25/03/20

Using an Arduino with the DS18B20 Waterproof Temperature Sensor

Aims:

- To connect an Arduino to The Things Network.
- To send sensor data from the DS18B20 Waterproof Temperature Sensor to the server gateway using LoRa technology.
- To use the sensor data to control a heater using a relay.

Research

Tutorial url: https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/

LoRa is a Long-Range radio technology developed by Semtech. It allows Internet of Things sensors to send data to remote locations. The radio network we will be using is the Things Network https://www.thethingsnetwork.org/docs/lorawan/. Use of this system is free.

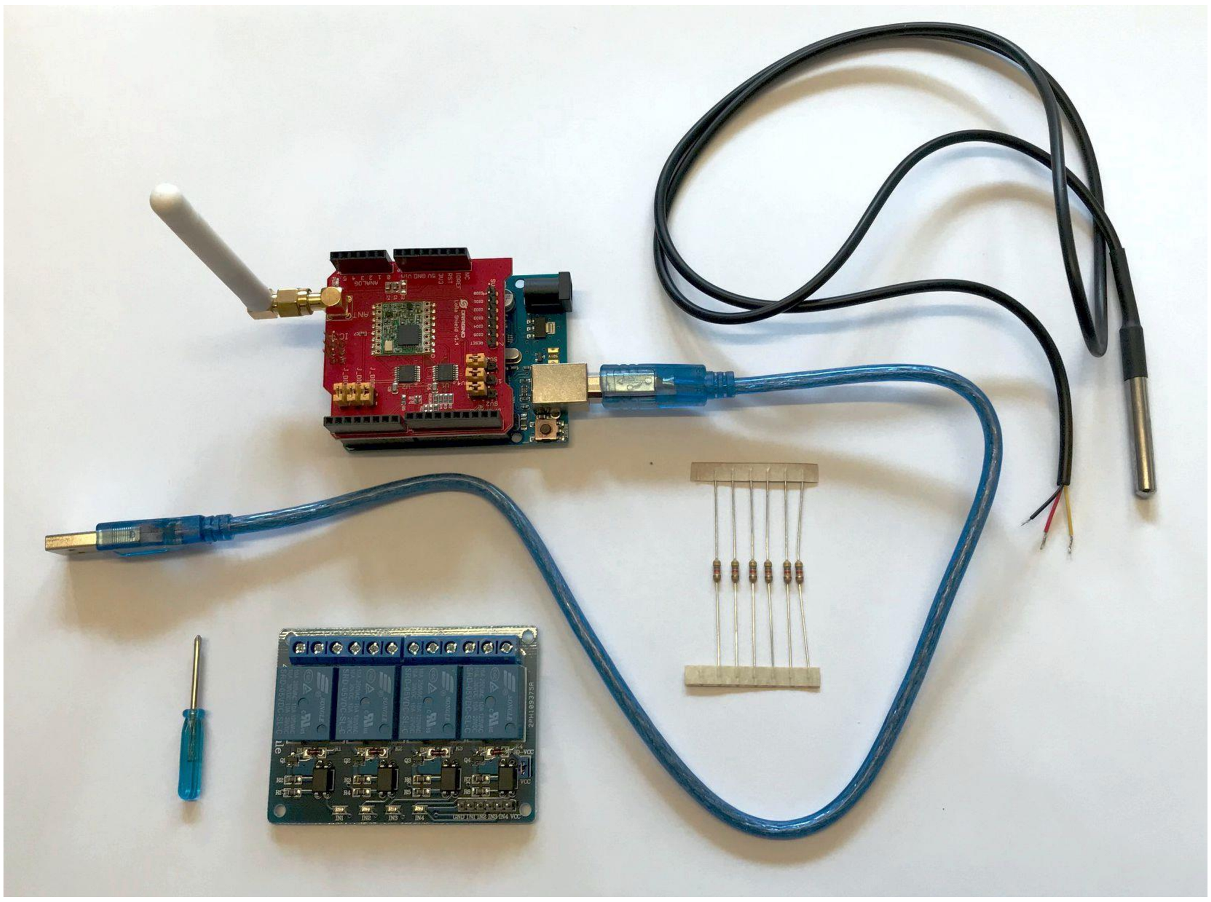
Equipment

My shopping list comprised the following items:

- Amazon: Arduino Uno £26.99 (Inc VAT)
  https://www.ukiot.store/product/lora-iot-development-kit/
- UKIoT Store: LoRa Shield for Arduino £20.83 (Ex VAT)
  https://www.ukiot.store/product/lora-shield-for-arduino/
- Amazon: DS18B20 waterproof temperature sensor £9.99 (Inc VAT)
  https://www.amazon.co.uk/Arduino-A000066-ARDUINO-UNO-REV3/dp/B008GRTSV6
  This product includes five temperature sensors, together with the 4.7K Ohm resistors and some jumper wires.
- Amazon: ELEGOO Relay module for Arduino £7.49 (Inc VAT)
  https://www.amazon.co.uk/gp/product/B06XK6HCQC/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
  This product comes with a small screwdriver to tighten the terminals

Already in stock:

- Solderless breadboard
- USB cable to connect the Arduino with your laptop
- Assorted electrical components
- Plastic sandwich box (to contain the electronics)
- Cable ties (to secure the electronics in the sandwich box)
- A heater



The illustration above shows the LoRa shield already mounted on the Arduino, and with the USB cable connected. The temperature sensor to the right comes with three wires, the yellow is for data, and the black and red are for neutral and positive electrical connections. Also shown are the 4.7K Ohm resistors and the relay block (with free screwdriver).

TIP:

Here is a useful website that lets you check the rating of your resistors https://www.allaboutcircuits.com/tools/resistor-color-code-calculator/

And this website explains the pins on the Arduino https://roboticsbackend.com/arduino-uno-pins-a-complete-practical-guide/

The DS18B20 is a ‘one-wire’ digital sensor, which means it only needs one date wire (and a connection to ground GND) to communicate with the Arduino.

Procedure

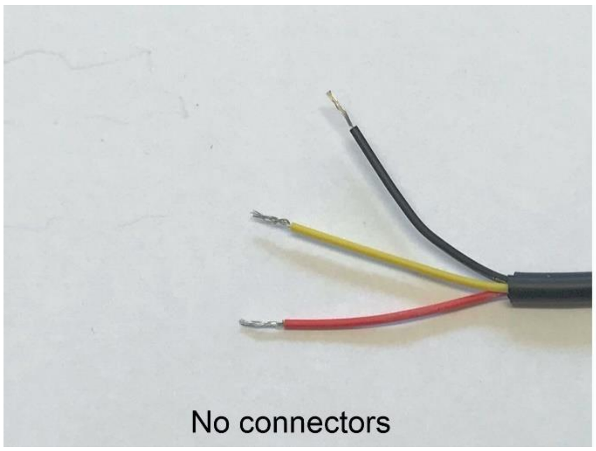
These instructions are based on the tutorial referred to above.

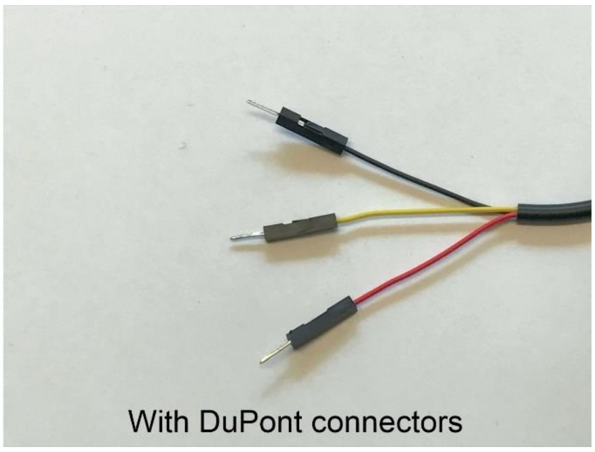
Wiring the DS18B20 sensor to the Arduino

The sensor can be wired up in one of two ways, parasitic mode or normal mode. In normal mode, the power for the sensor is supplied from a 5v (five volt) pin on the Arduino. In parasite mode, power is taken from the data line (the yellow wire in the illustrations). We shall use normal mode.

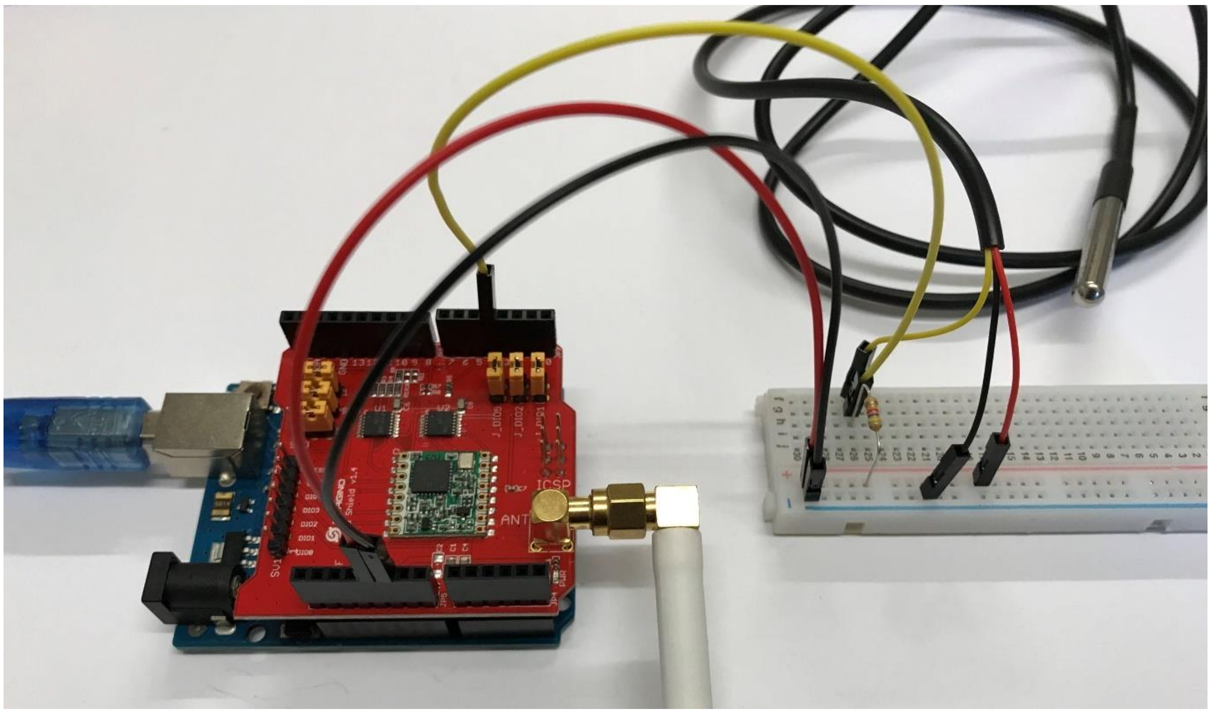
TIP:

Wiring up the sensor is easier if you put some DuPont connectors on to the ends of the wires.

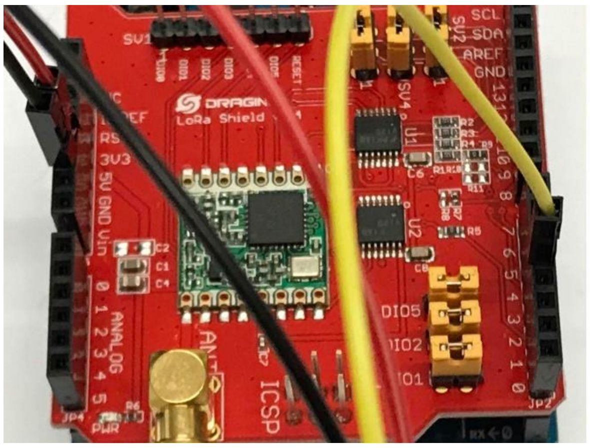


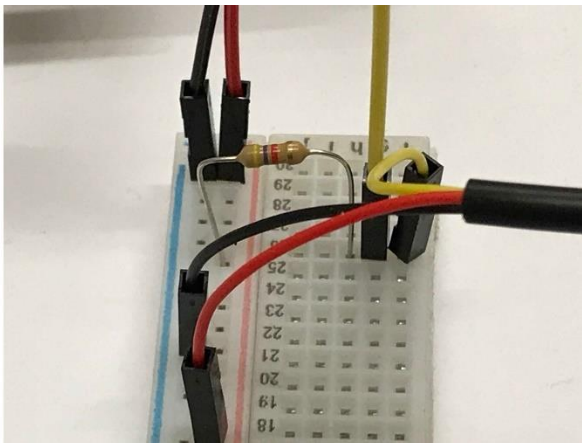


The wiring looks like this:



I have matched the sensor wire colours to connect the breadboard to the Arduino, so the red positive wire from the sensor connects to the positive rail on the breadboard, which takes power from the 5v pin on the Arduino. Similarly, the black ‘ground’ wire from the sensor connects to ground on the breadboard and then on to a GND pin on the Arduino. The yellow sensor wire connects to pin 4 on the Arduino by means of a yellow jumper wire from the breadboard. The sensor wire takes power from the positive rail on the breadboard through the 4.7K Ohm resistor.

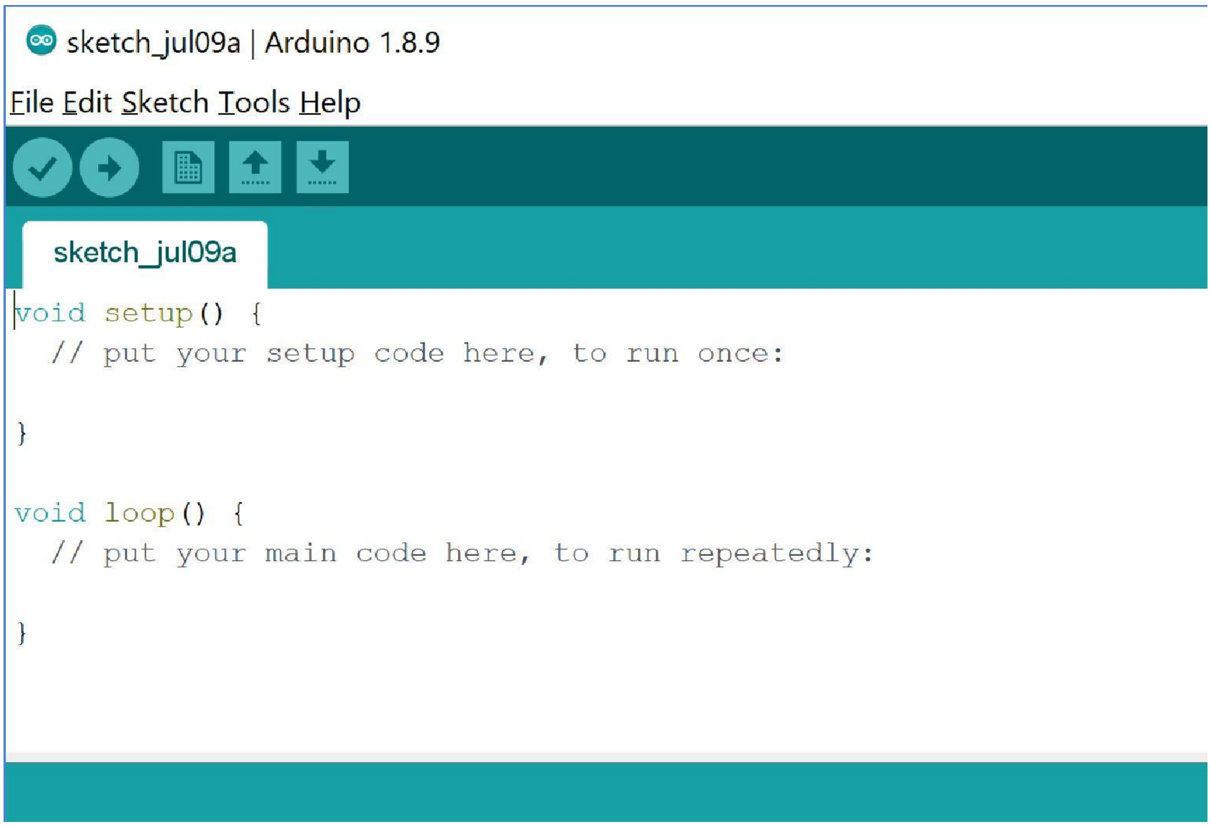




Programming the Arduino

In order to program the Arduino, you must first download and install the Arduino IDE (integrated development environment) onto your laptop. This is available here: https://www.arduino.cc/en/Main/Software. From the list on the left of the page, select the 'Windows Installer, for Windows XP and up' option. I am running Windows 10. The file I downloaded was called 'arduino-1.8.9-windows.exe'.

Once downloaded, double click the .exe file and follow the installation instructions. If all goes according to plan, you should see something like this when you open the IDE.

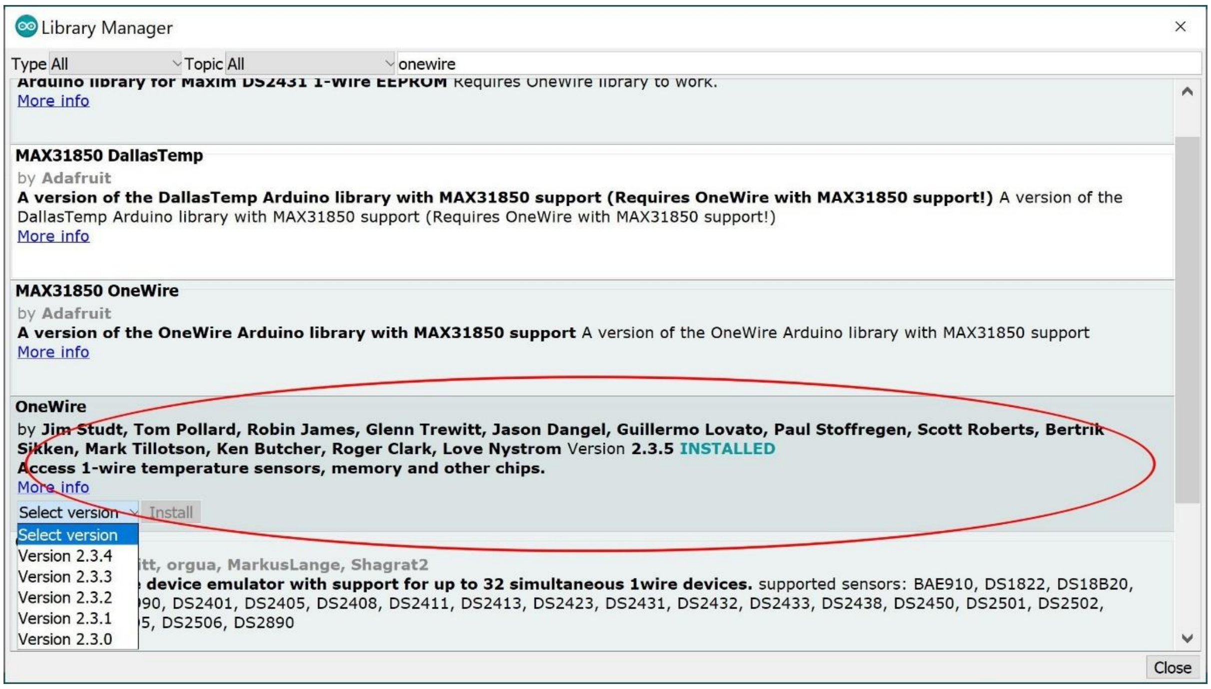


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Before we can write the program (sketch) we have to install some libraries to work with the DS18B20 sensor. We have to install two libraries.

Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries. The Library Manager should open.

Type "OneWire" in the search box and install the OneWire library by Paul Stoffregen.



Library Manager

Type All | Topic All | onewire

Arduino library for Maxim DS2431 1-Wire EEPROM Requires OneWire library to work.
More info

MAX31850 DallasTemp
by Adafruit
A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!) A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!)
More info

MAX31850 OneWire
by Adafruit
A version of the OneWire Arduino library with MAX31850 support A version of the OneWire Arduino library with MAX31850 support
More info

OneWire
by **Jim Studt, Tom Pollard, Robin James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Paul Stoffregen, Scott Roberts, Bertrik Sikken, Mark Tillotson, Ken Butcher, Roger Clark, Love Nystrom** Version **2.3.5** **INSTALLED**
Access 1-wire temperature sensors, memory and other chips.
More info

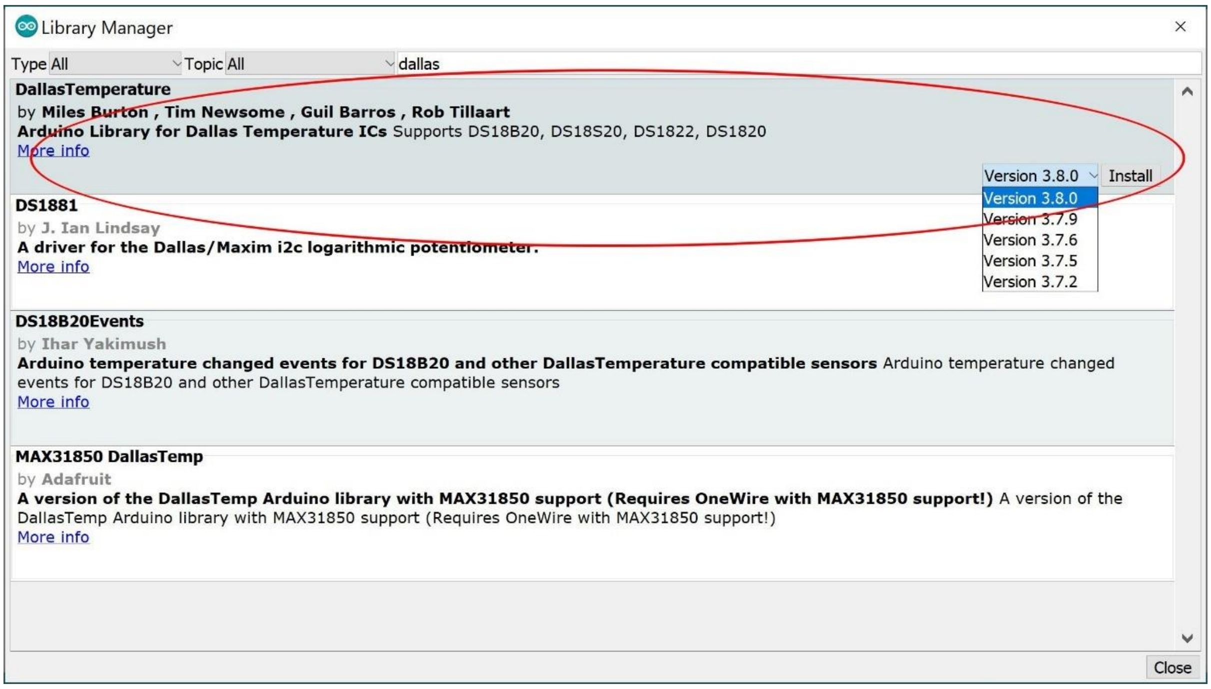
Select version | Install
Select version
Version 2.3.4
Version 2.3.3
Version 2.3.2
Version 2.3.1
Version 2.3.0

itt, orgua, MarkusLange, Shagrat2
device emulator with support for up to 32 simultaneous 1wire devices. supported sensors: BAE910, DS1822, DS18B20, 90, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433, DS2438, DS2450, DS2501, DS2502, 5, DS2506, DS2890

Close

Select the latest version and click install.

Next search for "Dallas" and install the Dallas Temperature library by Miles Burton.



Library Manager

Type All | Topic All | dallas

DallasTemperature
by **Miles Burton , Tim Newsome , Guil Barros , Rob Tillaart**
Arduino Library for Dallas Temperature ICs Supports DS18B20, DS18S20, DS1822, DS1820
More info

Version 3.8.0 | Install
Version 3.8.0
Version 3.7.9
Version 3.7.6
Version 3.7.5
Version 3.7.2

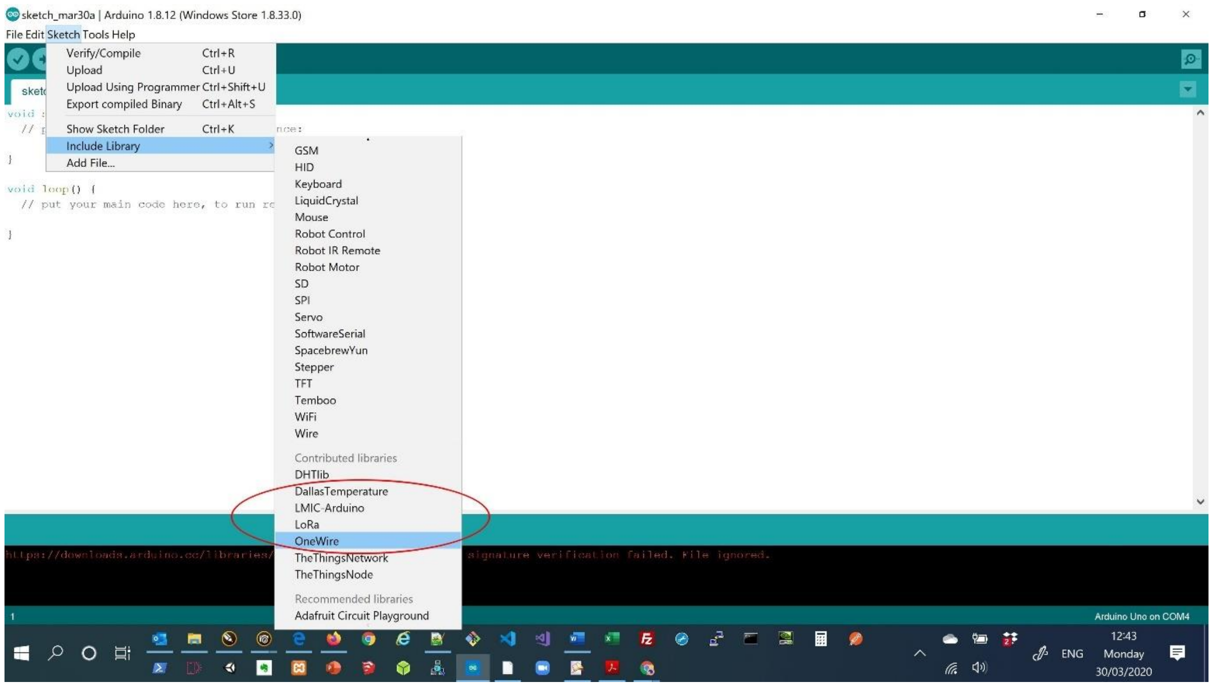
DS1881
by J. Ian Lindsay
A driver for the Dallas/Maxim i2c logarithmic potentiometer.
More info

DS18B20Events
by Ihar Yakimush
Arduino temperature changed events for DS18B20 and other DallasTemperature compatible sensors Arduino temperature changed events for DS18B20 and other DallasTemperature compatible sensors
More info

MAX31850 DallasTemp
by Adafruit
A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!) A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!)
More info

Close

Once you have installed the libraries, check that they are there by opening the Library Manager.



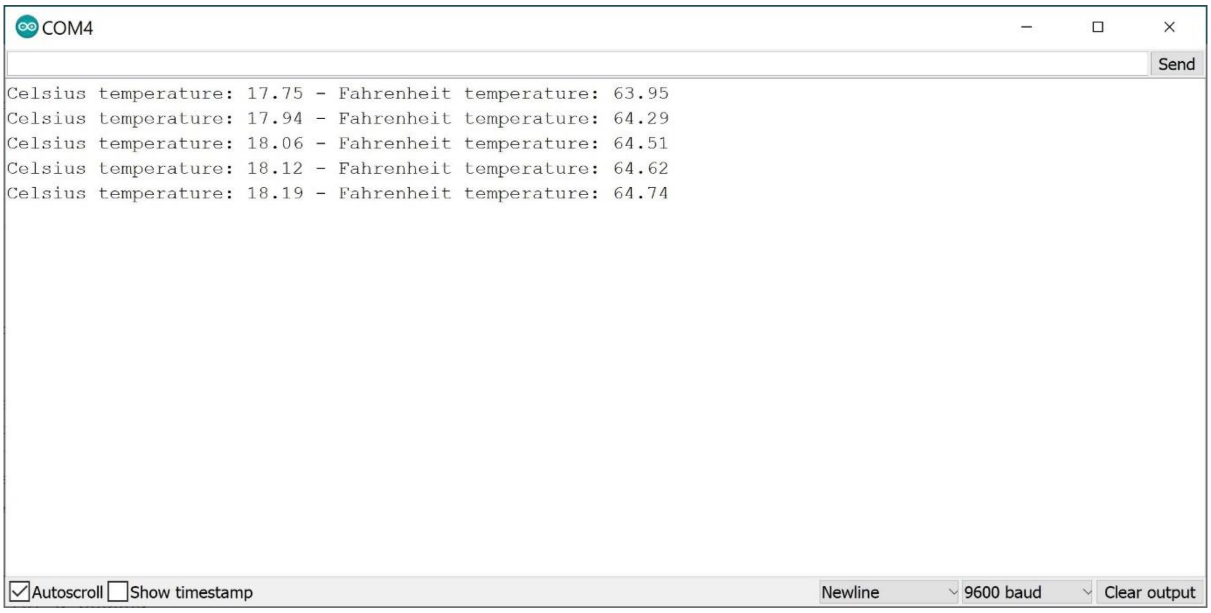
After installing the libraries, upload the following code to your Arduino board. This sketch is based on an example from the Dallas Temperature library.

```
/*********
  Rui Santos
  Complete project details at http://randomnerdtutorials.com
  Based on the Dallas Temperature Library example
*********/

#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is conntec to the Arduino digital pin 4
#define ONE_WIRE_BUS 4

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

void setup(void)
{
  // Start serial communication for debugging purposes
  Serial.begin(9600);
  // Start up the library
  sensors.begin();
}

void loop(void){
  // Call sensors.requestTemperatures() to issue a global temperature and
Requests to all devices on the bus
  sensors.requestTemperatures();

  Serial.print("Celsius temperature: ");
  // Why "byIndex"? You can have more than one IC on the same bus. 0 refers
to the first IC on the wire
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print(" - Fahrenheit temperature: ");
  Serial.println(sensors.getTempFByIndex(0));
  delay(1000);
}
```

You can find the original code here: https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/DS18B20_Temperature_Sensor_Arduino.ino

When you are ready to upload the code, plug your Arduino into your laptop, open your IDE and click on the 'upload' button at the top left of the IDE. You will see a confirmation message at the foot of the IDE was the code is compiled and uploaded.

To view the output of the temperature sensor, click on the Serial Monitor button at the top right of the IDE. You should see something like this:



```
Celsius temperature: 17.75 - Fahrenheit temperature: 63.95
Celsius temperature: 17.94 - Fahrenheit temperature: 64.29
Celsius temperature: 18.06 - Fahrenheit temperature: 64.51
Celsius temperature: 18.12 - Fahrenheit temperature: 64.62
Celsius temperature: 18.19 - Fahrenheit temperature: 64.74
```

You now know the system is working.

Register your device with the things network

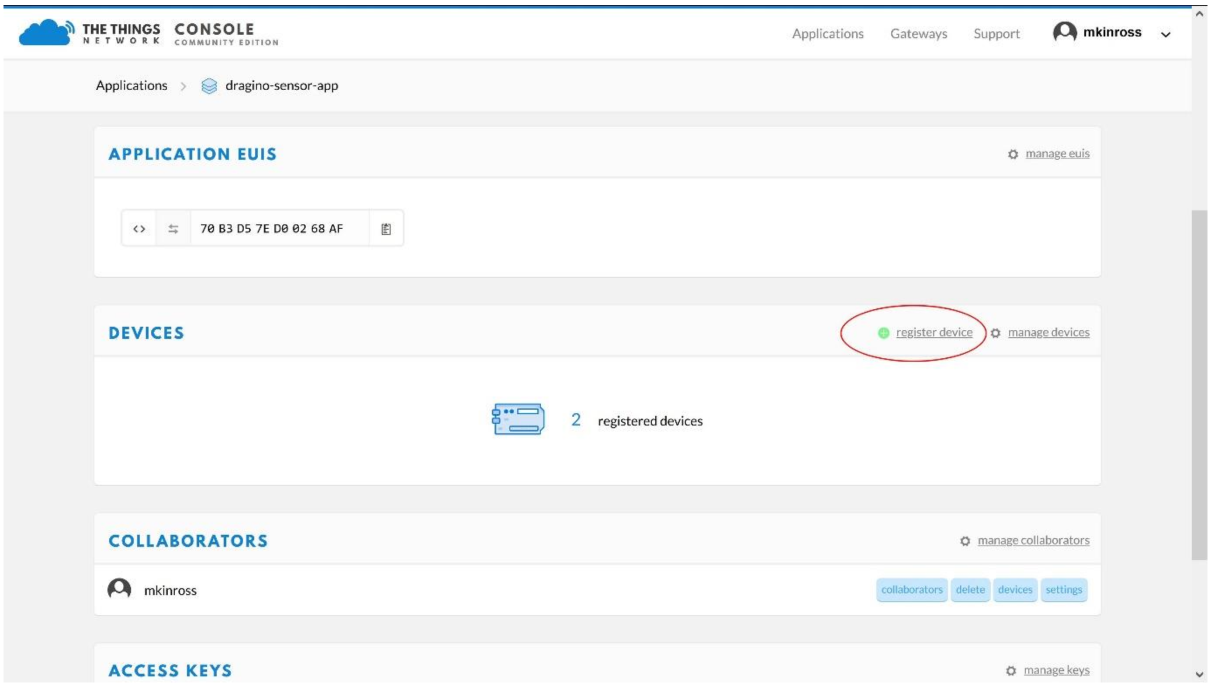
We now want to connect our device to the LoRa Network so that we can send temperature data. We will then use this data to control a relay, which will turn our heater on or off.

We will connect our device with The Things Network. See the article 'Using an Arduino with the LoRa IoT Development Kit.docx' to find out more about how the system works.

We need to add our device to TTN. Each application you create can have multiple devices associated with it.

We have already created an application in the article mentioned above, it is called 'dragino-sensor-app (My sensor application)'. We will add our device to this application.

Go to The Things Network console https://console.thethingsnetwork.org. Click on 'Applications' then click the above-mentioned app. In the DEVICES section, click 'register device'.


THE THINGS NETWORK CONSOLE COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > dragino-sensor-app

APPLICATION EUIS

manage euis

70 B3 D5 7E D0 02 68 AF

DEVICES

register device

manage devices

2 registered devices

COLLABORATORS

manage collaborators

mkinross

collaborators delete devices settings

ACCESS KEYS

manage keys

When the page opens, you need to complete the following fields:

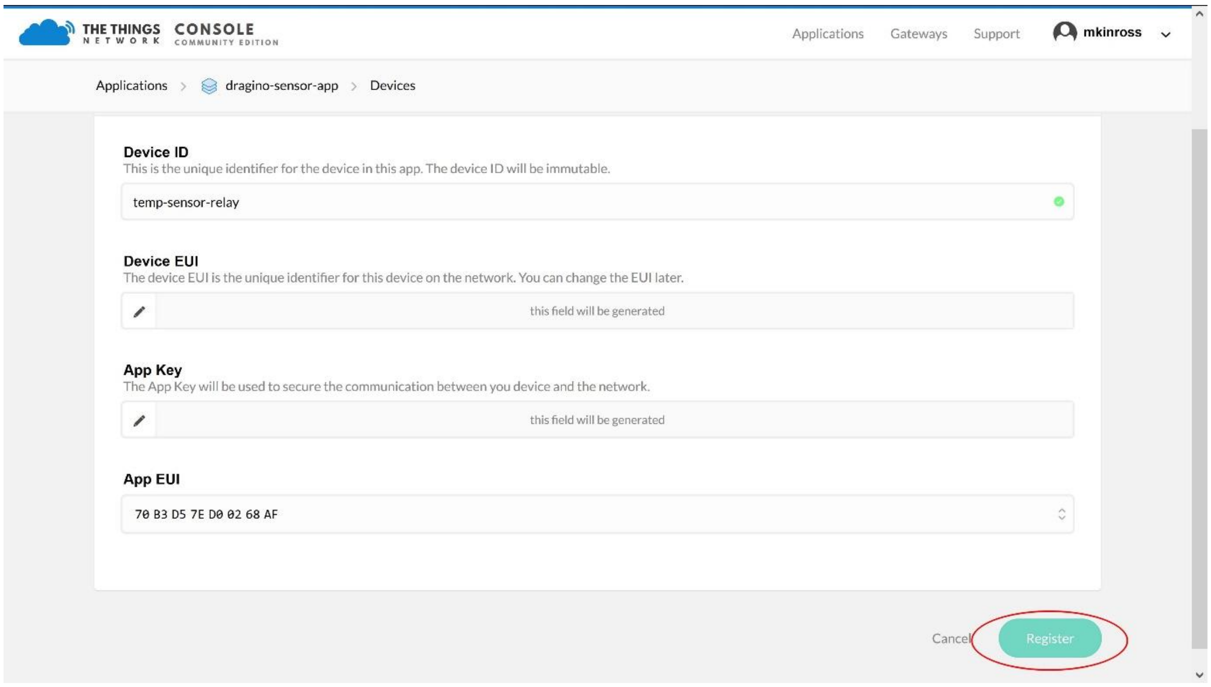
Device ID – This must be unique within the application

Device EUI – Click the 'squiggle' to the left of the box and it will turn into a pencil, which means it will be generated automatically

App Key – This will be generated automatically too

App EUI – This is the App EUI from the application overview and should be populated automatically

Click the green 'Register' button when you are ready


THE THINGS NETWORK CONSOLE COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > dragino-sensor-app > Devices

Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.
temp-sensor-relay

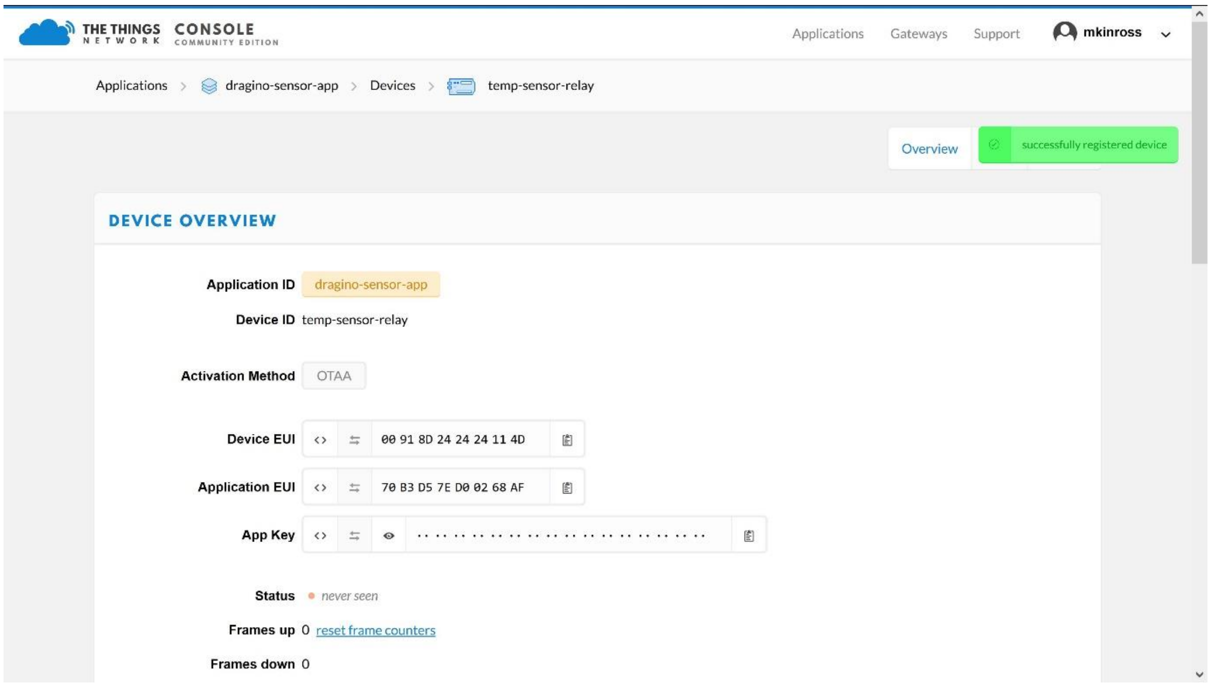
Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
this field will be generated

App Key
The App Key will be used to secure the communication between you device and the network.
this field will be generated

App EUI
70 B3 D5 7E D0 02 68 AF

Cancel Register

You should then see this page:


THE THINGS NETWORK
CONSOLE
COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > dragino-sensor-app > Devices > temp-sensor-relay
Overview
successfully registered device

DEVICE OVERVIEW

Application ID dragino-sensor-app

Device ID temp-sensor-relay

Activation Method OTAA

Device EUI 00 91 8D 24 24 24 11 4D

Application EUI 70 B3 D5 7E D0 02 68 AF

App Key ··

Status never seen

Frames up 0 reset frame counters

Frames down 0

You will notice that the activation method OTAA (over the air activation) is already selected.

This device will be using the Cayenne payload, so TTN can parse the sensor data properly.

All the devices within this application are already set up to use the Cayenne LPP payload format.

We now need to update the sketch to work with LoRa.

Download and open the sketch from Github https://github.com/dragino/Arduino-Profile-Examples/blob/master/libraries/Dragino/examples/IoTServer/Cayenne%20and%20TTN/cayenne%20and%20ttn%20example/lora_shield%20DHT11%20and%20Relay%20examples/lora_shield_cayenne_and_ttn-otaaClient/lora_shield_cayenne_and_ttn-otaaClient.ino.

When you open the sketch in your Arduino IDE, you will see something like this:

This article continues the same was as the article ‘Using an Arduino with the LoRa IoT Development Kit.docx’ from page 36.

The above sketch needs to be modified to work with the DS18B20 temperature sensor.

This article to be continued…