Lab 5: Week 3 - Writing Data to the Cloud

Purpose

In this third part of the thermocouple lab you will add to your Python script to record data on the Could so that it can be accessed by any device, at any time, and from any location. You will also create a Graphical User Interface (GUI) to present the data clearly to the user.

Learning Goals

The goals of this portion of the Lab are to:

- Become familiar with Python programming used to control sensors and physical systems
- Recognize Cyber Security Requirements used to protect Cloud computing
- Read/Write Data to the Cloud
- Create a Graphical User Interface (GUI) to present data to users on any device

Background

Cloud computing is revolutionizing data storage and remote access. Already many entities have migrated to the Cloud, so that data, files, and photos can be accessed by any device, at any time, and from any location. Google, Amazon (AWS), Apple, and many other high-profile tech companies are making it clear that this is the way of the future because Cloud storage eliminates the need for local storage systems (i.e. - hard drives, flash storage, etc.) that can *fill up* personal computers. In addition to essentially unlimited data storage, Cloud computing also offers enhanced (centralized) security systems and more powerful data processing that can evolve over time. A potential drawback is that many Cloud storage systems are implementing *subscription services* where users are required to pay annual fees to access their data – and will need to do so for perpetuity unless they decide they no longer need to access their photos/files/etc.

We will be using Google Sheets for our Cloud Computing. To do this, we will need to activate the API. **An API is an Application Programming Interface**. We will also need to download security Credentials to authorize the RPi, create a Python script to receive a security token, and use both to allow the RPi to read and write data to your Cloud Account.

! Important

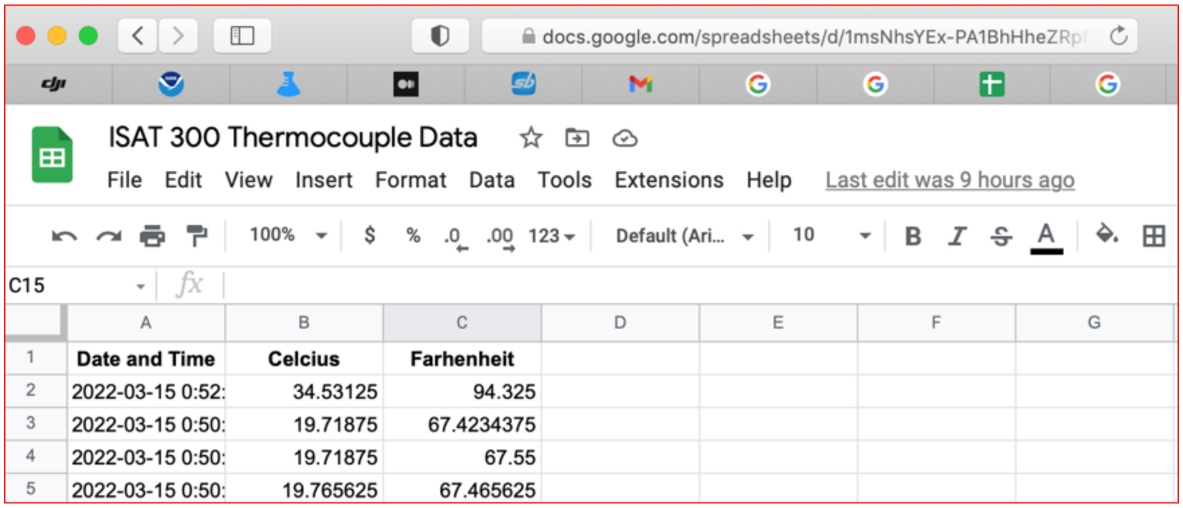
This lab assumes you have already completed Steps 1 and 2

- Step 1: Setting up a new google account
- Step 2: Establishing Cloud-based data storage - establishing Google Sheets API

These were done previously

Methods: Establishing Cloud-based data storage - Step 3: Creating the Google Sheets

1. Open the web browser on the RPi and log-in to Google Sheets using the google account with the activated API.
2. Create a new blank spreadsheet.
3. Give your spreadsheet a name by clicking on the top left corner where it says Untitled Spreadsheet.
4. Make Headers for the first three columns as shown below (Date and Time, Celsius, Fahrenheit).


docs.google.com/spreadsheets/d/1msNhsYEx-PA1BhHheZRp
ISAT 300 Thermocouple Data
File Edit View Insert Format Data Tools Extensions Help Last edit was 9 hours ago
100% $ % .0 .00 123 Default (Ari... 10 B I S A
C15

	A	B	C	D	E	F	G
1	Date and Time	Celcius	Farhenheit				
2	2022-03-15 0:52:	34.53125	94.325				
3	2022-03-15 0:50:	19.71875	67.4234375				
4	2022-03-15 0:50:	19.71875	67.55				
5	2022-03-15 0:50:	19.765625	67.465625				

Figure 1: Google Sheet with Columns

Methods: Establishing Cloud-based data storage - Step 4: Add credentials to Pi

When you set up the google sheets API you downloaded a file named `credentials.json` that was generated by Google as our first step in navigating the Google API Cyber Security.

If you don't have your file you can re-download it from the `google.cloud` Console.

You need to copy the `credentials.json` into the same folder that contains your *Python* scripts.

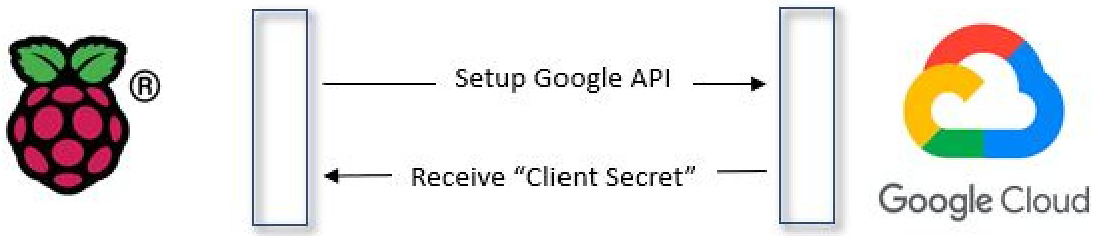
In the following steps, we will create a *Python* script that will show this credentials certificate to Google. When we run the script, it will open a web browser window and Google will then ask you to log-in to your Google account and verify that this new entity is allowed to read/write to your Google Sheets.

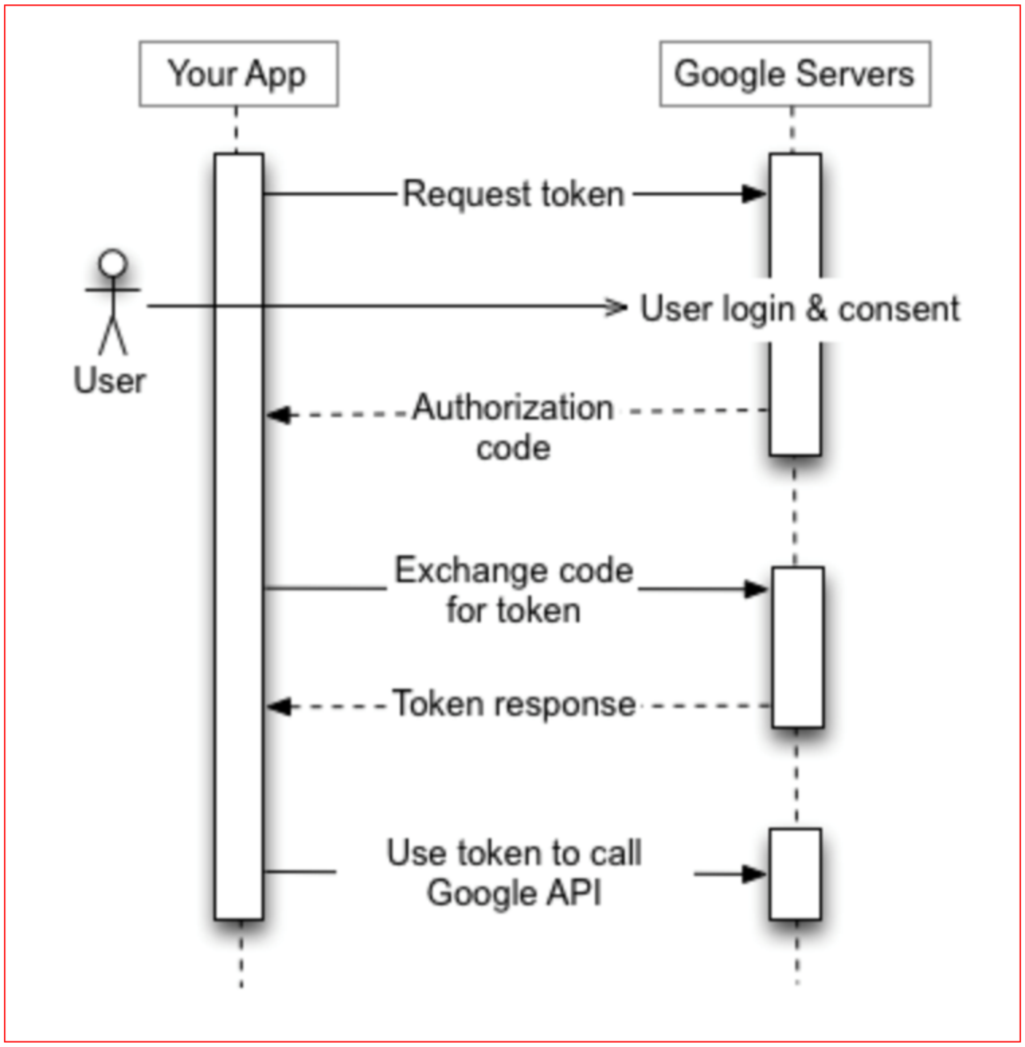
Once you acknowledge that this new entity is allowed to read/write to your Google Sheets, Google will provide you with a `token.json` file (in your downloads). You will need this `token.json` file to complete the process.

We will next update the RPi and install the ***Google Client Library*** that will be used by our Python script.

Methods: Establishing Cloud-based data storage - Step 4: Understanding the Process Flow

The *Client Secret* that we just generated using the Google API Dashboard is just the first step in a complex security process that is intended to ensure that only truly authenticated participants are able to access your data. We have completed the first two exchanges in the flow diagram below.





The next section of the Lab will setup the Raspberry Pi to interface with the Google Servers. We will install the Google library for Python Authentication and run a python script to request a token. You will need to verify that it is ok to provide the token to the RPi. We will use this token every time we want to access the Cloud to either read or write data.

Methods: Establishing Cloud-based data storage - Step 5: Preparing the RPi for Google API

Update the Raspberry Pi and install the Google Client Library using the command line in the Terminal.

1. **Update the RPi**: Power up and open the *Terminal* (command line). Execute the below commands:

```
sudo apt-get update
sudo apt-get upgrade
```

   When asked, confirm to proceed with `Y`.

2. Install the *Google Client Library* by typing (or copying) this into the command line:

   ```
   pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
   ```

Methods: Establishing Cloud-based data storage - Step 6: Python Authentication Script

1. **Open *Thonny***: We will be creating a script to download the essential security credentials and security tokens needed for permission to access Google's API and write data to the Google Cloud.

2. **Go to GitHub**: I have placed the base-code for security measures in the ISAT300 repository.

   https://github.com/TobGerken/ISAT300/tree/main/LabCode

   Copy and paste the code named `quickstart.py` into your *Thonny* development environment.

3. Save `quickstart.py` and close *Thonny*. We will be running the script from the command line.

   The `quickstart.py` file must be located in same directory as the `credentials.json` file.

   > Note
   >
   > If you need to re-download the `credentials.json` file, you can do so from the *google Cloud Console* following the last step from here.

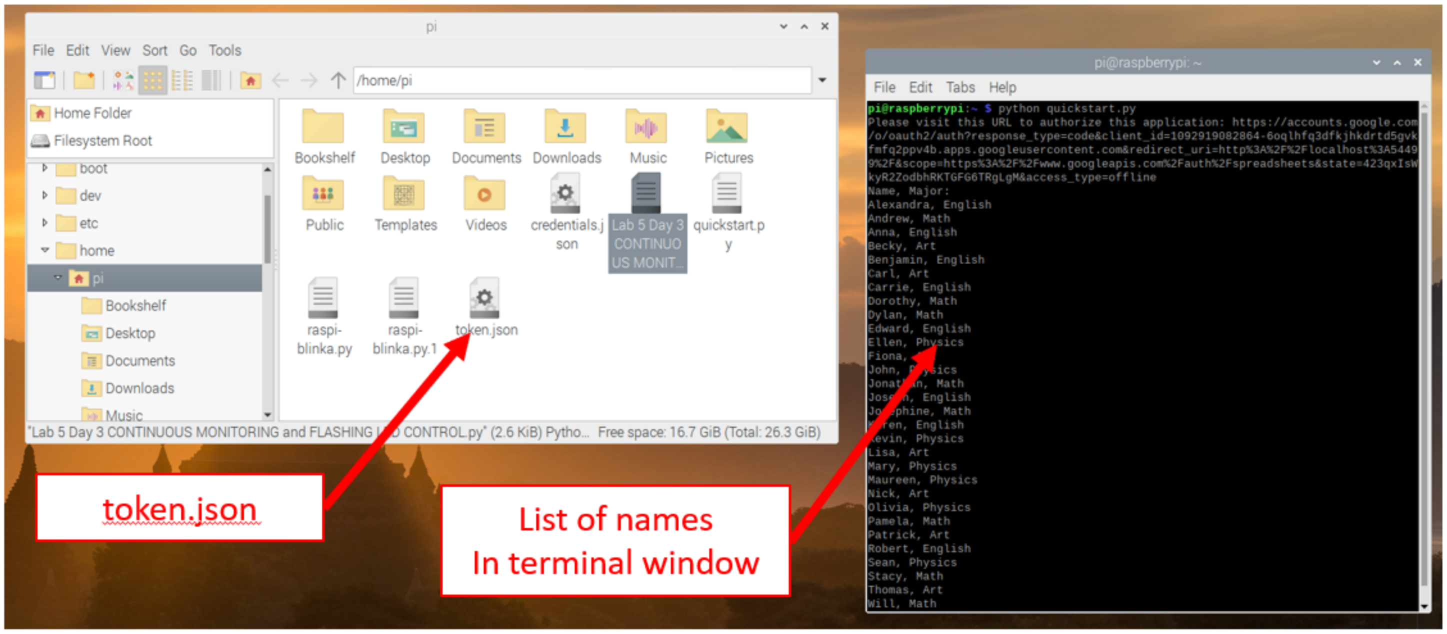
4. Go back to the *Terminal* window and run the `quickstart.py` *Python* program by typing:

   ```
   python quickstart.py
   ```

5. The `quickstart.py` program is provided by google to access the *google sheets API* you have previously set up and to retrieve the access token needed to authenticate your RPi with google.

   Running the program should prompt the web browser to open and a Google webpage to appear. ***Log-in to the Google Account that you authorized as a test user with your username and password.*** Agree to everything to allow `quickstart.py` to have access to read and write data to your Google sheets. Google will warn you that the app is not authorized and that you should only continue if the know the developer. Select Continue.

   If everything was successful, you should see a list of names in the *Terminal* window that were read from a public google sheet and a new file in the home/pi directory called `token.json`. You should now be able to read and write to Google Cloud.



```
pi@raspberrypi:~ $ python quickstart.py
Please visit this URL to authorize this application: https://accounts.google.com
/o/oauth2/auth?response_type=code&client_id=1092919082864-6oqlhfq3dfkjhkdrtd5gvk
fmfq2ppv4b.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A5449
9%2F&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fspreadsheets&state=423qxIsW
kyR2ZodbhRKTGFG6TRgLgM&access_type=offline
Name, Major:
Alexandra, English
Andrew, Math
Anna, English
Becky, Art
Benjamin, English
Carl, Art
Carrie, English
Dorothy, Math
Dylan, Math
Edward, English
Ellen, Physics
Fiona, [illegible]
John, Physics
Jonathan, Math
Joseph, English
Josephine, Math
Karen, English
Kevin, Physics
Lisa, Art
Mary, Physics
Maureen, Physics
Nick, Art
Olivia, Physics
Pamela, Math
Patrick, Art
Robert, English
Sean, Physics
Stacy, Math
Thomas, Art
Will, Math
```

Figure 2: Obtaining the access token using `quickstart.py`

Methods: Step 7 – Thermocouple hardware connection

Connect the RPi to the MAX31856 and thermocouple as you did in week 1

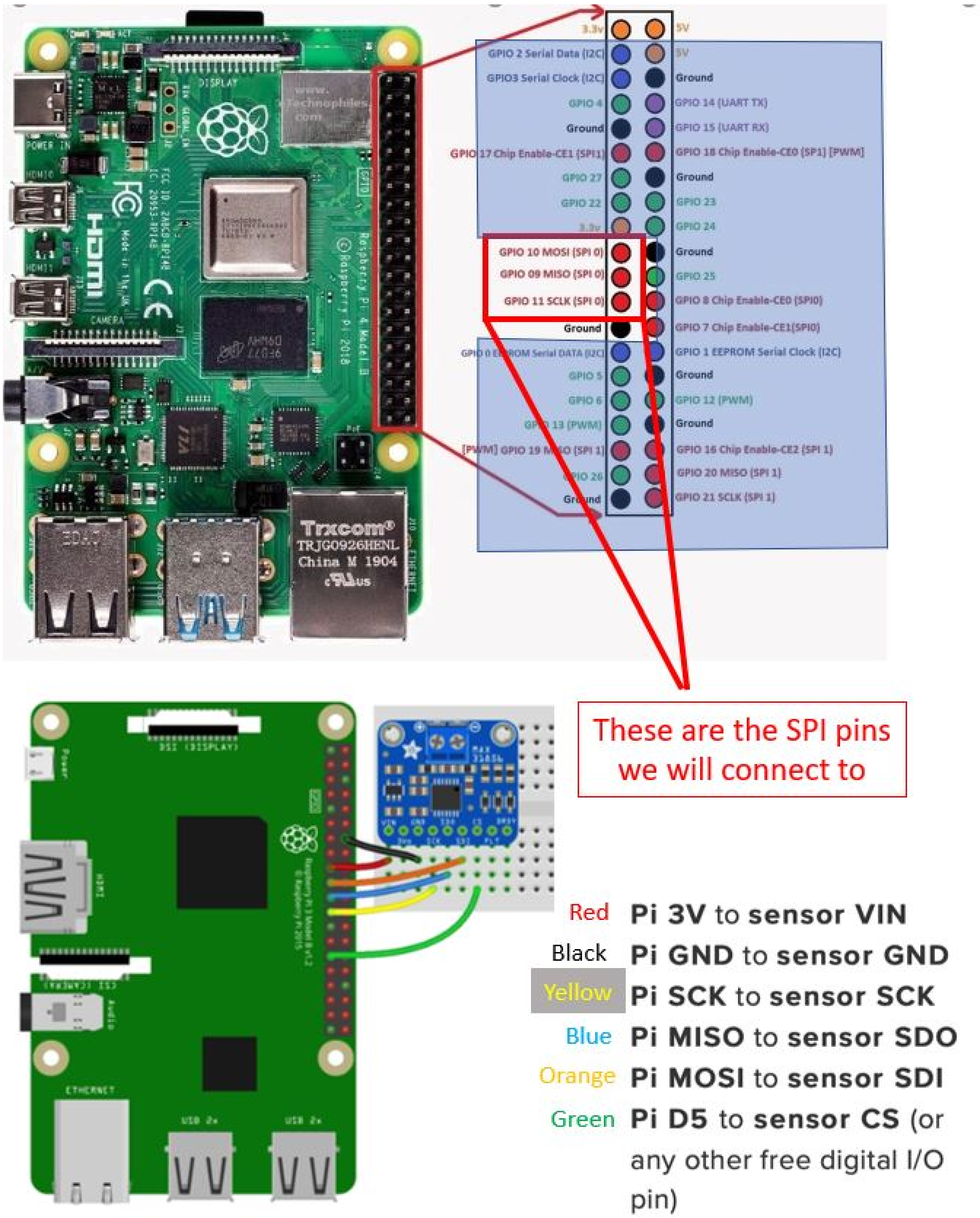

3.3v
5V
GPIO 2 Serial Data (I2C)
5V
GPIO3 Serial Clock (I2C)
Ground
GPIO 4
GPIO 14 (UART TX)
Ground
GPIO 15 (UART RX)
GPIO 17 Chip Enable-CE1 (SPI1)
GPIO 18 Chip Enable-CE0 (SP1) [PWM]
GPIO 27
Ground
GPIO 22
GPIO 23
3.3v
GPIO 24
GPIO 10 MOSI (SPI 0)
Ground
GPIO 09 MISO (SPI 0)
GPIO 25
GPIO 11 SCLK (SPI 0)
GPIO 8 Chip Enable-CE0 (SPI0)
Ground
GPIO 7 Chip Enable-CE1(SPI0)
GPIO 0 EEROM Serial DATA (I2C)
GPIO 1 EEPROM Serial Clock (I2C)
GPIO 5
Ground
GPIO 6
GPIO 12 (PWM)
GPIO 13 (PWM)
Ground
[PWM] GPIO 19 MISO (SPI 1)
GPIO 16 Chip Enable-CE2 (SPI 1)
GPIO 26
GPIO 20 MISO (SPI 1)
Ground
GPIO 21 SCLK (SPI 1)
These are the SPI pins we will connect to
Red Pi 3V to sensor VIN
Black Pi GND to sensor GND
Yellow Pi SCK to sensor SCK
Blue Pi MISO to sensor SDO
Orange Pi MOSI to sensor SDI
Green Pi D5 to sensor CS (or any other free digital I/O pin)

Figure 3: MAX31856 Wiring Instructions

Methods: Step 8 – Physical connection for the LED (optional)

You may want to connect an LED like you did in week 2. This is optional today, but will be required next week.

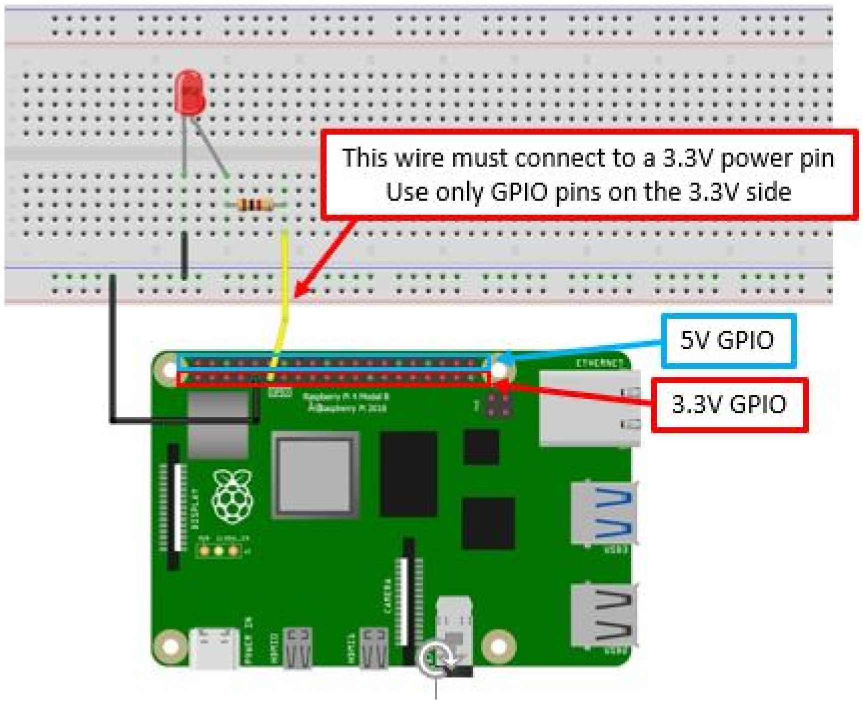

This wire must connect to a 3.3V power pin
Use only GPIO pins on the 3.3V side
5V GPIO
3.3V GPIO

Figure 4: Wiring diagram to connect the LED to the RPi for automated control

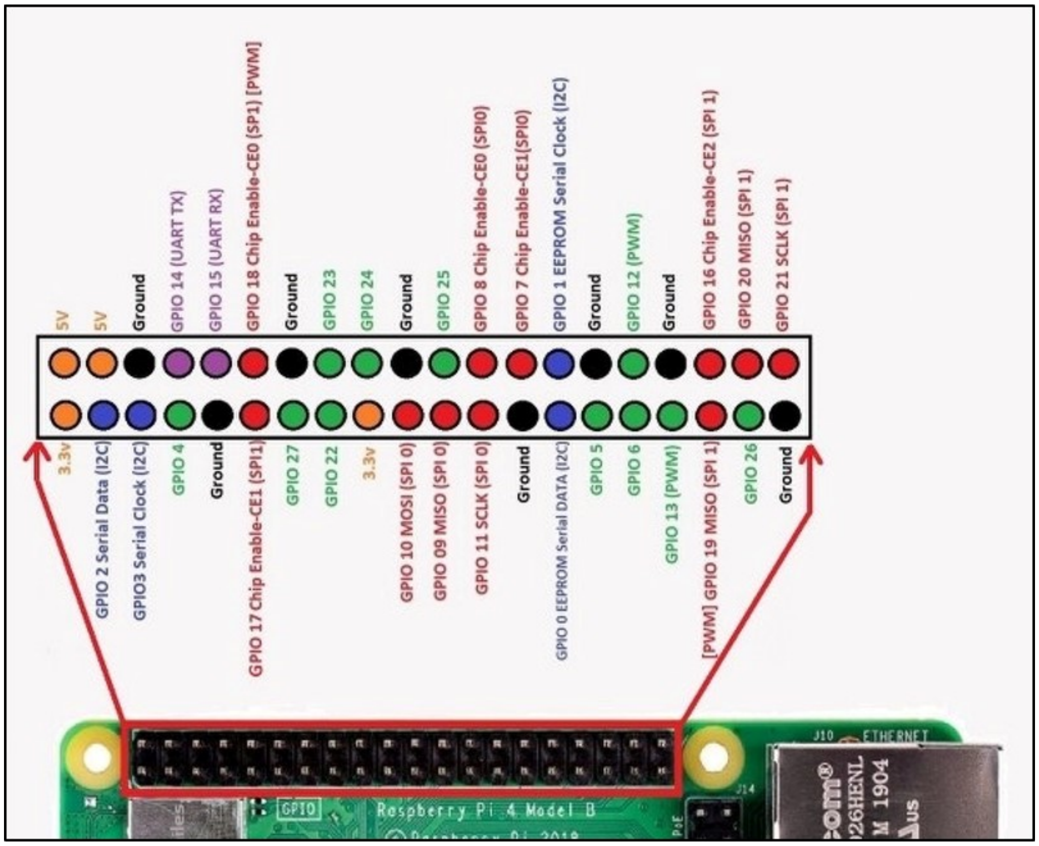


Figure 5: GPIO Overview

Methods: Step 9 – Software connection to write data to the Cloud

Note

You will need to modify the code that is being provided for today's lab. First, you must change the Spreadsheet ID to match your own Google Sheet and then you must modify the code to write the temperature data to the cloud.

1. Open the *Thonny* development environment
2. Go to GitHub and navigate to the ISAT-300 page

   https://github.com/TobGerken/ISAT300/tree/main/LabCode
3. Open the `ISAT 300 Lab 5 DAY 3 WRITING TO THE CLOUD Start.py` *Python* script. Download the file or copy it into *Thonny*.

   Note

   The starter code you downloaded contains a starter script to write data to a google sheet using the google sheets API.
   The full scope of the API can be found in the *official* API documentation
4. Open the Google Sheets spreadsheet you created. The *Spreadsheet ID* is located in the web address.

   Click on the web address for your Google Sheet and scroll until you can see the end.

   The *Spreadsheet ID* is located between two `/ ... /` just before the `/edit#gid=0`

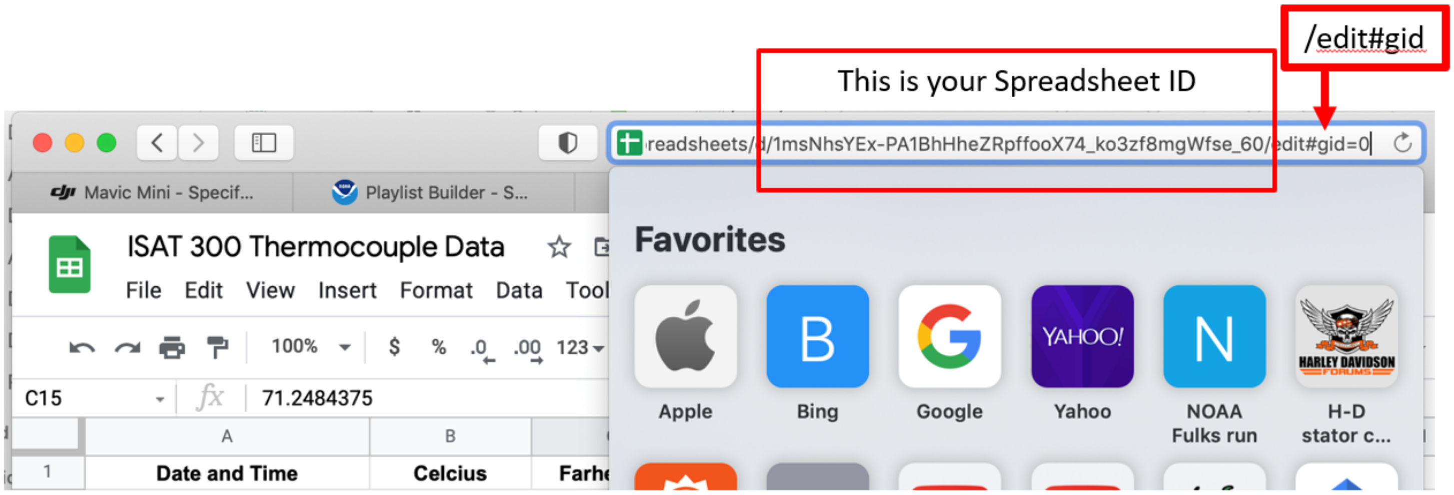
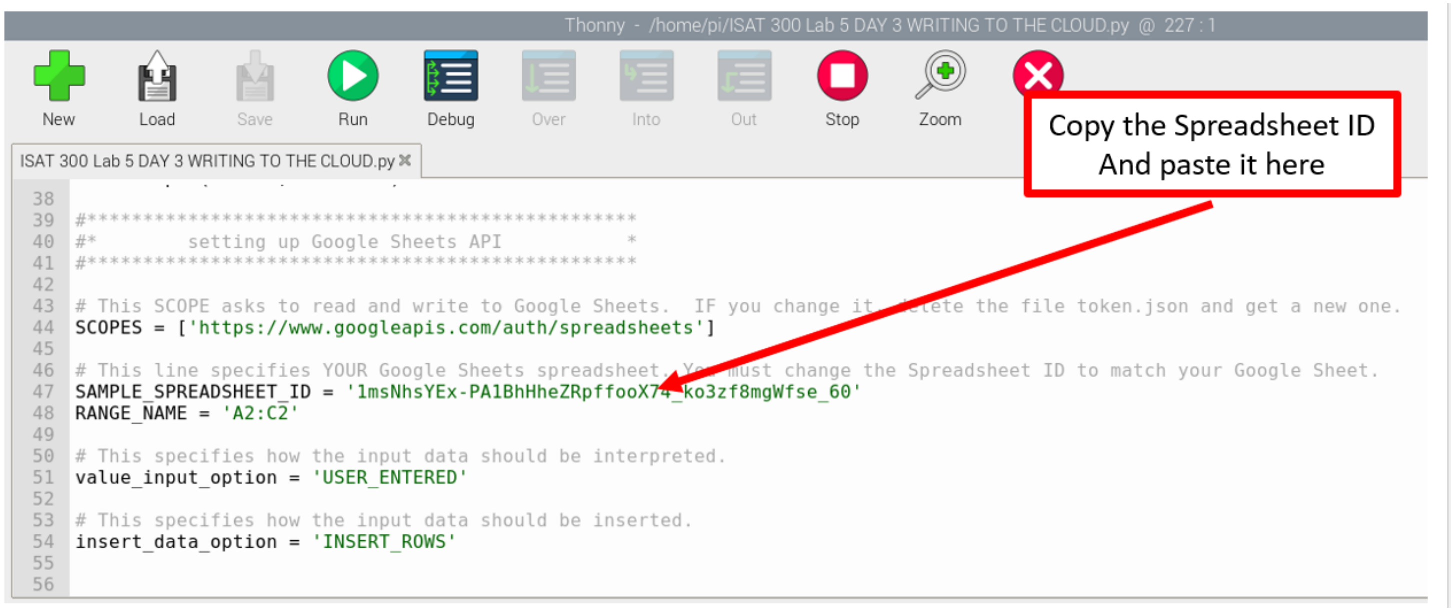

/edit#gid
This is your Spreadsheet ID
readsheets/d/1msNhsYEx-PA1BhHheZRpffooX74_ko3zf8mgWfse_60/edit#gid=0
Mavic Mini - Specif...
Playlist Builder - S...
ISAT 300 Thermocouple Data
File Edit View Insert Format Data Tool
100% $ % .0 .00 123
C15 71.2484375
A B
1 Date and Time Celcius Farhe
Favorites
Apple
Bing
Google
Yahoo
NOAA Fulks run
H-D stator c...

Figure 6: Locating the *spreadsheet ID*

5. Copy the *Spreadsheet ID* and paste it into the *Python* script in the correct location

   IF you have setup the Google Sheet to have the same three headings, you do not need to change anything else.


Thonny - /home/pi/ISAT 300 Lab 5 DAY 3 WRITING TO THE CLOUD.py @ 227 : 1
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
ISAT 300 Lab 5 DAY 3 WRITING TO THE CLOUD.py
Copy the Spreadsheet ID
And paste it here

```
#*************************************************
#*        setting up Google Sheets API          *
#*************************************************

# This SCOPE asks to read and write to Google Sheets.  IF you change it, delete the file token.json and get a new one.
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']

# This line specifies YOUR Google Sheets spreadsheet. You must change the Spreadsheet ID to match your Google Sheet.
SAMPLE_SPREADSHEET_ID = '1msNhsYEx-PA1BhHheZRpffooX74_ko3zf8mgWfse_60'
RANGE_NAME = 'A2:C2'

# This specifies how the input data should be interpreted.
value_input_option = 'USER_ENTERED'

# This specifies how the input data should be inserted.
insert_data_option = 'INSERT_ROWS'
```

Figure 7: Updating the code with the *spreadsheet ID*

6. Save the *Python* code and run it. You should see values appear in your Google Sheet.
7. The data that has appeared in the google sheet ***is not*** the thermocouple data. Examine the code you executed to find identify the location where it:
   - Sets up the connection to the google sheets API
   - Authenticates the user
   - Sets the values that are written to the google sheet
   - Actually writes the values to the google sheet

   Don't worry if you don't understand what each line does exactly. The goal is to have a broad strokes understanding

Updating the program to include your thermocouple measurements

8. Use your code from the previous week to add to the starter code. I have provided comments in the starter code that show you where you have to add certain lines. You can copy these over from the program you used in the previous lab.

   Go through your program from week 2 and identify, where in the code each of the following happens:
   - Import the packages used by the thermocouple and LED GPIO
   - Set up the LEDs
   - Set up the thermocouple
   - Modify the loop such that each iteration of the loop reads the uncalibrated thermocouple temperature
   - Applies the calibration
   - Prints uncalibrated and calibrated temperature to screen in °C and °F
   - Writes the calibrated temperature data to a `.csv` file
   - Sets the LED based on the condition of the calibrated temperature
   - Writes the calibrated temperature to the google sheet using the API

   Then copy over the code into the program to write data to the cloud. As you modify your program, make sure that:

- Initialization and setup statements - that only need to run a single time - are placed outside the loop
- You have a single loop for conducting the measurements and recording your data
- Have a final measurement and write step after the loop to ensure that a final temperature is measured and written to the csv-file as well as the google sheet.

9. Your code currently uses a `for loop` which executes a loop a fixed number of times.

   ```
   for i in range(10):
       <do something>
   ```

   You may be remembering from your programming class, that there are also `while loops` that may look like this:

   ```
   while (temp < 20):
       <do something>
   ```

   Think about the following questions:

   - What is the crucial difference between a `for` and a `while` loop?
   - How long will a while loop run?

   Replace your `for loop` with a `while loop` and set it up that the loop stops running when the calibrated temperature reaches a certain value.

   Test your code and watch data being written to the cloud.

10. Stop your code from running.

    Tip

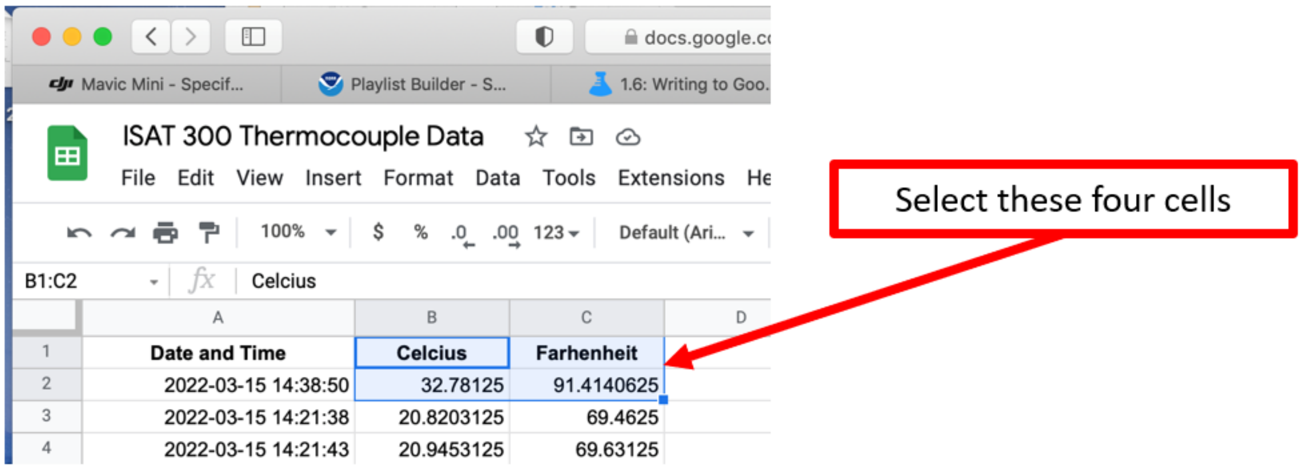
    I recommend using a `for loop` for any testing and to switch over to a while loop once your software works.
    You will need a working `while loop` next week, when we will do continuous temperature monitoring.

11. Test your code by writing 100 or so values to the cloud. If you set the measurement frequency (`time.sleep(<wait time in seconds>)`) too high, your code will produce an error message. Read the error message and try to understand why this is happening. Select an appropriate measurement frequency for your alert system.

Methods: Step 10 – Creating a Graphical User Interface (GUI)

We will create two Graphical User Interfaces in our Google Sheet for easy visualization of the data. The first will be a dial-gauge that changes in real-time. The second will be a graph of data over time.

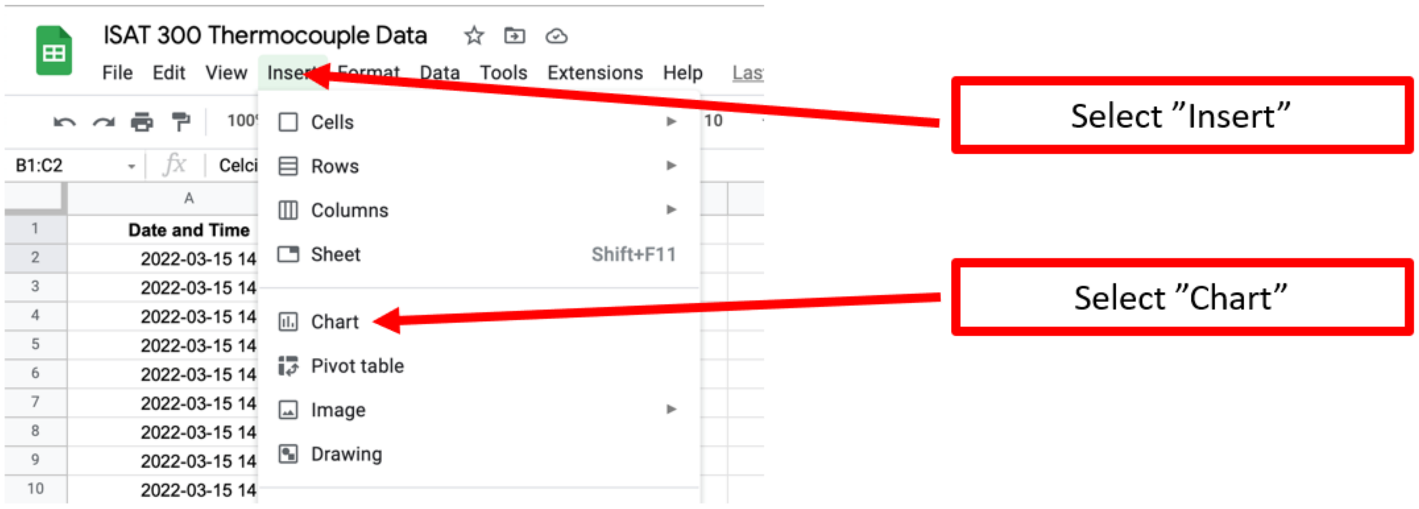
1. Stop the *Python* script and go to your Google Sheets.
2. Select the Celsius and Fahrenheit headers and the 1st two lines of data below them.



	A	B	C	D
1	Date and Time	Celcius	Farhenheit	
2	2022-03-15 14:38:50	32.78125	91.4140625	
3	2022-03-15 14:21:38	20.8203125	69.4625	
4	2022-03-15 14:21:43	20.9453125	69.63125	

Figure 8: Selecting the Data

3. Click on `Insert` from the toolbar. Select `Chart`.



	A
1	Date and Time
2	2022-03-15 14
3	2022-03-15 14
4	2022-03-15 14
5	2022-03-15 14
6	2022-03-15 14
7	2022-03-15 14
8	2022-03-15 14
9	2022-03-15 14
10	2022-03-15 14

Figure 9: Inserting the Chart

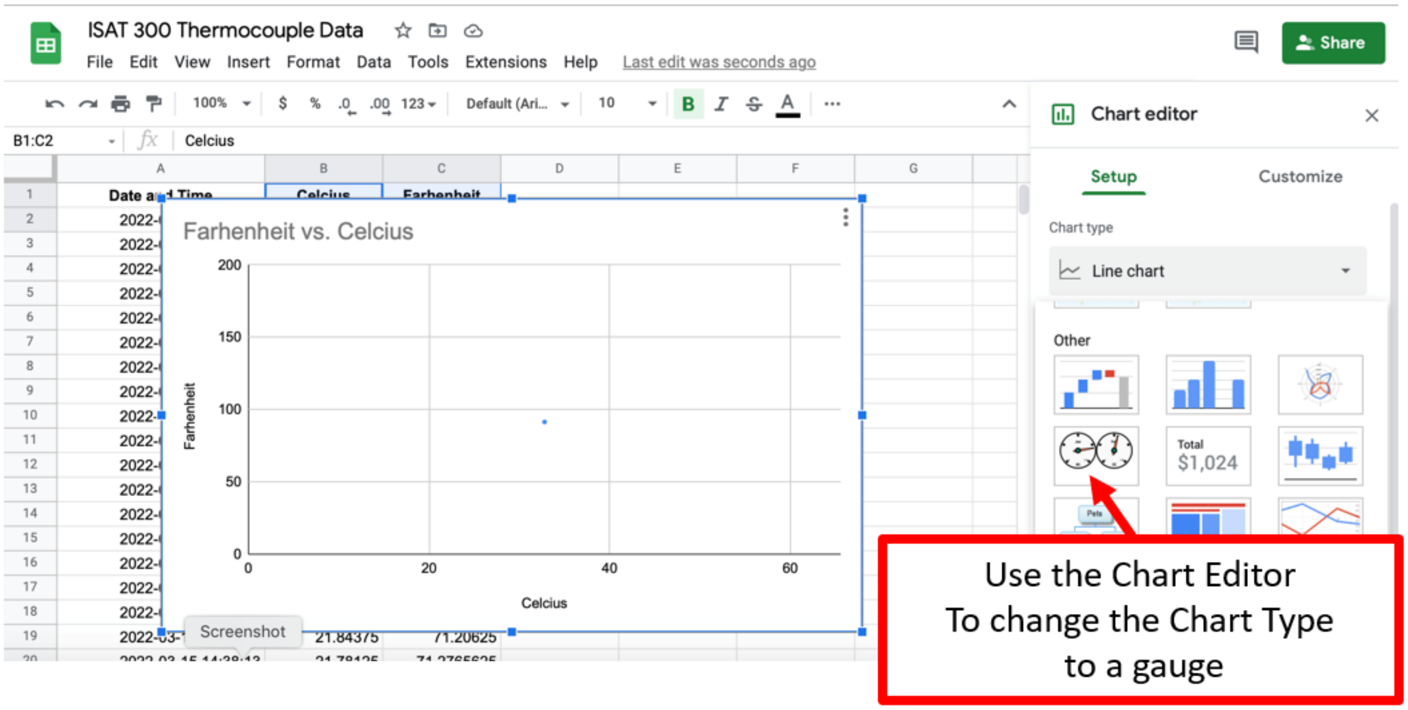


Figure 10: Setting the Chart Type to Gauge

4. Adjust the settings to make the Gauges look *good*

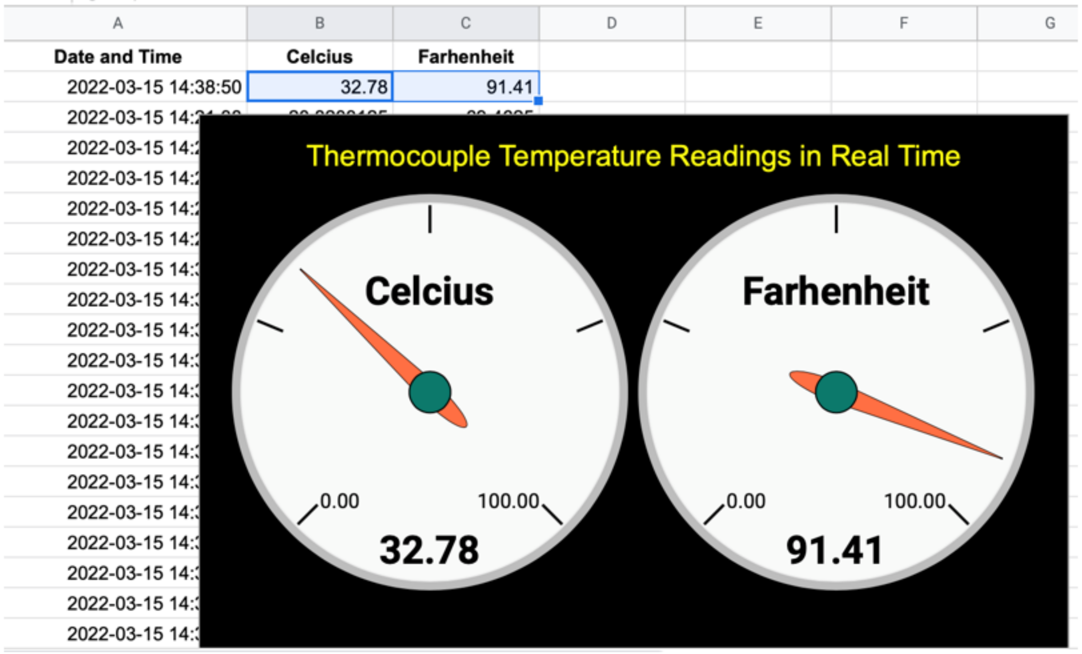
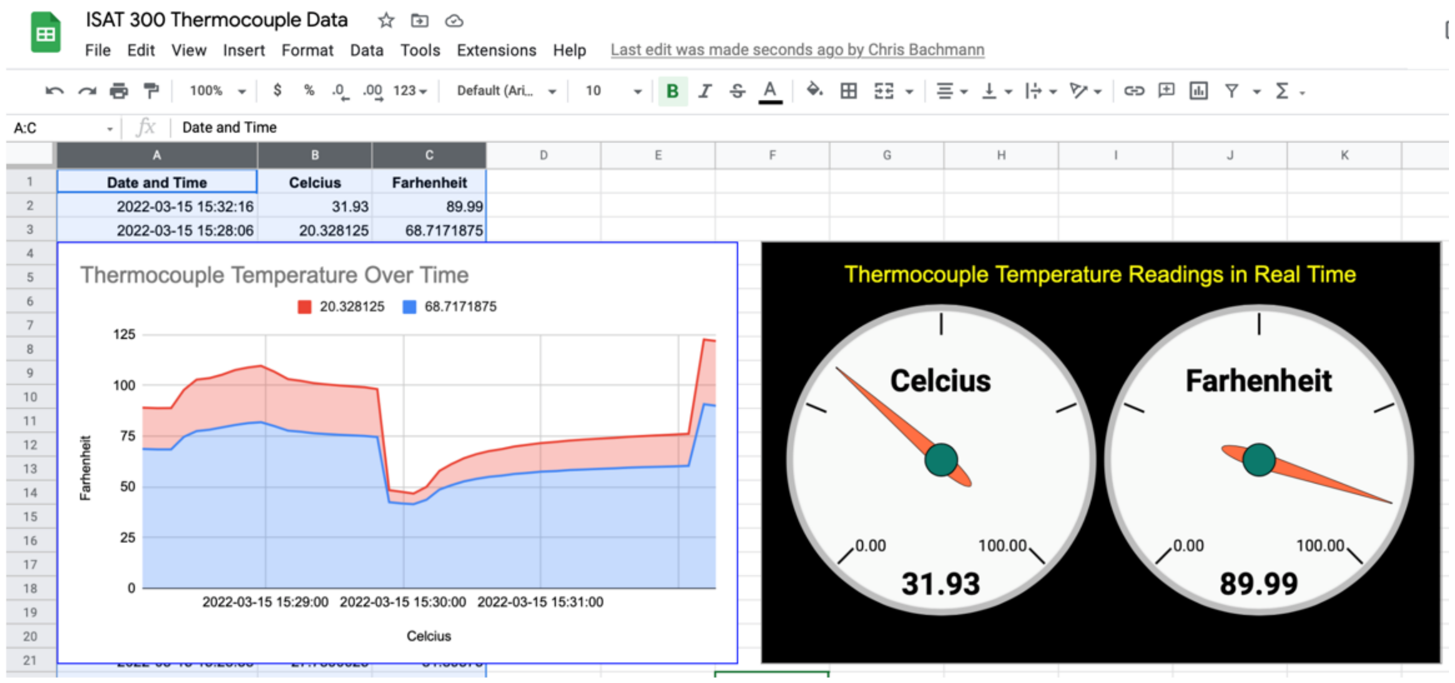


Figure 11: Adjusting the Gauges

5. Go back to *Thonny* and run the *Python* script again. You should see the gauge change in real time. (However, there will be a slight delay for the security authentication and data transfer)
6. Create a graph that changes over time by highlighting the first three columns and creating an area chart.

   Adjust the cells so the data starts with `A3` to eliminate the final data point used for the gauges.

Make this graph look *nice*.


ISAT 300 Thermocouple Data

Last edit was made seconds ago by Chris Bachmann

	A	B	C
1	Date and Time	Celcius	Farhenheit
2	2022-03-15 15:32:16	31.93	89.99
3	2022-03-15 15:28:06	20.328125	68.7171875

Figure 12: Finalizing the Dashboard

7. Share the link to the Google Sheet with everyone on your Team. Install the Google Sheets App on your phone and watch the gauge and graph change on many different devices all at the same time!

You are now writing data to the Cloud, which can be viewed by anyone, anywhere, at any time!

Acknowledgements

This Lab was designed by Dr. Chris Bachmann with modifications by Tobias Gerken. Special thanks to Joe Rudmin for helping secure necessary equipment and for providing lab support.

Revision	Description	Author
2025-03-19 (S25)	Updates for clarity and code structure	Tobias Gerken
2024-03-15 (S24)	Updated to Web and updates for clarity	Tobias Gerken
	Initial Version	Chris Bachmann