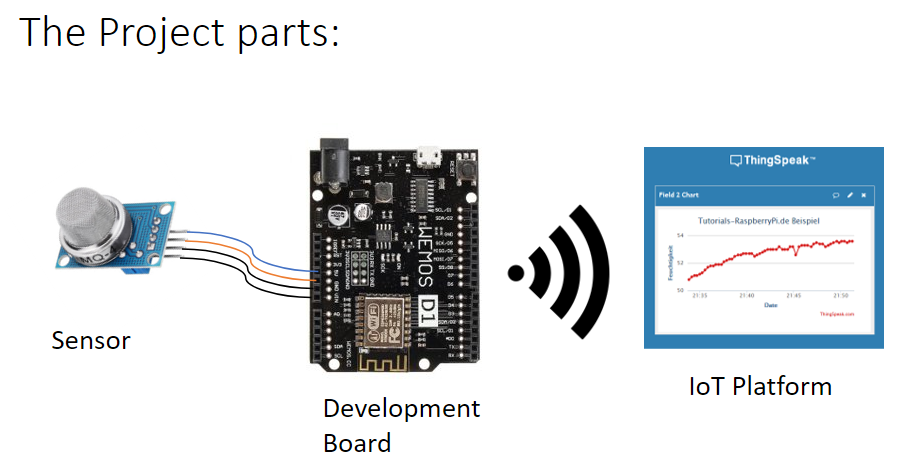
**Work Experience Week 2022 Air Quality Project**

**Intro:**

You’re going to create a device that will measure the air quality and will send that data to an online platform where you can visualise and compare it to others.



We’ll talk through sensors later. Thingspeak is an IoT (Internet of things) platform where we’ll collect all our data from our sensors and process that data.

**Development Board**

This board is called a **wemos** and is a wifi enabled microcontroller – which is great for prototyping internet enabled electronics projects. It runs on the **arduino** platform – which is a great program for writing and uploading C++ code.

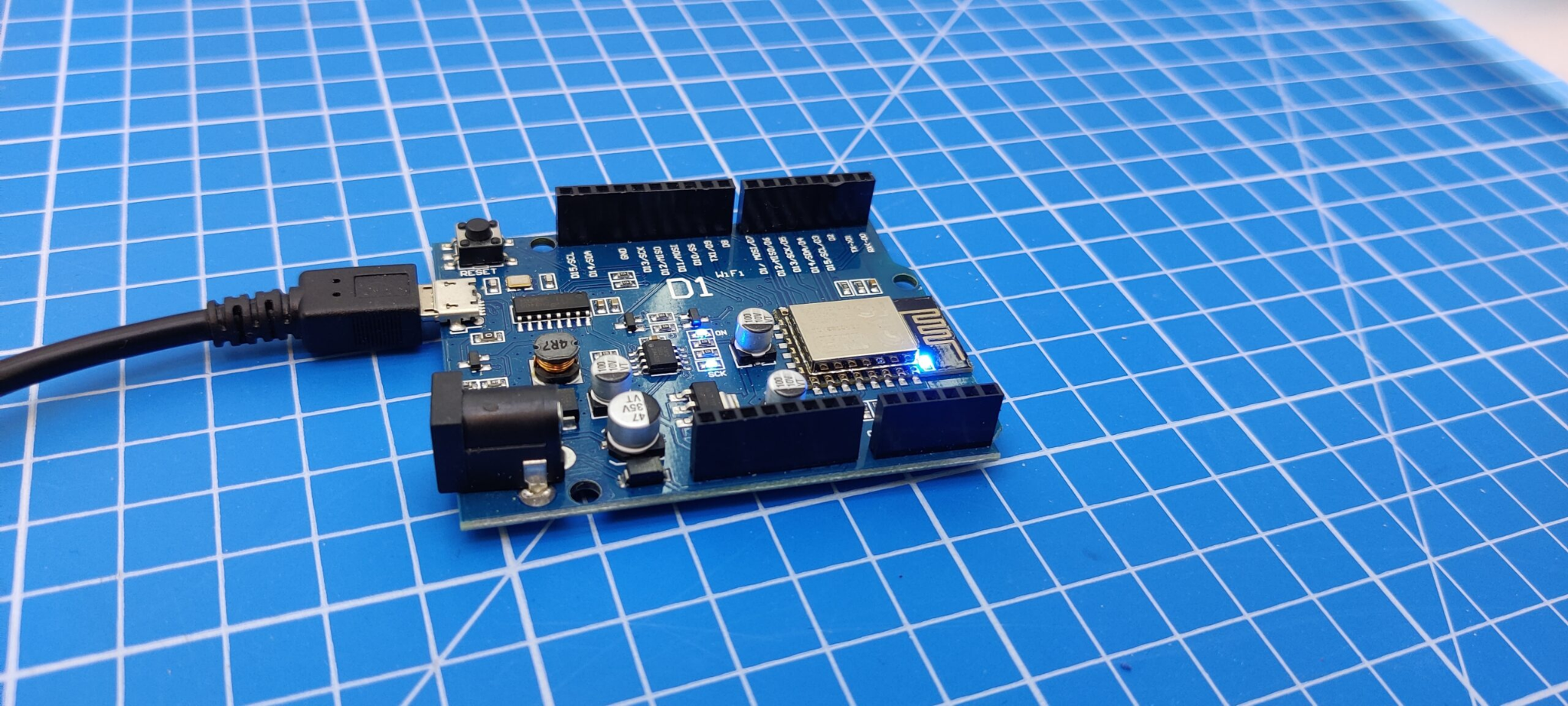
**1. Setting up your Wemos to program:**

1. Download and install Arduino from [here](https://www.arduino.cc/en/software) (mac, windows & linux) *-* ***if you’re using the outreach laptops this will be installed for you.***
2. Open Arduino
3. Click File > Preferences
4. Paste the following into the Additional Boards Manager URLs box:

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

1. Then click OK.
2. Click Tools > Boards > Boards Manager
3. Click Type > Contributed
4. Find **esp8266 by ESP8266 Community** and click install
5. Click Tools > Board > ESP8266 Boards > LOLIN(WEMOS) D1 R2 & mini
6. Click Tools > Upload Speed > 115200
7. Plug in wemos to computer using the USB cable, give your computer a few seconds to recognise the board (windows should make a noise when plugged in)

*Drivers for windows and mac* ***only*** *if USB doesn’t recognise:* [*https://www.wemos.cc/en/latest/ch340\_driver.html*](https://www.wemos.cc/en/latest/ch340_driver.html)



1. Click Tools > Port > COMx, where x is any number greater than 1.   
   If you can’t see a COM port, check your USB, possibly try a different USB port on your computer.

If you can’t see a suitable COM port show up at this point check back through the steps above and try again. If you still can’t, ask one of our helpers for help.

**2. Blink Code:**

1. Click File > Examples > ESP8266 > Blink. This will load an example piece of code:

void setup() {

pinMode(LED\_BUILTIN, OUTPUT); // Initialize the LED\_BUILTIN pin as an output

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED\_BUILTIN, LOW); // Turn the LED on (Note that LOW is the voltage level

// but actually the LED is on; this is because

// it is active low on the ESP-01)

delay(1000); // Wait for a second

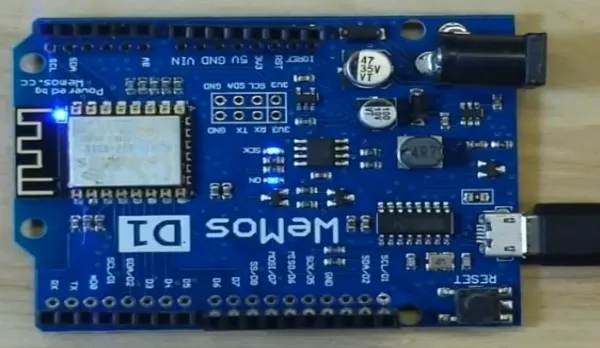
digitalWrite(LED\_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH

delay(2000); // Wait for two seconds (to demonstrate the active low LED)

}

1. Click the  upload button. You should see the sketch compiling at the bottom of the arduino window, and then a writing % in red text. It may take a little while to upload. Once it’s Done uploading you should see a blue LED light blink **on for 1 second and off for 2 seconds** and repeat over and over.

*Note: when you first plug in your wemos the light may already be flashing on and off* ***every second*** *– now it should be off for two seconds, then on for one if the upload succeeded.*

  
*The blue LED on the* ***right*** *is the one that should be flashing!*

Below is the code you’ve loaded on to the Wemos board. We’re blinking on and off an LED light (Light Emitting Diode) in the loop in the code below.

void setup() {

pinMode(LED\_BUILTIN, OUTPUT); // Initialize the LED\_BUILTIN pin as an output

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED\_BUILTIN, LOW); // Turn the LED on (Note that LOW is the voltage level

// but actually the LED is on; this is because

// it is active low on the ESP-01)

delay(1000); // Wait for a second

digitalWrite(LED\_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH

delay(2000); // Wait for two seconds (to demonstrate the active low LED)

}

Let’s talk through the main features of the code:

* **void setup()**  - function called at the **start** of the program, everything within the { } will run only once. In this case we’re setting up the LED pin as an output
* **void loop()** –everything within the{ } is **looped over and over** after setup has finished - in this case turning the LED on and then off over and over.
* **//** - means **comment**, we use this to annotate our code with notes
* **DigitalWrite –** we're telling a digital **pin** on the wemos to turn on or off – in this case the LED light, but in future our air quality sensors!
* **delay(x)** – pause program for x number of milliseconds. 1000ms = 1 second.

**Why don’t you change the delay values to smaller or larger values**

**what do you think will happen to the LED?**

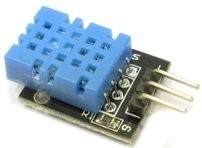
**Upload the code and see what happens!**

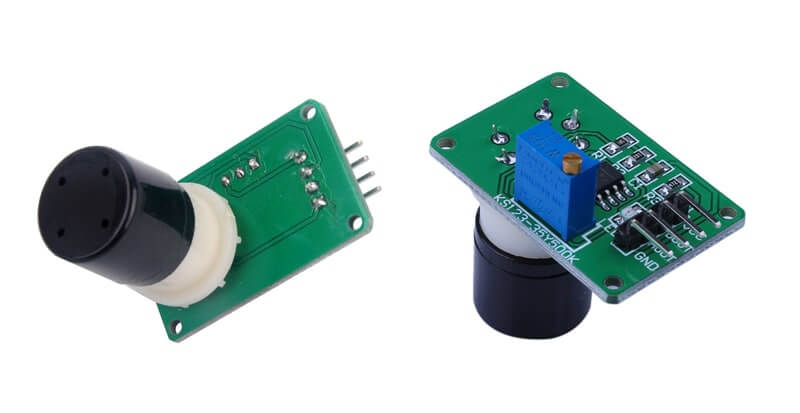
**3. Sensors Research**You’ve been given a sensor - A sensor is a device that detects and responds to some type of input from the physical environment.   
Different groups will have different sensors for detecting various air quality related aspects.

Find your sensor from the list below and research using the datasheet, library (chunks of code to help us read the sensors) and web searches: We advise splitting up in your team to answer the various parts:

* How do your sensors work? - What are you trying to measure? See page below.
* how will you wire them up?
  + analog or digital pins with the wemos? (check the library)
  + What voltage does the sensor run on - 5V or 3V3? (check the datasheet)
  + Sketch a wiring diagram for your sensor & wemos
* how do you think the code will work (see examples)

**MQ-135** - [Datasheet](https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf) - [Library](https://github.com/ViliusKraujutis/MQ135) - [Code Guide](https://docs.google.com/document/d/1-IcmL0PFMLWscSxfrf83bC4acDqrkmR5jGGRL1-vqUM/edit?usp=sharing)

**DHT11** - [Datasheet](https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf) - [Library](https://github.com/beegee-tokyo/DHTesp) - [Code Guide](https://docs.google.com/document/d/1Er8DJXRYOe-j82qNtZJ3iiSVNrDEQ1L5Dd7ooW7rPTk/edit?usp=sharing)

**MQ-131** - [Datasheet](https://cdn.sparkfun.com/assets/9/9/6/e/4/mq131-datasheet-low.pdf) - [Library](https://github.com/ostaquet/Arduino-MQ131-driver) - [Code Guide](https://docs.google.com/document/u/0/d/1xAPgOu6WURDJY_ohFiXQO--e6pOUWQmvoSLwsl4P7ls/edit)

**MQ-7** - [Datasheet](https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf) - [Library](https://github.com/fjebaker/MQ7) - [Code Guide](https://docs.google.com/document/u/0/d/1SOxqunTeVBkYko7gWsXOzr5aM6tMGpSdKJTmaW5Ckjg/edit)

The following tasks below would be good to divide up amongst your group:  
**4. Wiring** - Electronic Engineer,

**5. Coding** - Computer Scientist

**6. Setting up the IoT data** - Engineering Mathematics  
 **4. Sensor wiring**

1. Firstly **wiring** your sensor -
   1. it will need some power (is it 5V or 3.3V?),
   2. a ground (also called GND)
   3. a digital or analog pin (Ax or Dx)

have a look at the datasheets and library and see if you can work it out. Draw out a diagram of how you’ll wire up the sensor, and then get one of our helpers to check over your wiring plan.

1. **Ensure your wemos is unplugged from the USB cable** and do your wiring with the jumper leads provided - get a helper to check before you plug it in.

**5. Library setup & reading your sensor**

1. Download the library zip folder (link above) for your sensor from github by clicking Code > Download ZIP
2. In Arduino click Sketch > Include Library > Add .ZIP Library and add the zip file from your downloads folder.
   1. *On my outreach laptop this crashes arduino! If it crashes for you too, this is an alternate way to install the library:*
      1. *Extract the zip by right clicking on zip > extract all*
      2. *Find the extracted folder, double click it, copy the folder inside*
      3. *Go to Documents > Arduino > libraries and paste the folder here.*
      4. *You may need to restart arduino if it doesn’t detect the library*
3. Start a new sketch in Arduino (File > New) then look at the **code guide** for your sensor above and work through it.

**6. Connecting the Wemos To Thingspeak**

**Sign in to Thingspeak:**

1. Go to Thingspeak.com. Create an account, verify your email.
2. Click Channels > New Channel
3. Set Channel name as your name
4. Click Save channel
5. Click Sharing > Share channel view with everyone
6. Click API Keys and copy your Write API Key. You’ll need this API key for the code below.

Once your team has got your sensor wired up, you’re reading values from your sensor and your thingspeak account is set up it’s time for your team to rejoin together and get that data sending to thingspeak:

**7. Sending data to thingspeak from wemos**

This is a good time to regroup. Keep your sensor code open in one window and start a new sketch for this section where you can copy in some of the sensor code as you go. So click File > New.

The parts of this code you’ll have to modify are in red. Read the following carefully as you create this new thingspeak code and copy over lines code from your sensor code.

1. Add the following below to the **top of your code**:

#include "ESP8266WiFi.h"

String writeApiKey = "thingspeakWriteApiKeyHere";

const char\* ssid = "wifiNameHere";

const char\* password = "wifiPasswordHere";

const char\* server = "api.thingspeak.com";

WiFiClient client;

// sensor library etc here

* 1. Fill in your wifi name, wifi password and write api key within the appropriate “” quotemarks. This is to tell your wemos what wifi to connect to and which thingspeak channel to send the data to:
     1. Fill in your thingspeak write API key (from your thingspeak account)
     2. Wifi name: “**WEX22aq**”
     3. Wifi password “**airquality**”
  2. From your sensor code, copy over your sensor library include and any other bits of code you may have had **before void setup** to where it says   
     “*sensor library etc here”*

1. Add the following as **void setup()**:

void setup(){

Serial.begin(115200);

delay(10);

WiFi.begin(ssid, password);

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED){

delay(500);

Serial.print(".");

}

Serial.println("WiFi connected");

// sensor setup / calibration code here

}   
  
This code is to connect to the wifi before we can start working with the sensor.

* 1. Add your sensor calibration code (code you had in **void setup** of your sensor code) to where it says “// *sensor setup / calibration code here*” - within the { } of void setup.
  2. Make sure you don’t add another serial.begin - we’ve got one here already!

3. Add the following as **void loop** - this code will read your sensor (once you add that bit in!) and then send that data to Thingspeak every 20 seconds:

void loop()

{

if (client.connect(server,80)) {

// sensor(s) reading to float(s) here

String postStr = writeApiKey;

postStr +="&field1=";

postStr += String(FLOATNAMEHERE);

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+writeApiKey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(postStr.length());

client.print("\n\n");

client.print(postStr);

// serial prints here

Serial.println("Sending data to Thingspeak");

}

client.stop();

Serial.println("Waiting 20 secs");

// thingspeak needs at least a 15 sec delay between updates

// 20 seconds to be safe - change this to 60 if using MQ-131

delay(20000);

}

* 1. Paste your sensor code that reads the sensor and write to the float (code you had inside **void loop**) where it says *“sensor reading to a float here”*
  2. Change *FLOATNAMEHERE* to the variable name for your sensor from your sensor code
* If you’re using the DHT-11 you’ll want to add a field2 here and a second float for temperature
  1. Add your serial print for the air quality value(s) from your sensor code to where it says “*serial prints here*”
  2. we’re going to have a 20s delay here for thingspeak - make sure you don’t add your sensor delay here too.
* If you’re using MQ-131 change this to 60s delay.

4. Click the  verify button to check your code for any errors. If it’s ok it should say ”Done Compiling” at the bottom. If not, double check your code against the code above for any mistakes, if you’re not sure, do ask for help.

5. Now we’ve created some code to read the sensor lets test it! Plug the USB cable into the wemos and click the upload button.

6. Once the code is fully uploaded click the  serial monitor button. Make sure at the bottom right of the serial monitor the baud is set to 115200 baud.

If all goes to plan, you should see it connect to wifi, give you some sensor data and send it to thingspeak. Check the thingspeak page to see the data show after a few seconds.

If you have extra time at this point we can potentially give your group a second sensor to add on to the setup - often the air quality sensors get improved data accuracy from temperature / humidity sensors.

In tomorrow’s session, we’ll take a look at the data that’s been running on thingspeak and analyse it with some Eng Maths tools!