 model.
- Take care with the 5V / 3.3V issue when selecting your interface card. Some are 5V, some are 3.3V but many can be switched between 5 and 3.3V
- The pin assignment varies with the card but is normally printed on the circuit board silk screen
- You will not require this card if you are using an ESP8266 card that has a built in serial interface.
- Resistors 2 x 10K0 (actual value not so critical) for use as pull up resistors to the two pushbutton switches. They are not critical and any type can be used.

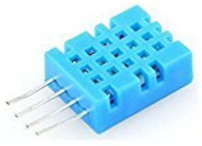
Power consumption monitoring

- Current transformer (CT). Many types are available with or without an internal shunt resistor and for different current ranges. The one that I used was an SCT013 30A/1V purchased from Amazon and is a 30A device with a 1V output (meaning that there is already a built in shunt resistor).
- Shunt resistor if one is not included in the CT. The resistance value will depend on the choice of CT and there are instructions included as to how to calculate this. The recommended type is metal film 1% for its stability and tolerance.
- Resistors 1 x 10K, 1 x 56K. The recommended type is metal film 1% . Carbon resistors are not recommended.
- 10uF 16V Electrolytic capacitor
- A 2.5mm or 3.5mm audio jack socket depending on what type of plug is attached to the CT.

Alarm sensing

- Note that although I have included this Passive Infra Red movement sensing module (PIR) device in the parts list as a recommended item, it is not actually required to test the software as this is not a main focus. A simple SPST switch would be perfectly acceptable during the prototype build. The alarm device used can be any type with a voltage or relay output when activated. Care must be taken with the power supply as unless you use one that runs off 3V3, you will need to provide a separate power supply and ensure that voltage of the output signal is limited also to 3.3 volts.
  The SR501 requires a 5V power supply but the output is already limited to 3V when active high. It is available from the usual variety of online suppliers.
- Any SPST switch used as an alarm enable signal. Here, I used a simple PCB mounting slide switch.
- 2 x 10K pull up resistors to the 3.3 V rail for the alarm and alarm enable switching.

Temperature and humidity measurement

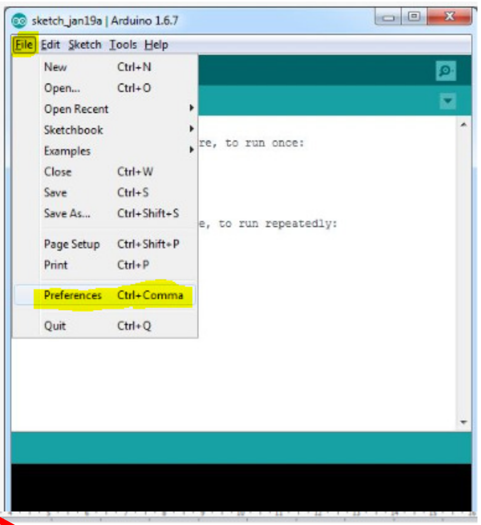


1. A DHT11 or DHT22 (higher performance) sensing module. This is available from a variety of online suppliers including Ebay, BidorBuy and Amazon
   - A 4.7K pullup resistor to the 3.3V rail for the output pin of the device

Power supply

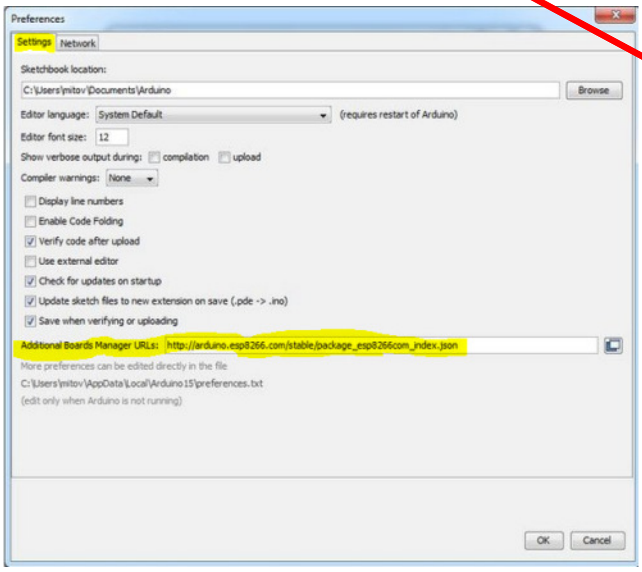
- I have not included a specific 3.3 V DC power supply directly in the project because your choice of solution will depend heavily on your final specification, packaging and configuration. There are many pre-built 3.3V power modules available and of course you can always build your own or regulate down from an already existing DC supply. Nevertheless, the topic of power is important and will be discussed again later.

Step 3. Ensuring that the Arduino IDE is set up correctly



1. In the Arduino IDE open the FILE – PREFERENCES - SETTINGS window and enter the URL (highlighted in yellow in the above photo) into the Additional Boards Manager URLs field, and select OK.
   - http://arduino.esp8266.com/stable/ package_esp8266com_index.json

2. Select the MENU option Tools → Board → Boards Manager...and scroll down and to locate the option esp8266 by ESP8266 Community which should be the last item on the list, and click INSTALL


Preferences

Settings | Network

Sketchbook location: C:\Users\mitov\Documents\Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

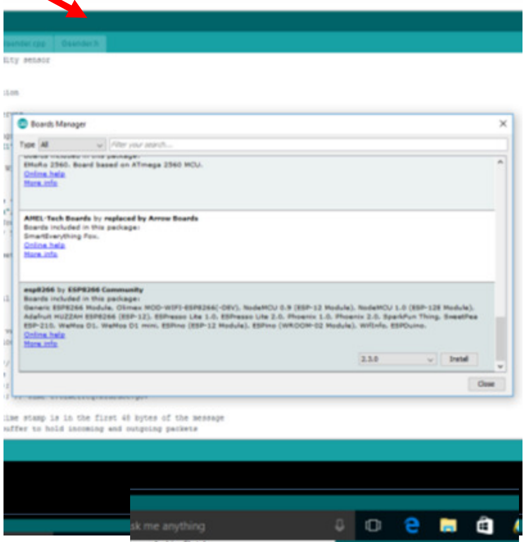
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

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

(edit only when Arduino is not running)

[OK] [Cancel]


Boards Manager

Type All | Filter your search...

Arduino Mega 2560. Board based on ATmega 2560 MCU.
Online help
More info

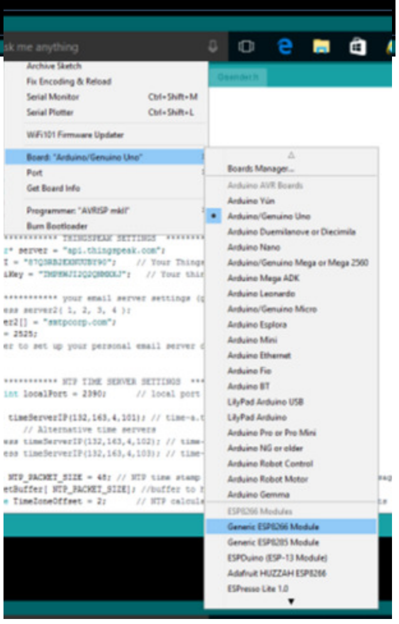
AMEL-Tech Boards by replaced by Arrow Boards
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by ESP8266 Community
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, Phoenix 1.0, Phoenix 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

2.3.0 [Install]

[Close]

After the files are downloaded close down and re-start the Arduino IDE we can now select the board we're using from the menu option
Tools → Board →
Generic ESP8266 Module.



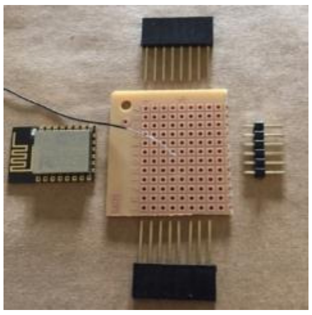
3. When you have plugged the FT232RL FTDI USB interface cable into the computer, select the COM port number that it is assigned to.

4. Note the required library files that you will need to have installed in the Arduino IDE library setup

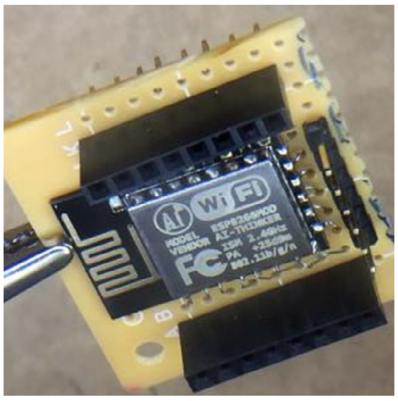
```
#include <DHT.h>   // for DHT temp/humidity sensor
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include "Gsender.h"  // for email function
#include <WiFiUdp.h>  // for NTP Time server
```

5. At this point, the IDE is "ready to go" and you can start with the sketch ...

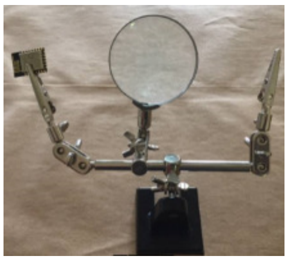
Step 4. Prepare the ESP-12E for use on a breadboard



The ESP_12 or ESP-12E module as supplied is unfortunately not compatible with a breadboard or stripboard usage without some work to make it more user friendly. Fortunately with the correct tools and a steady hand, this is not so difficult and I include the following instructions to allow you to adapt it for easy prototype work.



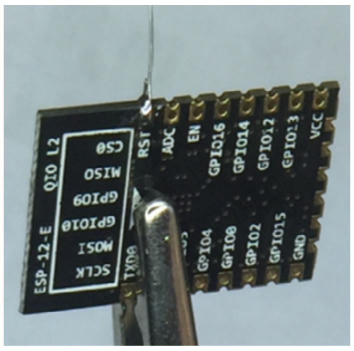
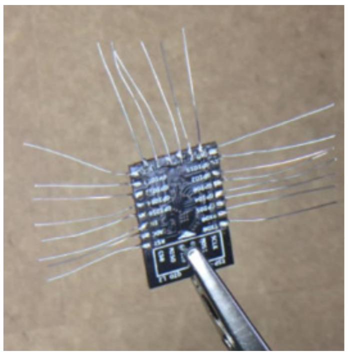
You will need a pair of pointed nose pliers/pointers, wire cutters, a small piece of perforated board typically 30 x 35 mm, some thin single strand wire with the insulation removed, a soldering iron with a fine tip and solder and the PCB mounting connectors as per the parts list. Note that unlike the ESP-12E, the ESP-12 does not have the 6 connectors at the end opposite the aerial. In this case or if you do not want to use the additional IO that the ESP-12E offers, you can miss the end male header mounting.



I highly recommend the use of a "third hand" assembly tool with or without the magnifying glass as this can be quite fiddly without a holding fixture whilst you are fixing and soldering the wires.

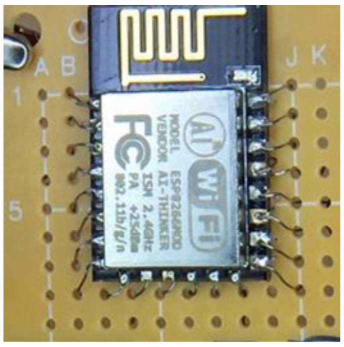
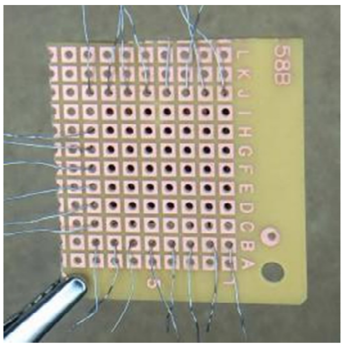
Before connecting each wire to the ESP-12 terminals, strip off the isolation of the single strand wire (you don't have to worry about them shorting as this will be handled later).

Push the wire through the ESP-12 board circuit board terminal and bend it back. Arrange the wire so that it points away from the board (see photo) and gently solder it ensuring that you do not apply too much solder (possible short circuit to other terminals or the ESP case) or you overheat the board. Take your time, it can be fiddly but you will soon become an expert with practice.

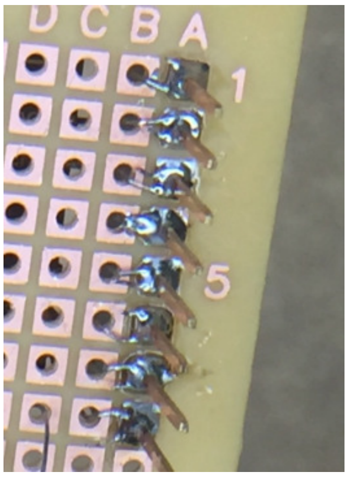
Cut the now attached wire so that there is 15 to 20 mm sticking out from the terminal pad.

Repeat this for each output pad on the ESP board.

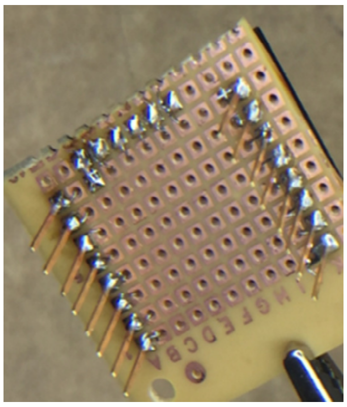



The next step is to mount the ESP board onto the non-copper side of the perforated board. This also can be a bit tricky as the wires are very thin and bendy but patience will serve you well. Making sure that you have selected the correct location on the board to later fit the headers, one at a time using the pliers / tweezers thread each wire into the corresponding hole in the perforated board and gently pull it through to take up the slack. Repeat for each wire as per the photo.

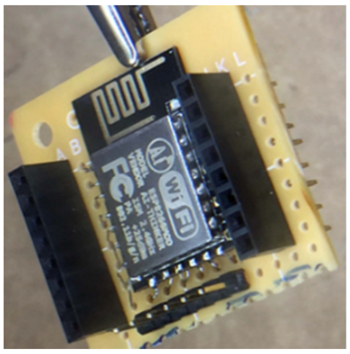
Take care not to snap the wires but if you soldered them correctly they will be able withstand a bit of force.
Then insert the female headers in the row next to where you brought the wires through – each pin should line up aside one of the wires. Solder just one of the header pins to allow you to fix the header in the board ensuring the header pin protrude at right angles through the board (a future concern when plugging the final assembly into the breadboard)
Next take each wire in turn and gently guide it around the pin next to it and wrap it around the base of the pin. Solder the wire and base of the pin to the board copper pad, trim any surplus wire with the cutters taking care not to cut the wire leading to the ESP module and that there is no short circuit to the adjacent pin.



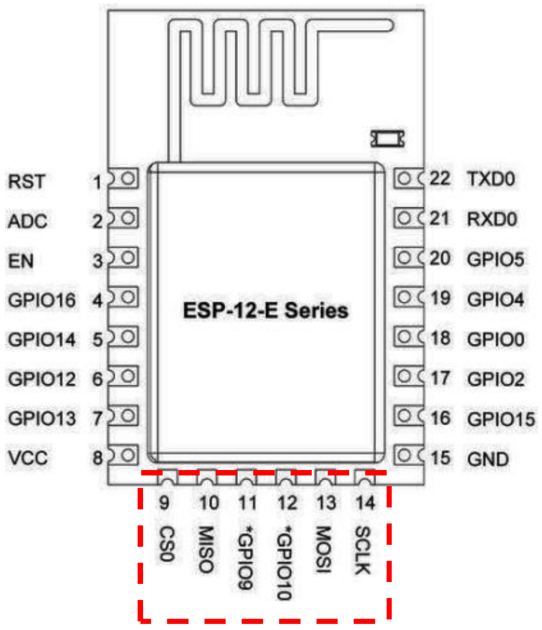
Repeat for each pin.
Fit the next header and repeat the process for each pin.
If you are fitting a third header for the ESP-12E extra IO pins then note that this time, the male or female header does not have extending pins as we do not want them to plug into the motherboard.



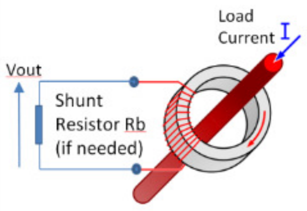
The photo shows a male header for this connection but a female header can be used if preferred.
You have now finished the conversion. Check the alignment and straightness of the pins by plugging the assembly into a breadboard. With the majority of standard breadboards, due to the large size of the plug in board, you will probably have to use the connectors on the assembly to make connections to the ESP board rather than the breadboard.



Once assembled on the perforated board, unfortunately you lose sight of the pin idents so use this diagram as a top view representation of the ESP-12E module.
The ESP-12 does not have the I/O of pins 9-14 however the remaining pins are the same

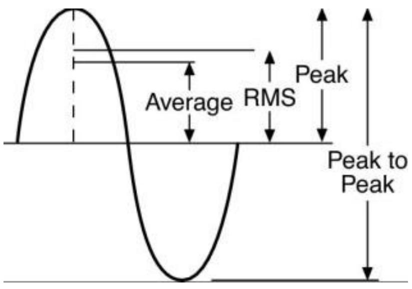


Step 5. Calculating the value of the shunt resistor (Power monitoring only)



The first question to answer is "do I need a shunt resistor as some current transformers (CT) have it built in. In my case I used a SCT013 30A/1V device which includes the burden resistor and so produces a 1 volt output at 30A load current however this is also available without the burden resistor.

So assuming that your device does NOT contain a burden resistor, you need to provide one and as such you need to **size** your burden resistor Rb, this converts your CT current into a voltage value.



First, …. Some basics … When we state numerical values of AC mains voltages and currents we are normally referring to their Root Mean Square values (V_{rms}, I_{rms}), their inter-relationsips with other important terms can be defined as:

V_{peak} = 1.414 x V_{rms}

$V_{peak\ to\ peak}$ ($V_{p\text{-}p}$) = 2 x V_{peak} = 2.824 x V_{rms}

The currents values ($I_{p\text{-}p}$, I_{rms} etc) also follow the same relationships.

The issue to take care of is that the ESP8266 analogue input convertor will be exposed to the $V_{p\text{-}p}$ of the CT output rather than the RMS values that we are more familiar and interested in.

This means that the maximum Vrms that the 0-1V DC analogue input input can cope with is (1 / 2.828) = 0.35

You need to determine from the CT specification its maximum primary current (E.g 30A rms) and the turns ratio of the transformer (E.g. typically 500 to 5000).

Start by dividing your maximum primary current by your CT's turns ratio. This will give you the maximum rms current to be developed across the burden resistor at the maximum current.

My device worked on 30A rms with a turns ratio 0f 1800:1 giving a secondary current of 0.01666A rms or 16.66mA rms. We use Ohms law to calculate the resistance required to develop 0.35 V_{rms} at 30A rms primary current (R=V/I) gives us R=0.35/0.01666=21R. This can be achieved with standard resistors by combining a 10R+10R+1.0R in series. (Note that I have not taken into account any tolerances in these calculations)

This gives the possibility of measuring upto 30A at your mains voltage. To calculate how many amps yours needs to sense, take the maximum continuous power you are expecting to sense and divide that by your mains line voltage (usually 110V or 220V depending on your country).

My mains supply is 220V AC giving me the possibility to measure upto (220 x 30) watts = 6.6 kW.

Clearly if you use a different transformer and/or have a different mains line voltage these numbers will be different so you need to chose your CT and size your resistor appropriately taking care with the RMS to peak to peak consideration for the ADC.

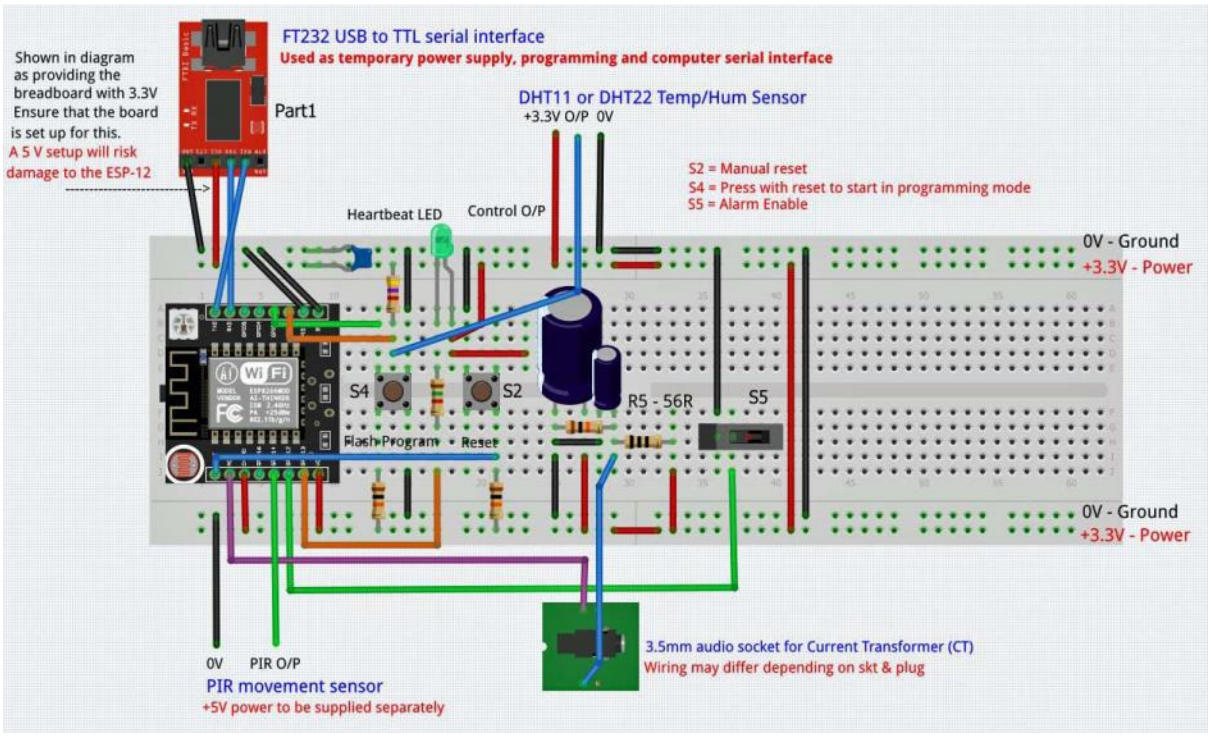
When the burden resistor is built in, it saves some work and a component but removes any design flexibility and risks limiting your power measurement range if you have a lower mains line voltage. (E.g. The built in resistor of the SCT013 is 62R so as it stands, I have a reduced sensitivity. I plan to lower the resistance by adding parallel resistors to the CT in the future)

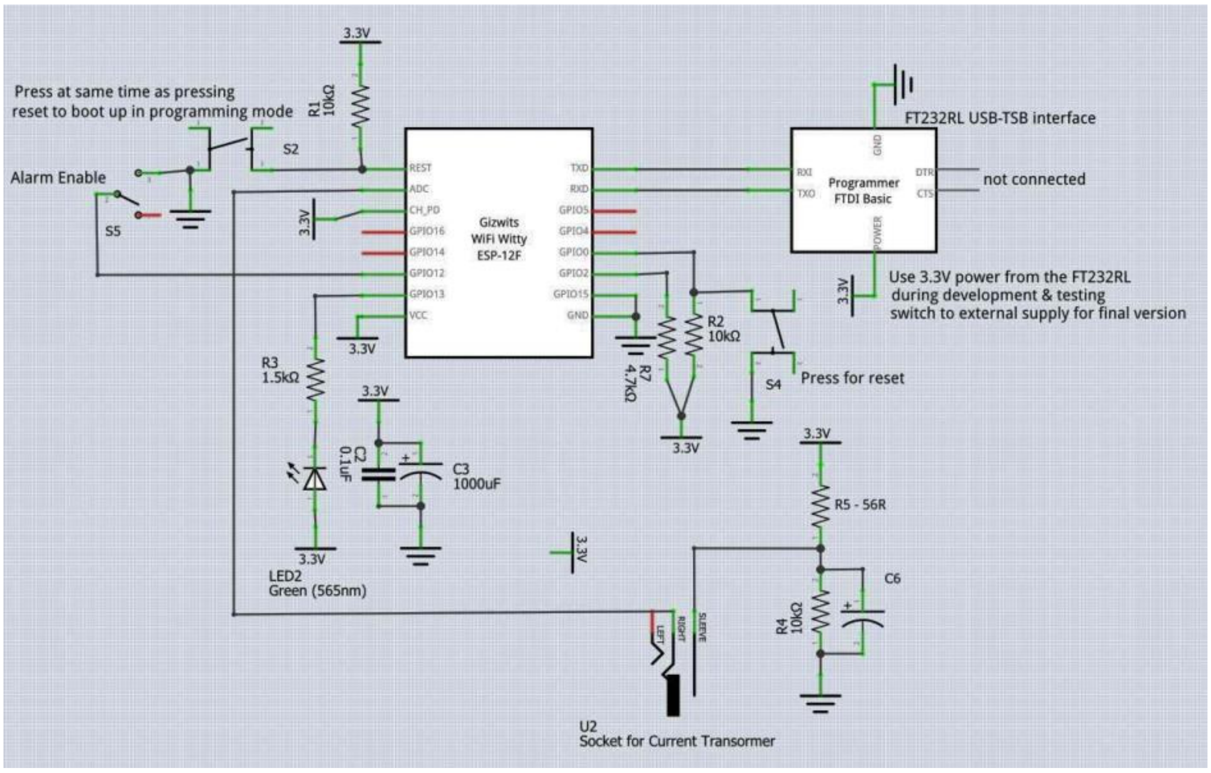
I lifted the following data from another instructable (see section 8) as an indicator to some commercially available CT's.

• Murata 56050C – 10A – 50:1	Talema AS-103 – 15A – 300:1
• Talema AC-1020 – 20A – 1000:1	Alttec L01-6215 – 30A – 1000:1
• Alttec L01-6216 – 40A – 1000:1	Talema ACX-1050 – 50A – 2500:1
• Alttec L01-6218 – 60A – 1000:1	Talema AC-1060 – 60A – 1000:1
• Alttec L01-6219 – 75A – 1000:1	Alttec L01-6221 – 150A – 1000:1

Step 6. Wiring the Breadboard with the required components

See the figures below for a complete breadboard layout and schematic.


FT232 USB to TTL serial interface
Used as temporary power supply, programming and computer serial interface
Shown in diagram as providing the breadboard with 3.3V Ensure that the board is set up for this.
A 5 V setup will risk damage to the ESP-12
Part1
DHT11 or DHT22 Temp/Hum Sensor
+3.3V O/P 0V
S2 = Manual reset
S4 = Press with reset to start in programming mode
S5 = Alarm Enable
Heartbeat LED
Control O/P
0V - Ground
+3.3V - Power
S4
S2
R5 - 56R
S5
Flash Program
Reset
0V - Ground
+3.3V - Power
3.5mm audio socket for Current Transformer (CT)
Wiring may differ depending on skt & plug
0V
PIR O/P
PIR movement sensor
+5V power to be supplied separately



The actual components used will depend on what functionality you require so simply install on the breadboard which modules are relevant to you.

Step 7. Hardware breakdown by function

Basic ESP-12 module & support components – common to all

This is the minimum configuration and essentially is the ESP8266 mounted on the perforated board as described in step 4 and then just plugged into the breadboard. Any wires to be connected to the module are to be plugged into the corresponding header connector soldered to the board. You should also mount and connect the two SPST push buttons S4 & S5 together with 10K pull up resistors and the FT232RL USB to TTL interface board. If you are not using the latter to provide the 3.3V supply then also connect an external 3V3 power supply instead of using the 3.3V output of the FT232 (leave this unconnected). The choice of power supply is very important - *the most important 1st point is that the basic power supply voltage requirement is strictly 3.3V*. Current consumption will depend on the hardware used and also the operating mode of the ESP8266 module (E.g. consumption is higher when the WiFi is operating).

The other very important point is that the supply voltage must be stable, I found that if the supply is not adequately regulated, the supply can suffer from occasional "glitches" or dips in the voltage for several milliseconds at a time. These are not normally a major problem but I found that occasionally a programming failure would occur during upload. The solution is to simply make sure you use a robust regulated supply and I suggest placing a 1000uF electrolytic capacitor across the 3.3V supply.

There is much discussion on the internet groups as to whether the USB to TTL serial interface board used can supply enough power to the ESP-12 directly with many saying "no" and recommending that a separate supply is always used. My own experience after testing 4 separate FT232 boards and using them to also power the components (ensuring that it is linked to 3V3 rather than 5V!) and 2 different external supplies, all worked fine albeit with the occasional programming upload failure (not a fatal problem, if this happens, you should just reset the board and repeat the upload).

Temperature and humidity sensor

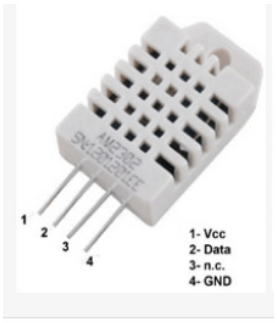
The sensor chosen for measuring temperature and humidity is the well proven and easily available DHT22 (or it's lower spec relative DHT11). It is a combined relative humidity and temperature sensor and it transmits its data serially in digital format on its data pin.

The DHT22 sensor can be powered directly from our local 3.3V voltage and measures from -40oC to +80oC with an accuracy of +/- 0.5oC for temperature and +/-2% for relative Humidity. Its sensing period is approximately 2 seconds and so we allow a minimum time between successive readings of approx. 4 seconds to be safe..

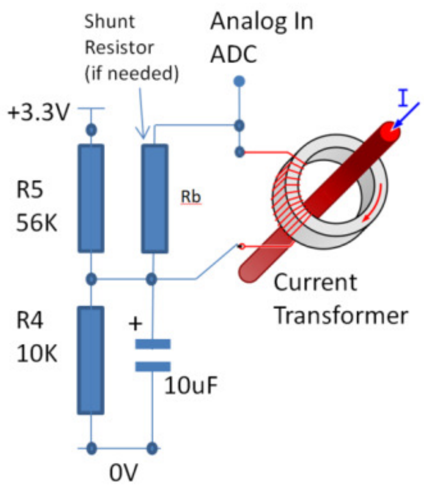
For more details of the sensors refer to the DHT11/22 data sheet enclosed with this project.

The DHT11/22 modules although having different dimensions and colours have identical pin layout as per below

1. VCC - Connected to the 3.3V supply
2. Data out. Connected to GPIO2 of the ESP-12 module with an 10K pull up resistor to the 3.3V power rail.
3. Not connected
4. Ground – connected to 0V

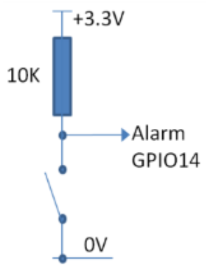


Power consumption measurement

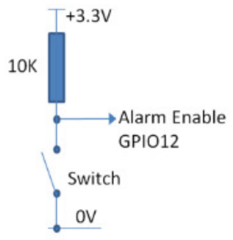


This is surprisingly easy and is focused on the Current transformer providing a 0-1 peak to peak sine voltage (at full resistive load).
To measure this correctly with the single ended DC analogue input converter, we reference the CT output to a voltage value ½ of the value of the range of the A/D converter. The latter being 1V gives us a reference value of 0.5V DC. This is achieved with the resistor chain R4 & R5 across the 3.3V power rails. The reference voltage at the junction of R4 & R5 is (3.3 x R4)/(R4+R5) = 33/66 = 0.5V DC. This will now allow the ESP8266 A/D input to measure a $1V_{pp}$ value sine wave.
Although measuring the peak to peak value of the output of the CT, this implementation of the software takes a simplified approach to the calculation of the RMS current (A) and resulting power calculation (W) including a totalised power consumption (kWh) which is perfectly adequate for our domestic performance needs. This is described in more detail in the sketch.

Alarm sensor



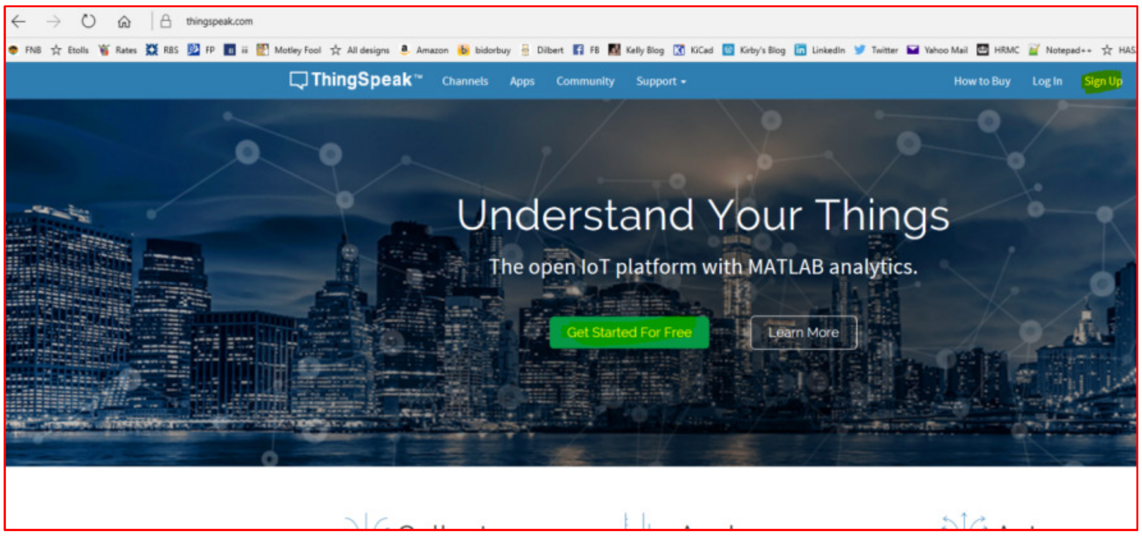
The alarm trigger is a simple configurable low to high or high to low digital input switch transition on GPIO14 with a debounce window of 1 second (I.e. it must be active for 1 second before responding). This means that the trigger device can be a PIR movement sensor, door or window switch, floor mat or simply anything that gives us a digital status change when our alarm threshold is activated. The only critical criteria here is that the input switching voltages must not exceed 3.3 V. If you have anything more than that, you should consider level conversion by whatever method is appropriate E.g. resistive ladder, opto isolator, transistor, relay etc.



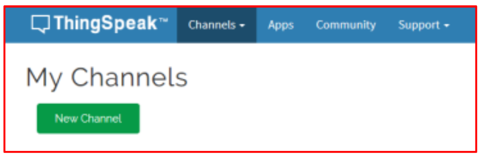
We do not want the alarm to be active all of the time so I have included an extra input switch on GPIO12 that acts as an alarm enable/disable. When this is OFF, the alarm input has no effect.
Note.
This is a "demo" program for the alarm function. To prevent unwanted repeats of the alarm message if the alarm condition persists, I have included a 3 minute "lock out window" that will prevent the alarm repeating if it is still activated. It is expected that you would refine the alarm procedure to suit your own particular requirements and preferences.

Step 8. Setup your Thingspeak account

1. First you must sign up for an account at www.thingspeak.com

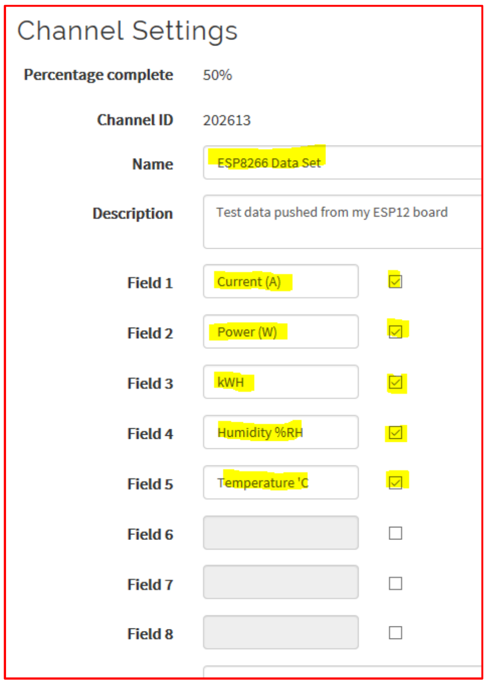


After creating your account, follow the onscreen instructions to create a new Channel ....



Set up the channel name and then tick the box on the right hand side of each field name to activate it and then type in the name of up to 8 data fields that you will be sending to the account from the sketch.. In our example, we have used 5 of the data fields Current, power, kWh, humidity and temperature.

For now you do not need to fill in any of the other data boxes for the channel.

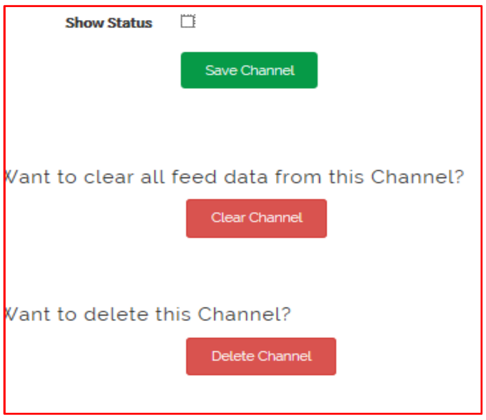


Channel Settings

Percentage complete	50%	
Channel ID	202613	
Name	ESP8266 Data Set	
Description	Test data pushed from my ESP12 board	
Field 1	Current (A)	✓
Field 2	Power (W)	✓
Field 3	kWH	✓
Field 4	Humidity %RH	✓
Field 5	Temperature 'C	✓
Field 6		☐
Field 7		☐
Field 8		☐

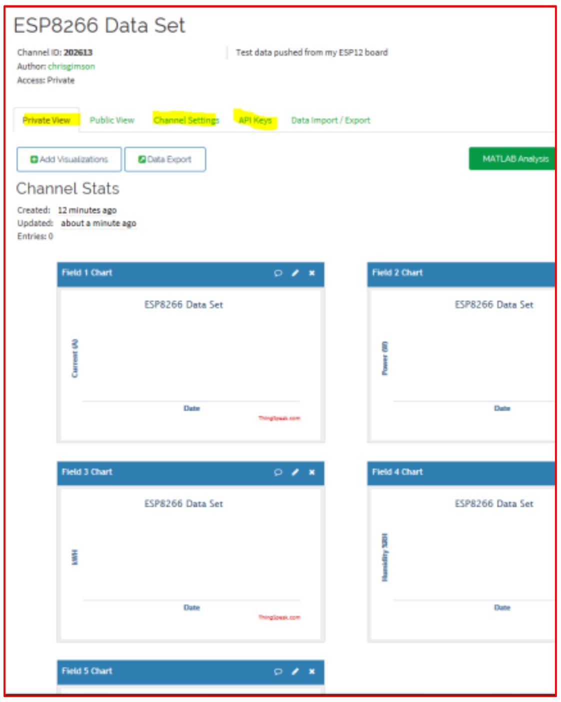
Pressing the "Save Channel" button will save the data and the channel is created

Note at the bottom of the screen, there is also a "Clear Data" button that allows you to clear out any stored data when required.


Show Status
Save Channel
Want to clear all feed data from this Channel?
Clear Channel
Want to delete this Channel?
Delete Channel

On saving the channel data, you are presented with the main channel screen and you will see a plot of each field that you have entered. You can set up each field plot separately by clicking on the settings button at the top right hand corner of each field plot.

Near the top of this screen, you will see the tabs that take you to each section of the channel settings …


ESP8266 Data Set

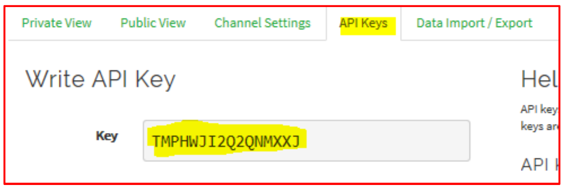
Channel ID: 202613
Author: chrisgimson
Access: Private

Test data pushed from my ESP12 board

Channel Stats

Created: 12 minutes ago
Updated: about a minute ago
Entries: 0

Selecting the API keys takes you to the "Write API key". You should copy and paste this to the API string value in the sketch …

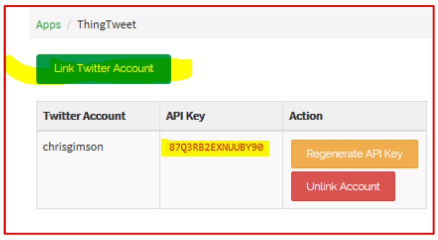


Write API Key

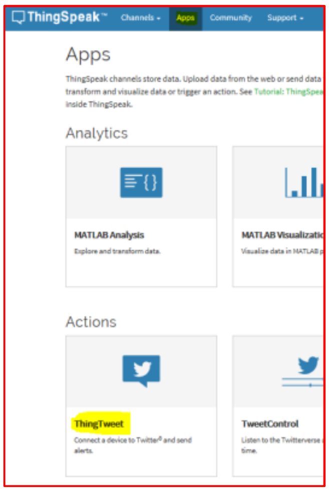
Key: TMPHWJI2Q2QNMXXJ

```
      // ************ THINGSPEAK   ********************************
const char* server = "api.thingspeak.com";
String API = "87Q3RB2EXNUUBY90";    // Your Thingspeak.com twitter account API key
String apiKey = "TMPHWJI2Q2QNMXXJ";    // Your thingspeak Data channel Write API key,
```

To use the Twitter function, of course you first need a Twitter account and then you can link it to the Thingspeak channel by the ThingTweet app accessible from the Apps tab on the main screen.



Twitter Account	API Key	Action
chrisgimson	87Q3RB2EXNUUBY90	Regenerate API Key Unlink Account



Apps

ThingSpeak channels store data. Upload data from the web or send data transform and visualize data or trigger an action. See Tutorial: ThingSpea inside ThingSpeak.

Analytics

MATLAB Analysis
Explore and transform data.

MATLAB Visualizatio
Visualize data in MATLAB p

Actions

ThingTweet
Connect a device to Twitter® and send alerts.

TweetControl
Listen to the Twitterverse time.

Enter your Twitter details on pressing the "Link Twitter Account" button and then copy and paste the resulting API key into the sketch …

```
      // ************ THINGSPEAK   ********************************
const char* server = "api.thingspeak.com";
String API = "87Q3RB2EXNUUBY90";    // Your Thingspeak.com twitter account API key
String apiKey = "TMPHWJI2Q2QNMXXJ";   // Your thingspeak Data channel Write API key,
```

Note that there are many other interesting Apps with Thingspeak worthy of some checking out on your own.

Step 9. Setup your gmail email account

Setting up the gmail account is pretty straightforward but care must be taken in ensuring that the correct log in details are entered into the sketch.

The parameters to be set up can be found in the Gsender.h tab in the sketch and are highlighted in yellow in the diagram.

Open up the website https://www.base64encode.org/ and enter your account log in username into the top box.

Press the "ENCODE" button.

A base64 encoded version of your username will appear in the second box.

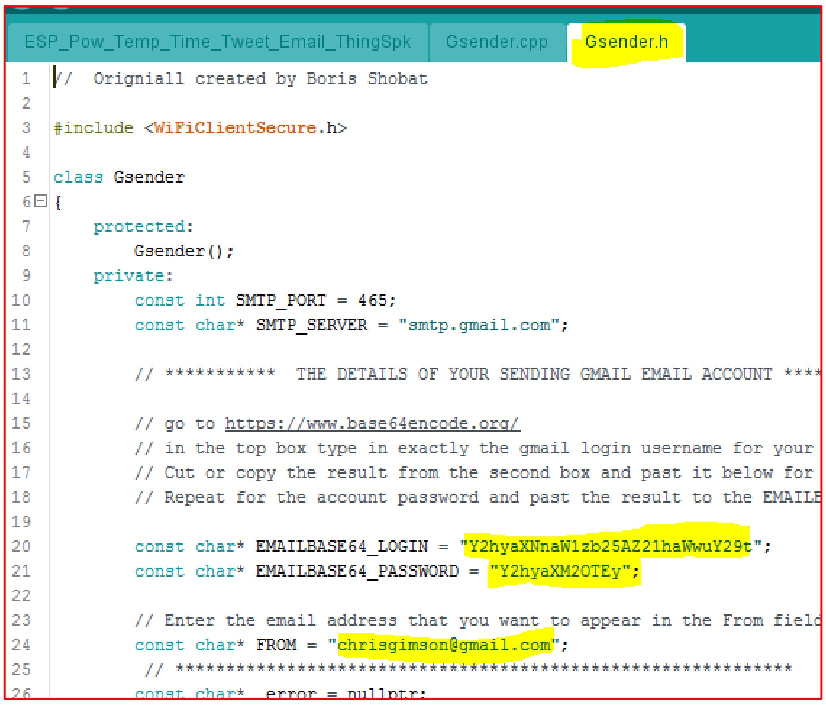
Cut and paste this into the highlighted part of the EMAILBASE64_LOGIN value.

Make sure that the value is enclosed in quotation marks.

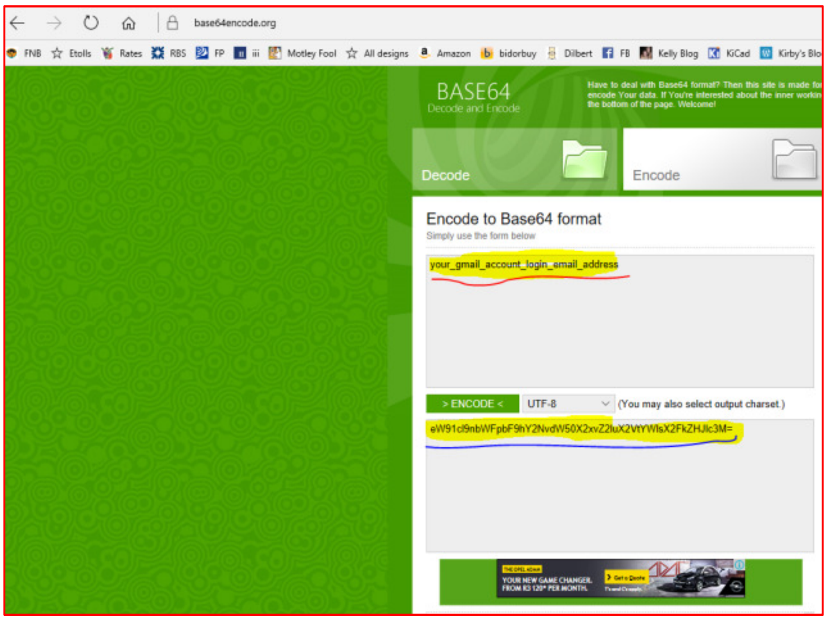
Repeat the process by typing your account password into the top box and cut and pasting the encoded version into the EMAILBASE64_PASSWORD field.

Replace the FROM field with the email address that you want to appear in the From field of the email that is sent out (note that this can be any email address).

The account part of the sketch is now set up.



```
//  Origniall created by Boris Shobat

#include <WiFiClientSecure.h>

class Gsender
{
    protected:
        Gsender();
    private:
        const int SMTP_PORT = 465;
        const char* SMTP_SERVER = "smtp.gmail.com";

        // ***********  THE DETAILS OF YOUR SENDING GMAIL EMAIL ACCOUNT ****

        // go to https://www.base64encode.org/
        // in the top box type in exactly the gmail login username for your
        // Cut or copy the result from the second box and past it below for
        // Repeat for the account password and past the result to the EMAILE

        const char* EMAILBASE64_LOGIN = "Y2hyaXNnaW1zb25AZ21haWwuY29t";
        const char* EMAILBASE64_PASSWORD = "Y2hyaXM2OTEy";

        // Enter the email address that you want to appear in the From field
        const char* FROM = "chrisgimson@gmail.com";
         // *************************************************************
        const char*  error = nullptr;
```


base64encode.org
FNB
Etolls
Rates
RBS
FP
iii
Motley Fool
All designs
Amazon
bidorbuy
Dilbert
FB
Kelly Blog
KiCad
Kirby's Blo
BASE64
Decode and Encode
Have to deal with Base64 format? Then this site is made fo encode Your data. If You're interested about the inner workin the bottom of the page. Welcome!
Decode
Encode
Encode to Base64 format
Simply use the form below
> ENCODE <
UTF-8
(You may also select output charset.)

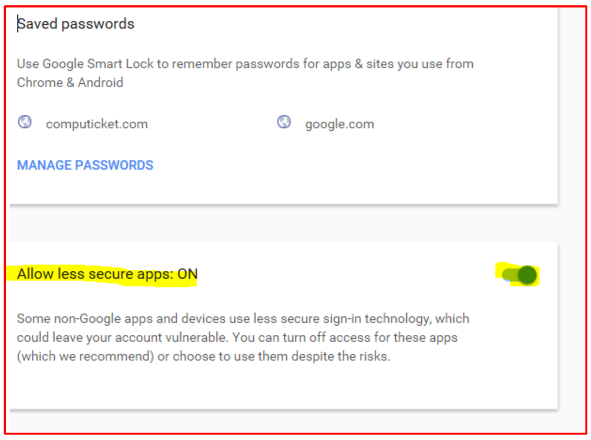
your_gmail_account_login_email_address

eW91cl9nbWFpbF9hY2NvdW50X2xvZ2luX2VtYWlsX2FkZHJlc3M=

The last task for you is to change the security of your gmail account to allow the sketch to connect to it.

Go to your Google account settings at https://myaccount.google.com/security#activity and enable "Allow less secure apps" at the bottom right hand side of the page.

This means that apps only need your email and password when logging in to your gmail account.



Saved passwords

Use Google Smart Lock to remember passwords for apps & sites you use from Chrome & Android

computicket.com google.com

MANAGE PASSWORDS

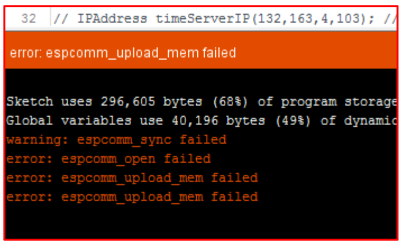
Allow less secure apps: ON

Some non-Google apps and devices use less secure sign-in technology, which could leave your account vulnerable. You can turn off access for these apps (which we recommend) or choose to use them despite the risks.

If you are concerned about security, set up a different account just to use with this sketch

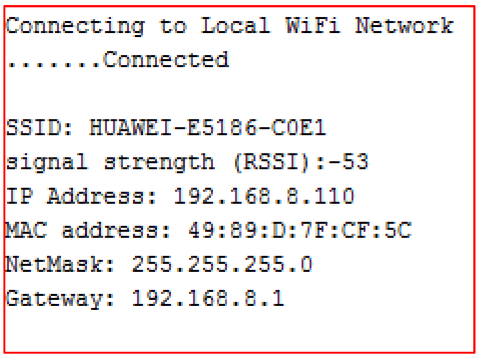
Step 10. The sketch and uploading to the ESP-12

Make sure that before you compile and upload the sketch, you press the S4 switch connected to the GPIO0 pin and then press the S2 Reset button connected to the RST pin. This will force the ESP-12 to boot up in the correct mode for flash programming.
Compile and upload the sketch to the ESP12 board.
If you get an upload message at the end of uploading like this then you have not pressed the switches in the correct sequence or incorrectly wired the switches.



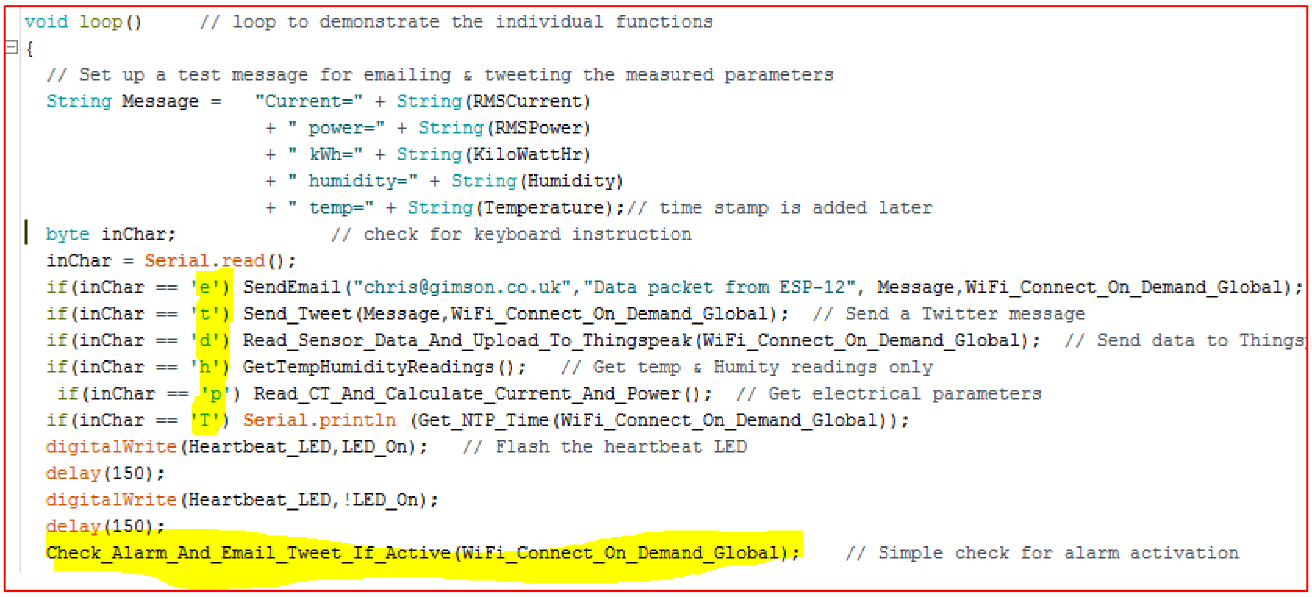
```
// IPAddress timeServerIP(132,163,4,103); //
error: espcomm_upload_mem failed

Sketch uses 296,605 bytes (68%) of program storage
Global variables use 40,196 bytes (49%) of dynamic
warning: espcomm_sync failed
error: espcomm_open failed
error: espcomm_upload_mem failed
error: espcomm_upload_mem failed
```

Open up the serial monitor window of the IDE ensuring that the baudrate is set to 115200 or your value if you changed it in the sketch and press the hardware reset button momentarily.
Assuming that you have correctly loaded the WiFi network SSID and password in the sketch, you should see something like this .....
The screen will show all of the essential parameters of your connection to your own network connection.
The sketch is loaded with a simple void() application that checks the keyboard for selected key depressions. Depending on the selected key, the routine will run one of the standalone functions.



```
Connecting to Local WiFi Network
.......Connected

SSID: HUAWEI-E5186-C0E1
signal strength (RSSI):-53
IP Address: 192.168.8.110
MAC address: 49:89:D:7F:CF:5C
NetMask: 255.255.255.0
Gateway: 192.168.8.1
```

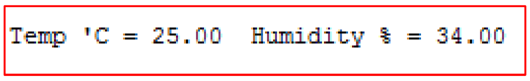
It also calls the Check_Alarm_And_Email_Tweet_If_Active function for possible alarm activity and flashes the heart beat LED once per loop to indicate healthy activity.



```
void loop()     // loop to demonstrate the individual functions
{
  // Set up a test message for emailing & tweeting the measured parameters
  String Message =   "Current=" + String(RMSCurrent)
                    + " power=" + String(RMSPower)
                    + " kWh=" + String(KiloWattHr)
                    + " humidity=" + String(Humidity)
                    + " temp=" + String(Temperature);// time stamp is added later
  byte inChar;               // check for keyboard instruction
  inChar = Serial.read();
  if(inChar == 'e') SendEmail("chris@gimson.co.uk","Data packet from ESP-12", Message,WiFi_Connect_On_Demand_Global);
  if(inChar == 't') Send_Tweet(Message,WiFi_Connect_On_Demand_Global);  // Send a Twitter message
  if(inChar == 'd') Read_Sensor_Data_And_Upload_To_Thingspeak(WiFi_Connect_On_Demand_Global);  // Send data to Things
  if(inChar == 'h') GetTempHumidityReadings();   // Get temp & Humity readings only
   if(inChar == 'p') Read_CT_And_Calculate_Current_And_Power();  // Get electrical parameters
  if(inChar == 'T') Serial.println (Get_NTP_Time(WiFi_Connect_On_Demand_Global));
  digitalWrite(Heartbeat_LED,LED_On);   // Flash the heartbeat LED
  delay(150);
  digitalWrite(Heartbeat_LED,!LED_On);
  delay(150);
  Check_Alarm_And_Email_Tweet_If_Active(WiFi_Connect_On_Demand_Global);    // Simple check for alarm activation
```

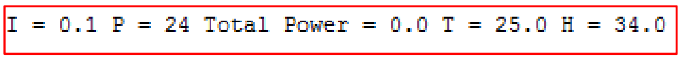
The keyboard actions result in …

1. Press "h" (lowercase) followed by the enter key to get and then view on the serial monitor the temperature & humidity data



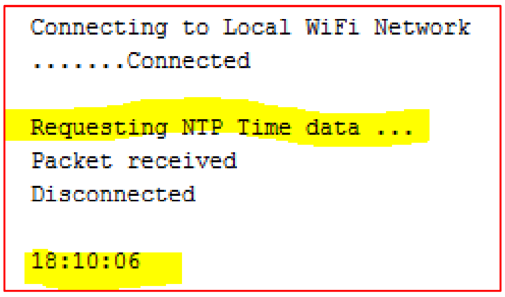
```
Temp 'C = 25.00  Humidity % = 34.00
```

2. Press "p" (lowercase) followed by the enter key to get and view the current (I), power (P), totalised power value (since last reset or power on) and the last stored temperature and humidity data



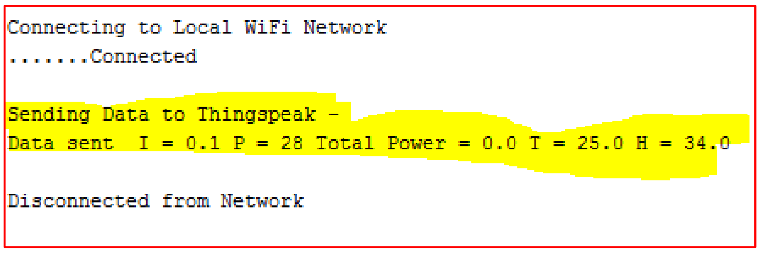
```
I = 0.1 P = 24 Total Power = 0.0 T = 25.0 H = 34.0
```

3. Press “T” (lowercase) followed by the enter key to go online to the NTP server and get & display the latest time in hours, minutes and seconds



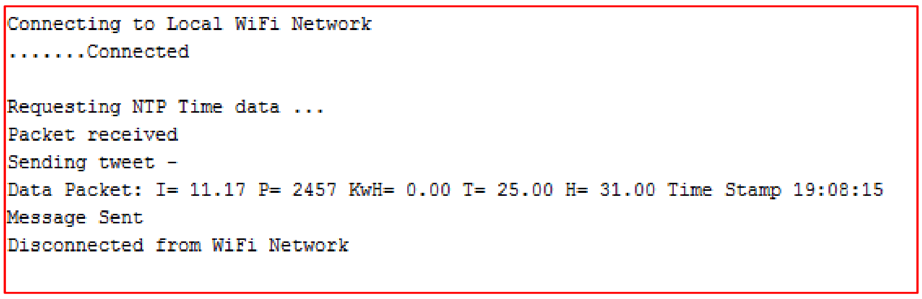
```
Connecting to Local WiFi Network
.......Connected

Requesting NTP Time data ...
Packet received
Disconnected

18:10:06
```

4. Press “d” (lowercase) followed by the enter key will get fresh sensor data and send it to your enabled Thingspeak data channel on the cloud.



```
Connecting to Local WiFi Network
.......Connected

Sending Data to Thingspeak -
Data sent  I = 0.1 P = 28 Total Power = 0.0 T = 25.0 H = 34.0

Disconnected from Network
```

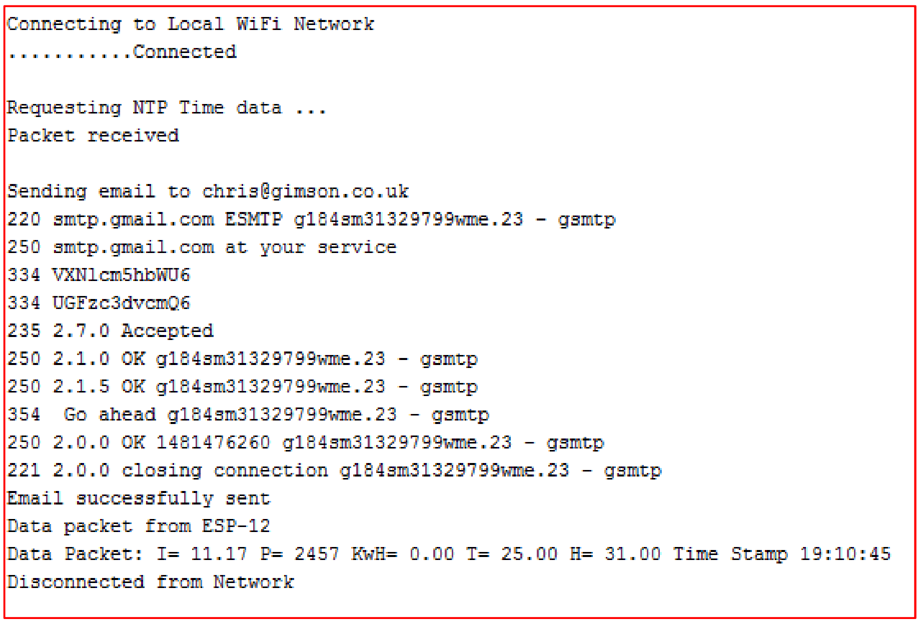
5. Press “t” (lowercase) followed by th e enter key to get fresh sensor data and then send it to your twitter account that you will have linked to your Thingspeak account.



```
Connecting to Local WiFi Network
.......Connected

Requesting NTP Time data ...
Packet received
Sending tweet -
Data Packet: I= 11.17 P= 2457 KwH= 0.00 T= 25.00 H= 31.00 Time Stamp 19:08:15
Message Sent
Disconnected from WiFi Network
```

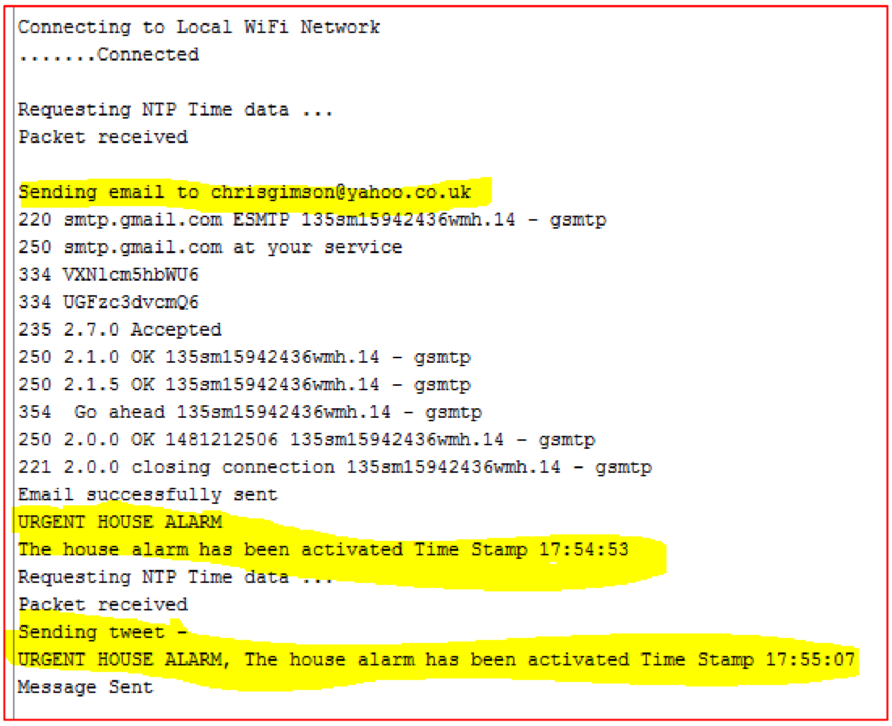
6. Press “e” (lowercase) followed by the enter key to get fresh sensor data and then email it to your chosen email address.

   If on running the sketch, you see “rejected” at line 235, you have loaded in correct login details or not reduced the security settings of the gmail account



```
Connecting to Local WiFi Network
..........Connected

Requesting NTP Time data ...
Packet received

Sending email to chris@gimson.co.uk
220 smtp.gmail.com ESMTP g184sm31329799wme.23 - gsmtp
250 smtp.gmail.com at your service
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 2.7.0 Accepted
250 2.1.0 OK g184sm31329799wme.23 - gsmtp
250 2.1.5 OK g184sm31329799wme.23 - gsmtp
354  Go ahead g184sm31329799wme.23 - gsmtp
250 2.0.0 OK 1481476260 g184sm31329799wme.23 - gsmtp
221 2.0.0 closing connection g184sm31329799wme.23 - gsmtp
Email successfully sent
Data packet from ESP-12
Data Packet: I= 11.17 P= 2457 KwH= 0.00 T= 25.00 H= 31.00 Time Stamp 19:10:45
Disconnected from Network
```

7. Activate your designated alarm input for greater than 1 second and then release it. The unit will send a customisable message to both your email address and your Twitter account.



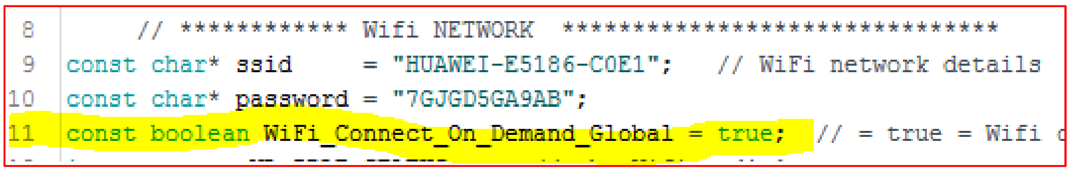
```
Connecting to Local WiFi Network
.......Connected

Requesting NTP Time data ...
Packet received

Sending email to chrisgimson@yahoo.co.uk
220 smtp.gmail.com ESMTP 135sm15942436wmh.14 - gsmtp
250 smtp.gmail.com at your service
334 VXNlcm5hbWU6
334 UGFzc3dvcmQ6
235 2.7.0 Accepted
250 2.1.0 OK 135sm15942436wmh.14 - gsmtp
250 2.1.5 OK 135sm15942436wmh.14 - gsmtp
354  Go ahead 135sm15942436wmh.14 - gsmtp
250 2.0.0 OK 1481212506 135sm15942436wmh.14 - gsmtp
221 2.0.0 closing connection 135sm15942436wmh.14 - gsmtp
Email successfully sent
URGENT HOUSE ALARM
The house alarm has been activated Time Stamp 17:54:53
Requesting NTP Time data ...
Packet received
Sending tweet -
URGENT HOUSE ALARM, The house alarm has been activated Time Stamp 17:55:07
Message Sent
```

Notes

As per the original compliation , the sketch remains offline when it is not performing one of the required functions (I.e it connects to the WiFi before performing its task and then disconnects. This is controlled by the Boolean flag "WiFi_Connect_On_Demand" that is passed to each of the WiFi sensitive functions.

If WiFi_Connect_On_Demand is true, the function will automatically connect and disconnect the network as it is required. If false, the functions will assume that the Wifi is permanently connected.

Should you wish to run the system with the WiFi permenantly on (E.g. f you also have a remote control function running that requires continual connection) then you should set the flag "WiFi_Connect_On_Demand_Global" to false in the WiFi start up parameters at the start of the sketch



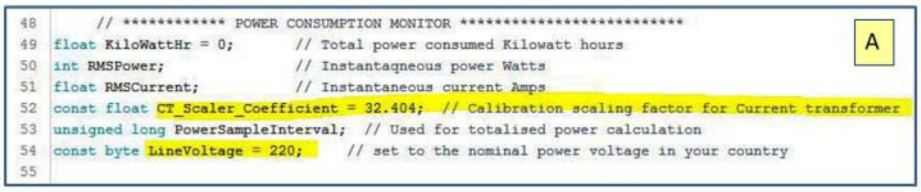
```
    // ************ Wifi NETWORK  ********************************
const char* ssid     = "HUAWEI-E5186-C0E1";   // WiFi network details
const char* password = "7GJGD5GA9AB";
const boolean WiFi_Connect_On_Demand_Global = true;   // = true = Wifi
```

Step 11. Calibration of the power monitoring channel

So, you have compiled the sketch after loading in all of the required configuration parameters but when you run the Read_CT_And_Calculate_Current_And_Power() function, you get incorrect readings for current and power!

Assuming that your load is within the range that you have calculated for your CT shunt resistor there are two possible reasons for such an error

A. You have not set the LineVoltage parameter to the correct value for your mains supply or more likely the calibration parameter CT_Scalar_Coefficient is not correct for the CT configuration that you are using.



```
    // ************ POWER CONSUMPTION MONITOR ***************************
float KiloWattHr = 0;        // Total power consumed Kilowatt hours
int RMSPower;                // Instantaqneous power Watts
float RMSCurrent;            // Instantaneous current Amps
const float CT_Scaler_Coefficient = 32.404;  // Calibration scaling factor for Current transformer
unsigned long PowerSampleInterval;  // Used for totalised power calculation
const byte LineVoltage = 220;      // set to the nominal power voltage in your country
```

This needs setting to an appropriate value for your set up and the only way that we can really do this is by either comparing the sketch calculated value under measurement conditions with another power or current monitoring unit that you conveniently have or by calibrating with a know electrical load - preferably a resistive one so that we do not introduce reactive or inductive losses. Acceptable test loads could be several incandescent lamps, a kettle, iron.

B. The "must have" is a knowledge of what the load actually is in terms of amps and/or power consumption in Watts. The best method is certainly to be able to used a reference measurement device. An AC clamp meter is ideal for measuring the current - the "clamp" part of the meter is actually a CT and you simply clamp it around the live power feed to your load. The domestic power meters available on the market also use a CT device separate to the display device in the same way that we are using ours. Last but not least you can also get socket power monitors that plug into your wall socket and you plug the load into a socket that is integral to the monitor.

Any of these devices allows you to test a range of test loads and you can directly compare the sketch reading with the test meter to obtain the the two readings necessary to check the calibration and then correct it. (Switch the load on and take a reading by pressing the "p" key followed by the "Enter" key.

We need the "sketch reading" and the "correct reading" from the test meter (Amps or Watts) for a range of variable loads preferably at the low end and at the high end E.g. 200W and 2000W.

If you are not using a test meter but simply relying on the stated load (E.g 2x100W lamps, 2000 heater etc) then the "correct reading" will simply be the known load.

The link between the correct scaling factor(**CSF**), the present scaling factor (32.4), the "sketch reading" and the correct reading is given by:

(CSF/32.4) = (sketch reading) / correct reading) (same equation with the current or power readings)

This gives us **CSF = (sketch reading x 32.4) / (correct reading)**

If possible, do this several times with different loads (without changing the sketch coefficient) and take the average of the calculated CSF and then replace the current value of 32.4 in the sketch with your calculated one.

Repeat the test and check that the readings with the new coefficient are satisfactory

Note:
If you start with negative readings from the sketch then either increase the test load and/or change the CT_Scalar_Coefficient to an arbitrary value until you obtain a positive measurement reading.

SAFETY WARNING!

It is clear that you are dealing with dangerous mains voltages that can be fatal if not handled properly. The CT transformers used in both the sketch and the test equipment if used do not connect directly to the mains supply. It is sufficient to clamp the measuring device or CT around either the single live or neutral wire of the mains circuit (not both!) however even this act may bring you into close proximity with exposed live terminals.

TAKE CARE!! *if you are not sure what you are doing, seek help from a more qualified person.*

Step 12. Support Stuff

This section points you to additional resources to help you build this project.
The following data sheets can be downloaded in the zip file Datasheets.zip
(You can also easily google their names and you will not be short of information!)

- DHT11/22 temperature and humidity sensor
- SCT013 30/1V Current transformer
- SR501 PIR sensor

Support information …

- A formatted version of the sketch – it provides easier reading and understanding of the key elements including those parts that you need to customise. ESP8266_formatted_Toolbox_sketch.zip
- The complete project document as a pdf file Instructable_ESP-12_Toolbox_Project_Documentation.zip
- The ESP_12_Toolbox.ino sketch with Gsender.h and Gsender.cpp support files - ESP_12_Toolbox_sketch.zip Unzip these files and put them together in a sub-directory (using the same name of the sketch) where you normally put your sketches.

Useful URL's

As well as a lot of experimentation, the key to any successful build is …. research. Not everything has to be original, the whole point of open source is to allow cutting and pasting of parts or complete modules. The instructables site alone is a wealth of information on all interesting topics and I list just a few here that helped me towards this instructable. I include them as acknowledgement to just some of the enablers of my work. This project is not the final item but it is meant to be an enabler for other project builders

- http://www.instructables.com/id/ESP8266-GMail-Sender/
- Needed for encoding your login information for the email sending function - https://www.base64encode.org/
- http://www.instructables.com/id/Make-ESP8266-REV-12-prototype-friendly/
- http://www.instructables.com/id/Getting-Started-with-the-ESP8266-ESP-12/
- http://www.instructables.com/id/ESP8266-With-DHT11-Temperature-Humidity-Monitor/
- http://www.instructables.com/id/Simple-Arduino-Home-Energy-Meter/
- FT232RL https://www.sparkfun.com/products/12731
  http://www.hobbytronics.co.uk/ftdi-basic-plus