Promethean Temperature Sensor

Joe Collins

November 22, 2020

Contents

1 Objective

A temperature sensor that can be scraped/polled using Prometheus.

Prometheus is excellent for monitoring server metrics, so it makes sense to use it to monitor other metrics such as temperature and humidity. There doesn't appear to be anything available off the shelf. There are temperature monitors but some effort would be required to get them to integrate with Prometheus. So we might as well build a bespoke temperature sensor that can be scraped/polled directly by Prometheus.

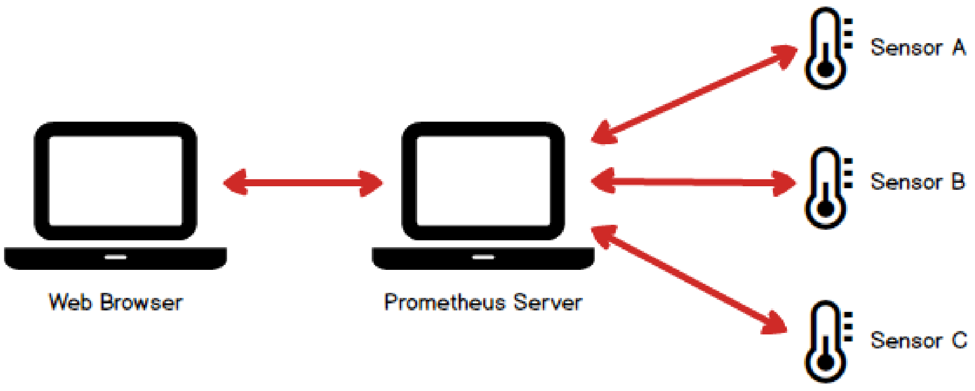

Figure 1: Prometheus scraping sensors

With Prometheus scraping the temperature sensors, a web browser can be used to few graphs of the scraped data.

2 Components

Each sensor is an Arduino microcontroller with a temperature sensore attached, total cost around £40.

Component	**Cost**
Arduino Uno Rev3, ATmega328P, CH340G Compatible Board	£5.79
UK 9V AC/DC Power Supply Adapter Plug for Arduino Uno	£7.95
Ethernet Shield LAN W5100 for Arduino Uno	£7.75
DHT22 AM2302 Digital Temperature and Humidity Sensor	£6.90
0.25 Watt Metal Film Resistor 10K Ohm	£0.99
Uno Ethernet Shield Case	£10.36
Total cost in November 2020	**£39.74**

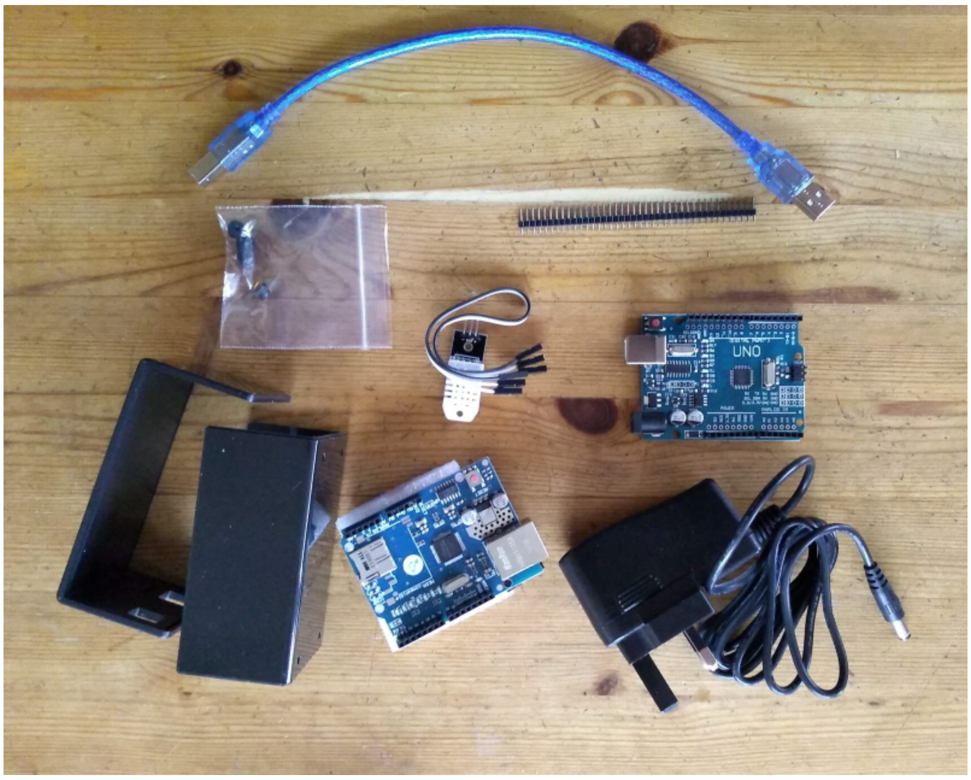

Figure 2: Components

3 Wiring the Sensor

The sensor is a DHT22 AM2302 capacitive humidity sensing digital temperature and humidity module. It is has a calibrated digital signal output of the temperature and humidity sensors. Other sensors are available such as the cheaper DHT11, but supposedly it is less sensitive and less durable. Whilst sensitivity is not important durability probably is.

The sensor is wired with a pull up resistor. This ensures that the signal wire has a small but consistent current, so it is less susceptible to electrical interference. In theory you can also set the pin mode with `pinMode("pin", INPUT_PULLUP);` to use a built in pull up resistor.

The sensor works without the pull up resistor but is probably less accurate (though I haven't tested it).

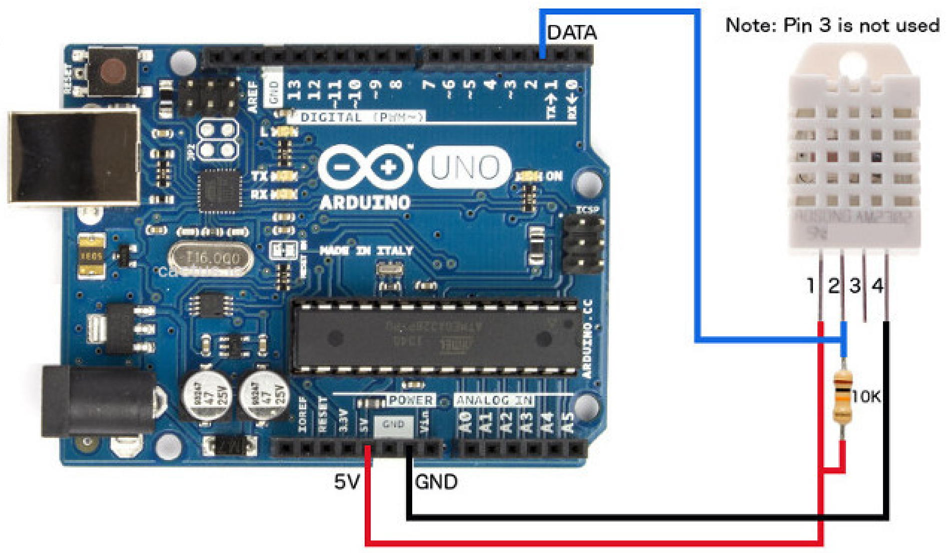

Figure 3: Wiring diagram with pull up resistor

4 Programming

Tooling:

- VSCode
  https://code.visualstudio.com/
- Platformio
  https://marketplace.visualstudio.com/items?itemName=platformio.platformio-ide

All Arduino programs have the same format with a `void setup()` and `void loop()` functions. The `loop()` runs continuously and the `setup()` is run when the Arduino is turned on or when the reset button (red button near the USB socket) is press.

```
"../src/shit.cpp" 5a≡
  #include <Arduino.h>
  ⟨ libraries 5b, ... ⟩
  ⟨ configuration 6a, ... ⟩
  ⟨ functions 6c, ... ⟩
  void setup()
  {
    Serial.begin(9600);
    while (!Serial){;}
  ⟨ setup 6b ⟩
  }
  void loop()
  {
  ⟨ loop 8a ⟩
  }
  ◇
```

Since we're using Platformio we need the Arduino header (`<Arduino.h>`). For normal Arduino programs it is not required.

It's useful to be able to connect to the Arduino over the USB cable, so the `setup()` opens

Include serial for output to monitor serial communications and wait for the port to open. Waiting for the serial port to connect is only needed for native USB ports.

4.1 Sensor

To interact with the DHT22 sensor we need specific drivers (`<DHT.h>`), which rely on the Adafruit Unified Sensor Driver (`<Adafruit_Sensor.h>`), The Adafruit Unified Sensor Driver is an abstraction layer which makes creating reliable drivers is easier.

The Serial Peripheral Interface (`<SPI.h>`) is a synchronous serial data protocol used by microcontrollers for communicating with peripheral devices quickly over short distances.

```
⟨ libraries 5b ⟩ ≡
  // Sensor libraries:
  #include <Adafruit_Sensor.h>
  #include <DHT.h>
  #include <SPI.h>
  ◇
```

Fragment defined by 5b, 7b.
Fragment referenced in 5a.

The sensor communicates using pin 2 (see wiring) and the sensor type need to be initialized. The temperature and humidity are stored in global variables since they'll be used by pretty much every part of the program.

⟨ *configuration* 6a ⟩ ≡

```
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht = DHT(DHTPIN, DHTTYPE);
float temperature = 0;
float humidity = 0;
◇
```

Fragment defined by 6a, 7c.
Fragment referenced in 5a.

The sensor needs to `begin()`. Formerly this method as used to pass in parameters relating the the speed of the Arduino. Now the sensor automatically adapts but `begin()` is still needed.

⟨ *setup* 6b ⟩ ≡

```
    Serial.println("Set␣up␣sensor");
    dht.begin();
  ◇
```

Fragment referenced in 5a.

Set the values in the globals, could have passed stuff around. Read the humidity as a percentage Read the temperature as Celsius:

⟨ *functions* 6c ⟩ ≡

```
void readSensor()
{

  humidity = dht.readHumidity();

  temperature = dht.readTemperature();
  // Check if any reads failed and exit early (to try again):
  if (isnan(humidity) || isnan(temperature))
  {
    Serial.println("Failed␣to␣read␣from␣sensor");
    return;
  }
}
◇
```

Fragment defined by 6c, 7a, 8b.
Fragment referenced in 5a.

Not necessary in use but handy for development.

⟨ *functions* 7a ⟩ ≡

```
void serialPrintReadings()
{
  Serial.print("Humidity:␣");
  Serial.print(humidity);
  Serial.print("␣%␣|␣");
  Serial.print("Temperature:␣");
  Serial.print(temperature);
  Serial.println("␣C");
}
◇
```

Fragment defined by 6c, 7a, 8b.
Fragment referenced in 5a.

4.2 Ethernet Client

aWOT is in version 3

`lasselukkari/aWOT@0.0.0-alpha+sha.bf07e6371c`

⟨ *libraries* 7b ⟩ ≡

```
// Ethernet shield
#include <Ethernet.h>
#include <aWOT.h>
◇
```

Fragment defined by 5b, 7b.
Fragment referenced in 5a.

⟨ *configuration* 7c ⟩ ≡

```
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(10, 0, 21, 211);
// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);
Application app;
◇
```

Fragment defined by 6a, 7c.
Fragment referenced in 5a.

No need to over do the rate besides it takes abot about 250 milliseconds to poll the sensor. Prometheus normally scrapes every minute.

⟨ *loop* 8a ⟩ ≡

```
    // Wait a couple of seconds between measurements.
    delay(2000);
    // Reading temperature or humidity takes about 250 milliseconds
    // Sensor readings may also be up to 2 seconds 'old'
    readSensor();
    serialPrintReadings();
    EthernetClient client = server.available();
    if (client.connected()) {
      app.process(&client);
      client.stop();
    }
  ◇
```

Fragment referenced in 5a.

Prometheus style metrics for scraping.

⟨ *functions* 8b ⟩ ≡

```
  void metricsCmd(Request &req, Response &res)
  {
    Serial.println("Request␣for␣metrics");
    res.set("Content-Type", "text/plain");
    res.print("#␣HELP␣temperature␣is␣the␣last␣temperature␣reading␣in␣degrees␣celsius\n");
    res.print("#␣TYPE␣temp␣gauge\n");
    res.print("temperature␣"+ String(temperature) + "\n");
    res.print("#␣HELP␣humidity␣is␣the␣last␣relative␣humidity␣reading␣as␣a␣percentage\n");
    res.print("#␣TYPE␣humidity␣gauge\n");
    res.print("humidity␣"+ String(humidity) + "\n");
  }
  ◇
```

Fragment defined by 6c, 7a, 8b.
Fragment referenced in 5a.

4.3 Upload

Drivers on PC.

read back doesn't always work.

4.4 Testing

```
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Webserver set up
Sensor set up
Humidity: 55.90 % | Temperature: 22.10 C
Humidity: 56.30 % | Temperature: 22.20 C
```

Rather than link up to a router

- Assign a manual IP address to the laptop's ethernet connection say 10.0.21.1.

- Subnet mask 255.255.255.0.
- Assign a manual IP address to the Arduino's ethernet, say 10.0.21.211.
- Subnet mask 255.255.255.0.
- Leave the default Gateway empty.
- Use an ethernet patch cable to link the two (since 100BaseT onwards it doesn't have to be a special cross over cable).
- You should then be able to get your Arduino site up on `http://192.168.0.2` from the laptop.

This is the endpoint at `http://10.0.21.211/metrics`.

```
> curl 10.0.21.211
# HELP temperature is the last temperature reading in degrees celsius
# TYPE temp gauge
temperature 23.30
# HELP humidity is the last relative humidity reading as a percentage
# TYPE humidity gauge
humidity 47.60
```

```
docker-compose up
```

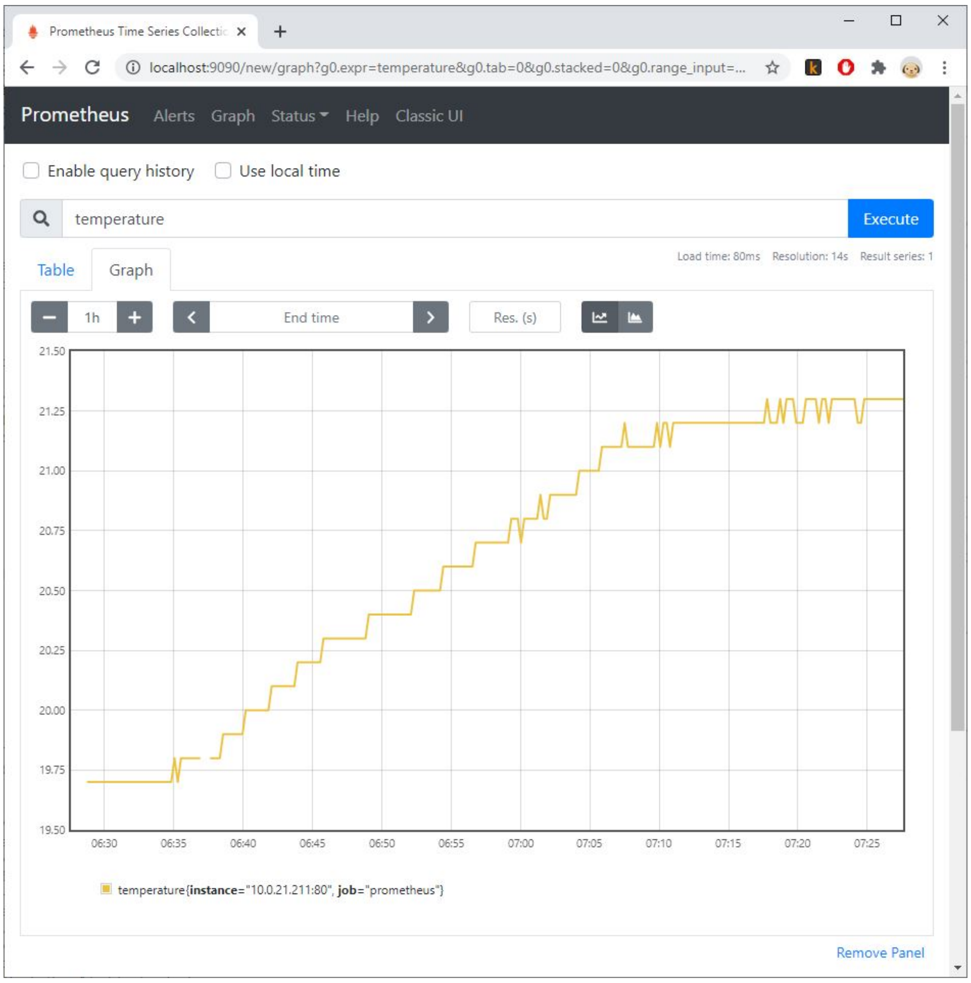

Figure 4: Temperature graph

5 Packaging

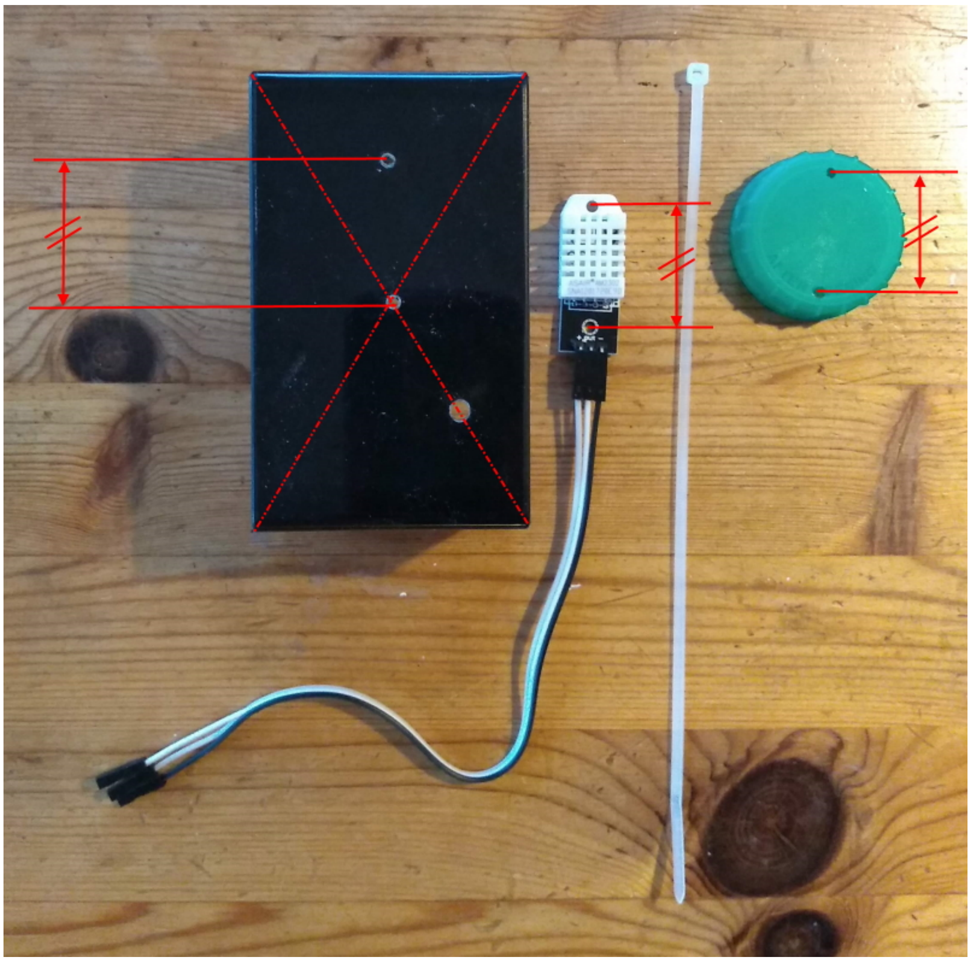

Figure 5: Components for mounting the sensor

2.5 mm holes and 4 mm holes.

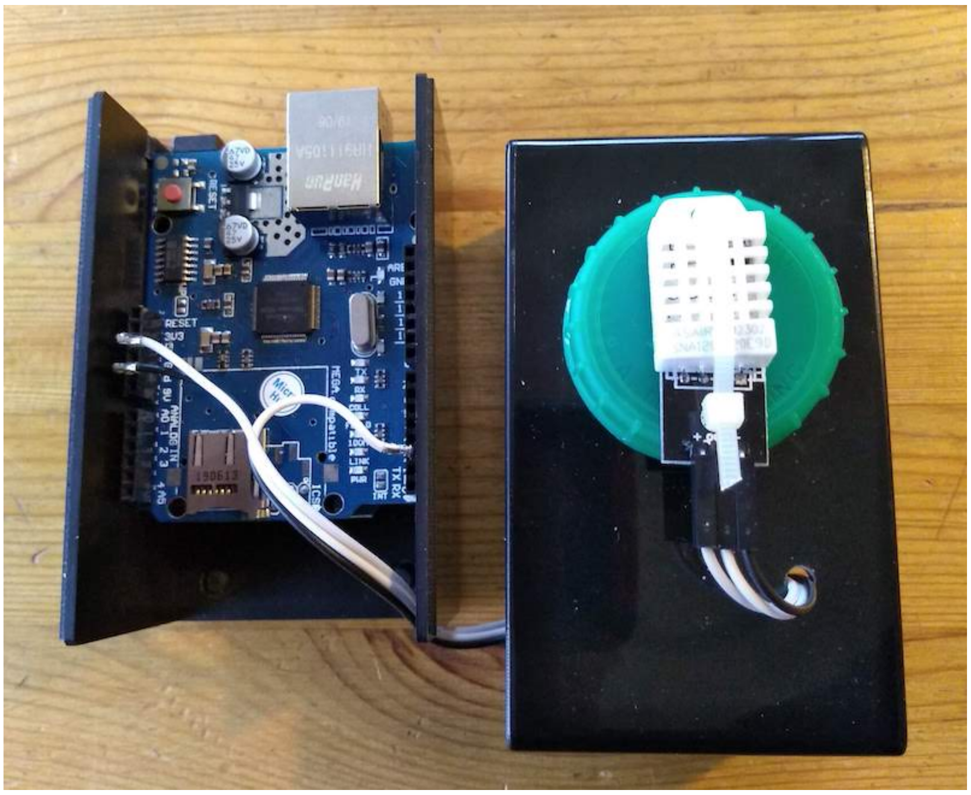

Figure 6: Mounted sensor and wiring