# Enabling IoT technology with low-cost platforms and software tools

## Introduction

Computing is about more than the PC on your desktop! Embedded devices are everywhere – from wireless telecommunications infrastructure points to electronic point of sale terminals. One definition of an embedded system is:

> "An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints."
>
> (http://en.wikipedia.org/wiki/Embedded_system)

The purpose of this workshop session is to introduce you to building Internet-of-Things applications using the popular MQTT protocol and super low-cost hardware. In less than an hour you will build an IoT enabled room monitoring system that measures the humidity and temperature and transmits the data using open IoT protocols over the Internet.

We will use the open source Arduino platform – a cheap, simple to program, well documented prototyping platform for designing electronic systems. The total cost of the hardware used in this tutorial is less than £40 (excluding networking equipment!).

## Background

Using HTTP for data exchange with embedded systems is tricky due to the resource constraints inherent in low-cost hardware platforms. For this reason, many Internet-of-Things devices (particularly those with strict resource constraints such as small micro-controllers) avoid HTTP and use alternative protocols such as CoAP and MQTT.

The official MQTT 3.1.1 specification[1] says:

> "MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium."

A key benefit of MQTT is the **publish – subscribe** model which reduces network traffic by allowing a server to push updates to subscribed clients when there is new information of interest available. Figure 1 illustrates this interaction.

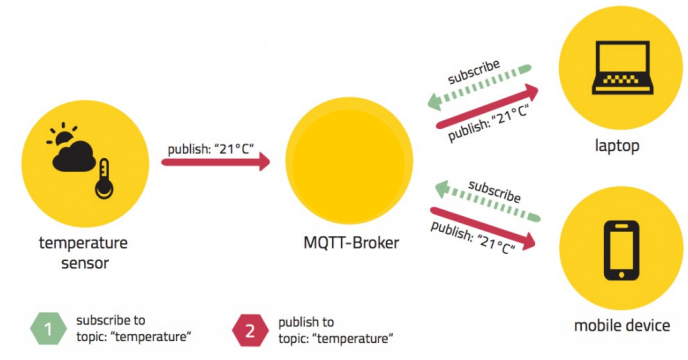


Figure 1 – MQTT's publish – subscribe model

1 http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

**Instructions**

In this session you are going to develop a simple, low-cost, Internet-of-Things-enabled room environment monitoring system. This will measure temperature and humidity within the room and send that data over the Internet to a remote server where it can be accessed from anywhere in the world!

The first thing we need to do is to build the circuit shown in Figure 2.

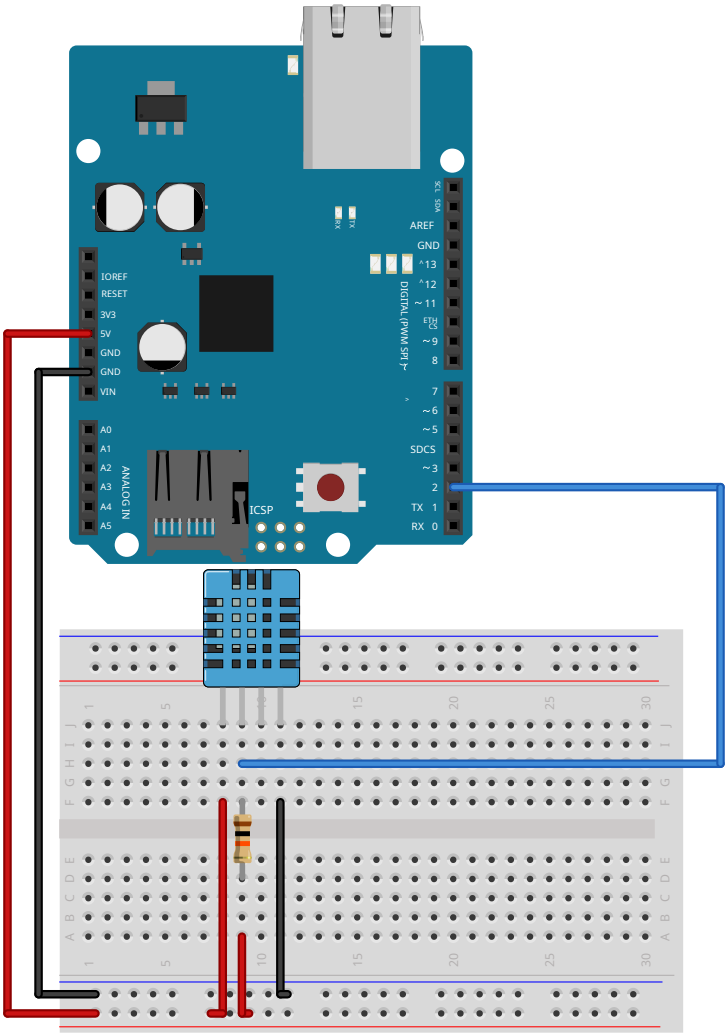


Figure 2 – The IoT-enabled room monitoring circuit

If you need a hand with this – please shout out :)!

You might notice looking around the room that some of you have blue sensors and some of you have white sensors – they are pretty much the same, but the white sensors have slightly better range and accuracy!

(We didn't have enough of a single type to go around!)

Figure 3 shows a photograph of my completed system.

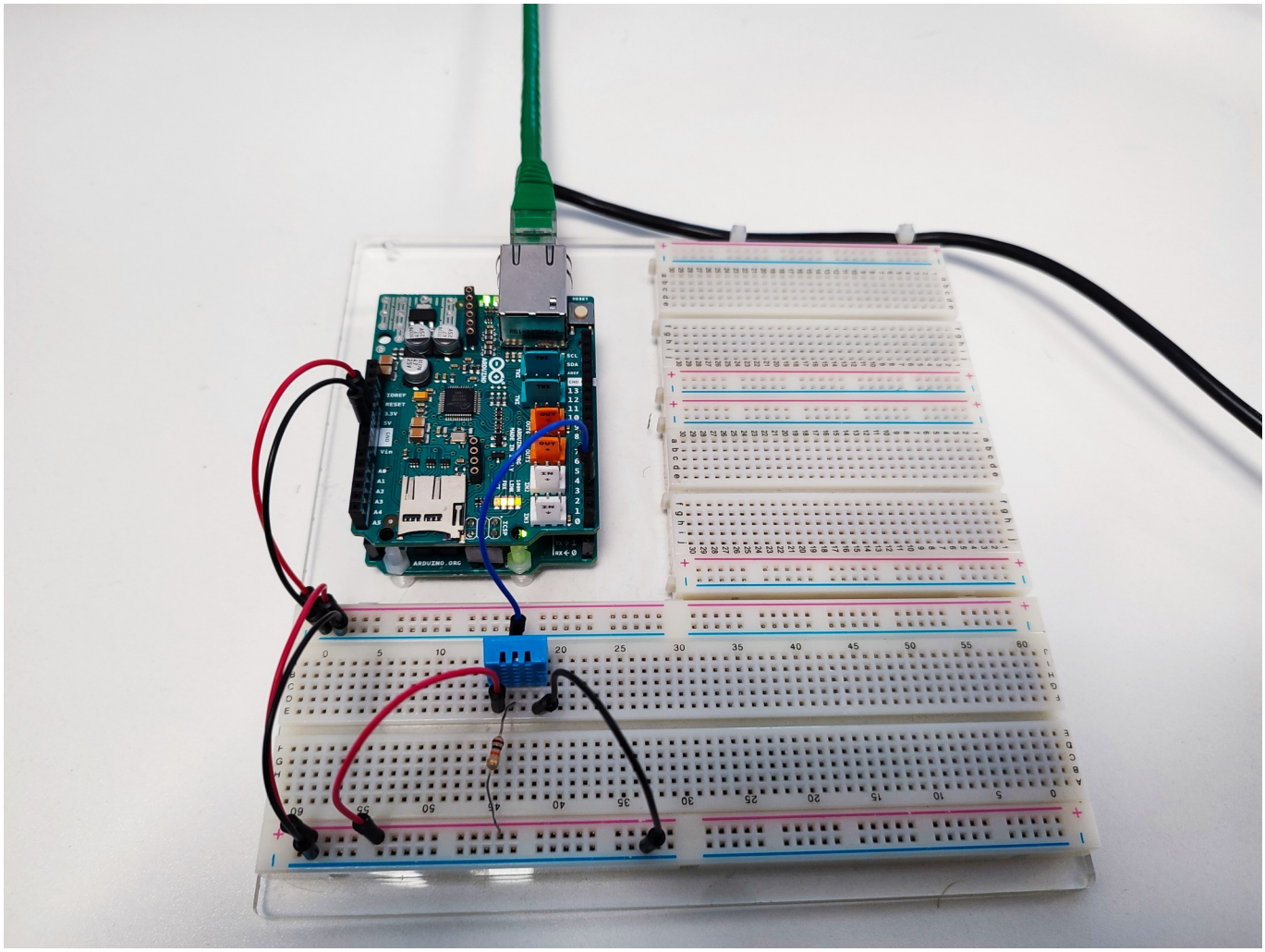

Figure 3 – My completed IoT enabled room monitoring system

The next thing we need to do is to connect the Arduino Uno and Ethernet shield to the network port on the bench in front of you (see Figure 4). Please use the empty network port and make a note of the number on the socket!

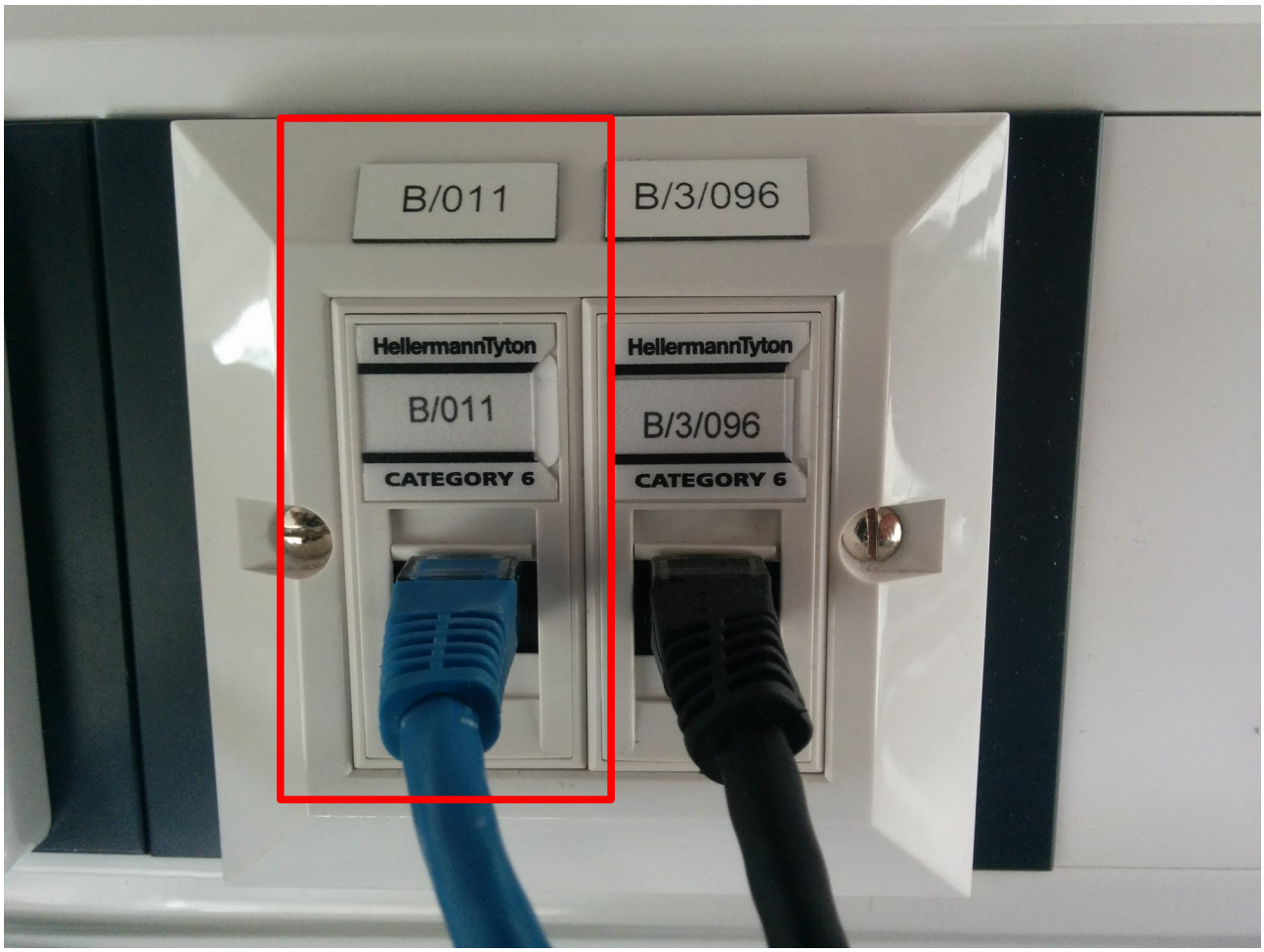


Figure 4 – The network ports

You will also need to connect up the patch panel to the network switch at the side of the room! Use the socket number here to identify the correct port on the patch panel, and then connect that to an empty space on the network switch.

Again, if you need a hand with this – please shout out :)!

We can now proceed to develop the code for our IoT system!

The point of this application is to make a connection to our MQTT broker, post the sensor data to the remote MQTT broker, and receive status updates back. If the connection is broken for any reason then the Arduino MQTT client will keep trying to reconnect and then send a status message when it does.

There are a few places in the supplied code that you will need to make some edits to make everything work correctly!

If at any point you need a hand with any of this – please shout out :)!

Code listing 1 shows some of the initialisation code. You can see that the MAC address is highlighted – this needs changing to the correct address of your hardware (otherwise we might not be able to get an IP address). This MAC address is written on a post-it note by the hardware!

Code listing 1:

```
/*
 * iot_data_node.ino
 *
 * arduino sketch to send temperature and humidity data to a public mqtt broker
 * in this example we are using the arduino ethernet shields for communication
 *
 * author:  alex shenfield
 * date:    21/02/2022
 */

// INCLUDES

// we are using the Ethernet library (which supports the WizNet w5500 Ethernet
// controller)
#include <SPI.h>
#include <Ethernet.h>

/// include the pubsub client
#include <PubSubClient.h>

// include the sensor library
#include "dht.h"

// ETHERNET CONNECTION DECLARATIONS

// mac address
byte mac[] = {0x90, 0xA2, 0xDA, 0x11, 0x44, 0xC7};

// MQTT DECLARATIONS

// mqtt server details
const char server[] = "test.mosquitto.org";
const long port     = 1883;

// get an Ethernet client object and pass it to the mqtt client
EthernetClient ethernet_client;
PubSubClient client(ethernet_client);

// timing variables - send data approximately every 10 seconds
long previous_time = 0;
long connection_interval = 10000;

// SENSOR DECLARATIONS

// dht sensor pin
#define dhtpin  2
```

The next place we have to edit the code is in the network connection method (see highlighted parts of code listing 2).

Firstly, we need to ensure that we provide a unique name for our system when we connect to the MQTT broker otherwise we will run in to connection problems! This is shown in the green highlighted part.

Secondly, we should change the name of the topic we are going to use to send and receive data to something unique (you can see I've just used my name). This will allow us to distinguish our data from everybody else's! This is shown in the yellow highlighted part.

Code listing 2:

```
// reconnect to server
void reconnect()
{
  // loop until we're reconnected
  while (!client.connected())
  {
    Serial.println("attempting mqtt connection...");

    // try to connect to the mqtt broker
    if (client.connect("axs_arduino_190310"))
    {
      Serial.println("... connected");

      // once connected, subscribe to the appropriate feeds
      client.subscribe("alexshenfield/shu-iot-demonstrator/#");

      // ... and publish an announcement
      client.publish("alexshenfield/shu-iot-demonstrator/status-messages", "we are alive!");
    }
    else
    {
      // print some error status
      Serial.print("connection failed, rc = ");
      Serial.print(client.state());
      Serial.println();
      Serial.println("we will try again in 5 seconds");

      // wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

And finally, we need to change two parts of our data sending method (see highlighted parts of code listing 3).

Here we need to change the green highlighted part dependent on which sensor we are using. If we have a blue sensor, we can leave the code as it is (this is the DHT11 sensor). If we have a white sensor (this is the DHT22 sensor), we need to change the green highlighted line to:

```
int chk = DHT.read22(dhtpin);
```

The second change we need to make is to the name of the topic shown in the yellow highlighted part. This first part should match the name you used in the previous step!

Code listing 3:

```
// function to send the sensor values
void send_sensors()
{
  // if the client is already connected then send data
  if (client.connected())
  {
    // dht sensor

    // create a dht sensor instance and read it
    dht DHT;
    int chk = DHT.read11(dhtpin);

    // if the return code indicates success we get the data, turn it into a string, and
    // send it out via mqtt
    if(chk == DHTLIB_OK)
    {
      // get the data from the sensor object
      float h = DHT.humidity;
      float t = DHT.temperature;

      // convert the sensor readings to character arrays
      char humidity_level[6];
      char temperature_level[6];
      dtostrf(h, 5, 2, humidity_level);
      dtostrf(t, 5, 2, temperature_level);

      // send via mqtt
      client.publish("alexshenfield/shu-iot-demonstrator/humidity", humidity_level);
      client.publish("alexshenfield/shu-iot-demonstrator/temperature", temperature_level);
    }
  }
}
```

That's it – you're done with the programming! Now you need to compile the code and load it onto the board.

First, select the correct board (Arduino Uno) and serial port (see Figures 5 and 6).

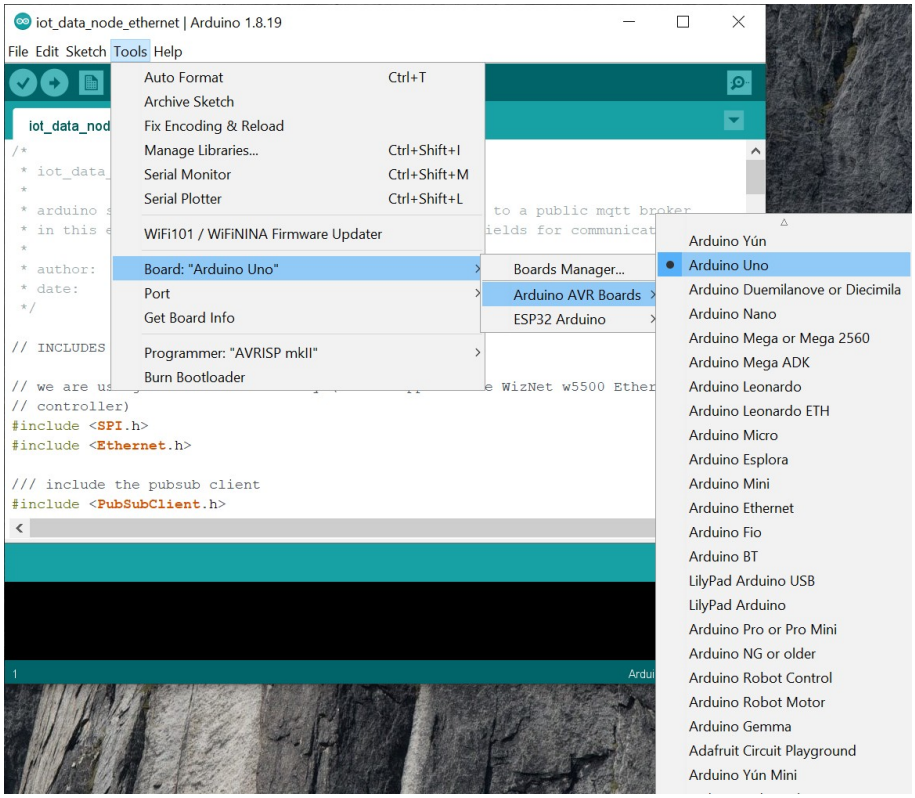


Figure 5 – Selecting the correct Arduino board

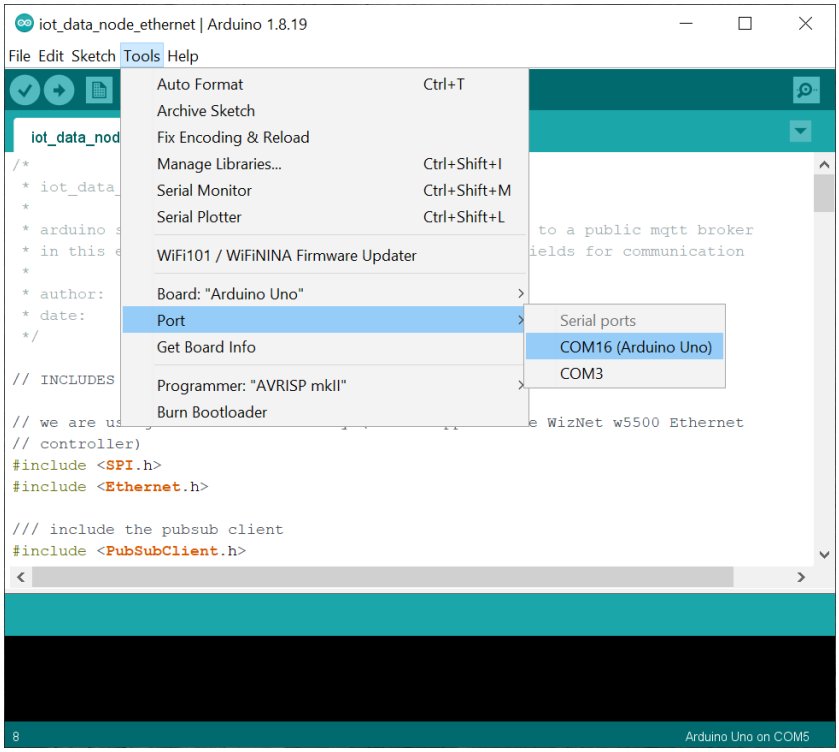


Figure 6 – Selecting the serial port

Then upload the program to the Arduino board (by pressing the right facing arrow icon in the toolbar of the Arduino IDE – see Figure 7).

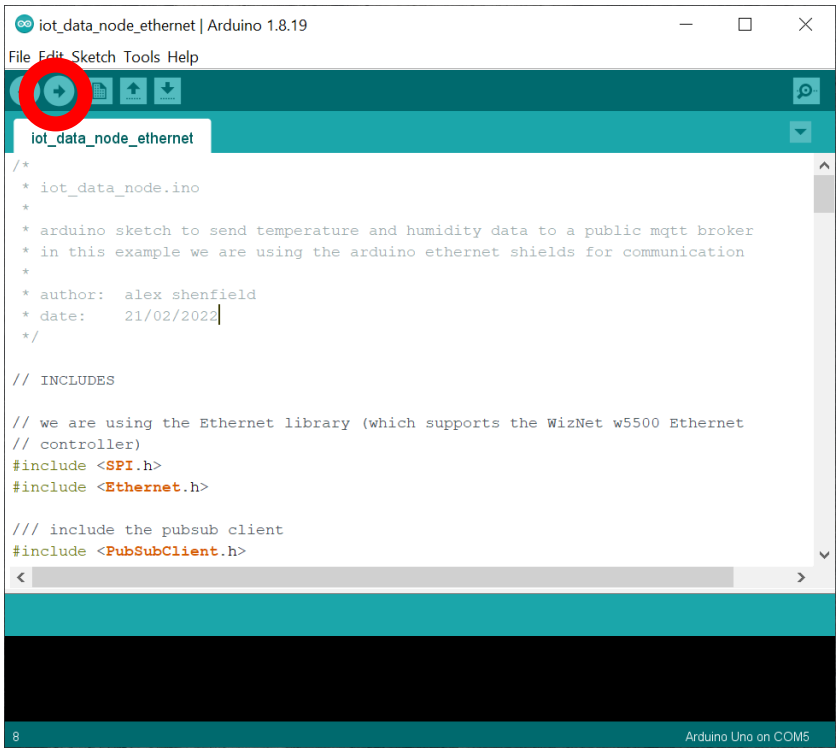


Figure 6 – Uploading the program to the board

To see what's going on, we can use the serial monitor in the Arduino IDE (we access this by clicking on the magnifying glass icon on the top right of the Arduino IDE – see Figure 7). This shows us progress updates as the application runs (see Figure 8).


Figure 7 – Opening the serial monitor

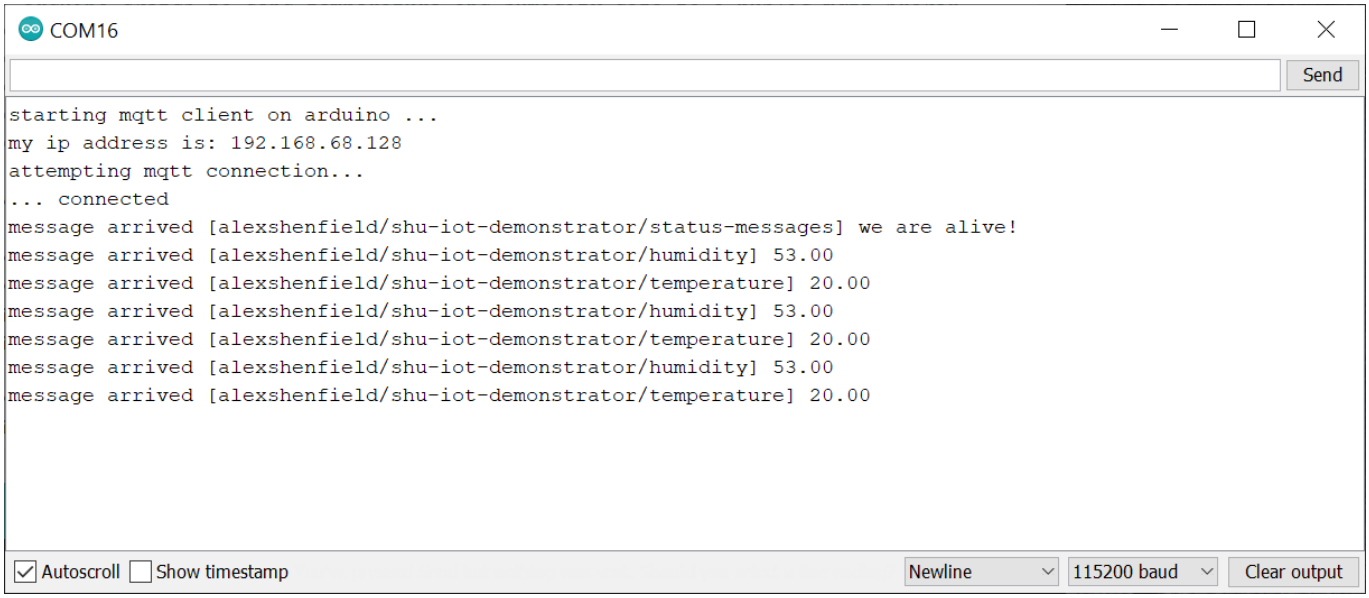


Figure 8 – Monitoring application progress through the serial monitor

**Extra credit**

So, how can you access this data you're sending off to a remote MQTT broker??

I like to use a program called MQTT Explorer for connection to the broker and examining the messages we are sending – particularly as it provides the neat feature of being able to graph numerical values (and who doesn't love graphs!!).

MQTT Explorer should be available and set up on the desktop. When you start it up, you should see a screen that looks something like that shown in Figure 9 – don't press connect just yet!

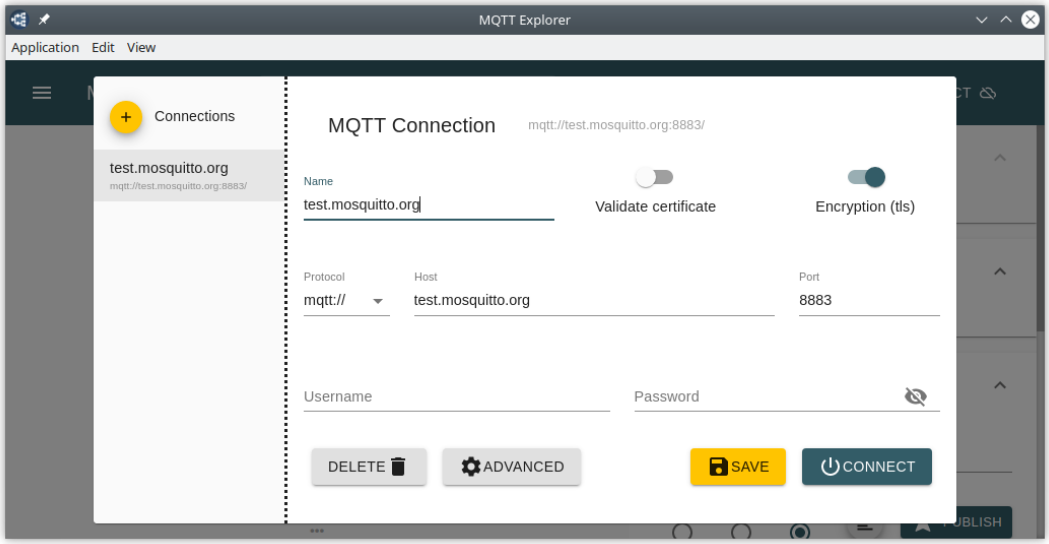


Figure 9 – MQTT Explorer connection window

Before we connect to the broker, we need to make sure we are subscribed to the topics we specified earlier (in code listings 2 and 3). Figure 10 shows what this looks like for me. The topic should be something like:

`<identifying name>/shu-iot-demonstrator/#`

(this "#" means "get notifications for everything above the specified root topic").

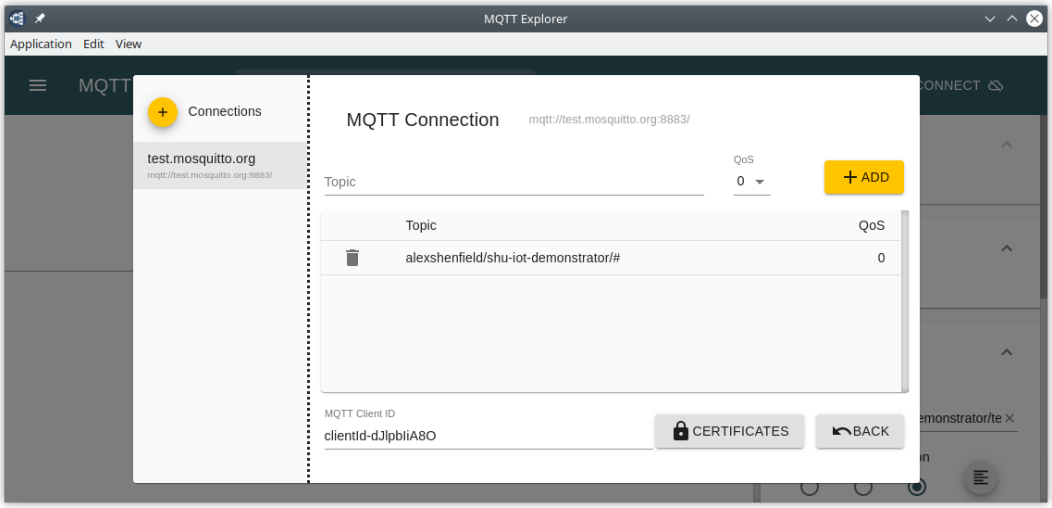


Figure 910– MQTT Explorer "Advanced" window for topic subscriptions

And Figure 11 then shows the humidity and temperature values for my system (and the corresponding graph). Try breathing on the sensor to make both temperature and humidity go up!

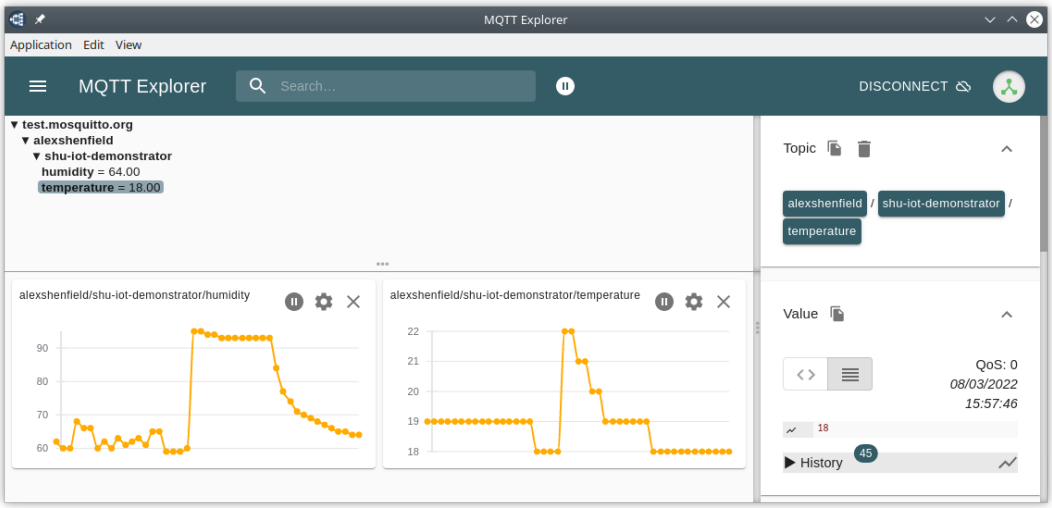


Figure 11 – MQTT Explorer window with data visualisations

So that's it! In less than 1 hour you have built an IoT enabled room monitoring system capable of reading temperature and humidity and transmitting them over the Internet. And all for less than £50 worth of parts (substantially cheaper if you use non branded hardware!).