

Brewing: A Thermostatically Controlled Mash Tun using an Arduino and a DS18B20 Temperature Sensor

One of the trickiest process when brewing all grain beer, is controlling the temperature of the mash for the hour or so it takes to complete. This project hopes to solve this problem by putting the mash tun on a temperature controlled hot plate. We will be using a DS18B20 waterproof temperature sensor linked to an Arduino micro controller which will maintain the temperature of a hotplate using a relay.

Aims:

- To construct a mash tun, which can be used on a hotplate and can be fitted with a DS18B20.
- To measure the temperature of the mash using the DS18B20.
- To send sensor data from the DS18B20 to an Arduino which will then switch a relay either on or off as required.
- To control the heat of the hotplate through the relay.

Research

This article has been largely inspired by the video from Short Circuited Brewers "How To Brew Small Batch All Grain BIAB Beer | Brewing on Small batch Brewing Equipment"

<https://www.youtube.com/watch?v=AQr54jf8GuM>.

See also: Mashing, Lautering and Sparging <https://en.wikipedia.org/wiki/Lautering>

in order to wire up the temperature sensor, we are following this tutorial

<https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/>

Equipment

My shopping list comprised the following items:

- 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
- Amazon: Excelsteel 12 Quart 18/10 Stainless Steel 4 Piece Muti-Cookware Set with Encapsulated Base £55.78

https://www.amazon.co.uk/gp/product/B002CGSYB2/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1

- Amazon: 1pc Mini 12V DC 6W Food Grade Submersible Brushless Water Pump 2L/min
£7.39
https://www.amazon.co.uk/gp/product/B07C1Y63C6/ref=ppx_yo_dt_b_bia_item_fallback_o01_s00
- Amazon: 30W Universal AC to DC Power Adapter 100-240V to 3V-12V Multi Voltage Micro USB Plug Switching Power Supply Charger Plug £10.68
https://www.amazon.co.uk/Universal-100-240V-Switching-Household-Electronics/dp/B07BQR2QDF/ref=sr_1_6?crd=1CMRADPA0PB1Y&dchild=1&keywords=universal+ac+adapter&qid=1588064652&srefix=universal+ac+a%2Caps%2C162&sr=8-6
- Amazon: 5 pcs DS18B20 Waterproof Temperature Sensors Temperature Transducer £10.59
https://www.amazon.co.uk/DS18B20-Waterproof-Temperature-Sensors-Transducer/dp/B00CHEZ250/ref=sr_1_29?dchild=1&keywords=Bonega%C2%AE+5PCS+Temperature+Sensor+DS18B20&qid=1588064952&sr=8-29
- Amazon: Multicomp Pi-Blox Case for Raspberry Pi 3 & Raspberry Pi 2 Model B & Pi Camera £7.99
https://www.amazon.co.uk/gp/product/B017Z32E6M/ref=ppx_yo_dt_b_bia_item_fallback_o03_s00 (we are just using this as a junction box)
- Amazon: ELEGOO Relay Module 4 Channel DC 5V with Optocoupler for Arduino UNO R3 MEGA 2560 Project 1280 DSP ARM PIC AVR STM32 Raspberry Pi £6.99
https://www.amazon.co.uk/gp/product/B06XK6HCQC/ref=ppx_yo_dt_b_bia_item_fallback_o08_s00
- Amazon: Yuhtech 360 Pcs Male Female Hex Brass Spacer Standoff Screw Nut Assortment Kit M2 M3 M4 £20.79
https://www.amazon.co.uk/gp/product/B0811R9TWX/ref=ppx_yo_dt_b_bia_item_fallback_o04_s00
- Amazon: PVC Tube 6mm Internal Diameter 1/4" (5M) £6.03
https://www.amazon.co.uk/gp/product/B00H84LC4E/ref=ppx_yo_dt_b_bia_item_fallback_o05_s00
- Amazon: ARDUINO UNO REV3 £18.70
https://www.amazon.co.uk/Arduino-A000066-ARDUINO-UNO-REV3/dp/B008GRTSV6/ref=sr_1_7?dchild=1&keywords=Arduino+Uno&qid=1588069270&sr=8-7
- Amazon: Extra Large Reusable Drawstring Straining Brew in a Bag £12.99
https://www.amazon.co.uk/dp/B01D53H3W6/ref=pe_3187911_189395841_TE_dp_1

NOTE: Some of the illustrations below show the Arduino fitted with a red Lora Shield (<http://www.dragino.com/products/lora/item/102-lora-shield.html>). This shield is not used in this article and can be ignored. The pins referred to in the text connect to the same pins on the shield as on the Arduino.

Already in stock:

- Solderless breadboard
- USB cable to connect the Arduino with your laptop

- Assorted electrical components
- Plastic sandwich box (to contain the electronics)
- A hotplate
- A power drill

Procedure 1: Assembling the Mash Tun

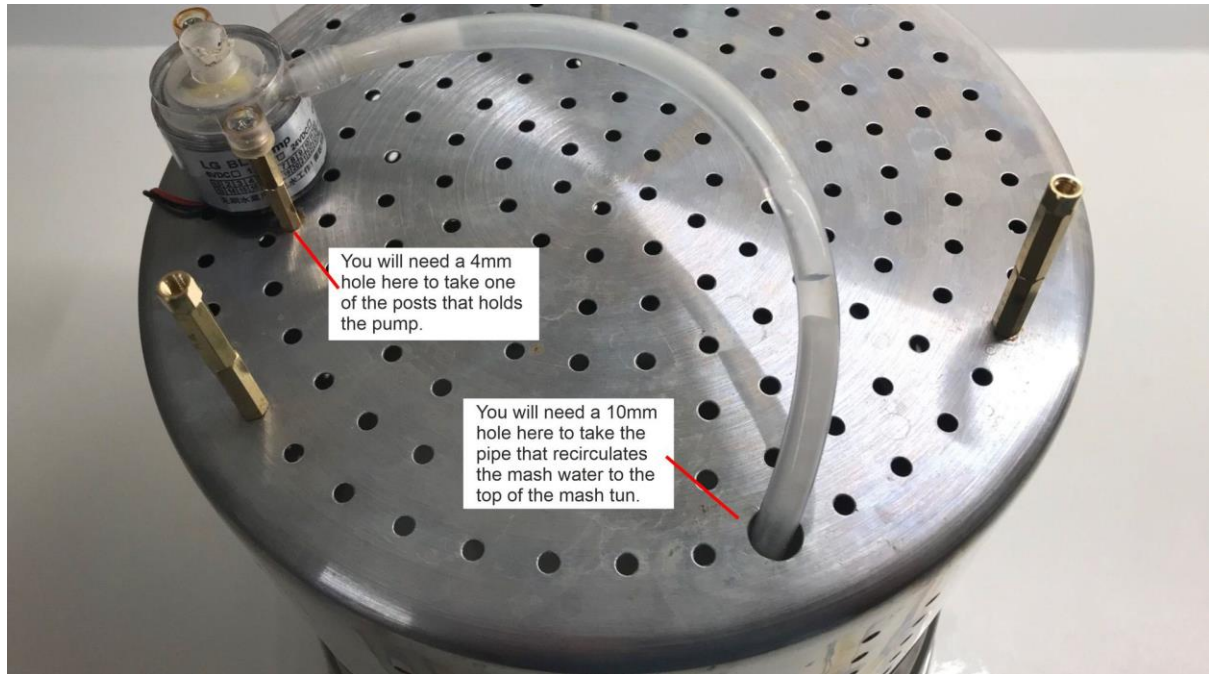
For this you will need the stainless-steel boiler, the submersible pump, some brass standoffs and a length of the PVC tube.

The knob has been removed from the lid so that we can use the hole to insert the temperature probe into the mash tun



Firstly, remove the knob from the top of the glass lid. It is only held on by a screw. This will reveal a hole, through which we can pass the temperature sensor.

In order to fit the pump, we will have to drill two holes in the base of the inner container.



The pump is fixed to the base of the inner container using brass standoff spacers and screws.

The inner container of the mash tun stands on three legs. For each leg you will need:

- M4*8 screw
- M4*20 20mm female-male spacer
- M4*20+6 10mm female-male spacer
- M4*5+6 15mm female-female spacer
- M4*8 screw

For each of the two pump supports you will need:

- M3* 6 screw
- M3*15 15mm female-female spacer
- M3*15 15mm male-female spacer
- M3*6 screw

The standoff spacers for the pump are a size down on the legs, so that the spacers fit into the screw holes on the pump.

You can adjust the speed of the pump by twisting a screw on the back of the universal power adaptor. Ours was set to 4.5v. If the pump runs too fast, the water will not drain through the mash quickly enough to stop it overflowing.

Procedure 2: Connecting the Arduino Controller

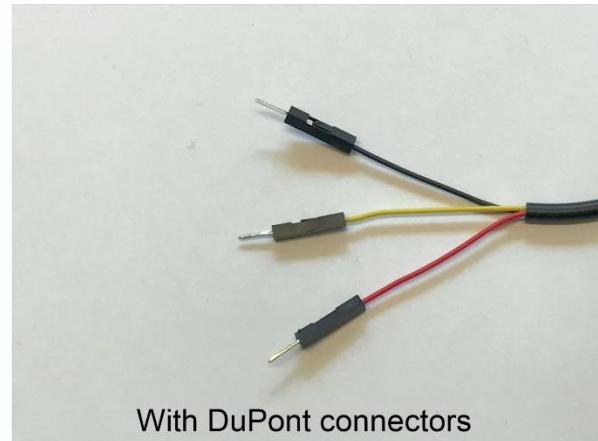
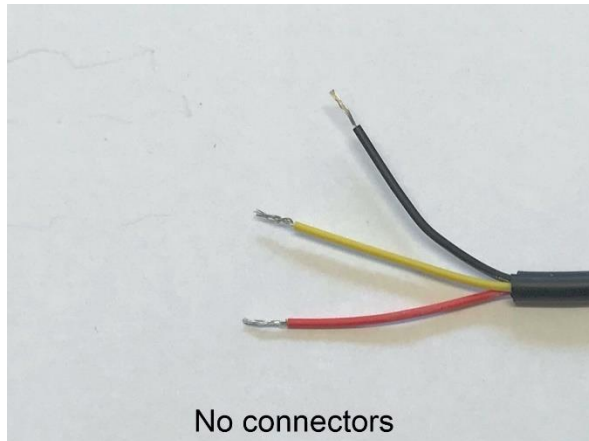
These instructions are based on the tutorial referred to above.

Wiring the DS18B20 sensor to the Arduino

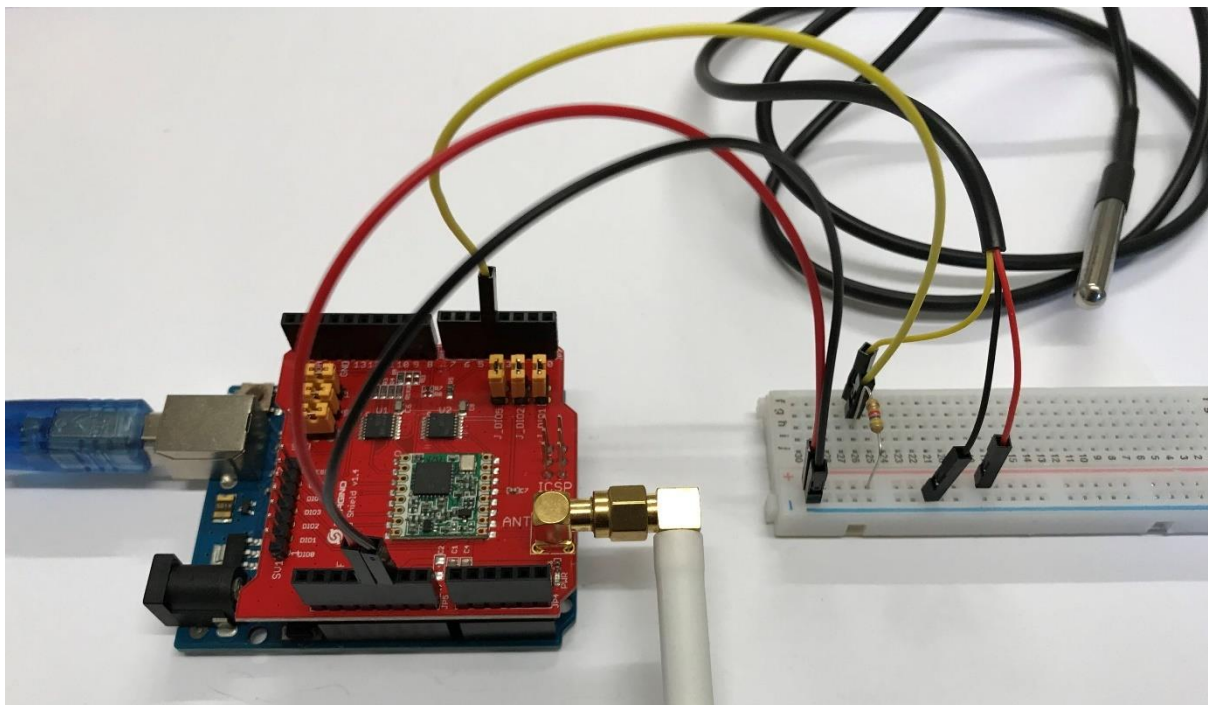
The sensor can be wired up in one of two ways, parasitic mode or normal mode. In normal mode, the power for the sensor is supplied from a 5v (five volt) pin on the Arduino. In parasite mode, power is taken from the data line (the yellow wire in the illustrations). We shall use normal mode.

TIP:

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



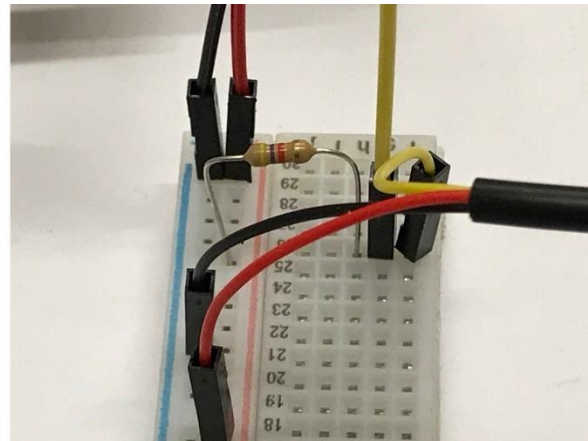
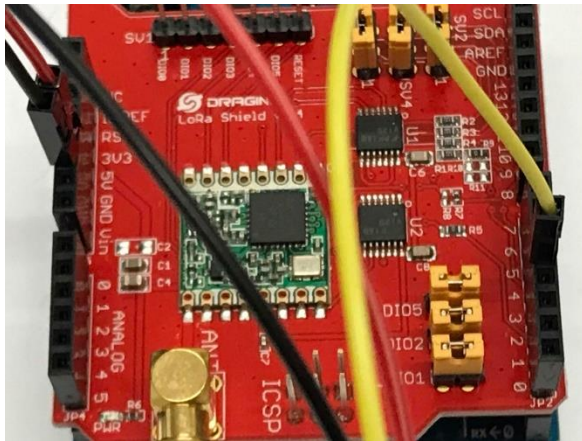
The wiring looks like this:



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.

You can see a diagram of the Arduino pins here <https://roboticsbackend.com/arduino-uno-pins-a-complete-practical-guide/>.



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'.

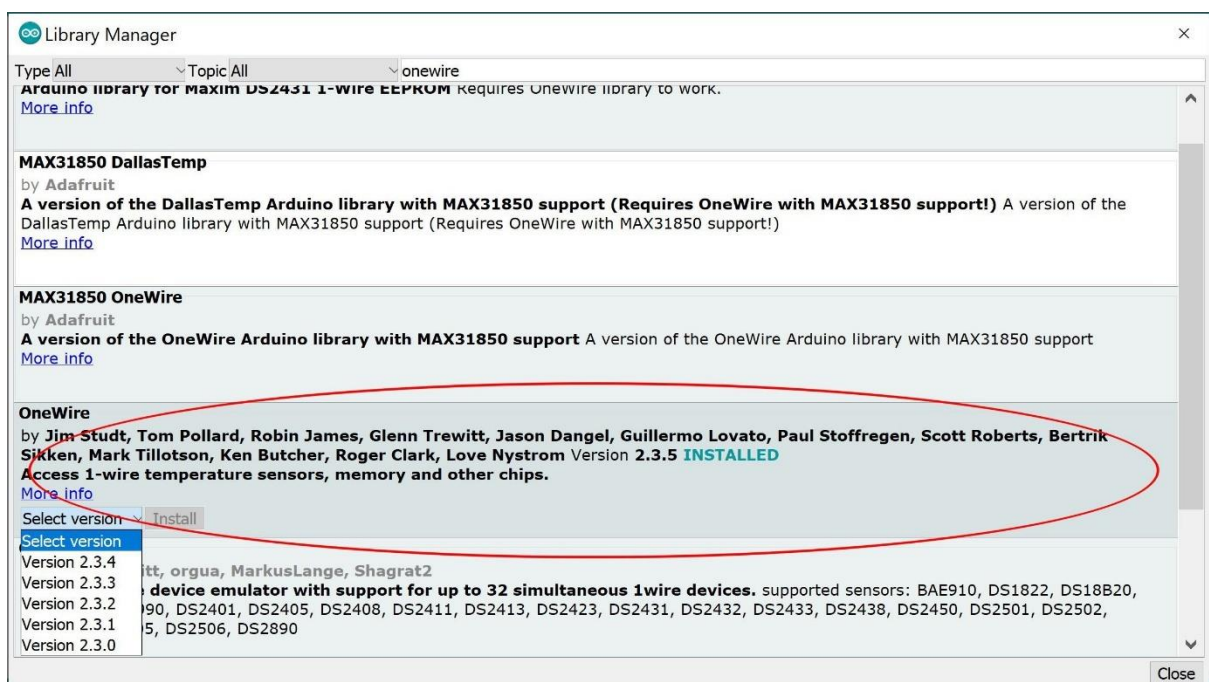
Once downloaded, double click the .exe file and follow the installation instructions. If all goes according to plan, you should see something like this when you open the IDE.



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.

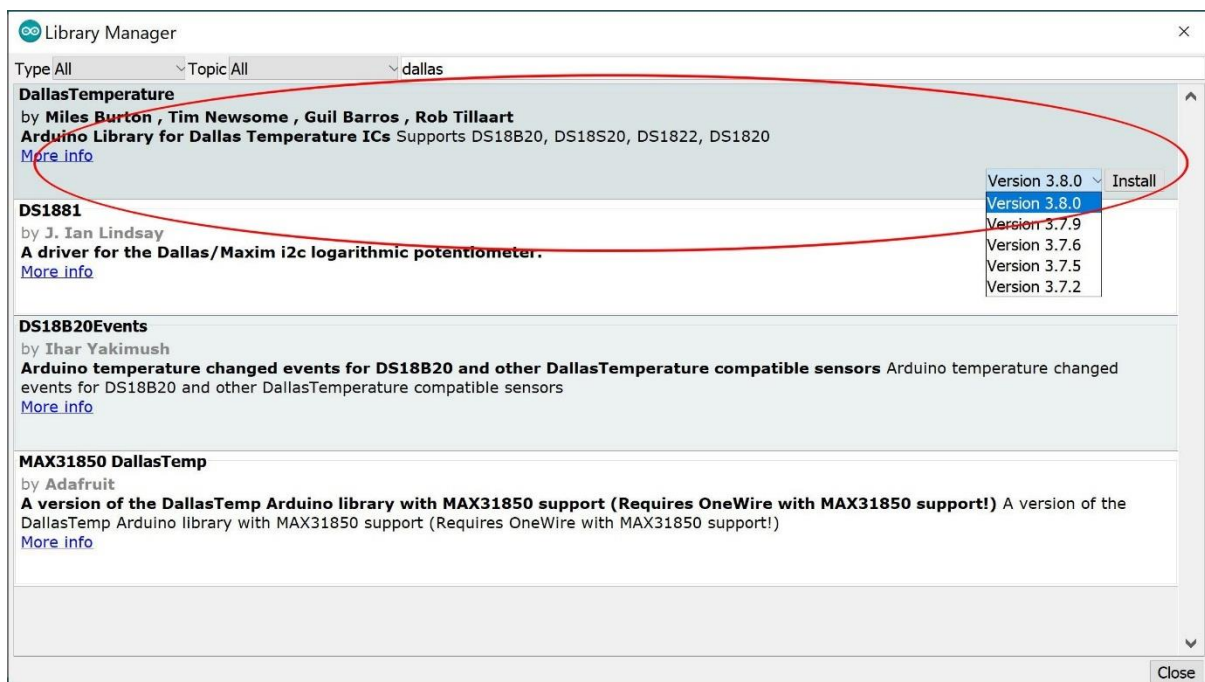
Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries. The Library Manager should open.

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

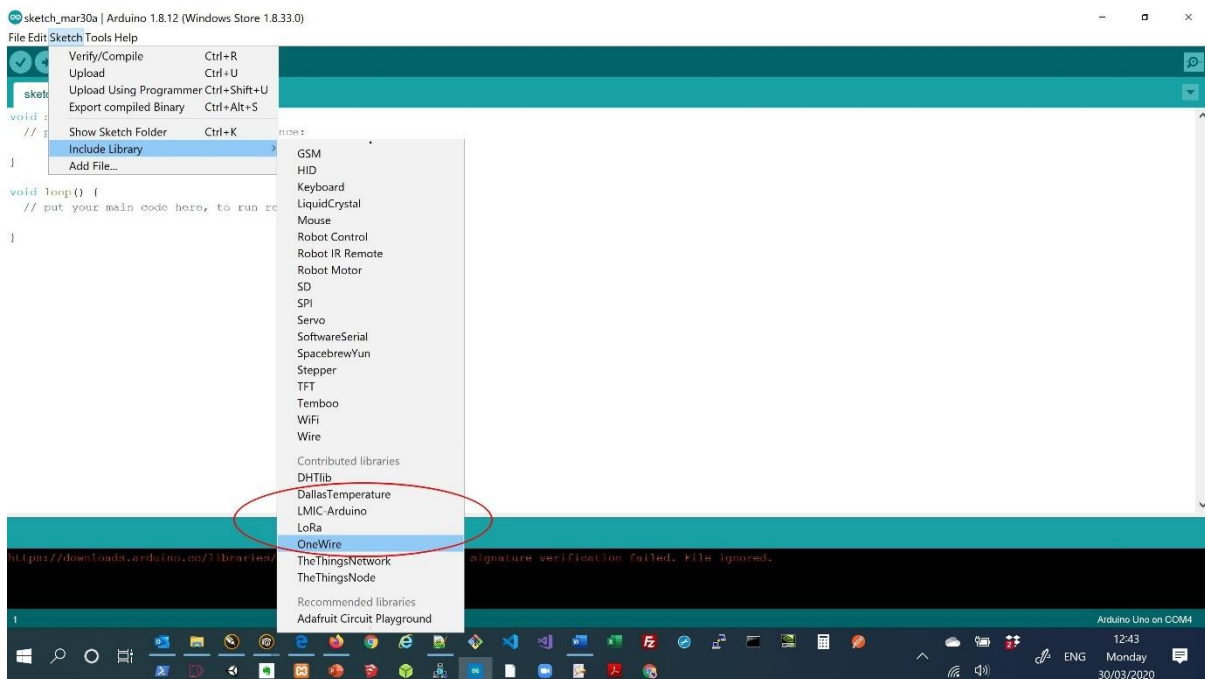


Select the latest version and click install.

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



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();

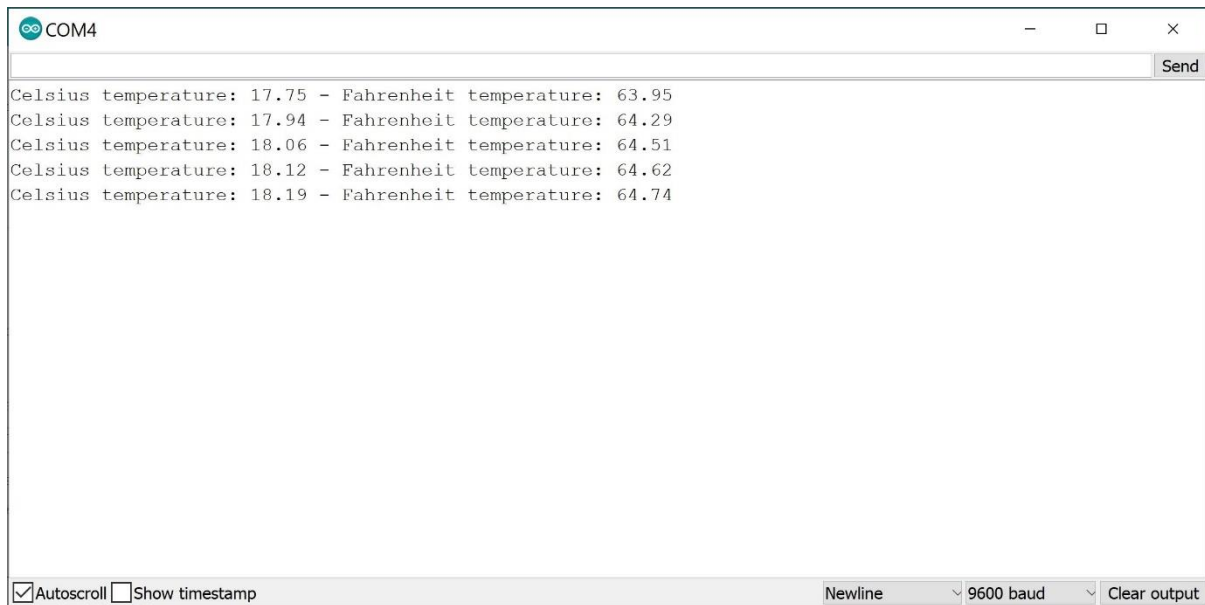
  Serial.print("Celsius temperature: ");
  // Why "byIndex"? You can have more than one IC on the same bus. 0 refers
  to the first IC on the wire
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print(" - Fahrenheit temperature: ");
  Serial.println(sensors.getTempFByIndex(0));
  delay(1000);
}

```

You can find the original code here: [https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/DS18B20 Temperature Sensor Arduino.ino](https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/DS18B20%20Temperature%20Sensor%20Arduino.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:



```
COM4
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

☒ Autoscroll ☐ Show timestamp
Newline 9600 baud Clear output
```

You now know the system is working.

Incorporating the relay

We are using the ELEGOO Relay module for Arduino, which can handle up to 10amps. This will be switched on when digital pin 7 on the Arduino is set to 'high' by the sketch, and switched off when set to 'low'.

Here is the sketch again, but modified to control the relay.

```
/*
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 connctec 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);

float tempVariable = 0; // initialises a variable to take
temperature data
float previousTempVariable = 0; // initialises a variable to
remember the previous temperature

// All temperatures are in °C
// Target temperature is the water temperature we are aiming for.
int targetTemperature = 66;
```

```

// We have to stop the heater before the target temperature is
// acheived or else it will overrun due to the heat within the hot
// plate
int cutOffTemperature = 56;

// If the system starts to cool, switch on the hot plate for a few
// seconds (value is in milliseconds)
int nudgeTime = 5000;

// We only need nudgeTime if the system is cooling, but we will
// start by assuming the system is cold
String systemState = "heating";

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

    pinMode(7, OUTPUT);    // sets the digital pin 7 as output
}

void loop(void){
    previousTempVariable = tempVariable;

    // 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));
    tempVariable = (sensors.getTempCByIndex(0));

    Serial.print("The previous temperature was: ");
    Serial.print(previousTempVariable);
    Serial.print("\n");

    Serial.print("The current temperature is: ");
    Serial.print(tempVariable);
    Serial.print("\n");

    if (tempVariable < previousTempVariable)
    {
        systemState = "cooling";

    } else if (tempVariable > previousTempVariable) {
        systemState = "warming";

    } else {
        systemState = "stable";
    }

    if (tempVariable < cutOffTemperature) {

```

```

    // The system is still warming up
    digitalWrite(7, LOW); // sets the digital pin 7 on
    Serial.print("Current temperature is below cut-off temperature
so heater is switched on");
    Serial.print("\n");
    Serial.print("The system is: ");
    Serial.print(systemState);
    Serial.print("\n");
    } else if ((tempVariable > cutOffTemperature) && (tempVariable <
targetTemperature)) {
    // Only turn on the hot plate if it is cooling, or we might
    overrun the target temperature
    if (systemState = "cooling") {
        // turn on the hotplate for a bit
        digitalWrite(7, LOW); // sets the digital pin 7 on
        Serial.print("The heater is briefly switched on as we are just
below the target temperature");
        delay(nudgeTime);
        Serial.print("\n");
        digitalWrite(7, HIGH); // sets the digital pin 7 on
        Serial.print("The heater is now switched off in case we over-
shoot the target temperature");
        Serial.print("\n");
    }
    } else {
        digitalWrite(7, HIGH); // sets the digital pin 7 on
        Serial.print("Target temperature has been acheive so the heater
has been switched off");
        Serial.print("\n");
    }
}

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

delay(15000);
}

```

There are four parameters that can be changed in this sketch.

1. The 'delay' in the last line controls the measurement interval in milliseconds. So, in the sketch above the temperature will be measured every 15 seconds.
2. We need to set the final temperature that we wish the thermostat to hold. In the sketch above this is called 'targetTemperature' and is set to 66°C. Change this number to whatever temperature you wish to hold your mash at.
3. There is also a parameter called 'cutOffTemperature', which above is set to 55°C. This is the temperature at which we start to slow down the temperature rise. If we just continued heating until we reach the target temperature, the residual heat in the hotplate will mean that we will carry on heating above the target temperature, even if the hotplate has been turned off.

4. We slow down the temperature rise by setting the 'nudgeTime', which above is set to 5000 milliseconds. This means that within each fifteen second measurement interval, the hotplate is only turned on for five seconds, and then turned off for the remaining ten seconds.

Putting it All Together

Assembly