Lab 3: Temperature Sensor Characteristics

Purpose

The purpose of this lab is to give you an opportunity to directly observe two characteristics of sensors with a first-order system response.The systems you will observe are using two common sensors to measure temperature – a thermistor and a thermocouple. When combined with signal conditioning and display components, these sensors become a useful temperature-measuring instrument. You will measure response times to a step input change in temperature for two cases: when the temperature suddenly increases and when the temperature suddenly decreases. You will calibrate the thermocouple using a two-point method and test your calibration by measuring body temperatures.

Learning Goals

Upon successful completion of this laboratory, you will be able to

- Describe the behavior of a first-order sensor response to a step change.
- Calculate the time constant for a first-order sensor from dynamic response output data.
- Explain the physical significance of a first-order time constant.
- Describe the practical consequences of a sensor or instrument having a large or small time constant.
- Report your measurement findings with the appropriate uncertainty or error limits.
- Calibrate the response of a thermocouple and use it to measure body temperature.

Background

One of the most common measurements is temperature. Temperature information is important in many different applications: environmental quality, industrial Processes and even measurement systems themselves, because instrument sensitivity and precision are often temperature dependent. Today you will be working with an environmental probe for temperature. A wide variety of sensors have been developed and used to sense temperatures in air, water, and other environmental media. In this activity, you will determine and compare the response times for several temperature sensors used in monitoring water temperatures.

Among the many characteristics that describe sensors, response time is an example of a dynamic characteristic that is important to consider. If you are monitoring air or water temperatures that change rapidly, you will want to select a sensor with a fast response time.

To determine the response time, we will use a step input and assume the sensor acts as a first-order system. For a first-order system, the time constant is a measure of the time it takes the system to reach 63.2% of its final value. The response time is dependent upon the tolerance band for the sensor (as defined by the manufacturer or user).

Thermocouples

Thermocouples are sensors that are used to measure temperature. Thermocouples are constructed by welding two wires of different metals together. Whenever two dissimilar metals are in contact, a very small voltage is generated across the junction. This is an example of a thermoelectric effect and is called the **Seebeck effect**. The voltage is on the order of a few millivolts and is a function of temperature. Because the voltage generated at this junction is very low, thermocouples are susceptible to electrical noise. A less-noise prone device utilizes a different type of temperature sensor known as a thermistor. Thermistors are semiconductors that change their resistance as a function of temperature. The advantage of thermocouples over thermistors, however, is that thermocouples are relatively inexpensive, easier to make, and respond more quickly to changes in temperature. That is because thermocouples have a relatively small time constant.

Calibration

Thermocouples and thermistors produce a voltage output that is proportional to the temperature they are exposed to. Therefore, it is necessary to calibrate this voltage to known temperature standards. Because thermocouples show linear sensor response behavior, it is possible to perform a two-point calibration. This means we can measure the voltage output of the sensor for two known temperatures (T_1, T_2). This allows us to measure the change in voltage output ($\Delta V = V_2 - V_1$) for the change in temperature ($\Delta T = T_2 - T_1$) and to calculate the sensor sensitivity (S) as:

$S = \frac{V_2 - V_1}{T_2 - T_1} = \frac{\Delta V}{\Delta T}$

Knowing the sensitivity allows for estimation of an unknown temperature T_x using:

$T_x = T_1 + \frac{(V(T_x) - V_1)}{S}$

Two-point calibration procedures are commonly used in measurement science. Often the calibration parameters are included in the sensor system, allowing the sensor system to calculate and output the quantity of interest. For example, the thermocouple you are working with today, comes with an amplifier and analog to digital converter, which will translate the voltage output into a calibrated temperature.

For many instruments, it is possible to manually update calibration constants and manufacturer specifications require periodic re-calibration of the system.

Materials

- Raspberry Pi Computer
- Adafruit MAX31865 Temperature Sensor Amplifier and A/D Converter (Spec Sheet)
- Thermocouple
- Breadboard and wires for connection
- Ice bath
- Hot water bath
- Digital Multimeter
- Instructor Demonstration:
  - LabQuest meter (Materials)
  - Temperature probe, stainless steel
  - Temperature probe, surface contact

Lab Exercise

> ⚠ Safety Caution
>
> **Hot water bath is boiling. Do not move it from the hot plate. Take care to avoid scalding yourself or others.**

Part I: Temperature Response Time Measurements

Connect the thermocouple sensor to the analog-to-digital converter

In this part of the lab, you will construct a temperature probe using the RPi computer and the thermocouple.

The thermocouple sensor will give an output voltage that corresponds to temperature. For this output voltage signal to understood by the RPi, it must be converted to a digital signal (the RPi does not accept analog signals, only digital). We will use an analog-to-digital converter (A/D board) to do this.

The A/D board we will be using is specific for thermocouples and can handle a wide range of types. This board (the MAX31856) will also provide the reference temperature shown in the above diagram, will convert the analog voltage signal to a digital signal, **and** will do the mathematical conversions necessary to convert voltage into a temperature reading before sending the data to the RPi. The process of determining the correct mathematical conversion is called **sensor calibration**.

You will directly measure the voltage output from the thermocouple in Part 2 of the lab and manually calculate the calibration performed by the A/D board

> ! Important: Connecting the thermocouple
>
> Connect the thermocouple to the MAX31856 A/D board.
> Annoyingly, Yellow is (+) and Red is (-).Be sure to get (+) and (-) correct.

Connect the thermocouple to the A/D board like in the figure below:

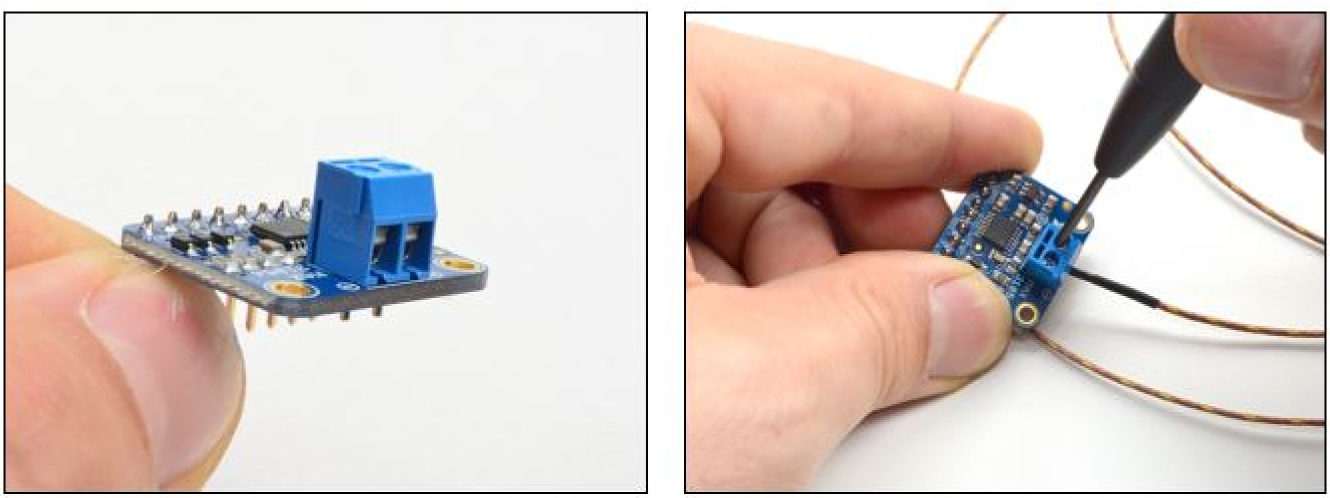

Figure 1: **Connecting the Sensor to the A/D board** - voltages from the thermocouple sensor must be converted from *analog* to *digital* to be used by the RPi

Software connection

We will be using *Python3.7* and CircuitPython in this lab. We have already used *Python* in the previous lab, but *CircuitPython* is new.

This means we have to install *CircuitPython*. We also want to make sure that we have a compatible *Python3* installation.

You will be using the command line to perform these installations and checks.

1. **Linux Terminal**: Open a terminal window by clicking on this icon in the upper left of the control bar. This opens the *command line*, which will be used to enter your commands.

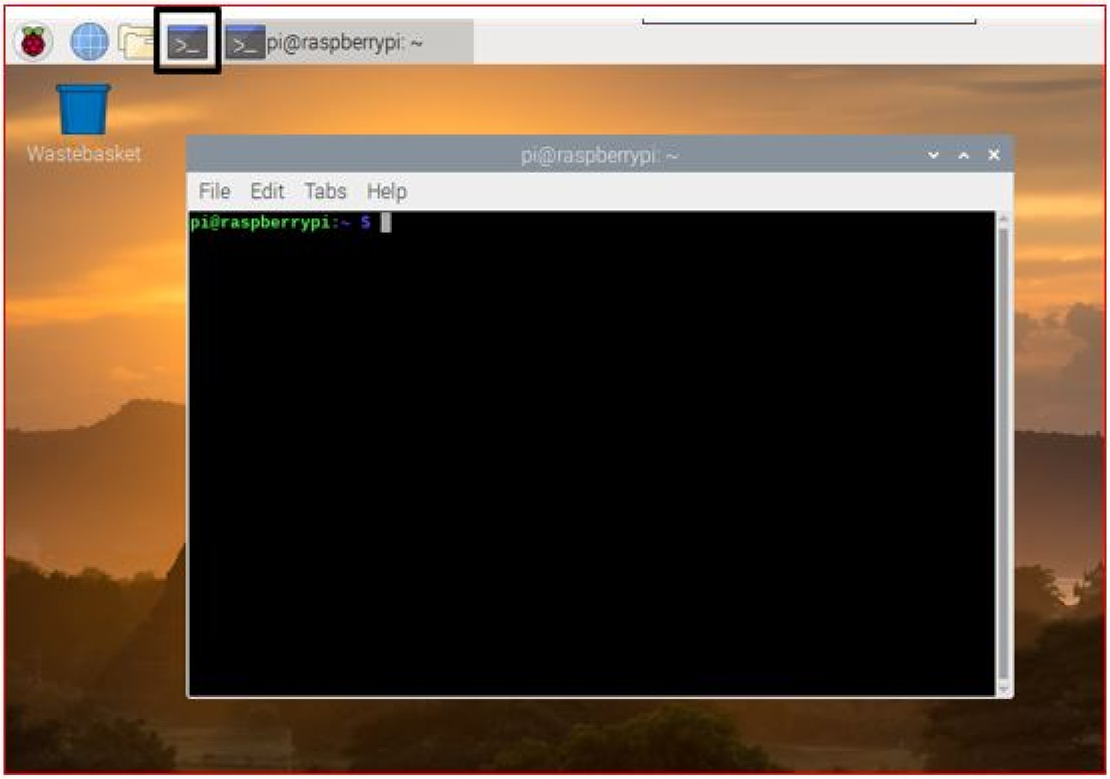

Figure 2: The Linux Terminal

2. **Update the RPi:** First we want to make sure that our system is up to date.

> Helpful terminology
>
> - `sudo`: *super user do*, which means following commands have *root*-level access. Anything executed with `sudo` can have serious consequences as system files are being changed.
> - `apt-get`: package manager for linus, which is used to install and maintain new software
> - `apt-get update`: `apt` checks for updates of installed software on the internet
> - `apt-get upgrade`: `apd` fetches and installs any available system updates

This means running the below commands will check for and install any system updates. *It is good practice to do this regularly.*

```
sudo apt-get update

sudo apt-get upgrade
```

3. **Python Maintenance**:

> **i** Helpful terminology
>
> - `pip3`: a package manager for Python v3. Similar to `apt`, `pip` is used to download and install python packages.
> - `setuptools`: `Setuptools` is a collection of enhancements to *Python* that allow developers to more easily build and distribute *Python* packages, especially ones that depend on other packages.

a) We install/ upgrade the `setuptools` expansion for `pip3` which is needed.

```
sudo pip3 install --upgrade setuptools
```

If the above does not work, we need to install `pip3` first (and then repeat the above step)

```
sudo apt-get install python3-pip
```

Software Connection - Blinka and CircuitPython

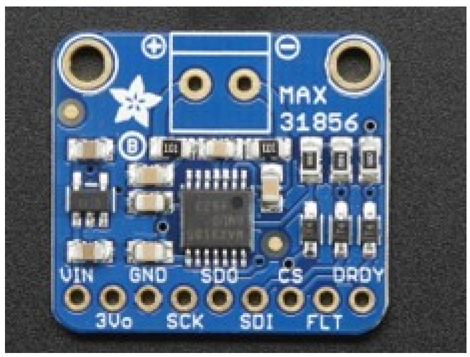

Figure 3: MAX31856 A/D Converter

The MAX31856 uses CircuitPython and SPI + I2C communication protocols. The Raspberry Pi will be using Python3.7 (which is different than CircuitPython).

In order to be able to communicate between the RPi and MAX31865, we will need to install a package called *Blinka* to connect the two systems.

> **i** Note
>
> ***CircuitPython and Python*** are similar, but not the same. CircuitPython is *lighter* version of Python – i.e. – it takes up less space – and is used on devices that have lower computing needs. CircuitPython was created by Adafruit (it is a version of MicroPython) and is used on many Adafruit products (such as our thermocouple board). Python is more robust.Blinka (from the creators of CircuitPython) allows the Python 3.7 on the RPi to communicate with the Circuit-Python scripts on the thermocouple analog-digital converter board.

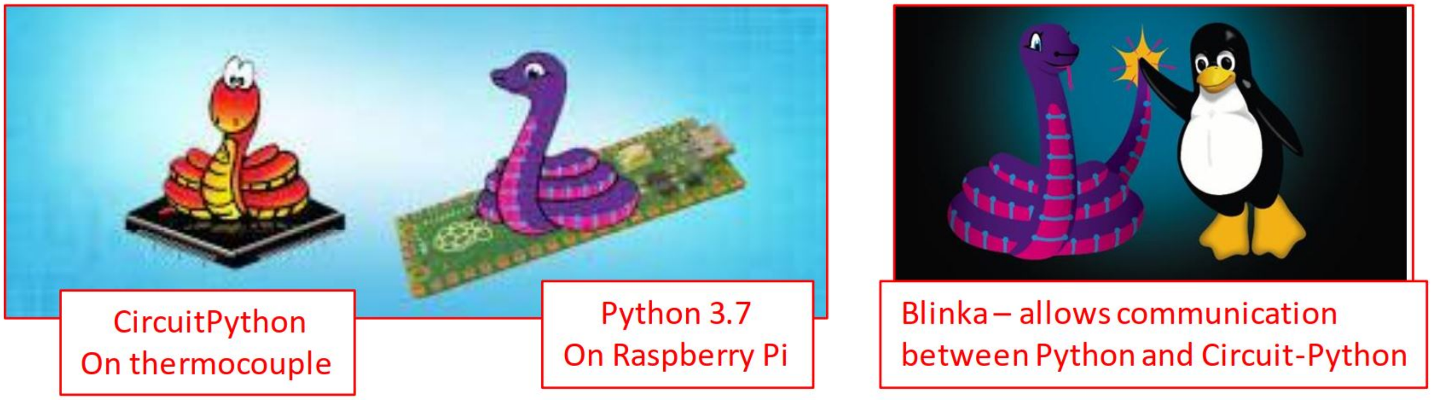

Figure 4: Blinka and Circuit Python

1. **Installing blinka**: Adafruit has put together script that configures your RPi and installs blinka. The next lines will walk you through the process.

   > **Helpful terminology**
   >
   > - `adafruit-python-shell`: A *shell*-script is a set of instructions that is run on the command line. Installing the `adafruit-python-shell` ensures that shell scripts can be executed.
   > - `wget` is a command to download (*get*) a file from the web.

   So the next two commands, will install the python-shell and then download a *python* script to install `blinka`.

   > **Tip**
   >
   > Running the commands may produce an error. Luckily the error-message will tell you what to do. Follow the instructions to resolve this.
   > **In general read the output in the terminal to ensure that things are working (or not) as planned**

   ```
   sudo pip3 install --upgrade adafruit-python-shell
   wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/raspi-blin
   ```

   > **Note**
   >
   > In the *PDF* version of the document the `wget` line cuts off. It should read:
   > wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/raspi-blinka.py

   Then we need to run the script to finish the setup.

   ```
   sudo python3 raspi-blinka.py
   ```

   It MAY (or may not) ask you to confirm. If it does, type ***Y*** to accept

2. Re-boot the Pi to finalize everything that has just been installed.

3. Wait until your RPi restarts before proceeding to the next step.

   In the meantime, here is more information on what you just installed.

   CircuitPython: open-source programming language developed by Adafruit designed to simplify experimenting and learning code on micro-controllers. It is based on MicroPython, which is a stripped-down version of

Python used commonly with micro-controllers because it doesn't require as much memory or computing power.

- **Blinka**: translates CircuitPython hardware Application Programming Interface (API) so that the hardware can communicate with the RPi (which is using Python3).
- **SPI**: Serial Peripheral Interface – communication protocol requiring 4 channels: a clock, output from computer to device, input from device to computer, and a Chip Select (for multiple devices)
- **I2C** – Inter-Integrated Circuit – communication requiring only 2 channels: in and out, but is still synchronizes data transfer between the two systems. Hardware using I2C is more complex than SPI, but wiring is simpler, especially when controlling multiple devices (only 2 wires instead of 4).

Software Connection Test

Before going further, we need to test the *Python3-Blinka-CircuitPython* install we just completed.

We can do this by running a simple program (named `ISAT 300 blinka test.py`), you can download from my github:

https://raw.githubusercontent.com/TobGerken/ISAT300/main/LabCode/ISAT 300 blinka test.py

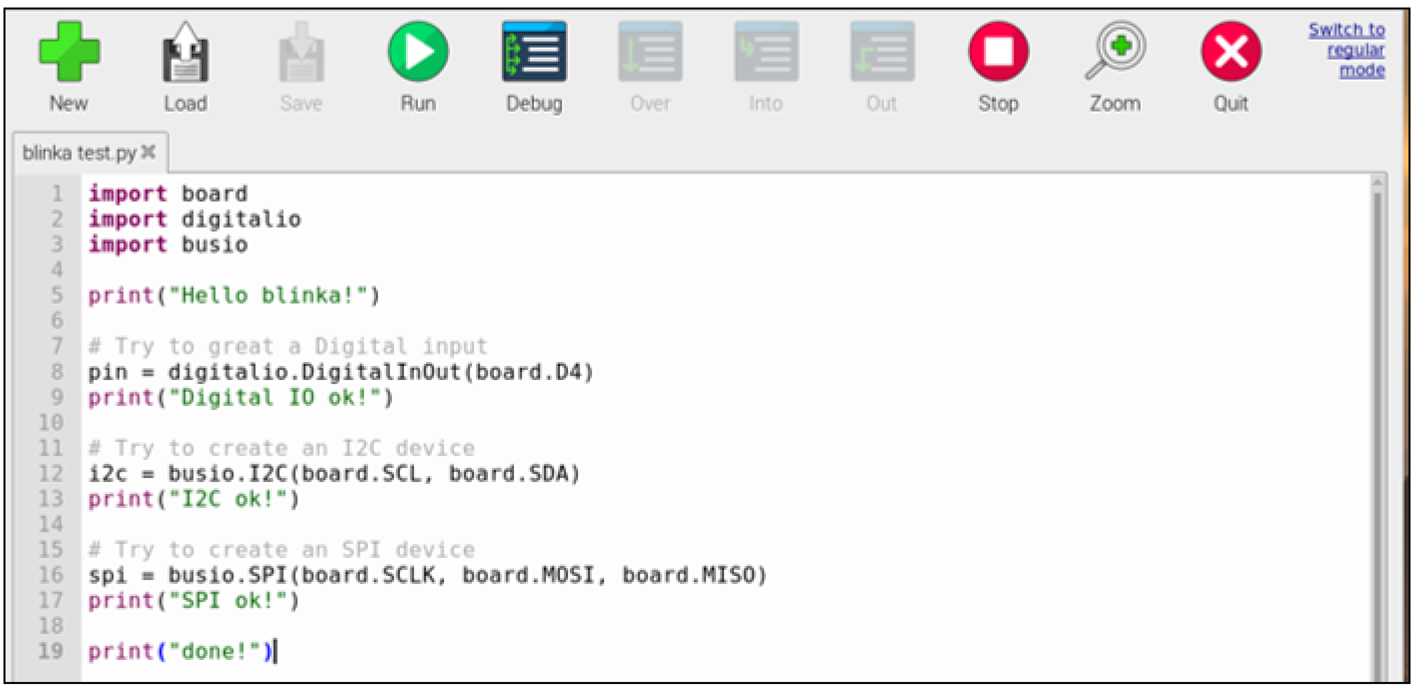

Figure 5: Blinka Test Program

If everything is working correctly, you should see this when you run it:

Figure 6: Blinka Test Output

> **i** Congratulations!
>
> You have just setup a computer so it can interface with external devices using both ***SPI*** and ***I2C*** protocols, *Python3*, and *CircuitPython*!

Software Connection Part II

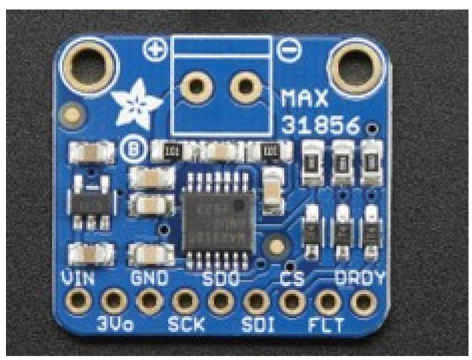

Figure 7: MAX31856 A/D Converter

The MAX31856 is more than just an analog to digital converter. The chips on this small circuit board will also provide a reference temperature for the thermocouple and are pre-programmed to convert the voltage signal into a temperature reading.

Adafruit (the company that makes this board) has also provided Python packages that we will need to install on the RPi to make the system work.

1. Open the `terminal` and execute the below command to install the *MAX31856*

   ```
   sudo pip3 install adafruit-circuitpython-max31856
   ```

After these software packages are install, we will need to physically connect the RPi to the thermocouple A/D board.

Before connection the thermocouple: Power down the Pi to avoid shorting any connections during the wiring process!

Hardware-Connection

> **!** Important
>
> Shut down the RPi before making the connection and ask an instructor to get the wiring checked before turning it back on!

Connect the RPi to the MAX31856 as shown in the diagram below.

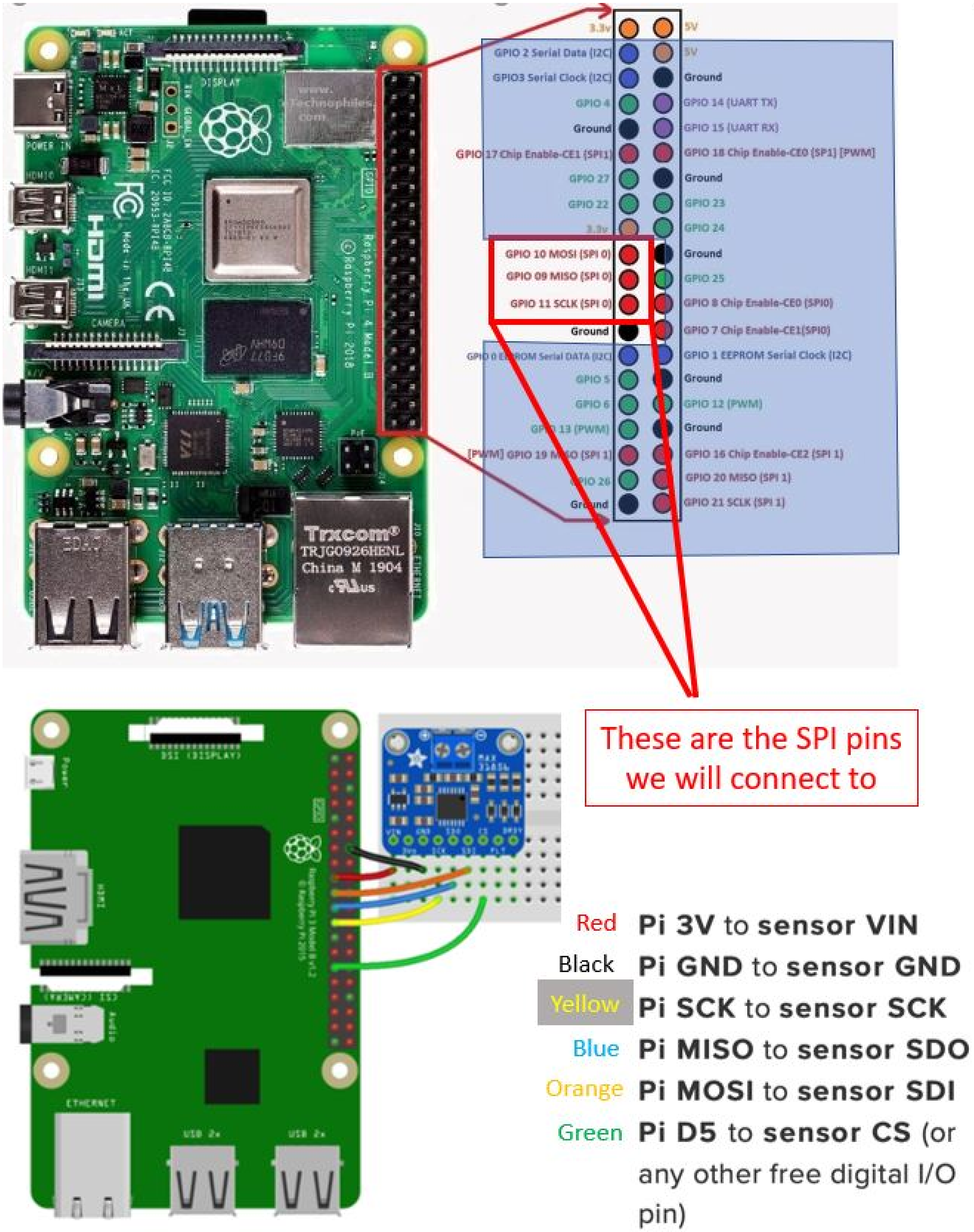

Figure 8: MAX31856 Wiring Instructions

Software Connection III: Python Code to interface with thermocouple

Tip: File and Code Hygiene

It it is good practice to

- create a dedicated folder for your work, where you save all files that belong to a given project. This means e.g. a folder for ISAT300 that contains subfolders for each lab.
- save all files to disk with sensible names
- use *typical* file extensions, such as `<filename>.py` for *Python* files.

1. Open the *Thonny* IDE and obtain the Lab 5 StarterCode from my GitHub:

   Lab 3 Starter Code

2. The starter code will set up the software connection to the thermocouple and will then read its temperature.

ISAT300_Lab5_PartI.py

```
#**********************************************************************
#   importing packages already on the RPi into this Python program   *
#**********************************************************************

import board
import busio
import digitalio
import adafruit_max31856

#***********************************
#   Setting up the thermocouple    *
#***********************************
# This part of the code is difficult to understand, but
# in essence it creates an object that allows phyton to communicate
# with the thermocouple.

# create an SPI object
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)

# allocate a Chip Select pin for the thermocouple (CS pin) and set the direction
cs = digitalio.DigitalInOut(board.D5)
cs.direction = digitalio.Direction.OUTPUT


# create a thermocouple object with the above
thermocouple = adafruit_max31856.MAX31856(spi, cs)

#*********************************************************************
#**    reading the thermocouple and printing output on screen      *
#*********************************************************************

temperature = thermocouple.temperature

print(temperature, "C")

```

Shell

```
Python 3.10.11 (C:\Users\gerkentx\AppData\Local\Programs\Thonny\python.exe)
>>>
```

Local Python 3 • Thonny's Python

Some helpful definitions to explain what's happening

- `Import board`: module from CircuitPython, provides access to board specific objects (pins)
- `Import busio`: module from CircuitPython, we will use for the SPI communication
- `Import digitalio`: module from CircuitPython, will let us set the input/output for our Chip Select pin

- Creating the object **thermocouple** allows python to query the temperature from the thermocouple object by accessing the **thermocouple.temperature** attribute, which provides the thermocouple temperature in ^{o}C.

3. Extending the code to conduct multiple measurements

   Instead of doing a single measurement, we want our thermocouple to continuously read the temperature. We can achieve this with a loop. We also want to save the temperature and the time the temperature was taken, so that we can later save this our data to a file.

   a) Write a `for-loop` to conduct 100 measurements. You can use the `range(100)` function for this.

   b) Within each loop you should:

      - read the thermocouple temperature and append it to a list of temperatures
      - get the current time and append it to a list of times.

   The below code will import the datetime module and return the current time:

```
import datetime # place before loop
datetime.datetime.now()
```

   The below code will append the value `1` to a list called `list`

```
list.append(1)
```

Tip

You should create an empty list for the temperatures and the times before the loop:

```
temperatures = []
times = []
```

4. Extending the code to write your measurements to a file:

   After the first loop finished, you should create a second loop to write your data to a csv file. You can do this as follows:

```
f = open("temperature_measurements.csv", "a")
f.write("Time,Temperature \n")
for i in range(100):
    f.write("{0},{1}\n".format(times[i], temperatures[i]))
f.close()
```

Experimental Procedure:

1. With the probes sitting in an ice water bath, start your program.
2. As quickly as possible transfer the probe to boiling water bath and gently stir for about 10 seconds.

   Warning

   Take care that you do not touch the water or have your hand very near the boiling water to avoid injury!

3. Quickly transfer the probe back to ice water bath and stir for another 10 seconds.

4. Finally, transfer probe back to boiling water and stir for another 10 seconds and wait for the data collection to end.
5. Confirm that your data was saved to a csv-file on the RPi and that 10 seconds where enough for the thermocouple to clearly equilibrate to the temperatures of the hot and icy water.
6. **Repeat** steps 3-6 for a total of 3 Runs (make sure to change the file name for saving the file)
7. Retrieve the csv files for all three runs
8. Wrap several layers of aluminum foil around your thermocouple and repeat steps 1-7 above.

Your instructor will conduct a demonstration of the same experiment using a commercial LabQuest sensor. For your analysis, you will be provided with the data from that experiment.

Part II: Thermocouple Calibration

Procedure:

1. Set the DMM to the highest sensitivity available for measuring DC voltage (400 mV).
2. Dip the thermocouple into the beaker with hot water. The water should be visibly boiling to ensure that the water temperature is 100 °C.
3. Wait for the thermocouple to read a steady voltage and record it into your lab notes
4. Transfer the thermocouple to the ice water with 0 °C and wait for the DMM reading to stabilize. Record it into your lab notebook.
5. **Repeat** steps 2-4 at least two more times.
6. Hold the thermocouple enclosed in your hand and record the measured voltage from the DMM, after the reading has stabilized

Deliverables

Your deliverable is a lab report that conforms to all lab report requirements.

Your results and discussion sections should ***at least*** contain the following items:

Part I:

1. Create graphs of the temperature vs time for each probe. Show all trials.
2. Compare the sampling rate from the RPi measurement system to the commercial LabQuest sensor and describe differences you see. How well is the shape of the curve represented in both measurement set-ups.
3. Fit exponential functions to each dataset and determine time constants and response times.
4. Create a table showing time constants for each probe and direction. Estimate the uncertainty of the time constants from your data. Show all trials.
5. Discuss the following questions:
    a) Were the time constants the same in each direction (i.e. cold-to-hot vs hot-to-cold)? Was there any evidence of hysteresis?
    b) How do your measured response times compare to the specifications provided by the manufacturers? (see spec sheets)
    c) Why are the reported time constants (see spec sheets) different for air versus water?
    d) Discuss advantages and disadvantages of high and low sampling rates and how this might relate to experimental errors and measurement uncertainty.

Part II:

1. Create a graph showing voltage vs. temperature for the thermocouple that includes a measure of uncertainty for the obtained measurements.
2. Include your table with all raw data into the report
3. Calculate the sensitivity of the thermocouple using your best estimate for the voltages at T2 and T1.
4. Answer these questions:

    a) Estimate the value of your body temperature using the calibrated thermocouple.
    b) Compare your measurement of body temperature to the average body temperature of an adult. How do they compare?
    c) What is the effective resolution of your calibrated sensor?
    d) Can you think of potential issues, if you were for example to use this calibration curve to estimate the temperature of an oil fryer ($\sim 220\,^{\circ}\mathrm{C}$)?
    e) What errors and sources of uncertainty in your measurements do you identify? Which of these do you think is most likely responsible for any differences between measured and expected body temperature? Explain.

Tips for writing your lab report

- Discuss the answers to the questions in your discussion section without copying the questions verbatim.
- Review the lecture materials about the importance about choosing appropriate models and curve fitting.
- Discuss importance of calibration for measurements *in general* and how it applies to this lab.
- Make sure to discuss sources of uncertainty and error

Revision	Description	Author
2025-02-12	Harmonized calibration with lecture content	Tobias Gerken
2025-02-04 (S25)	Update to use RPi for measurements rather than LabQuest	Tobias Gerken
2024-02-05 (S24)	Updated to Web, Clarification of deliverables, added procedures for sampling rate and data download	Tobias Gerken
	Initial Version	Chris Bachmann