

How to use Arduino UNO for low-cost temperature measurements with Pt 100 sensors

Veit, Martin; Johra, Hicham

Creative Commons License
CC BY 4.0

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

[Link to publication from Aalborg University](#)

Citation for published version (APA):

Veit, M., & Johra, H. (2022). *How to use Arduino UNO for low-cost temperature measurements with Pt 100 sensors*. Department of the Built Environment, Aalborg University. DCE Lecture notes No. 80

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy

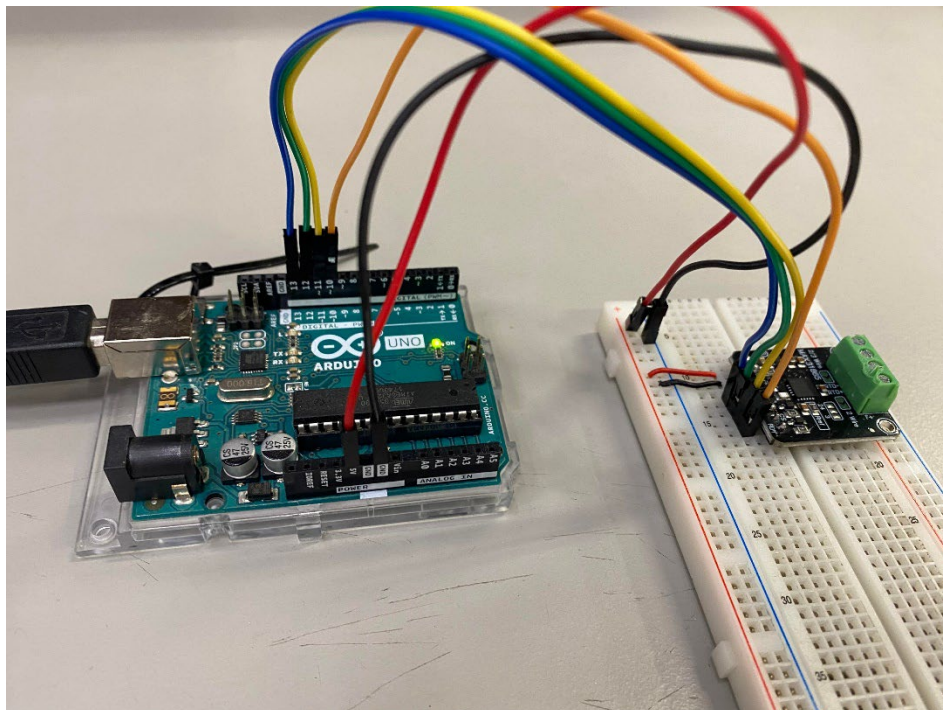
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.



DEPARTMENT OF THE BUILT ENVIRONMENT
AALBORG UNIVERSITY

How to use Arduino UNO for low-cost temperature measurements with Pt 100 sensors

Martin Veit
Hicham Johra



Aalborg University
Department of the Built Environment
Division of Sustainability, Energy & Indoor Environment

DCE Lecture Notes No. 80

How to use Arduino Uno for low-cost temperature measurements with Pt 100 sensors

by

Martin Veit
Hicham Johra

November 2022

© Aalborg University

Scientific Publications at the Department of the Built Environment

Technical Reports are published for timely dissemination of research results and scientific work carried out at the Department of the Built Environment at Aalborg University. This medium allows publication of more detailed explanations and results than typically allowed in scientific journals.

Technical Memoranda are produced to enable the preliminary dissemination of scientific work by the personnel of the Department of the Built Environment where such release is deemed to be appropriate. Documents of this kind may be incomplete or temporary versions of papers—or part of continuing work. This should be kept in mind when references are given to publications of this kind.

Contract Reports are produced to report scientific work carried out under contract. Publications of this kind contain confidential matter and are reserved for the sponsors and the Department of the Built Environment. Therefore, Contract Reports are generally not available for public circulation.

Lecture Notes contain material produced by the lecturers at the Department of the Built Environment for educational purposes. This may be scientific notes, lecture books, example problems or manuals for laboratory work, or computer programs developed at the Department of the Built Environment.

Theses are monographs or collections of papers published to report the scientific work carried out at the Department of the Built Environment to obtain a degree as either PhD or Doctor of Technology. The thesis is publicly available after the defence of the degree.

Latest News is published to enable rapid communication of information about scientific work carried out at the Department of the Built Environment. This includes the status of research projects, developments in the laboratories, information about collaborative work and recent research results.

Published 2022 by
Aalborg University
Department of the Built Environment
Thomas Manns Vej 23
DK-9220 Aalborg Ø, Denmark

Printed in Aalborg at Aalborg University

ISSN 1901-7286
DCE Lecture Notes No. 80

Table of Contents

1	Foreword	6
2	Introduction and motivations.....	7
3	Equipment	8
4	Description of the Arduino UNO, breadboard, and Pt 100	9
4.1	Hardware	9
4.2	Breadboard	10
4.3	Pt 100 sensor	11
4.4	Adafruit MAX31865 RTD amplifier	12
5	Connection of the hardware	15
5.1	Setting up a single temperature measurement Adafruit module.....	15
5.2	Setting up multiple temperature measurement Adafruit modules	20
5.3	Arduino IDE Installation.....	21
5.4	Arduino code	22
5.5	Description of code.....	26
6	Simple LabVIEW interface for the Arduino setup.....	29
7	Troubleshooting	30
7.1	Incorrect wiring	30
7.2	Insufficient depth of pins in breadboard.....	30
7.3	Problems with Pt 100 sensors and connection to the Adafruit MAX31865 amplifier	31
7.4	No power connected to Adafruit MAX31865.....	33
7.5	Malfunctioning amplifier.....	35
7.6	Short-circuiting of the Arduino UNO	36
7.7	When the troubleshooting guide fails.....	37
7.8	Code error in Arduino IDE	37
7.8.1	Board at COM# is not available	37
	Appendix A: Code to perform temperature measurements with Arduino.....	38
	References	41

1 Foreword

Temperature measurements are widely used for different applications, such as measuring air temperature in ventilation ducts, measurements for thermal comfort and other more specialized applications. Therefore, it is important to have access to reliable temperature sensors. However, this can be expensive, and the need for cheap robust temperature measurement setups is therefore high. This lecture note seeks to make this more approachable, by measuring temperature using an Arduino UNO, an amplifier and a Pt 100 sensor.

The aim of this lecture note is to detail the full procedure to make cheap temperature measurement setups using an Arduino UNO and Pt 100 sensors. These temperature measurement setups have been built and are used at Aalborg University, Department of the Built Environment (<https://www.en.build.aau.dk/>).

The guideline is meant for people with zero experience, so anyone can replicate the setup. Section 5 is a step-by-step guideline on how to connect the Arduino for measurement setup, and what code to use for logging the data. This section can stand alone and require no knowledge of the other sections. For a more thorough overview of the setup, read the previous sections as well. If any issues are experienced, refer to section 7 for troubleshooting common problems.

2 Introduction and motivations

The Arduino is a small computer board, which can be programmed with a relatively simple code to perform automated tasks with very low energy need, small volume and at a very low cost. In the present case, an Arduino UNO board is used to perform temperature measurements with a Pt 100 sensor and transmit the information to a “normal” personal computer (PC) via a USB cable.

The motivation behind the use of Arduino to perform temperature measurements is cost. Arduinos are relatively easy to use and are ideal for cheap simple setups such as continuous temperature measurements. The Arduino temperature setup is an interesting alternative to more costly equipment from, e.g., National Instruments. While National Instruments equipment is great for high-precision setups integrating multiple input and output modules with high actuation/measurement rates, simple integration and robust operation, the Arduino hardware is a great alternative for cost-effective simple setups like for stand-alone temperature measurement with a handful of Pt 100 sensors. For such application, the cost of an Arduino setup is about 10 times lower than that of a National Instruments setup (see *Table 1*).

Table 1. Price overview of components for temperature measurement setup with Pt 100 sensors for an Arduino hardware compared to a National Instruments hardware. The pricing (Danish Kroner: DKK) is based on 2022 prices for each component.

Arduino setup	
Hardware	Cost
Arduino UNO	1 x 200 DKK
Adafruit MAX31865	10 x 100 DKK
Breadboard	1 x 44 DKK
Pt 100	10 x 25 DKK
Cost per channel	149 DKK

National Instruments setup	
Hardware	Cost
NI 9216	1 x 11 330 DKK
Pt 100	8 x 25 DKK
Cost per channel	1 441 DKK

3 Equipment

In this section, the necessary hardware for making an Arduino-based temperature measurement setup is described:

1. Arduino UNO: Small programmable computer board, capable of performing automated measurements and actuations with a range of input/output channels and communication buses.
2. Adafruit MAX31865: Arduino-compatible temperature sensor amplifier for Pt 100 resistance temperature detectors (RTDs) [1].
3. Pt 100: High-accuracy resistance temperature detector (RTD).
4. Breadboard: Connection board for rapid and simple wiring of different electronic components.
5. Wires and miscellaneous: to connect the different electronic components together on the breadboard.

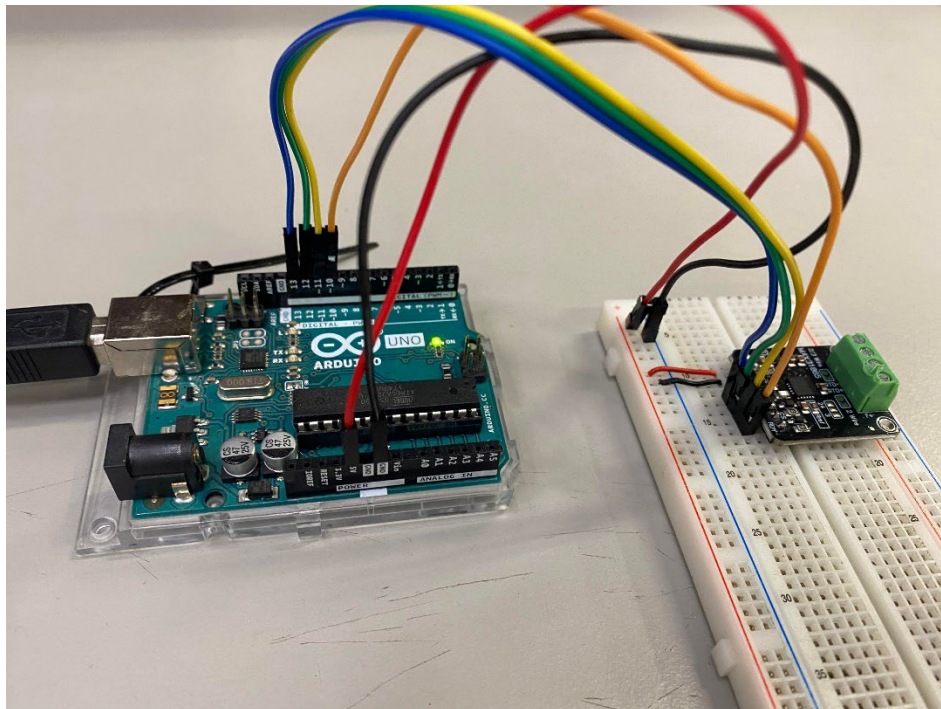


Figure 1. Overview of an Arduino UNO board connected to an Adafruit MAX31865 for the measurement of temperature with a Pt 100 resistance temperature detector (RTD).

4 Description of the Arduino UNO, breadboard, and Pt 100

4.1 Hardware

The Arduino UNO is a microcontroller board (see *Figure 2*) with 14 digital input/output pins (highlighted in red on the top of *Figure 2*) a USB connection (highlighted in blue on the left-hand-side of *Figure 2*), power pins (highlighted in yellow on the bottom of *Figure 2*) and more buses that are not used for the present application. This microcontroller is used to transmit the measured temperature to a computer via the USB port, using an RS232 communication protocol. This USB port is also used to power up the Arduino UNO during operation.

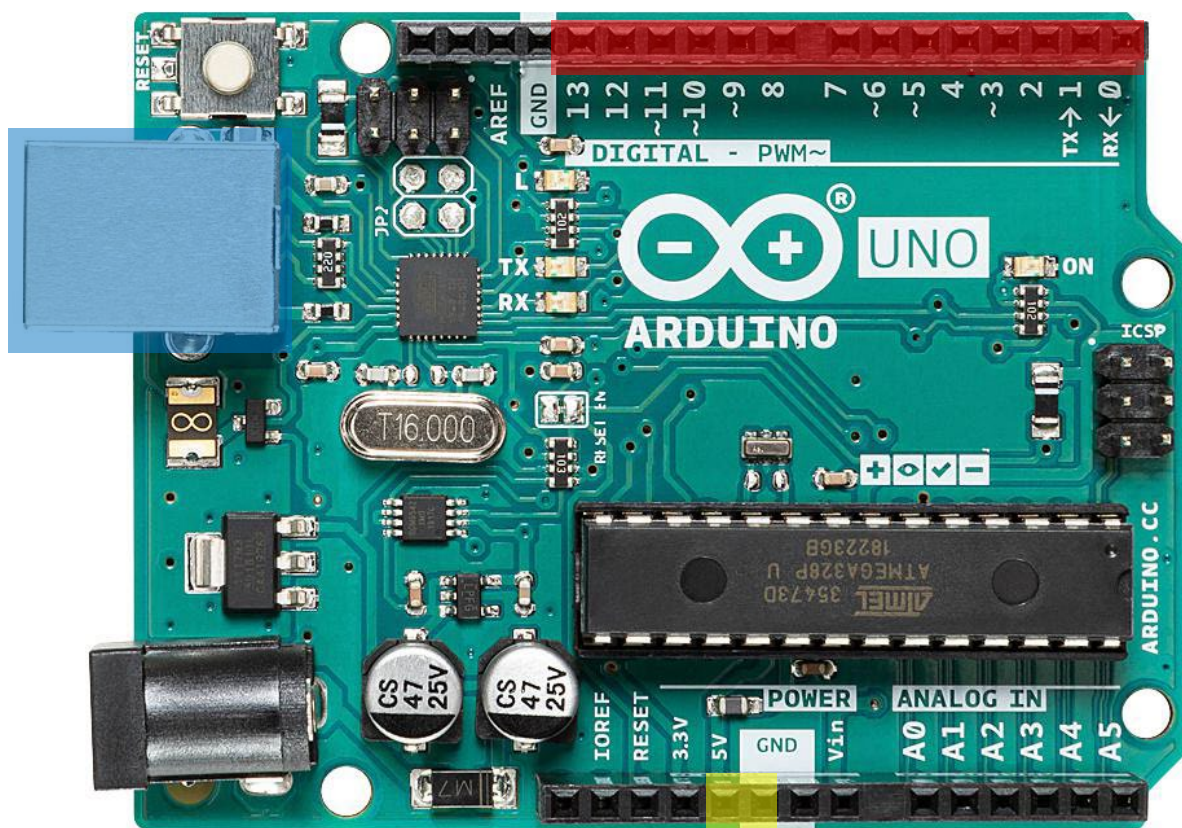


Figure 2. Overview of the Arduino UNO board.

When the Arduino UNO is connected to a PC, it is supplied with power from the PC. To connect power to a breadboard, the **5V** pin and ground should be connected.

Finally, the digital pins can be used as input or output channels. In the case of temperature measurements, they will be used as inputs channels to transmit the temperature measurements from the Pt 100 sensors, via the Adafruit MAX31865 to the Arduino UNO and then the computer.

4.2 Breadboard

A breadboard (see *Figure 3*) is at the core of the setup to connect the different components together. Therefore, understanding how a breadboard works is essential to understand the setup.

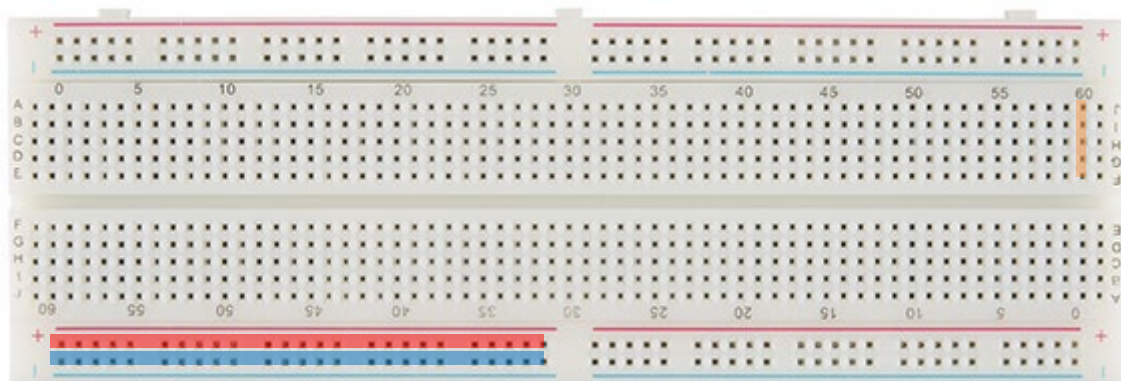


Figure 3. A breadboard and how the pins are connected.

The breadboard is organized as arrays of pins, which are labelled with letters on the sides and numbers at the top and bottom (see *Figure 3*). At the top and bottom, there are a plus and a minus arrays, which can be used as a common power source for components (see red (+) and blue (-) lines in *Figure 3*). The plus and minus arrays are all connected, so a power source connected to a plus and minus pin will power the entire row until the mid-section. Some breadboards have power lines that span the entire length of the breadboard, which are indicated by a full line on the entire length without discontinuous blue and red lines.

The pins on the same number line are all connected until the mid-section. E.g., the pins connected from F to J on the row 60 are all connected (see orange stripe in *Figure 3*).

4.3 Pt 100 sensor

A Pt 100 sensor is a resistance temperature detector (RTD). RTDs have a resistance that varies as a function of the temperature. Pt 100 are RTDs containing platinum with an electrical resistance of 100 Ohms at 0 °C. Pt 100 RTDs can be wired in 3 different ways: 2-wire, 3-wire and 4-wire configuration. The 4-wire configuration is the most accurate and precise configuration and, therefore, should be chosen if possible. For this setup, only 4-wire Pt 100 sensors are used (see *Figure 4*). These 4-wire Pt 100 RTDs have 2 negative wires and 2 positive wires. Each pair is connected to a single leg of the Pt 100 sensor



Figure 4. A 4-wire Pt 100 temperature sensor with a metal sheath.

For a detailed description of Pt 100 sensors and how to assemble them, read the lecture note “Assembling temperature sensors: thermocouples and resistance temperature detectors RTD (Pt100)” [2].

4.4 Adafruit MAX31865 RTD amplifier

The Adafruit MAX31865 [1] is used to measure the resistances of the Pt 100 RTDs, to amplify it, amplify it and transmit it to the Arduino UNO board through the digital input buses of the latter. The Adafruit MAX31865 can be used for 2-wire, 3-wire and 4-wire Pt 100 RTDs. It is used with the 4-wire Pt 100 configuration in this setup. The wires of the Pt 100 RTD should be connected to the Adafruit screw terminals highlighted in red in *Figure 5*. The connection to the Arduino UNO is made through the pins highlighted in blue in *Figure 5*.

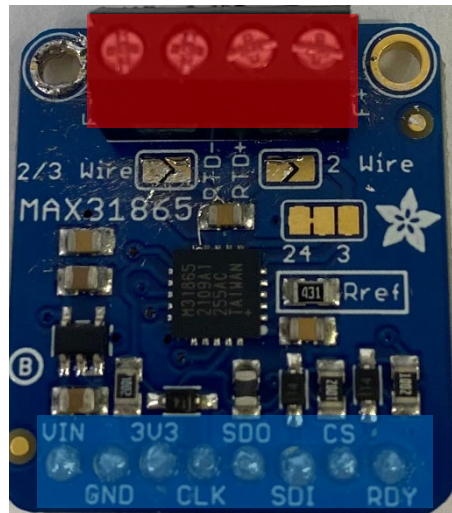


Figure 5. Adafruit MAX31865 and the relevant pins. Red) RTD screw terminals; Blue) power and communication pins to the Arduino UNO board.

If the Adafruit module does not have connector pins and screw terminals, they must be soldered manually. For assembly and soldering instructions, read Adafruit documentation [1].

The wiring of the Pt 100 RTD is shown in *Figure 6*: the 2 screw terminals on the left-hand-side of the Adafruit should be connected to the negative wires (-) of the Pt 100 RTD. The 2 screw terminals on the right-hand-side of the Adafruit should be connected to the positive wires (+) of the Pt 100 RTD.

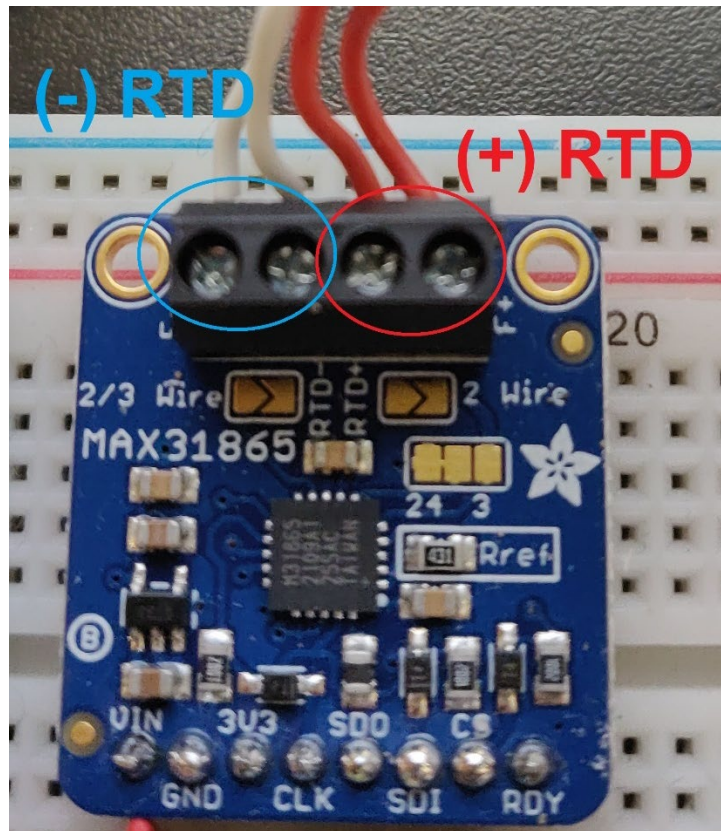


Figure 6. Wiring of the Pt 100 RTD.

The power supply pins are as follows:

Vin	Power supply positive pin (+): 3-5 VDC.
3V3	3.3 VDC output from the voltage regulator: <i>not used for this setup</i> .
GND	Common ground for power (-) and logic.

The Serial Peripheral Interface (SPI) logic pins are as follows:

SCK/CLK	SPI CLoCK pin, which is an input to the chip.
SDO	Serial Data Out.
SDI	Serial Data In.
CS	Chip Select pin, this is the communication pin between the Arduino and the amplifier.

When using multiple Pt 100 sensors and amplifiers, you should 'bit-bang' the SPI pins: each amplifier is connected in series with the **SCK**, **SDO** and **SDI** pins of the other amplifiers. This will be shown in section 5.

The last pin is used to speed up the reading of the measurement.

RDY | Data-ready indicator pin: *not used for this setup*.

5 Connection of the hardware

The setup is scalable with up to 9 sensors. Guidelines for single-sensor is given first, followed by multiple-sensor configuration to show the scalability of the setup.

5.1 Setting up a single temperature measurement Adafruit module

The experimental setup and code used is based on the Adafruit documentation [1]. The color of the wires presented in this document does not make any difference, but it is recommended to use the same color logic or similar, and to stay as consistent as possible in order to make the setup more readable and easier to troubleshoot.

The first step is to make sure that the Arduino UNO works. Insert the USB cable from the Arduino into the computer USB port and see if the ON LED turns bright green on the Arduino (see *Figure 7*).

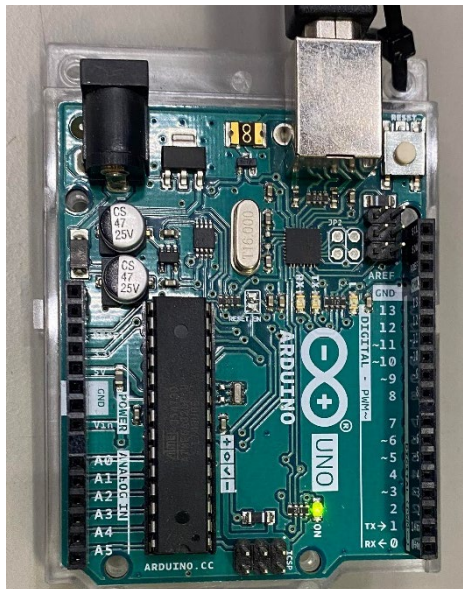


Figure 7. A Arduino UNO with a green LED light switched on, indicating that the Arduino is powered up.

The next step is to grab jumper cables, a breadboard, normal wires and the Adafruit MAX31865 amplifiers (see *Figure 8*).

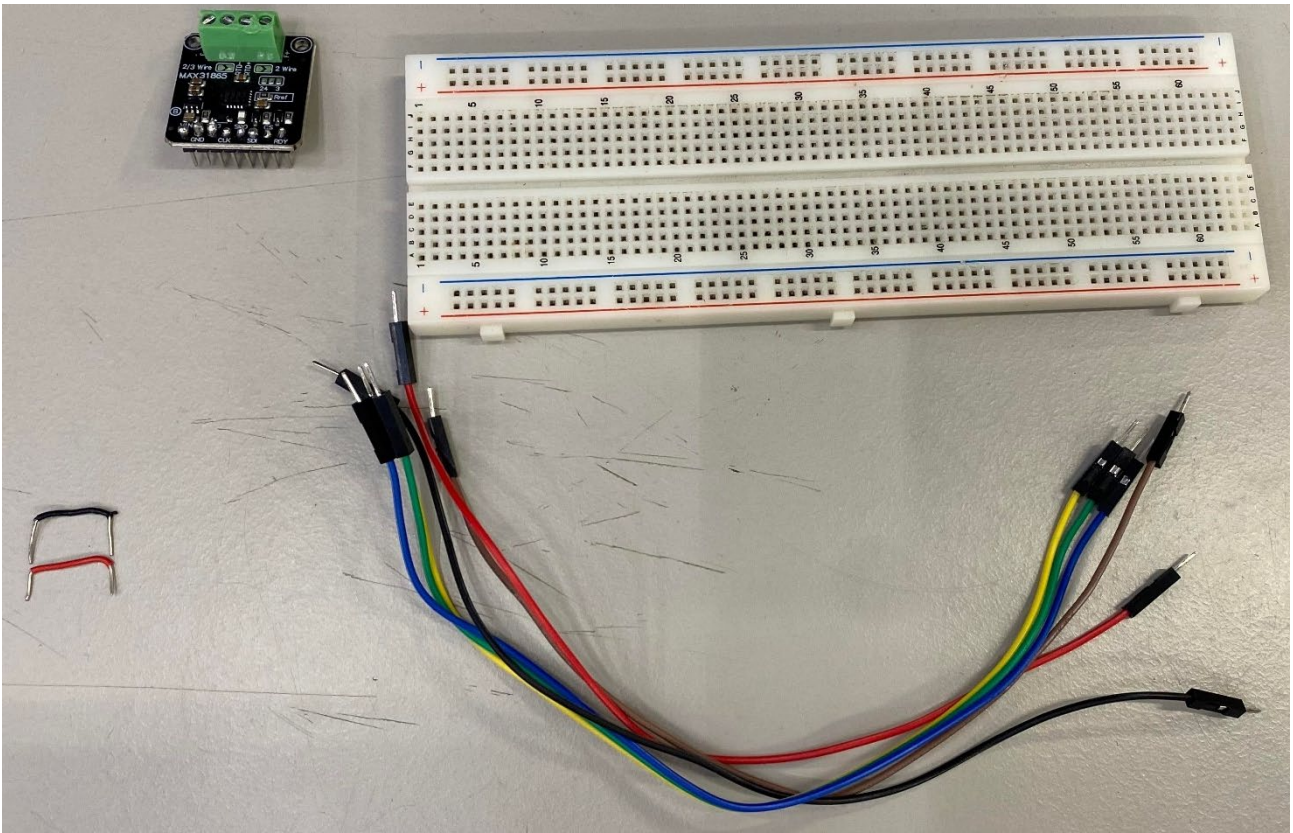


Figure 8. Relevant equipment to set up a single temperature sensor.

Connect the **5V** pin on the Arduino to the positive (+) column on the breadboard with a **red** jumper cable, and a **black** jumper cable from the **GND** pin on the Arduino to the negative (-) column (see *Figure 9*). This will power up the entire array so that all the remaining pins of that line (between the blue and red stripes) can be used to power up the Adafruit amplifiers.

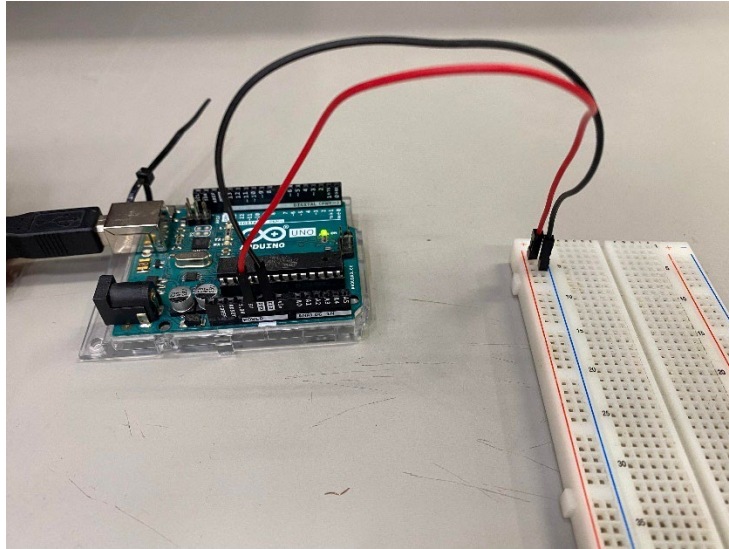


Figure 9. Breadboard powered by the Arduino UNO.

Place the first amplifier somewhere on the breadboard as illustrated in *Figure 10*. Column E can be used for this. Next, power the amplifier from the power lines with a **red** wire to (+) array connected to the **Vin** pin of the Arduino and a **black** wire to the (-) array connected to the **GND** pin of the Arduino (see *Figure 10*).

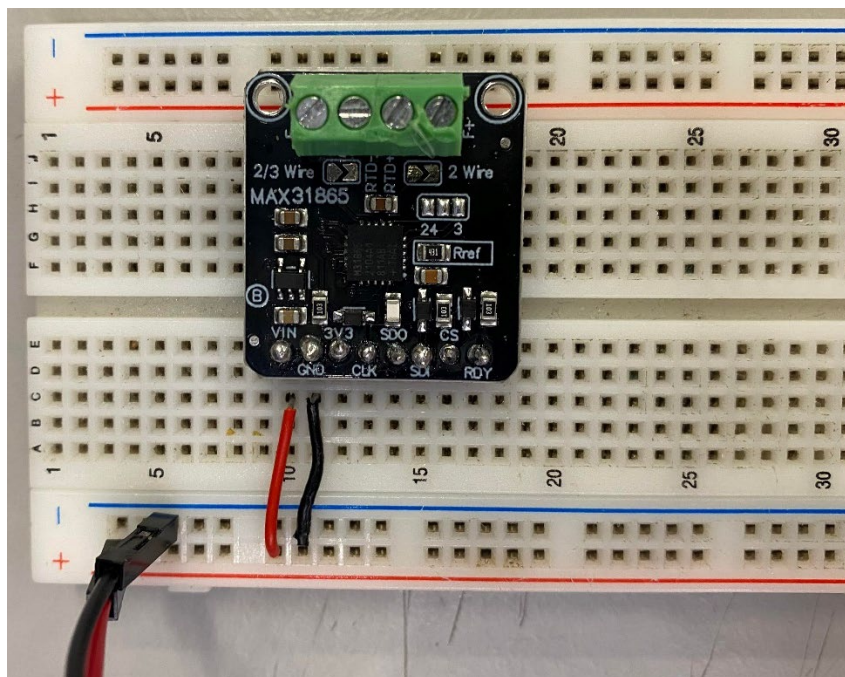


Figure 10. Breadboard with Adafruit MAX31865, powered by the power line connected to the Arduino UNO.

Next, the communication wires should be connected between the Arduino and the amplifier (see *Figure 11*):

- Connect channel 13 on the Arduino, to the same row as the **CLK** pin on the amplifier with a **blue** jumper cable.
- Connect channel 12 on the Arduino, to the same row as the **SDO** pin on the amplifier with a **green** jumper cable.
- Connect channel 11 on the Arduino, to the same row as the **SDI** pin on the amplifier with a **yellow** jumper cable.
- Connect channel 10 on the Arduino, to the same row as the **CS** pin on the amplifier with an **orange** jumper cable

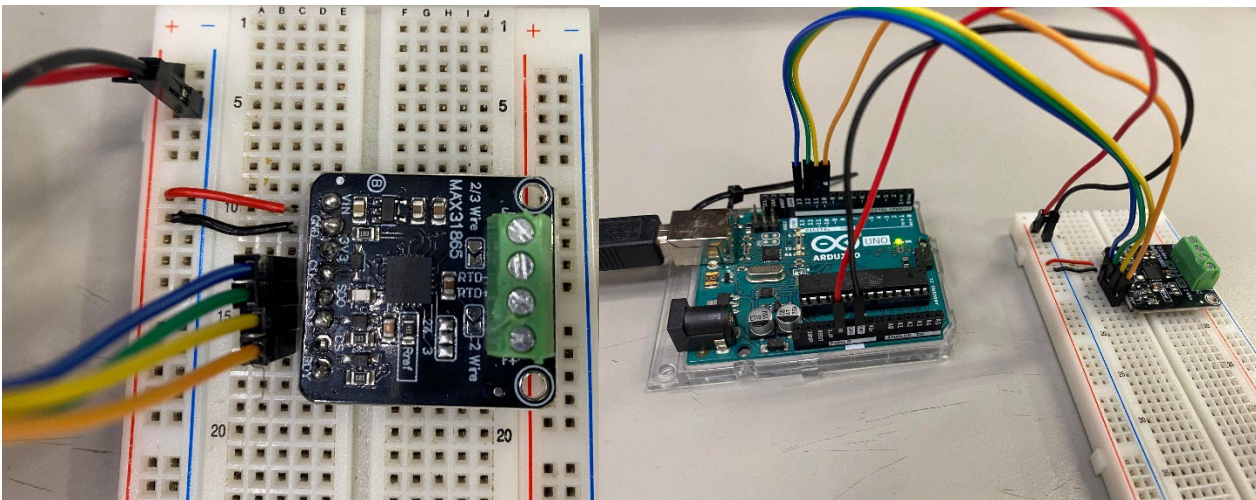


Figure 11. Connection of communication pins to the Adafruit MAX31865. Left) Close-up of the amplifier; right) overview with connection to the Arduino UNO.

The Arduino is now ready to communicate with the amplifier. Connect the two wires of one of the legs to one side of the screw terminal block, and the two others at the two other screw terminals (see *Figure 12*).

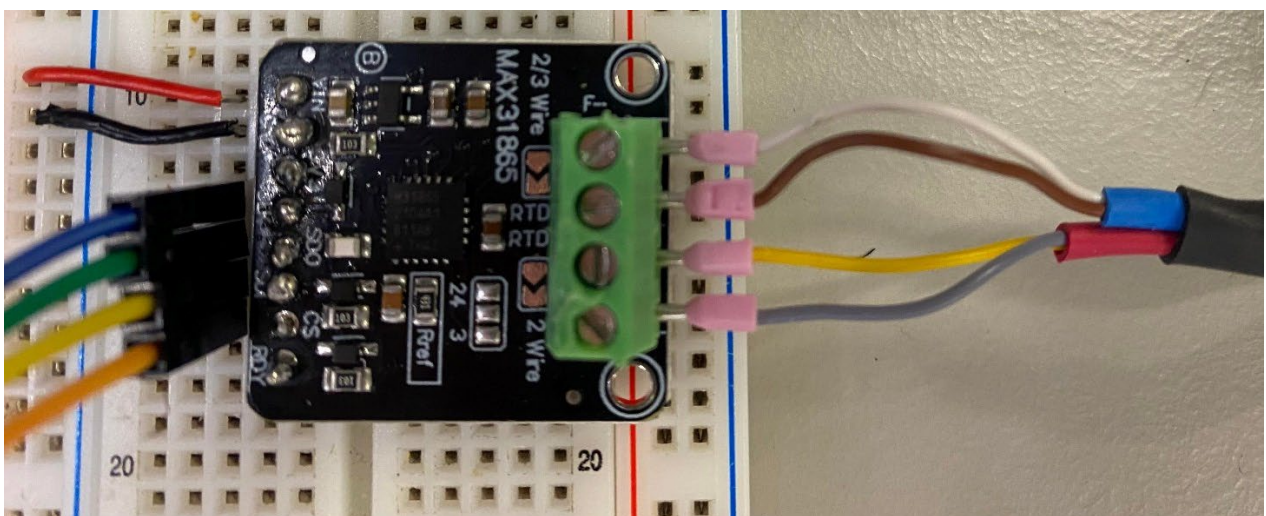


Figure 12. Connection of Pt 100 RTD on the Adafruit MAX31865.

If there is no clear separation of wires for each leg, use a multimeter and check for continuity. Set the knob of the multimeter to resistance measurement and use each of the measurement points of the multimeter at the RTD terminals. If the multimeter makes an audible high-pitched noise, those two wires constitute one leg, and should be placed next to each on the same side of the terminal block. Alternatively, you can check the electrical resistance between the different wires. The electrical resistance between 2 wires of the same leg should be very small (around 2 Ohms), while the electrical resistance between 2 wires of different legs should be around 102 Ohms (for connection to a Pt 100 RTD) (see *Figure 13*).

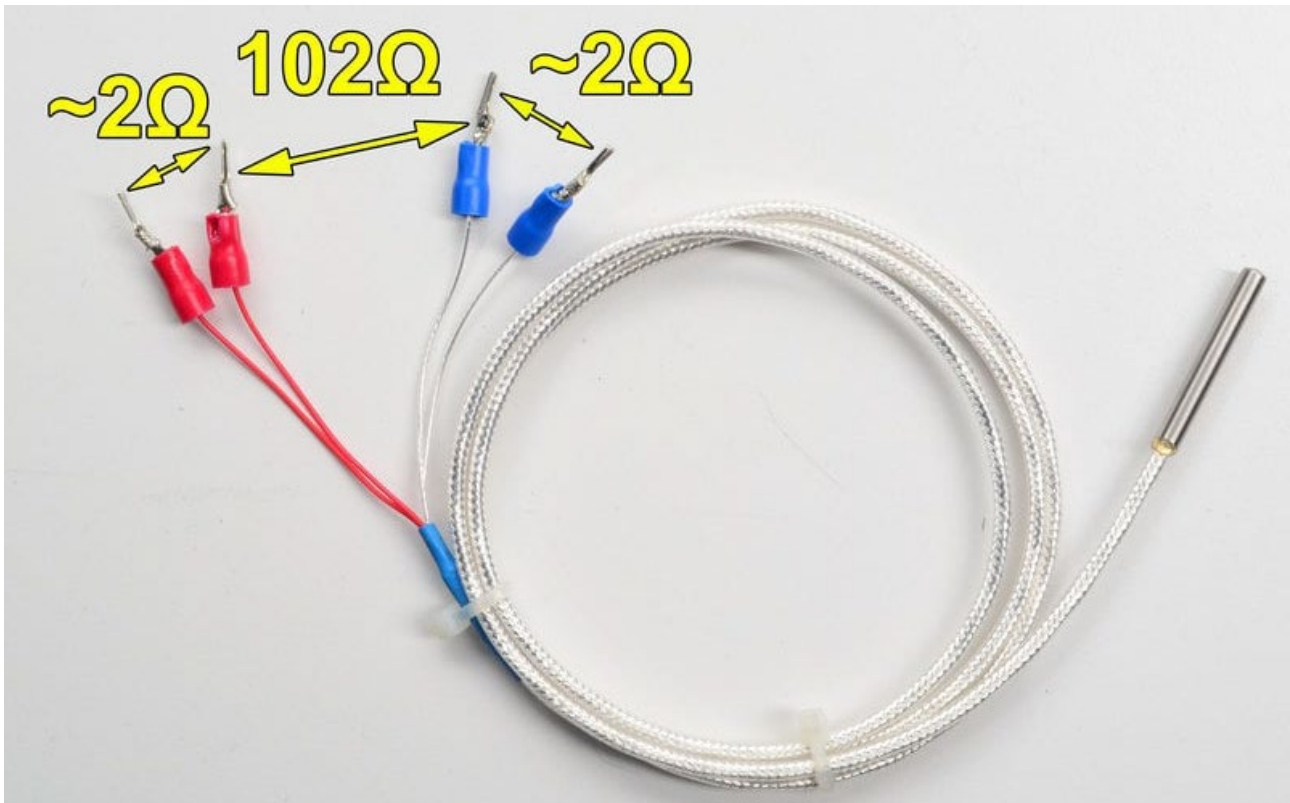


Figure 13. Electrical resistance between the different wires and legs of a 4-wire Pt 100 RTD [1].

You are now ready to use the provided code to make temperature measurement recordings with the Arduino IDE (Integrated Development Environment).

5.2 Setting up multiple temperature measurement Adafruit modules

For setups requiring multiple sensors, you can 'bit-bang' (connect in series) the **CLK** pins of the different Adafruit modules, and do the same for the **SDO** pins and the **SDI** pins (see *Figure 14*). This means that the **CLK** pin on the first amplifier can be connected to the **CLK** pin on the next amplifier, and so on. To do this, place a wire on the same column (same number) on the breadboard as the **CLK** pin on the first amplifier, and connect it to the column of the **CLK** pin on the next amplifier. Do the same for the **SDO** and **SDI** pins. Connect the **CS** pin of the 2nd Adafruit amplifier to the next digital channel on the Arduino (digital channel 9); connect the **CS** pin of the 3rd Adafruit amplifier to the Arduino digital channel 8; connect the **CS** pin of the 4th Adafruit amplifier to the Arduino digital channel 7; etc. A maximum of 9 Adafruit amplifiers can be connected to a single Arduino UNO on digital channels 10 to 2.

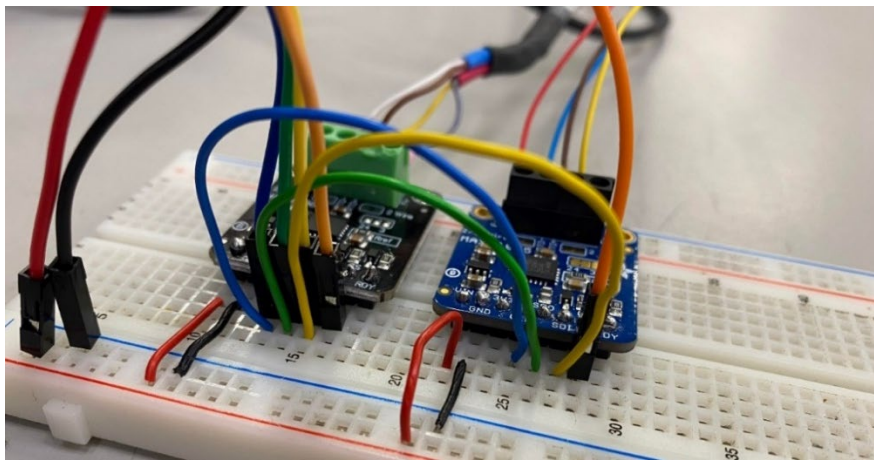


Figure 14. Connection of multiple Adafruit MAX31865 amplifiers by 'bit-banging' the communication pins.

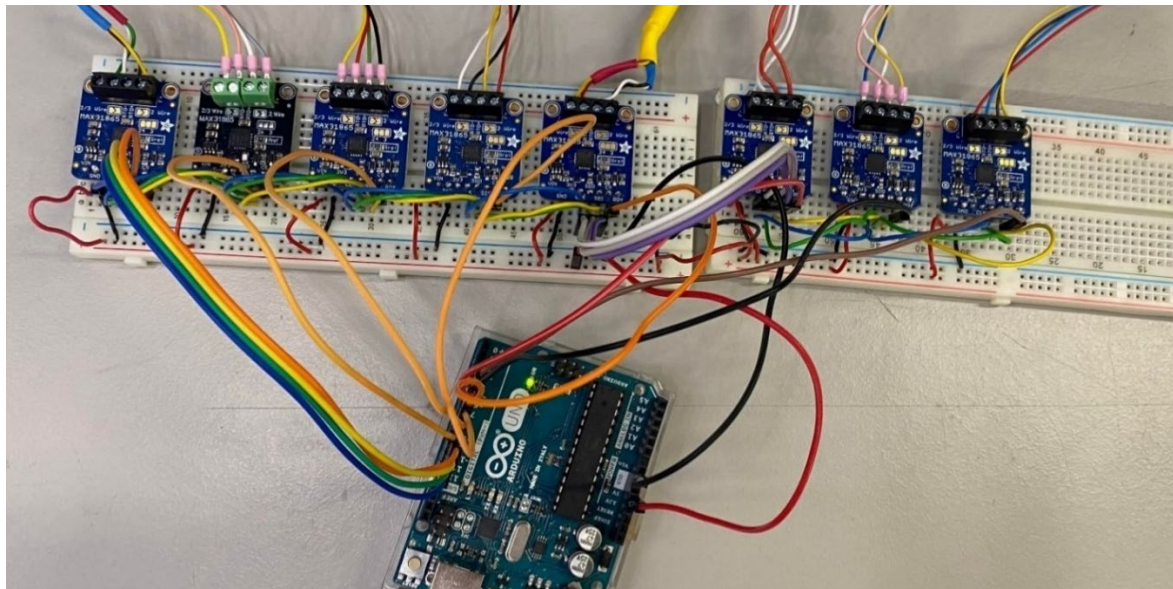
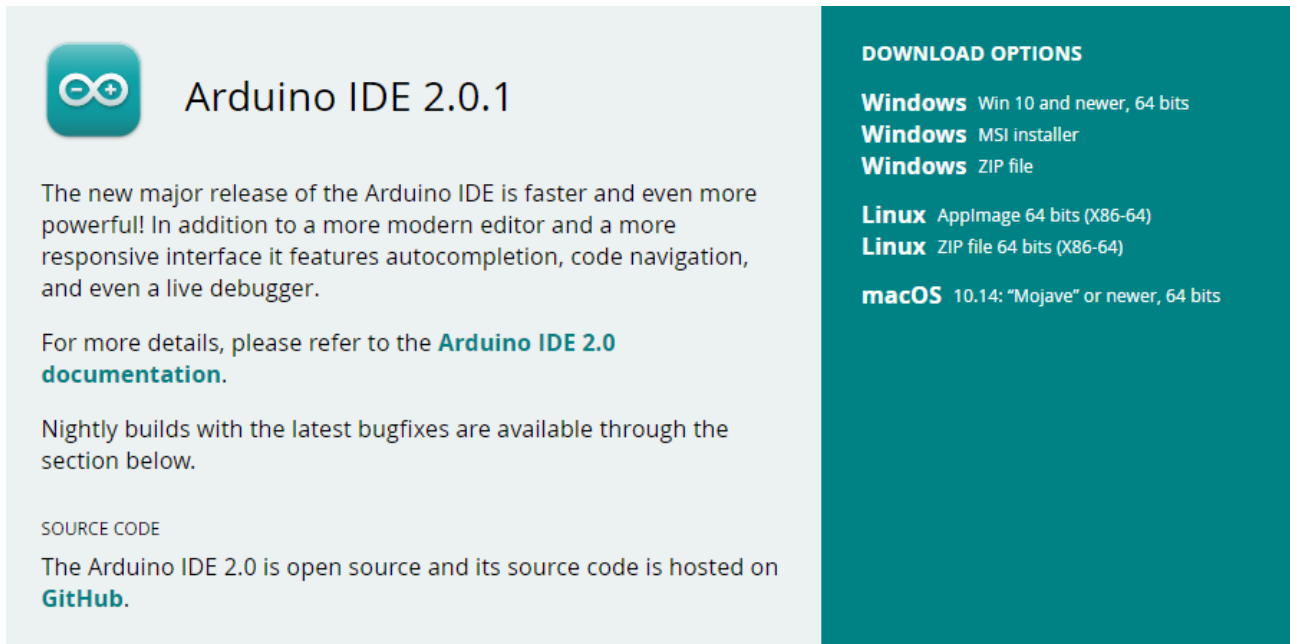



Figure 15. Setup with 8 amplifiers connected to a single Arduino UNO.

TIP: When constructing the setup, start with a single amplifier, make sure it works and then connect the next one.

5.3 Arduino IDE Installation

To download the Arduino IDE, go to <https://www.arduino.cc/en/software>, and select the correct download option. For *Windows 10* and newer, select the option shown in *Figure 16*. After download, run the installation file and follow the steps.



 **Arduino IDE 2.0.1**

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE
The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

- Windows** Win 10 and newer, 64 bits
- Windows** MSI installer
- Windows** ZIP file
- Linux** Applimage 64 bits (X86-64)
- Linux** ZIP file 64 bits (X86-64)
- macOS** 10.14: "Mojave" or newer, 64 bits

Figure 16. Download guide for how to install the Arduino IDE.

5.4 Arduino code

To perform the temperature measurement with the Arduino UNO, a code (program) must be uploaded onto the Arduino board. In this section, a walkthrough of the code is given, along with how to use this code on the Arduino.

Arduino has its own IDE, which uses simplified C++ code as the programming language. When a new script is initialized, two different blocks of code are written by default (see *Figure 17*).

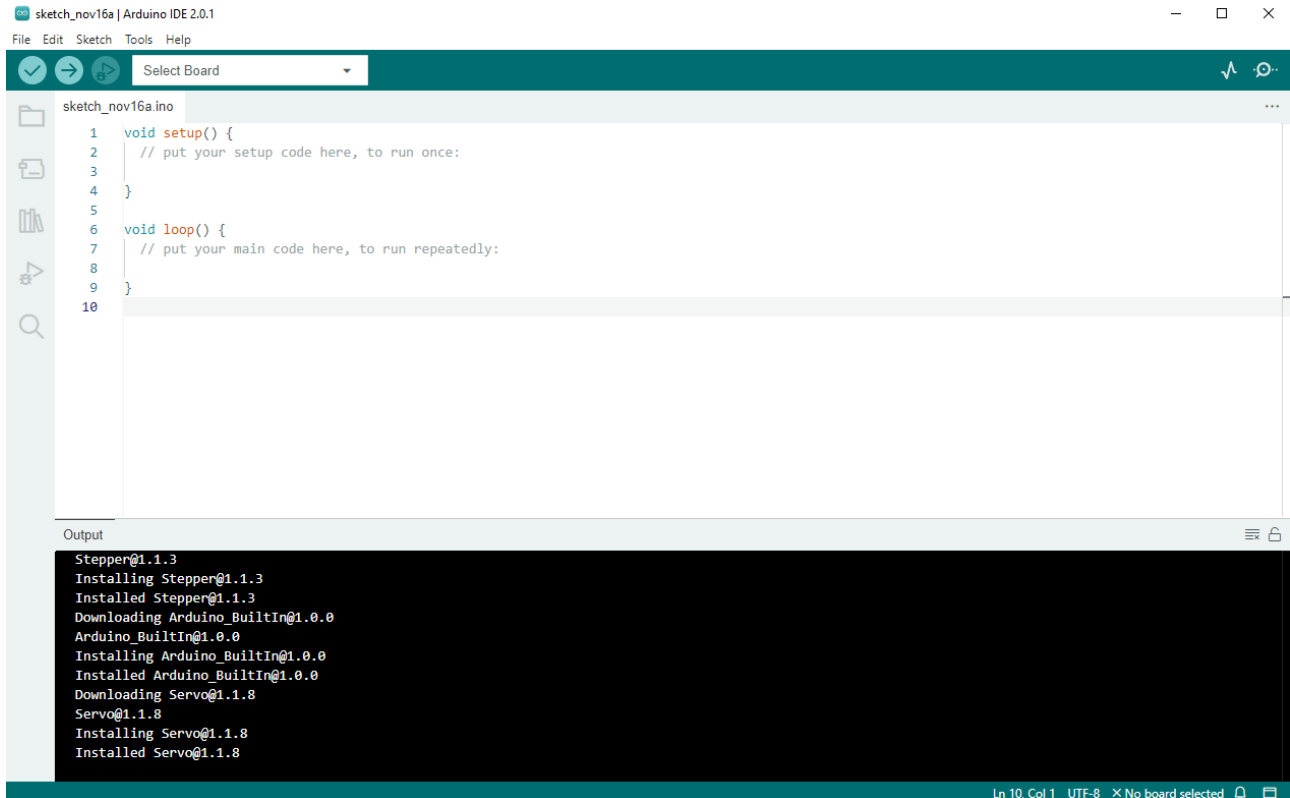


Figure 17. Default code when opening the Arduino IDE.

These two different functions have two different purposes. The `setup()` function is for code which is to be executed exactly once, while the `loop()` function runs continuously. Code that is preceded by “//” is comments, meaning that the corresponding code after the slash will not be executed.

To read the temperature from the Adafruit MAX31865 amplifiers and the RTDs, the library <Adafruit_MAX31865.h> needs to be included in the code. Good practice is to include packages at the beginning of the script for transparency and readability. To get this library, go to the toolbar and click on *Sketch* → *Include Library* → *Manage Libraries* and search for *Adafruit MAX31865* (V 1.6.0) and install it (see *Figure 18* and *Figure 19*). Install all dependencies of the library Adafruit MAX31865.

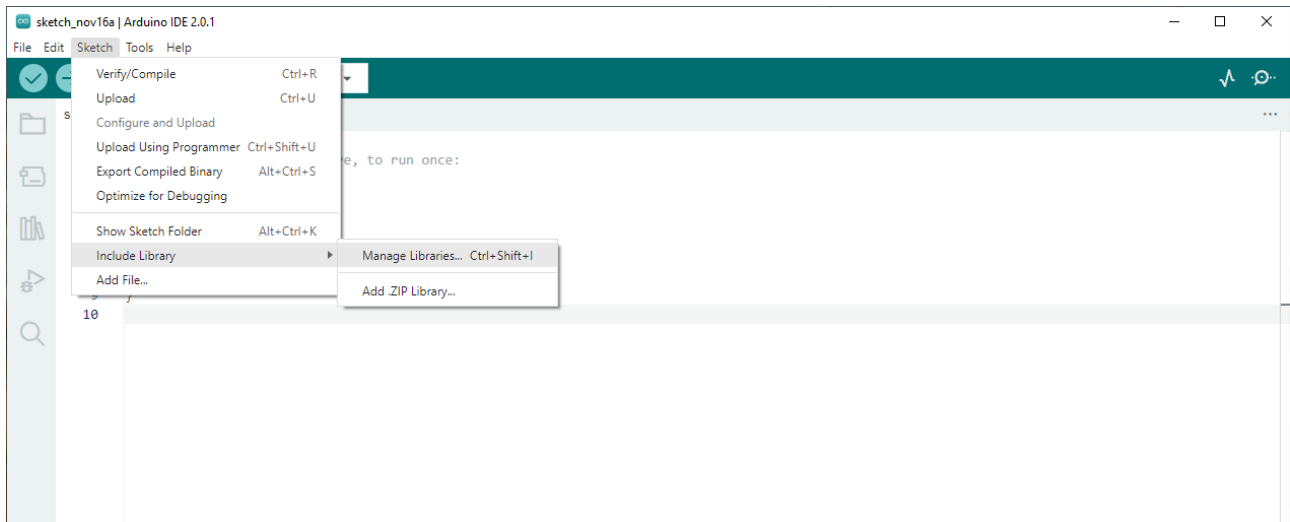


Figure 18. Manage Arduino libraries.

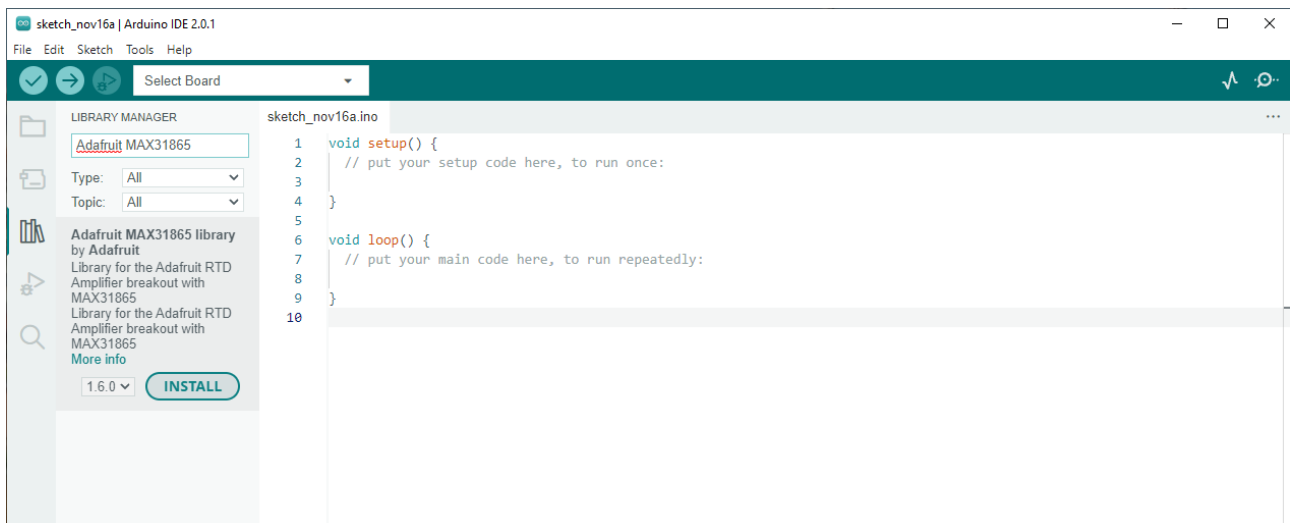


Figure 19. Library manager and the relevant library for the code used in this setup (Adafruit MAX31865 library).

Next, choose the correct COM-port. This is done at Tools → Port: → and then choose the port connected to the Arduino UNO (see *Figure 20*).

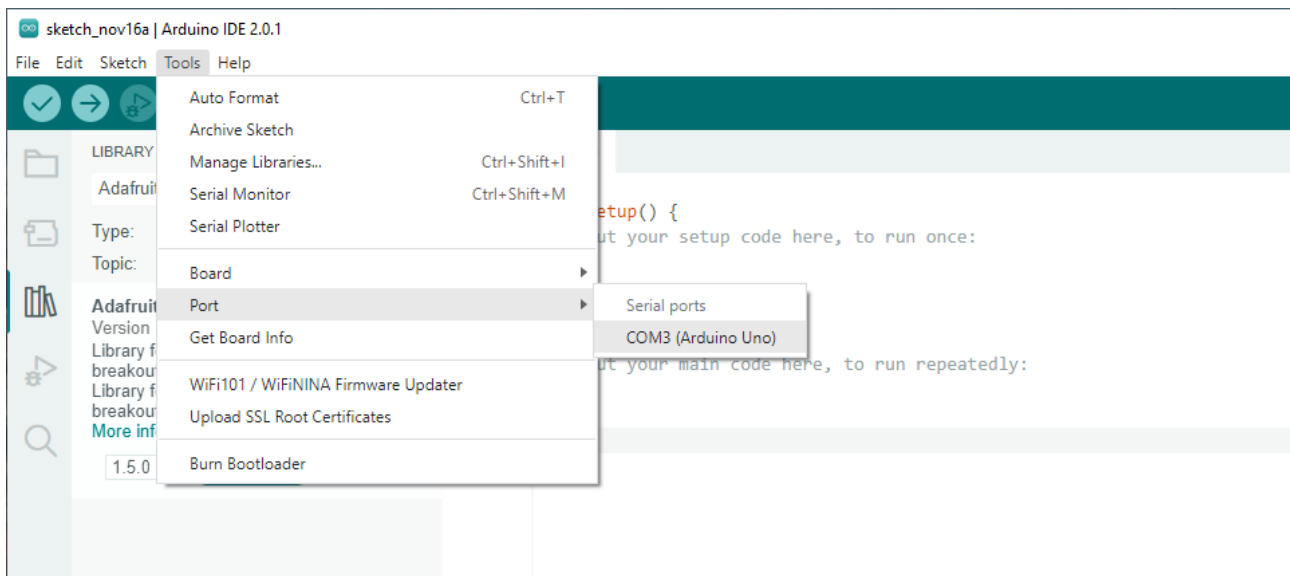


Figure 20. Select the correct COM port to the connected Arduino UNO board.

Select the correct board type (Arduino UNO): Tools → Board → Arduino AVR Boards → Arduino Uno (see *Figure 21*).

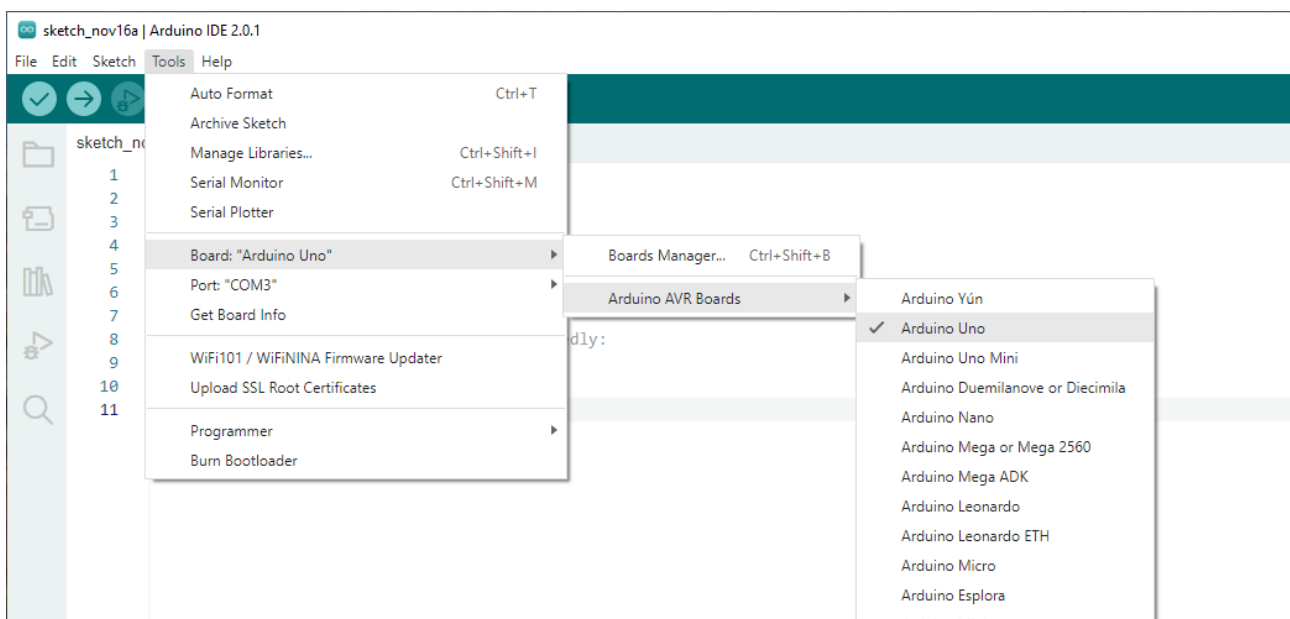


Figure 21. Select the correct Arduino board.

Enable the Serial Monitor and set the baud rate to 115200 in order to constantly display the data received from the Arduino (see Figure 22, highlighted in red).

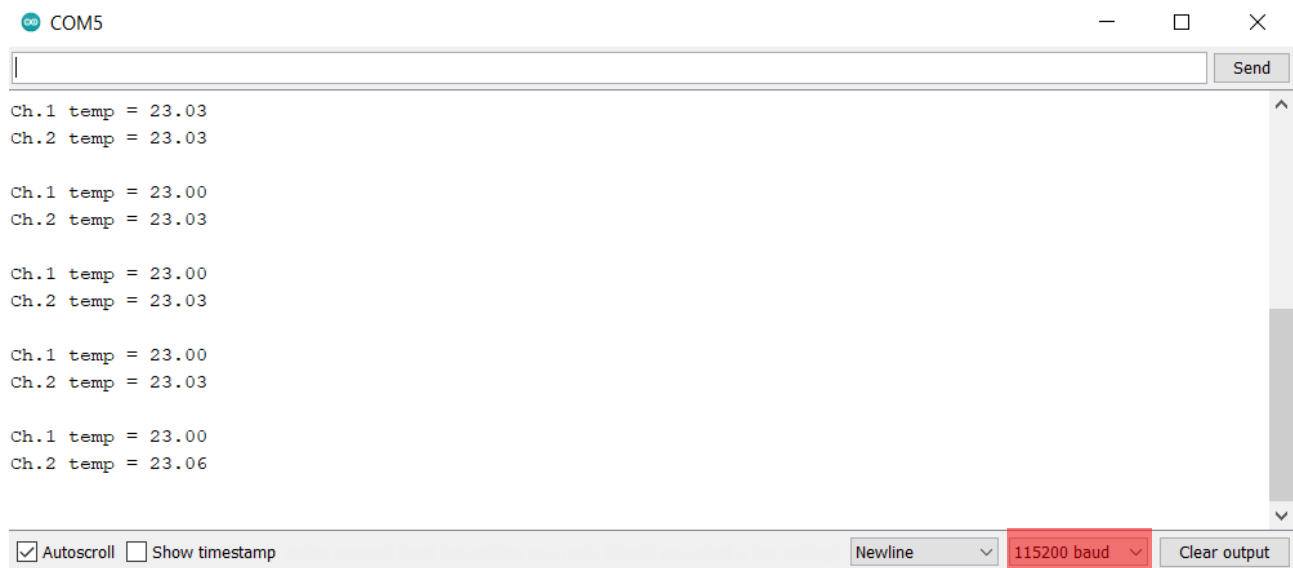


Figure 22. The serial monitor window, where the baud rate can be changed.

If the Arduino board does not send any data, maybe there is no running program on the board. Follow instruction in the following section to upload the correct program/code onto the Arduino board. The code for continuous measurement of temperature with the Arduino board and data transfer through the USB port with rs232 communication protocol can be found in Appendix A.

5.5 Description of code

A walkthrough of the code performing continuous temperature measurement with the Arduino board is given in this section. The code can be copied from appendix A directly and uploaded onto the Arduino.

IMPORTANT: *In the script, remember to change the number of sensors, in the function `num_sensors_func()`.*

The premise of the code is that the user defines the number of sensors present in the setup. This is the only place where the user should change code unless the user wants to change to delay time (sampling rate).

```
int num_sensors_func() {  
    int num_sensors = 4;  
    return num_sensors;  
}
```

This is an auxiliary function, such that the user can easily identify where to insert the number of sensors, and such that the number of sensors is only defined once and can be used in both the `setup()` and `loop()` functions.

Next, two values are defined, the RREF and RNOMINAL. The first defines the value of the reference resistor, which is soldered on the Adafruit MAX31865 amplifier, while the latter defines the nominal resistance of the RTD sensor. For Pt 100 RTDs, this value is 100, meaning a resistance of 100 Ohms at 0 °C.

```
// The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000  
#define RREF      430.0  
// The 'nominal' 0-degrees-C resistance of the sensor  
// 100.0 for PT100, 1000.0 for PT1000  
#define RNOMINAL 100.0
```

The next step is to define variables for each sensor, and their respective digital input channels. As the amplifiers are bit-banged, only the Chip Select (CS) wire should be connected and have a digital input channel defined. For a CS wire connected in the digital input channel 10, the following code applies:

```
Adafruit_MAX31865 thermol = Adafruit_MAX31865(10);
```

This code is then replicated for the remaining channels in the Arduino. If the user defines that only three sensors are used, the remaining code for the rest of the sensors will not be executed.

After this code, the sensors will be initialized, and a signal is given to begin sending the measured temperatures to the computer. Based on the number of sensors used, determined by the user, up to 10 sensors will be initialized. This code is only performed once, therefore, it is placed in the `setup()` function.

```

void setup() {
  int num_sensors;
  num_sensors = num_sensors_func();
  Serial.begin(115200);

  if (num_sensors >= 1){
    thermol.begin(MAX31865_4WIRE);
  }
}

```

Finally, the temperature measurement from the sensors will be “printed” (sent) on the serial port, which will be running continuously and therefore running in the `loop()` function. The same principle is used as before, where only the specified number of sensors will be executed.

```

void loop() {
  int num_sensors;
  num_sensors = num_sensors_func();

  if (num_sensors >= 1){
    Serial.print("Ch.1 temp = "); Serial.println(thermol.temperature(RNOMINAL, RREF));
  }
}

```

The last line is a delay function (wait a certain number of milliseconds before displaying the measured temperature once again). This can be modified to set the sampling rate.

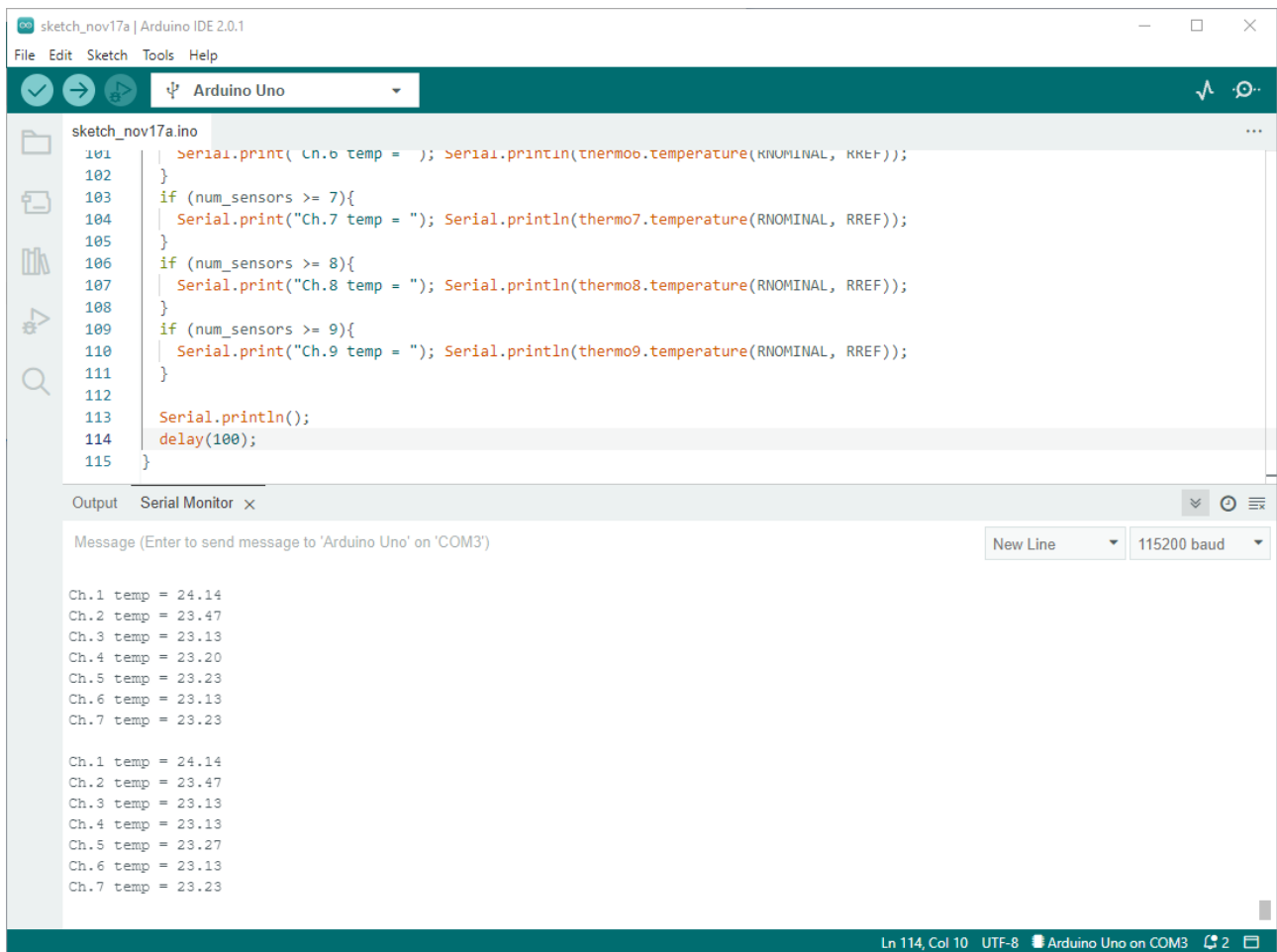
The code is replicated in its entirety in Appendix A.

To upload the code onto the Arduino board and have it running continuously, copy/paste it into the Arduino IDE, press “Verify” and then “Upload” button (see *Figure 23*).



Figure 23. Verify and upload code onto the Arduino board.

Open the *Serial Monitor* (Tools → Serial Monitor) and the data should be displayed continuously (see *Figure 24*).



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for checking, running, and uploading code, along with a dropdown menu set to 'Arduino Uno'. The main editor window displays the code for 'sketch_nov17a.ino'. The code includes comments and serial print statements for temperature readings from various sensors. The Serial Monitor window is open at the bottom, showing a continuous stream of temperature data for channels 1 through 7. The status bar at the bottom indicates the current line and column (Ln 114, Col 10), the encoding (UTF-8), and the connection (Arduino Uno on COM3).

```
101 | Serial.print( "Ch.6 temp = "); Serial.println(thermo6.temperature(RNOMINAL, RREF));
102 | }
103 | if (num_sensors >= 7){
104 |   Serial.print("Ch.7 temp = "); Serial.println(thermo7.temperature(RNOMINAL, RREF));
105 | }
106 | if (num_sensors >= 8){
107 |   Serial.print("Ch.8 temp = "); Serial.println(thermo8.temperature(RNOMINAL, RREF));
108 | }
109 | if (num_sensors >= 9){
110 |   Serial.print("Ch.9 temp = "); Serial.println(thermo9.temperature(RNOMINAL, RREF));
111 | }
112 |
113 | Serial.println();
114 | delay(100);
115 | }
```

Output Serial Monitor x

Message (Enter to send message to 'Arduino Uno' on 'COM3')

New Line 115200 baud

Ch.1 temp = 24.14
Ch.2 temp = 23.47
Ch.3 temp = 23.13
Ch.4 temp = 23.20
Ch.5 temp = 23.23
Ch.6 temp = 23.13
Ch.7 temp = 23.23

Ch.1 temp = 24.14
Ch.2 temp = 23.47
Ch.3 temp = 23.13
Ch.4 temp = 23.13
Ch.5 temp = 23.27
Ch.6 temp = 23.13
Ch.7 temp = 23.23

Ln 114, Col 10 UTF-8 Arduino Uno on COM3 2

Figure 24. Data from temperature measurements is continuously displayed in the Serial Monitor.

6 Simple LabVIEW interface for the Arduino setup

A simple graphical user interface has been developed to continuously display and log the temperature measurements from the Arduino setup described in this document. The interface detects automatically the Arduino UNO board, displays the temperature measurements on a graph and logs the data in a log file that the user can choose (see *Figure 25*).

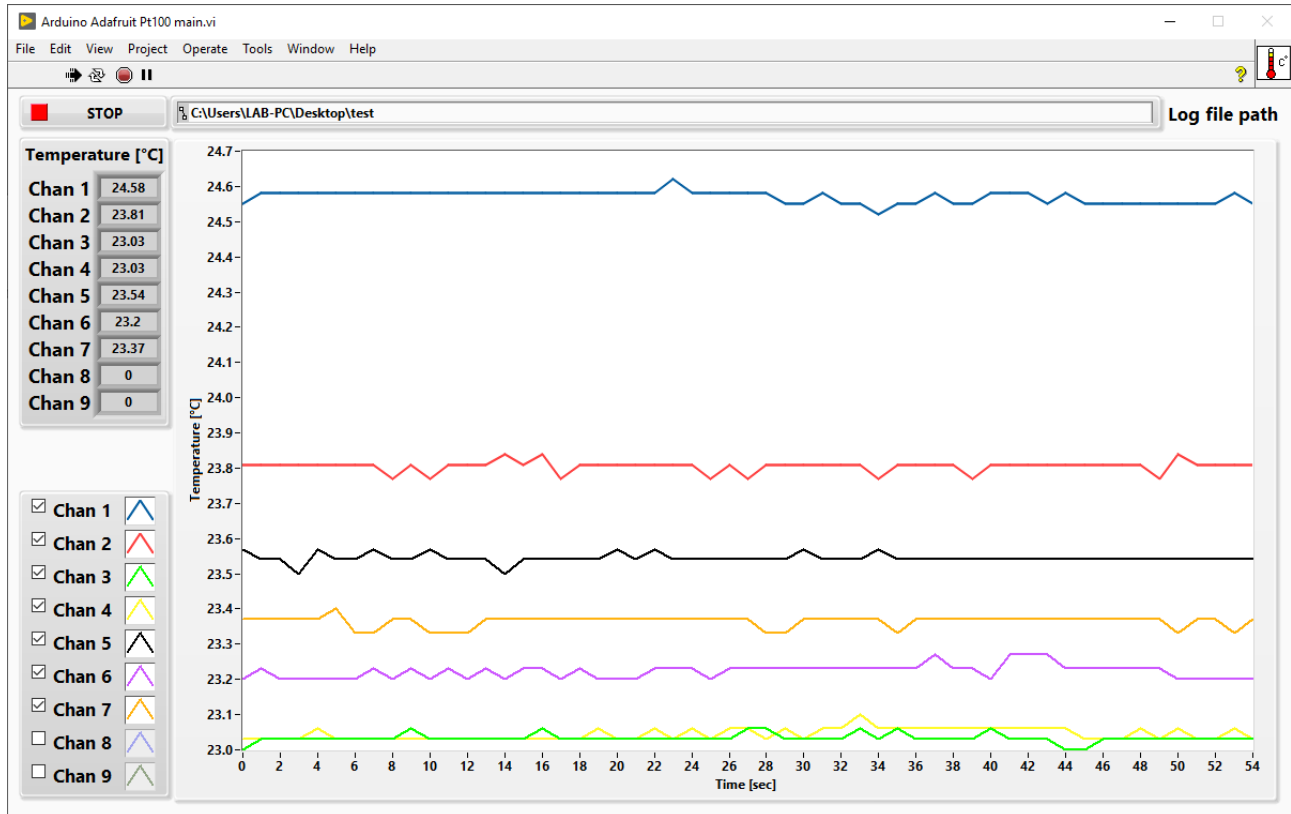


Figure 25. LabVIEW graphical user interface to visualize and log the temperature measurements from the Arduino UNO setup.

This LabVIEW interface can be found attached to this document (folder: **LabVIEW GUI for temperature measurement with Arduino**; file: **Arduino Adafruit Pt100 main.vi**).

Before using the LabVIEW interface, make sure that you have closed the Arduino IDE.

7 Troubleshooting

As this setup requires a substantial amount of preparation, a troubleshooting guide is given to help with the most common issues. The troubleshooting guide is from most to least common problems, so start from the top and work your way down.

Tip: If you have multiple sensors connected to a single Arduino UNO, and one of the amplifiers with a Pt 100 sensor works correctly, use this to test if the problem is with the circuit on the breadboard or with the amplifier and Pt 100 sensor.

7.1 Incorrect wiring

The most common problem encountered when setting up Arduino with breadboards is incorrect wiring. As there are many wires, it can be easy to lose the overview. Start from the beginning and work your way through the setup as described in the previous section. Even if a single pin is misplaced, the setup will not work.

7.2 Insufficient depth of pins in breadboard

A simple problem to fix is when wires in the breadboard are not entirely connected to the underlying metal in the breadboard. To fix this, simply press down on the wire or pin until it reaches the underlying metal (see *Figure 26*).

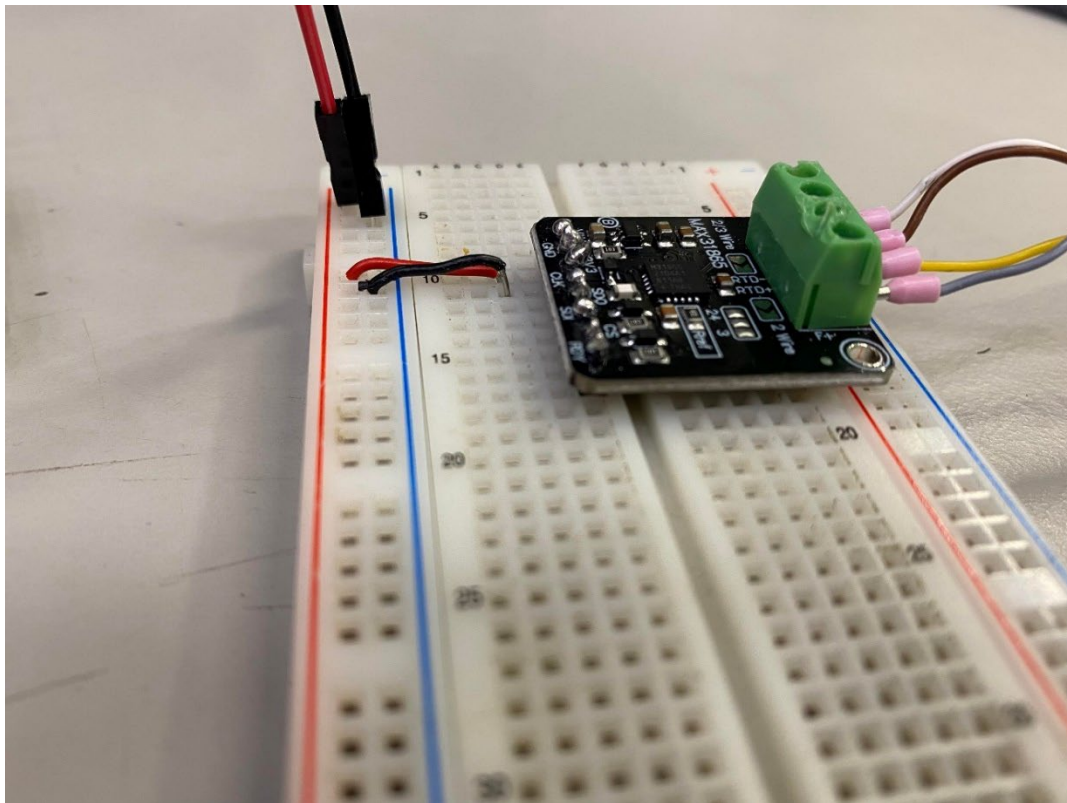


Figure 26. Example of a wire which has not been completely inserted into the breadboard.

7.3 Problems with Pt 100 sensors and connection to the Adafruit MAX31865 amplifier


Two very common problems are related to wrong connection of the Pt 100 RTDs to the amplifier, and broken Pt 100 sensors. These issues can be solved at the same time. Check to see if the Pt 100 RTD works correctly with a multimeter. Use the multimeter to measure the electrical resistance and to investigate if there is continuity in the circuit. Typically, continuity and resistance measurement share the same spot. The continuity symbol is denoted 



Figure 27. Multimeter with the knob placed to measure resistance and continuity.

After this, measure the resistance between the different wires on the Pt 100 RTD. You can measure on top of the screws on the Adafruit MAX31865, as shown in *Figure 28*.

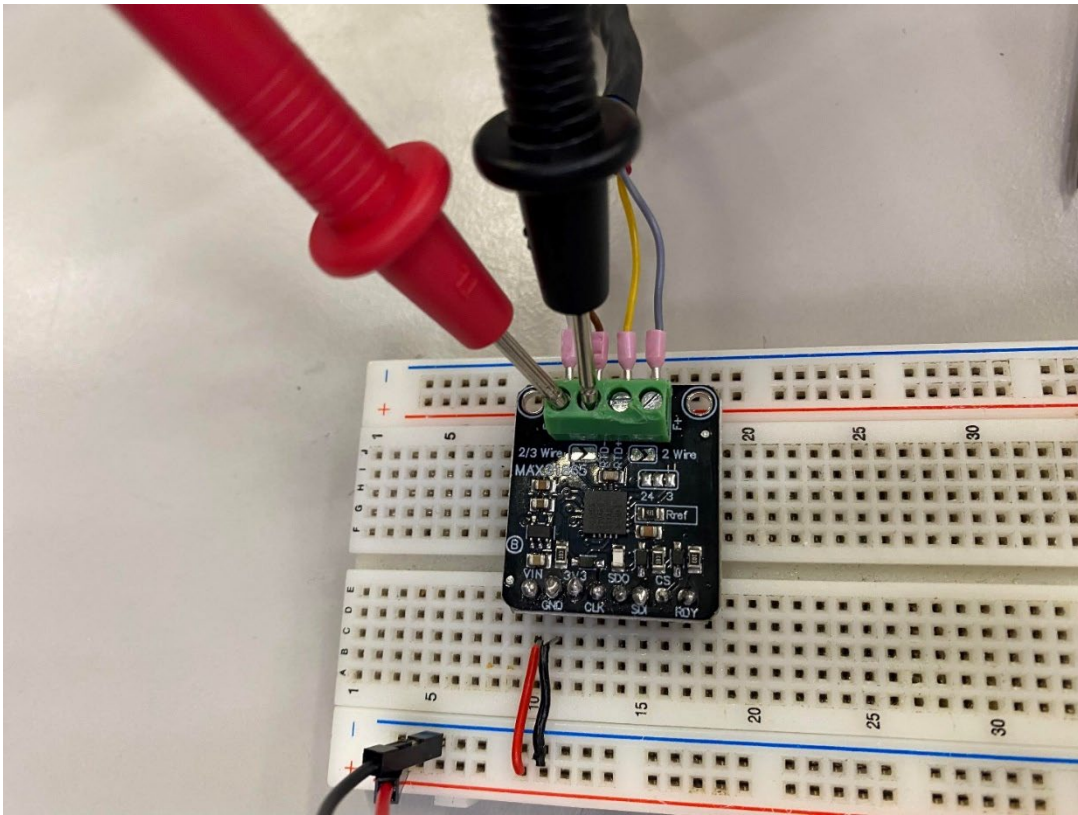


Figure 28. Illustration of how to measure if the Pt 100 sensor is connected correctly at the measurement tip. Do this for both legs of the Pt 100.

As shown in *Figure 28*, the Pt 100 sensor consists of two legs. In a four-wire configuration, there are two wires for each leg. When measuring the wires on the same leg, there is continuity if they are connected correctly. If there is continuity you will hear a high-pitched sound. On the amplifier, the wires for each leg should be in pairs, such that the wire for the same leg is next to each other on one side or the other.

It is not important which leg is connected to which spot on the amplifier, as long as the wire for each leg is next to each other.

When measuring across the Pt 100, meaning a wire from each leg, the resistance should be above 100 Ohms (if the sensor is at ambient temperature above 0 °C).

If both criteria are fulfilled, the Pt 100 works as intended and is connected correctly to the amplifier. If not, the wires might not be connected correctly.

7.4 No power connected to Adafruit MAX31865

If the Adafruit MAX31865 amplifier is not supplied with power, it will not work. To check if they are supplied with power, take a multimeter, and turn the knob to measure DC voltage (see *Figure 29*).



Figure 29. Multimeter with the knob dialed to measure DC voltage.

After that, take the positive (**red**) handle of the multimeter and place it on the **5V** pin on the Adafruit MAX31865, and the negative (**black**) handle of the multimeter and place it on the ground pin (see *Figure 30*).

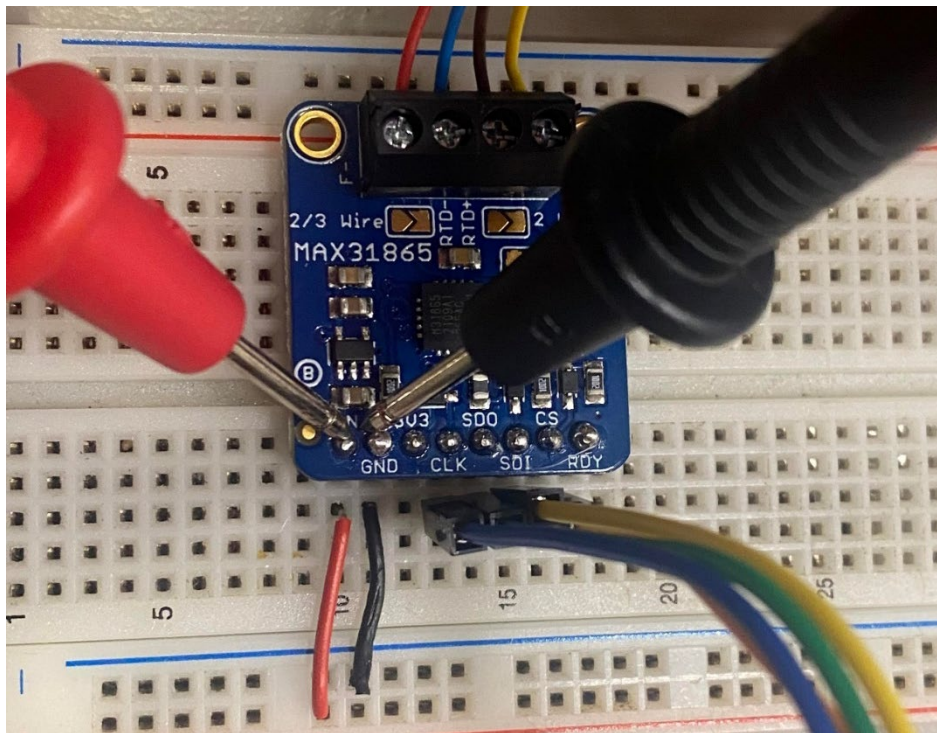


Figure 30. Illustration of how to measure DC voltage on the Adafruit 31865 amplifier.

If the multimeter shows approximately 5 V, it is powered up correctly. If this is not the case, it will not be powered and will not work. To fix this, check if the wires going to the ground and **5V** pin are on the same column on the breadboard as the corresponding pins on the amplifier. If they are, check to see if the row is powered from the (+) and (-) lines on the sides of the breadboard.

If this does not work either, the power supplied from the Arduino to the breadboard is not working correctly, and the Arduino could be broken. When the Arduino is powered on, the ON LED should be bright green.

7.5 Malfunctioning amplifier

If the wiring on the breadboard is correct and the Pt 100 sensors work correctly, check if the amplifier works correctly. The first thing to verify is if the chip emits a significant amount of heat. Quickly put a thumb, or another body part, on the chip. If it is broken, it will burn your finger.

Specifically, the chip above the V_{in} is known to have this issue and being a malfunctioning chip (see *Figure 31*).

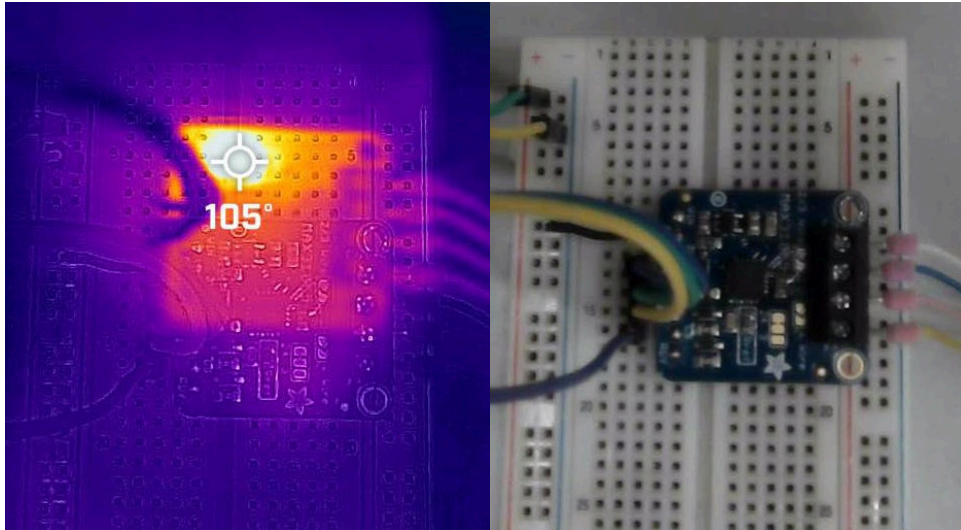


Figure 31. Thermography of malfunctioning chip.

7.6 Short-circuiting of the Arduino UNO

If the Arduino is connected to the computer with the USB, the ON LED should be bright green. If this is not the case, the Arduino might be short-circuited. This happens when the **5V** pin and the **GND** pin are connected to the same column on the breadboard (see example in 32). Notice that the ON LED is not turned on.

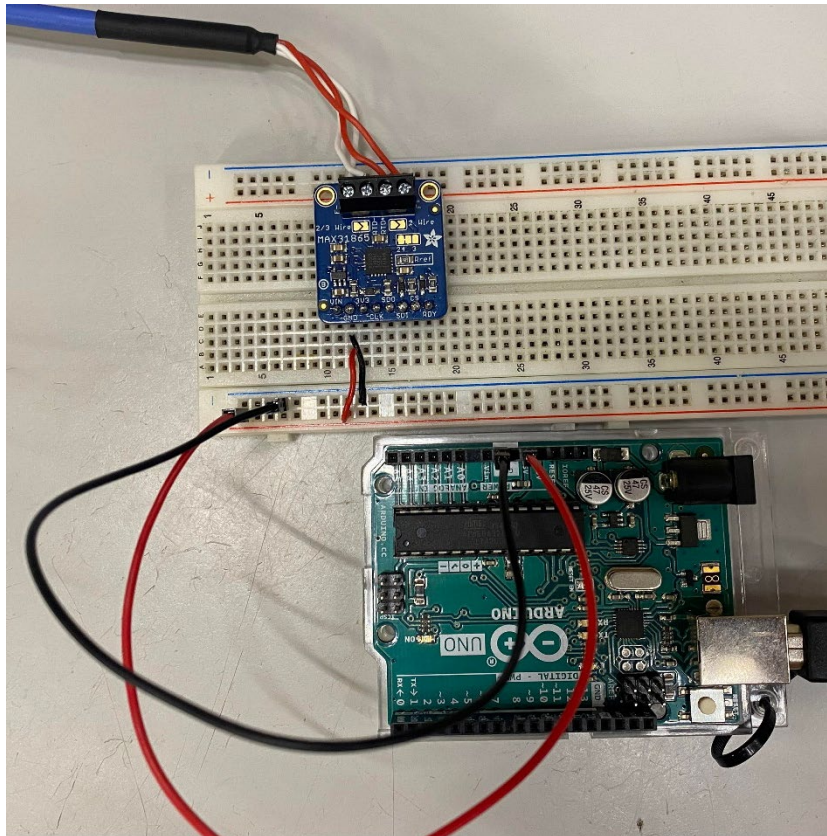


Figure 32. Short-circuited Arduino UNO. Notice that the ON LED is not bright green, as it should be when powered up correctly.

To fix this, simply power up the amplifier correctly without short-circuiting the power supply line from the Arduino.

7.7 When the troubleshooting guide fails

If the different troubleshooting steps fail, try to close and open the Serial Monitor (where the temperature data is displayed). If this fails as well, close and open the script and try again. If this does not work either, try the other troubleshooting steps again.

7.8 Code error in Arduino IDE

Some errors might also occur when trying to compile the code to be uploaded onto the Arduino board.

7.8.1 Board at COM# is not available

When changing computer and connecting the Arduino setup to a new computer, it is common that the COM port number of the Arduino board changes. To fix the problem, go to Tools → Port: → and choose the correct COM port.

Appendix A: Code to perform temperature measurements with Arduino

To perform the temperature measurements, copy this code into the Arduino IDE and run the code:

```
1  #include <Adafruit_MAX31865.h>
2
3  /*****
4  *****
5  * This is a script developed for measuring temperature using Arduino UNO and Adafruit
6  MAX31865 with Pt 100's.
7  * Uses the library for the Adafruit PT100/P1000 RTD Sensor w/MAX31865
8  ----> https://www.adafruit.com/products/3328
9
10 * Author: Martin Veit, Aalborg University
11 * For bugs, improvements or anything else related to this script and setup, contact me at
12 mveit@build.aau.dk
13
14 * INSTRUCTIONS:
15 * Follow the guide for how to assemble the setup on Moodle (Insert link)
16 * In this script, define the number of sensors used in the function "num_sensors_func"
17 (Maximum of 9 sensors)
18 * At the last line, define the delay wanted in milliseconds.
19 * To upload code on arduino, press ctrl + U or click on Sketch -> Upload
20 * To run script, press ctrl + shift + M or click on Tools -> Serial Monitor
21 */
22 /*****
23 *****
24
25 //////////////////////////////////// DEFINE NUMBER OF SENSORS USED
26 ////////////////////////////////////
27 int num_sensors_func(){
28     int num_sensors = 7;
29     return num_sensors;
30 }
31 ////////////////////////////////////
32 ////////////////////////////////////
33
34 // The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000
35 #define RREF 430.0
36 // The 'nominal' 0-degrees-C resistance of the sensor
37 // 100.0 for PT100, 1000.0 for PT1000
38 #define RNOMINAL 100.0
39
40 // The input channels used in descending order (Channel 1 is digital input 10, channel 2 is digital
41 input 9 etc.)
42 Adafruit_MAX31865 thermo1 = Adafruit_MAX31865(10);
43 Adafruit_MAX31865 thermo2 = Adafruit_MAX31865(9);
```

```

38 Adafruit_MAX31865 thermo3 = Adafruit_MAX31865(8);
39 Adafruit_MAX31865 thermo4 = Adafruit_MAX31865(7);
40 Adafruit_MAX31865 thermo5 = Adafruit_MAX31865(6);
41 Adafruit_MAX31865 thermo6 = Adafruit_MAX31865(5);
42 Adafruit_MAX31865 thermo7 = Adafruit_MAX31865(4);
43 Adafruit_MAX31865 thermo8 = Adafruit_MAX31865(3);
44 Adafruit_MAX31865 thermo9 = Adafruit_MAX31865(2);
45
46 void setup() {
47   int num_sensors;
48   num_sensors = num_sensors_func();
49   Serial.begin(115200);
50
51
52
53   if (num_sensors >= 1){
54     thermo1.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
55   }
56   if (num_sensors >= 2){
57     thermo2.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
58   }
59   if (num_sensors >= 3){
60     thermo3.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
61   }
62   if (num_sensors >= 4){
63     thermo4.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
64   }
65   if (num_sensors >= 5){
66     thermo5.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
67   }
68   if (num_sensors >= 6){
69     thermo6.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
70   }
71   if (num_sensors >= 7){
72     thermo7.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
73   }
74   if (num_sensors >= 8){
75     thermo8.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
76   }
77   if (num_sensors >= 9){
78     thermo9.begin(MAX31865_4WIRE); // set to 2WIRE or 4WIRE as necessary
79   }
80 }
81 void loop() {
82   int num_sensors;
83   num_sensors = num_sensors_func();
84
85   if (num_sensors >= 1){
86     Serial.print("Ch.1 temp = "); Serial.println(thermo1.temperature(RNOMINAL, RREF));
87   }

```



```

88     if (num_sensors >= 2){
89         Serial.print("Ch.2 temp = "); Serial.println(thermo2.temperature(RNOMINAL, RREF));
90     }
91     if (num_sensors >= 3){
92         Serial.print("Ch.3 temp = "); Serial.println(thermo3.temperature(RNOMINAL, RREF));
93     }
94     if (num_sensors >= 4){
95         Serial.print("Ch.4 temp = "); Serial.println(thermo4.temperature(RNOMINAL, RREF));
96     }
97     if (num_sensors >= 5){
98         Serial.print("Ch.5 temp = "); Serial.println(thermo5.temperature(RNOMINAL, RREF));
99     }
100    if (num_sensors >= 6){
101        Serial.print("Ch.6 temp = "); Serial.println(thermo6.temperature(RNOMINAL, RREF));
102    }
103    if (num_sensors >= 7){
104        Serial.print("Ch.7 temp = "); Serial.println(thermo7.temperature(RNOMINAL, RREF));
105    }
106    if (num_sensors >= 8){
107        Serial.print("Ch.8 temp = "); Serial.println(thermo8.temperature(RNOMINAL, RREF));
108    }
109    if (num_sensors >= 9){
110        Serial.print("Ch.9 temp = "); Serial.println(thermo9.temperature(RNOMINAL, RREF));
111    }
112
113    Serial.println();
114    delay(500);
115 }

```

References

- [1] Adafruit MAX31865 RTD PT100 or PT1000 amplifier. Adafruit Learning System. (n.d.). Retrieved September 6, 2022, from <https://learn.adafruit.com/adafruit-max31865-rtd-pt100-amplifier/overview>
- [2] Hicham Johra (2020). Assembling temperature sensors: thermocouples and resistance temperature detectors RTD (Pt100). DCE Lecture Notes No. 78. Aalborg University, Department of Civil Engineering. <https://doi.org/10.54337/aau449755797>.

Recent publications in the Technical Report Series

Hicham Johra. Thermal properties of common building materials. DCE Technical Reports No. 216. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra. Project CleanTechBlock 2: Thermal conductivity measurement of cellular glass samples. DCE Technical Reports No. 263. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra. Cleaning Procedure for the Guarded Hot Plate Apparatus EP500. DCE Technical Reports No. 265. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra. Long-Term Stability and Calibration of the Reference Thermometer ASL F200. DCE Technical Reports No. 266. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra, Olena K. Larsen, Chen Zhang, Ivan T. Nikolaisson, Simon P. Melgaard. Description of the Double Skin Façade Full-Scale Test Facilities of Aalborg University. DCE Technical Reports No. 287. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra. Overview of the Typical Domestic Hot Water Production Systems and Energy Sources in the Different Countries of the World. DCE Technical Report No. 288. Department of Civil Engineering, Aalborg University, 2019.

Hicham Johra. Thermal Properties of Building Materials - Review and Database. DCE Technical Report No. 289. Department of the Built Environment, Aalborg University, 2021.

Hicham Johra. Instant District Cooling System: Project Study Case Presentation. DCE Technical Reports No. 290. Department of the Built Environment, Aalborg University, 2021.

Hicham Johra. Performance overview of caloric heat pumps: magnetocaloric, elastocaloric, electrocaloric and barocaloric systems. DCE Technical Reports No. 301. Department of the Built Environment, Aalborg University, 2022.

M. Veit, H. Johra. Experimental Investigations of a Full-Scale Wall Element in a Large Guarded Hot Box Setup: Methodology Description. DCE Technical Reports No. 304. Department of the Built Environment, Aalborg University, 2022.

