Specific Gravity Sensor

Jiangyue Chu, Alex Weiss, Michael Chestnut

Final Design Report

4/8/2024

Introduction

Clients View

During the fermentation process, it is important to know how quickly the sugars are being digested in order to get an estimate of the alcohol content. This process typically requires the brewer to open the container which adds risk of ruining the batch. This is primarily an issue in the scale of home brewing rather than on a larger industrial scale. A potential solution to this issue would be a device that can track the rate of sugar being digested by yeast in a brew without having to open the container. From our client, we were able to gather data that would further develop the need for a new solution and how to proceed in designing it. A typical way that homebrewers would measure specific gravity is by taking measurements before fermentation begins and during fermentation using a hydrometer that floats in the liquid. [2] This existing solution works well but it is limited in multiple ways such as needing to physically be read and monitored by opening the brewing container, battery life, durability, and accuracy. These factors lead to the need for an improvement upon typical hydrometer designs used by homebrewers.

Existing Alternatives

An alternative product is on the market already. It has been done by a company called Tilt. [1] Their product uses a sensor to determine how far the sensor is tilting in the liquid, and determines the density and specific gravity based on that. Their product is somewhat expensive though, priced at $135.00. Besides that product and method, there are no other digital hydrometer products that provide live data and measurements on the market. Other products in this field still require the user to take manual measurements. We hope to implement the digital hydrometer in a more cost effective and efficient manner.

Team FermenTech

Our team is FermenTech. Our team consists of Alex Weiss, Jiangyue Chu, and Michael Chestnut. This group is led by our client and advisor Dr. Kyle Winfree. Dr. Winfree is the Associate Director for Graduate Programs for the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. Each team member is assigned a responsibility to ensure the team operates efficiently. Our treasurer is Alex Weiss. Alex is responsible for keeping track of the funding we have used, ordering necessary parts, picking up the delivered parts, and overseeing whatever else may involve the team's finances. Our secretary is Jiangyue Chu. Jiangyue is responsible for taking notes of the meetings, reporting what has been done so far, what is accomplished each time we meet, and keeping track of the prototypes. Our team leader is Michael Chestnut. Michael is responsible for organizing team meetings, communicating with our client Dr. Winfree, and ensuring the team is properly informed of the latest news.

Problem Statement

Statement of Needs

The problem this project is trying to solve is how to measure the specific gravity of a must or wort without opening the container to take measurements. This is a problem because when you open the container to make measurements there can be bacteria that enters the system and ruins the batch. The product to be developed will be unique in its ability to be used without opening the container and will have the ability to view real time data. The target market for this product will be homebrewer technology.

The marketing requirements include that the system must have the following qualities:

- Accuracy
- Measure specific gravity without opening the container
- Small size
- Provide real time data
- Durability (low frequency of maintenance)
- Easy to use
- Cost effective

Statement of Objectives

The project aims to design and implement a high-precision, user-friendly digital hydrometer/refractometer with the following key objectives and features:

1. The device must be capable of accurately measuring the specific gravity of the brew, with a precision of 0.0025 g/ml, ideally reaching 0.001 g/ml, essential for accurately calculating sugar content.

2. The device must measure temperature, with a precision of 1°C , and the temperature sensor works in a temperature range of 0°C -35°C (32°F - 95°F).

3. The device must be small enough to fit into a 5 gallon bucket.

4. The device must have batteries and continuously work for at least for four weeks.

5. The device must keep real-time measuring temperature and sugar content.

6. The device must be able to store real-time data.

7. Enable data transmission over common IoT protocols (e.g., Wi-Fi, Bluetooth) or include USB connectivity for manual data transfer.

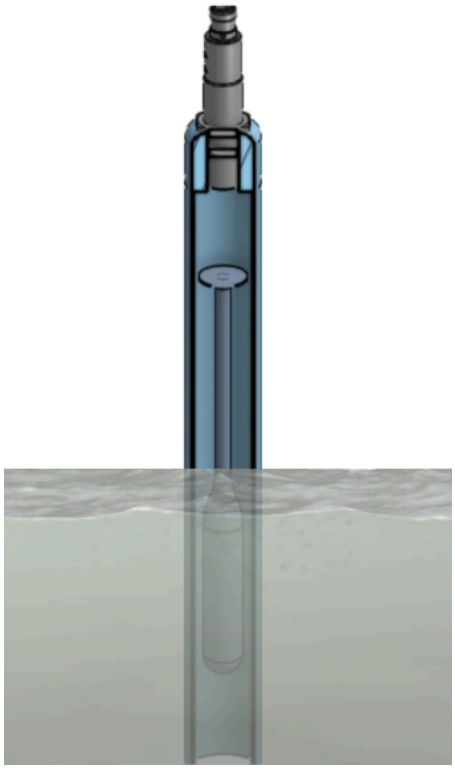
8. The device should include internal diagnostics to alert if any measurement anomalies are detected.

9. The device needs to measure the battery voltage and send a low battery warning when the voltage of a battery drops to a specific value.

11. The device must allow user calibration to accommodate various specific brewing conditions and requirements.

Concept

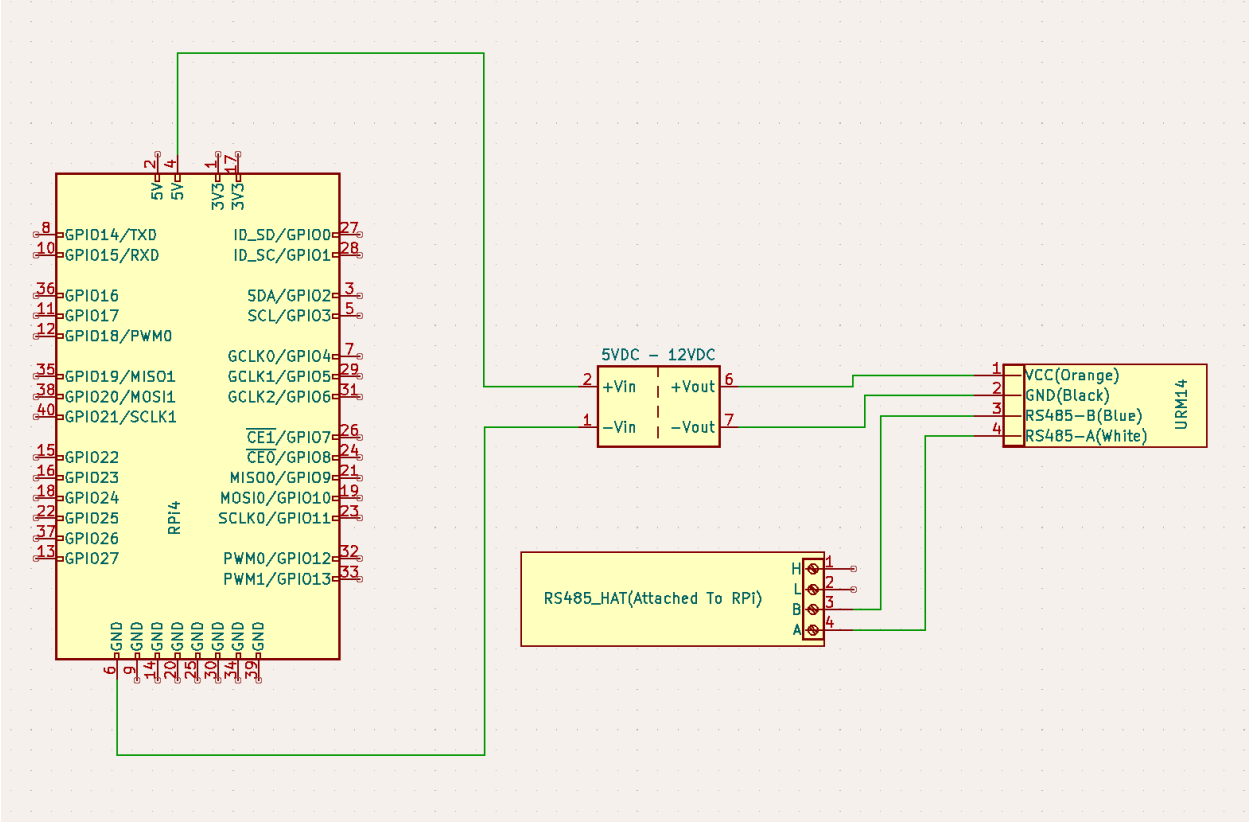
About Our Solution



Our solution to the problem at hand involves the use of a precision ultrasonic sensor. The plan for our team is to use the ultrasonic sensor to measure the distance to a bobber floating in the brewing liquid. The bobber will shift slightly in height as the density increases, just as a physical hydrometer does. The ultrasonic sensor will then measure the distance to the top of the bobber. As the height shifts, the raspberry pi will be able to interpolate the shift in height and provide a respective shift in specific gravity. Our ultrasonic sensor also includes a built in thermometer. This means we

will be able to satisfy both the specific gravity measurement and temperature measurement with one sensor. The raspberry pi will then take the data and store it into InfluxDB using a client called telegraf. Telegraf simply parses the outputted data and stores it into the database. Once the data is stored into the database, our team will create a live display of the data using Grafana. Grafana is an open source graphing software that we can use to give users a live view of the temperature and specific gravity.

Schematics



In the figure above, we can see the wiring schematic for our sensor. This sensor uses minimal components. The components involved are the RaspberryPi, the DC to DC converter, the RS485 HAT, and the URM14 ultrasonic sensor. 5v is taken from the raspberry pi and runs through a 5v to 12v DC to DC converter. This voltage is then supplied to the URM14. The

URM14 is connected to the RS485 HAT, which is connected to the RaspberryPi via header pins. The RS485 HAT handles all RS485 to TTL (UART) communication. Ideally we can convert the DC to DC converter into another HAT for the RaspberryPi. This would compact the design even further.

Bill Of Materials

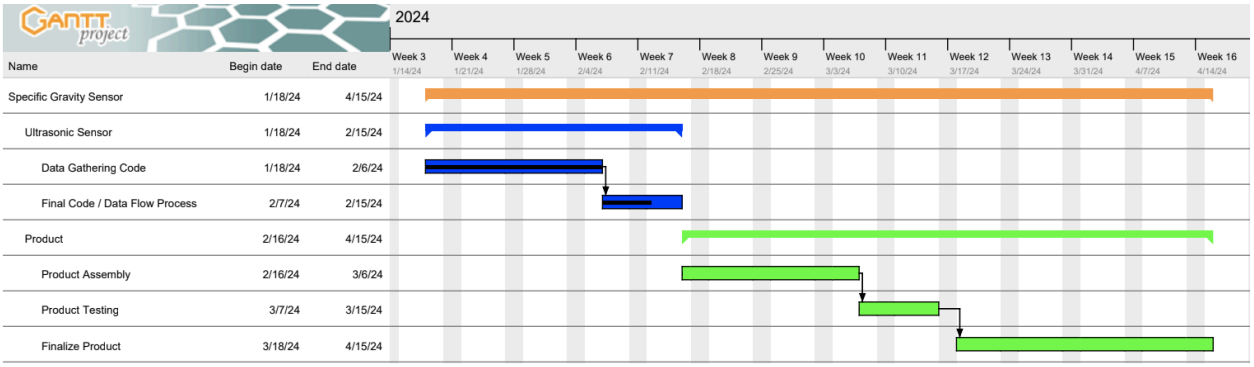
Phase	Item	Qty	Cost
Prototyping			
	Raspberry pi 4	1	$ 65.00
	D1 Mini Pro	1	$ 15.00
	Hall effect sensors (20pc)	1	$ 9.00
	Neodymium magnets (60pc)	1	$ 9.00
	Flex sensor	1	$ 14.00
	lm358 op amps	1	$ 8.00
	Flex sensor	1	$ 12.00
	Wireless Charging Receiver	1	$ 16.00
	RS485 to TTL adapter	1	$ 10.00
	IR sensors	1	$ 10.00
	Other IR sensors	1	$ 16.00
	Voltage Step Up	1	$ 6.00
	Precision Ultra Sonic Sensor	1	$ 110.00
	Wireless Charger	1	$ 16.00
	RS486 CAN HAT	1	$ 17.00
Testing			
	10 ft 1" PVC Pipe	1	$ 13.98
	1" PVC Male Adapter	1	$ 1.17
	1" PVC Female mdapter	1	$ 1.54
	1" PVC Cap	1	$ 1.17
	taxes on above materials	1	$ 2.00
	Sandpaper	1	TBD
	Rubber Gaskets	1	TBD
		Total Cost:	$ 352.86

The table above shows our team's current bill of materials. It can be observed that a majority of the funds were spent in the prototyping phase. This is because our team had many different ideas on how to approach the problem. We finally found a solution that we felt was best. This solution involves the ultrasonic sensor, the RS485 CAN hat, the voltage step up, and

all of the testing materials. Our bill of materials shows the many different avenues of prototyping our team took.

Project management

Gantt chart



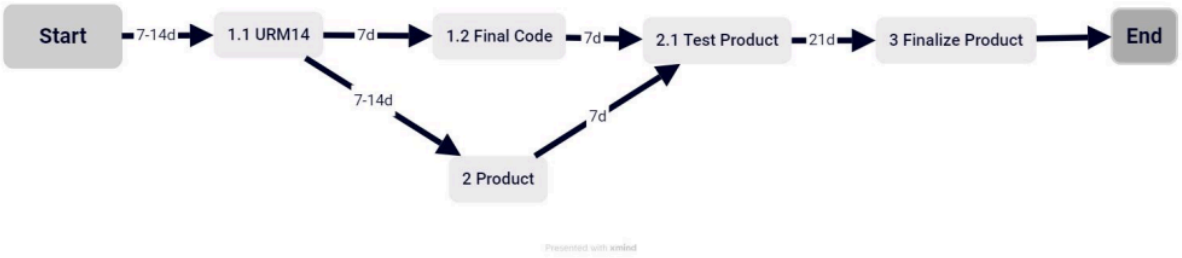
The Gantt chart shown above shows the various tasks that need to be completed to have the specific gravity sensor done on time. The time frame in the Gantt chart is from 1/18/24 to 4/15/25 and begins at that point as that is the time when the ultrasonic sensor was received. Before completing any assembly of the product we must first complete the code for gathering and displaying data. For the product section of our timeline we must first assemble before testing and finalizing the product as shown by the dependencies in the chart.

WBS

ID	Activity	Description	Deliverables	Duration (Days)	People	Resources	Dependencies
1	Specific Gravity Sensor						
1.1	URM14	figure out how to get data off URM14	None	14 Days	Michael	None	None
1.2	Final Code	Finish code and data flow process	None	7 Days	Michael	None	1.1
2	Product	gather parts and assemble product	"prototype"	14 Days	Alex, Jiangyue	None	1.1, 1.2
2.1	Test Product	Test reliability, accuracy, functionaility	improved product	7 Days	Alex, Jiangyue	None	2
3	Finalize Product	Make final improvements	Final Product	21 Days	Michael, Alex, Jiangyue	None	2.1

The work breakdown structure table shown above includes the same tasks as in the Gantt chart, however includes descriptions of each activity, who will complete them, and the deliverables. The coding section is assigned to Michael and is to be completed over three total weeks. The product section is assigned to Alex and Jiangyue and is to be completed over 6 weeks with the main deliverable being the finalized specific gravity sensor.

Pert Network

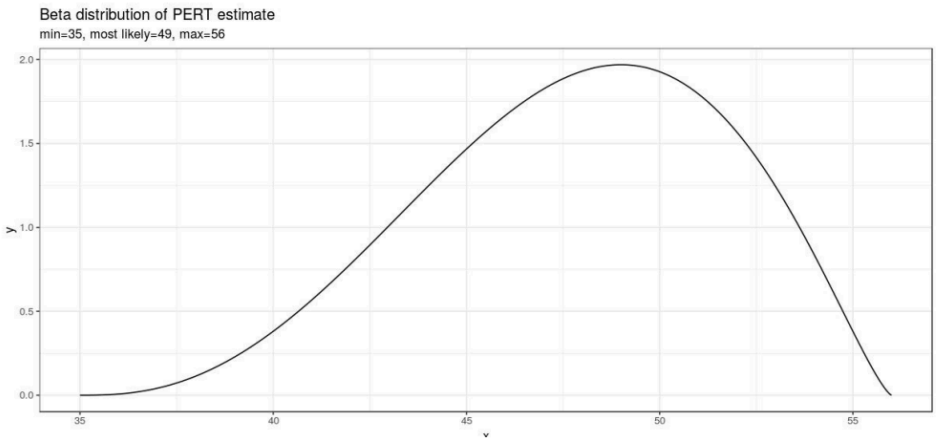


Optimistic time: 35 days

Pessimistic time: 56 days

Most Likely Estimation: 49 days

Pert Distribution Plot:



In the PERT network analysis, we have detailed our project flow and the time required for each part. Specifically, we pointed out that throughout the project process, completing the final code and product assembly can be done in parallel, which significantly increases our adaptability to project changes, allowing us to flexibly adjust the code to deal with emergencies. Additionally, we mentioned the detailed work breakdown from February to April 15th, including the optimistic time, pessimistic time, and the most likely estimation time. These time estimates help us assess the possibility of completing the project under different circumstances and provide us with a data-based method to prepare for the best and worst scenarios. Moreover, according to the PERT distribution plot, this chart visually shows the possible distribution of time required to complete the project, enabling us to better understand the uncertainty and range of variation in the project completion time.

Testing

Preliminary Testing Data

water (mL)	sugar (g)	bobber measurement (mm)	no bobber measurement (mm)	spec grav (g/ml)
210 ml	0	189.7	242.3	1
210 ml	0	189.9	242.5	1
210 ml	0	190.1	242.7	1
210 ml	0	189.9	242.7	1
210 ml	0	189.9	242.5	1
210 ml	4	183.5	242.1	1.01904
210 ml	4	184	241.1	1.01904
210 ml	4	184.2	240.9	1.01904
210 ml	4	184.2	240.9	1.01904
210 ml	4	183.8	240.9	1.01904
210 ml	8	176.2	239.4	1.038095
210 ml	8	176.4	239.1	1.038095
210 ml	8	176.4	238.9	1.038095
210 ml	8	176.2	239.2	1.038095
210 ml	8	176.2	239.2	1.038095
210 ml	12	169.3	236.8	1.057143
210 ml	12	169	236.8	1.057143
210 ml	12	169.1	236.8	1.057143
210 ml	12	169.3	236.5	1.057143
210 ml	12	169.1	236.5	1.057143
210 ml	16	161.7	233.4	1.07619
210 ml	16	161.2	233.4	1.07619
210 ml	16	161.2	233.2	1.07619
210 ml	16	161.6	233	1.07619
210 ml	16	161.4	233	1.07619
210 ml	28	142.9	226.3	1.1333333
210 ml	28	142.9	226.1	1.1333333
210 ml	28	142.7	226.3	1.1333333
210 ml	28	142.9	226.3	1.1333333
210 ml	28	142.6	226.1	1.1333333

In the initial data testing phase, we conducted our first test with an ultrasonic sensor. We utilized a hydrometer as a floating object, placing a 3D-printed cap on top of the hydrometer to facilitate measurements by the ultrasonic sensor. We altered the solution's density by adding sugar, measuring the change in the cap's height due to differences in buoyancy caused by the varying densities. In adding sugar, we adopted a strategy of incrementally adding 4g of sugar starting from 0g up to 16g, then directly increasing to 28g. In terms of code design, we measured once per second and calculated an average every sixty seconds as the final value. For each sugar addition, we collected data five times to ensure accuracy, gathering and analyzing the original experimental data. To ensure data was accurate, each time a measurement was taken, the sensor and testing materials were disassembled and then reassembled before the measurement was taken. This ensured that no unknown variables were influencing the results.

Final Testing Data

	Water (mL)	Sugar (g)	Distance (mm) w/bobber	Distance (mm) w/o bobber	Specific Gravity (g/mL)
	23000	0	281	299.9	1
	23000	0	281.1	299.8	1
	23000	0	281.2	299.9	1
	23000	0	281	300	1
AVERAGE	23000		281.075	299.9	
	23000	354	274.1	297.6	1.015391304
	23000	354	276	298.5	1.015391304
	23000	354	276.4	297.6	1.015391304
	23000	354	276.4	297.6	1.015391304
AVERAGE	23000		275.725	297.825	
	23000	708	270	296.2	1.030782609
	23000	708	270.8	296.2	1.030782609
	23000	708	270.2	296.2	1.030782609
	23000	708	270.1	295.4	1.030782609
AVERAGE	23000		270.275	296	
	23000	1067	263.6	293.5	1.046391304
	23000	1067	263.4	294.4	1.046391304
	23000	1067	264.3	293.7	1.046391304
	23000	1067	263.4	293.6	1.046391304
AVERAGE	23000		263.675	293.8	
	23000	1415	256.5	289.8	1.061521739
	23000	1415	257	289.8	1.061521739
	23000	1415	256.8	290.5	1.061521739
	23000	1415	257.3	290.2	1.061521739
AVERAGE			256.9	290.075	

When conducting our final tests we used the same methodology as with the initial testing by adding sugar in increments then measuring the change in bobber height in order to calculate the accuracy. The main difference between the initial and final testing was that we used our

product fully assembled rather than the small scale graduated cylinder in the initial testing. For each increment of adding sugar we measured the bobber height four times by taking the average over 60 seconds with a measurement every second. By using our product fully assembled to make these tests we are able to make conclusions about the precision and accuracy of our device as it compares to our specifications.

Preliminary Test Matrix

Tester: Michael Chestnut					
Test case: URM14 sensor					
Set up: measure bob float height with URM14 and observe change in specific gravity					
Test	**Total Sugar Added**	**Expected change in bobber height**	**Obtained**	**Pass**	**Fail**
1	4	> 6mm	~6 mm	✔	
2	8	> 6mm	~8mm	✔	
3	12	> 6mm	~7mm	✔	
4	16	> 6mm	~7.5mm	✔	
5	28	> 18mm	~19mm	✔	

The table above shows our test matrix, showing that we expected a change in bobber height of at least 1.5mm per gram of sugar added. This change in height would ensure that the sensor is accurate enough to detect a small change in specific gravity. As the table shows, we observed changes in bobber height greater than expected on all tests conducted. This indicates a passing grade for all tests.

Final Test Matrix

Tester: Alex Weiss					
Test Case: URM14 Sensor			Date: 03/28/24		
			Time: 10:00 AM		
Setup	Change specific gravity and measure bobber height				
Test	Sugar Added (g)	Expected change in bobber height (mm)	Measured (mm)	Pass	Fail

1	354	6.16	5.35		X
2	354	6.16	5.45		X
3	359	6.24	6.6	X	
4	348	6.05	6.775	X	

The table above shows the final testing matrix. The expected change in bobber height was based on the .0025 g/mL accuracy specification and two out of the four tests passed this mark. The two tests that did not meet the accuracy specification were within an acceptable range to still be considered accurate.

Preliminary Testing Results

Change in bobber distance per gram added (mm/g)
1.682142857
Change in water height per gram added (mm/g)
0.578571429
Change in specific gravity per mm change in distance (0-4g) (g/ml)
0.003227119
Change in specific gravity per mm change in distance (4-8g) (g/ml)
0.002610274
Change in specific gravity per mm change in distance (8-12g) (g/ml)
0.002682817
Change in specific gravity per mm change in distance (12-16g) (g/ml)
0.002411013
Change in specific gravity per mm change in distance (16-28) (g/ml)
0.002991796
Change in specific gravity per mm change in distance (0-28g) (g/ml)
0.002830856

Average Change In Specific Gravity Per mm of Bobber Height Difference (g/ml): 0.002792313

Through our preliminary testing we were able to extract some valuable information including the smallest change in specific gravity we can measure, change in bobber distance per gram of sugar, and change in water height per gram of sugar. We found that the average change in specific gravity per mm of bobber height difference was approximately 0.0028 g/mL. This

information tells us that we still have some work to do on improving the accuracy to the clients expectation of 0.0025 g/mL. By finding the change in bobber distance per gram of sugar that was added we are able to determine how much of a change in height we can expect in real application which will impact our overall design including the height of the tube required. The change in water height per gram of sugar added that was measured without the bobber will tell us how much we can expect to offset our distance measurements by accounting for sugars or fruits that are added to the liquid. Overall, our preliminary results will guide our next steps in designing our product to meet requirements.

Final Testing Results

Change in bobber distance per gram added (mm/g)
0.01708480565
Change in water height per gram added (mm/g)
0.006943462898
Change in specific gravity per mm change in distance (0-350g)
0.002876879317
Change in specific gravity per mm change in distance (350-700g)
0.002824092541
Change in specific gravity per mm change in distance (700-1050g)
0.002364953887
Change in specific gravity per mm change in distance (1050-1400g)
0.002233274507
Change in specific gravity per mm change in distance (0-1400g)
0.002544849602
Average Change in Specific Gravity per mm of bobber height difference: 0.002574800063

Our final testing results allowed us to figure out what the average accuracy of our device is over different changes in specific gravity. Through this testing we were able to find that our

product has an average accuracy of .00257 g/mL which meets our accuracy specification. Additionally we were able to find out that the change in water height per gram of sugar added remained similar to our preliminary testing and allows us to factor that change into our code to ensure a more accurate result.

Live Data Display

The figure above shows an example of the live data display. This data was acquired from an actual fermentation test. 16 liters of water was combined with 2 kg of sugar to bring the solution to 1.058 specific gravity. Unfortunately, the yeast that was added likely died, and fermentation never occurred. Because fermentation never occurred, specific gravity never changed. However, we can see the temperature changing and producing an accurate graph of the data.

Calibration

The calibration of our device requires two simple steps. First the user must take an initial specific gravity measurement using a physical hydrometer. Next the user must update this measurement within the calibration script upon starting up the device. The device will not run without this measurement and will continue to prompt the user for this information until it is received.

Measurements

The precision of our device can be determined through our final testing results by comparing the accuracy of each test. The four tests that were performed remained within .0005 g/mL of each other which displays that our device has a high level of precision. The accuracy of our device can be confirmed by comparing our final results of a long term brew to a physical hydrometer reading. After adding yeast to a sugar water solution we were able to find that the final specific gravity output from our device matched the actual final specific gravity measured by a physical hydrometer.

User Manual

FermenTech Specific Gravity Sensor

Components:

RASPBERRY PI 4:

- Used to run power monitoring python script

RS485 CAN HAT:

- Used to facilitate UART to RS485 conversion

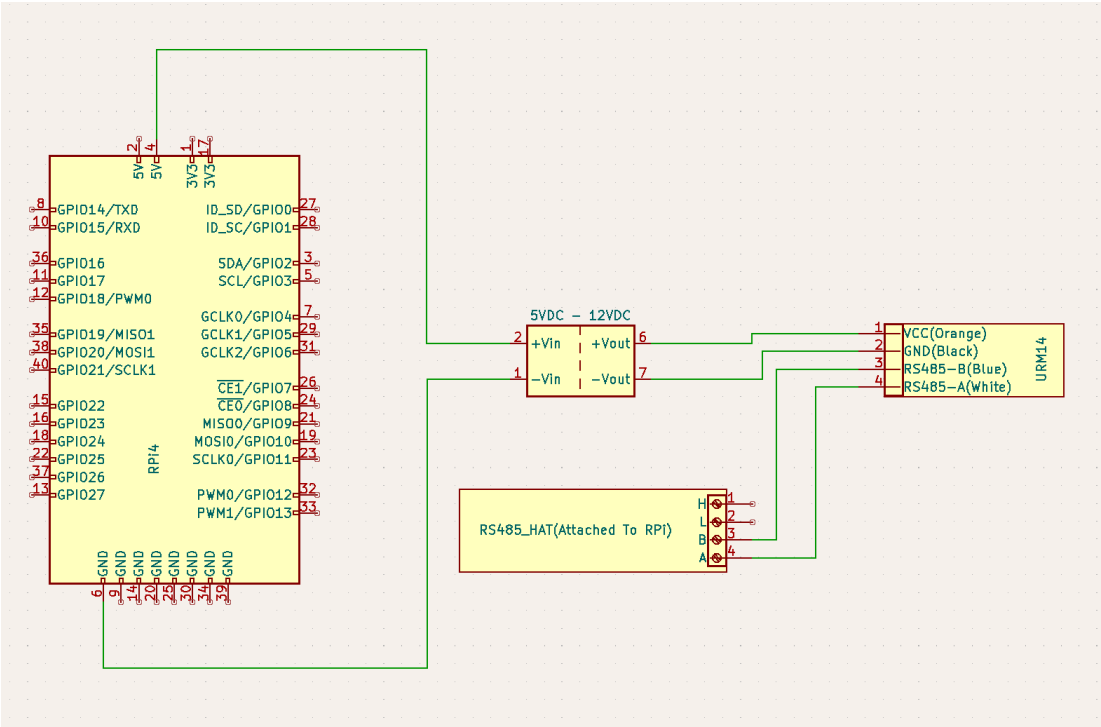
URM14 Ultrasonic Sensor:

- Sensor to gather distance and temperature measurements

MT3608:

- Voltage booster to converse pi4's 5v output to a suitable 7v-15v for the URM14

Schematic



Using The Sensor

Dependencies

- Install pyserial:

sudo pip3 install pyserial

- Install modbus-tk:

sudo pip3 install modbus-tk

- Install the GPIO package:

- On Raspbian:

sudo apt-get install rpi.gpio

- On Ubuntu:

sudo apt install python3-lgpio

- Install google sheets api:

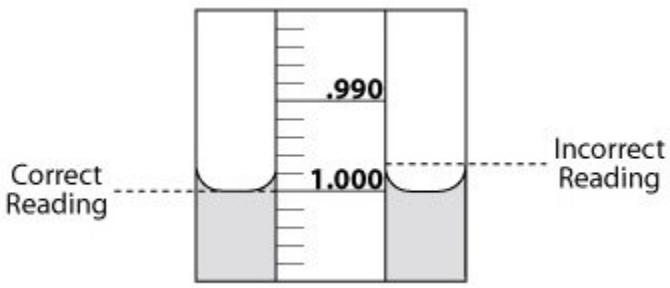
sudo pip install --upgrade google-api-python-client oauth2client

Calibrating The Sensor

- Modify value in Calibration script, following the commented instructions:

nano Calibrate.py

- Tip for reading hydrometer:



Running With Python

After installing dependencies:

- Clone this repo to get necessary files:

    git clone https://github.com/MichaelChestnut/FermenTech.git

- Change into FermenTech directory:

    cd FermenTech

- Verify that permissions are set so that the script is executable by running:

    chmod +x Calibrate.py

    chmod +x Measure.py

- Execute the script:

    python3 Measure.py

- Done!

Installing InfluxDB

Official InfluxDB Documentation: https://docs.influxdata.com/influxdb/v2/

- Get .deb file:

    curl -O https://dl.influxdata.com/influxdb/releases/influxdb2_2.7.5-1_arm64.deb

    sudo dpkg -i influxdb2_2.7.5-1_arm64.deb

- Start InfluxDB:

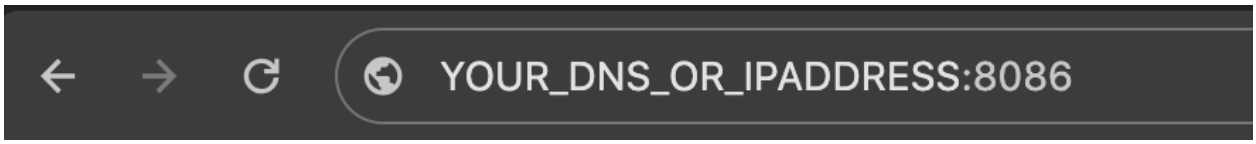
    sudo service influxdb start

- Check status of InfluxDB:

    sudo service influxdb status

- If Influx is running, access it through a web search bar, InfluxDB on port 8086:

  - example: device_DNS_or_IPaddress:8086



- Follow initial set up of influxdb and copy api token for later

Running with Telegraf

Official Telegraf Documentation: https://docs.influxdata.com/telegraf/v1/

NOTE: Install InfluxDB prior to attempting telegraf steps

NOTE: For telegraf to work, data must be printed to Standard Output. In this case, Data must be in CSV format.

Telegraf installation:

    curl -s https://repos.influxdata.com/influxdata-archive_compat.key > influxdata-archive_compat.key

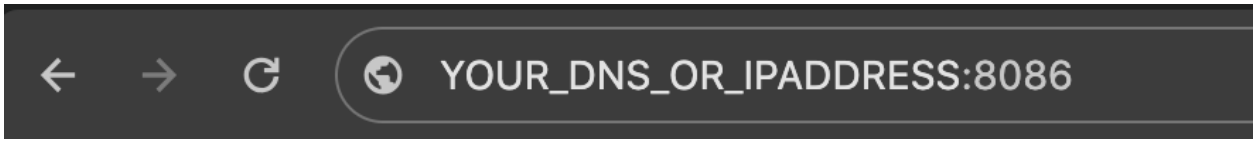
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-archive_compat.key' | sha256sum -c && cat influxdata-archive_compat.key | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg > /dev/null

echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debian stable main' | sudo tee /etc/apt/sources.list.d/influxdata.list

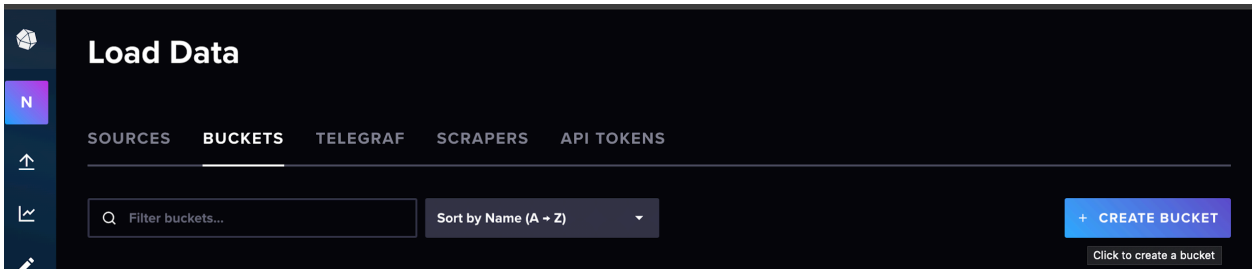
sudo apt-get update && sudo apt-get install telegraf

- Access telegraf through InfluxDB on device through a web search bar, InfluxDB/Telegraf on port 8086

  - example: device_DNS_or_IPaddress:8086



- Create bucket in InfluxDB for data:

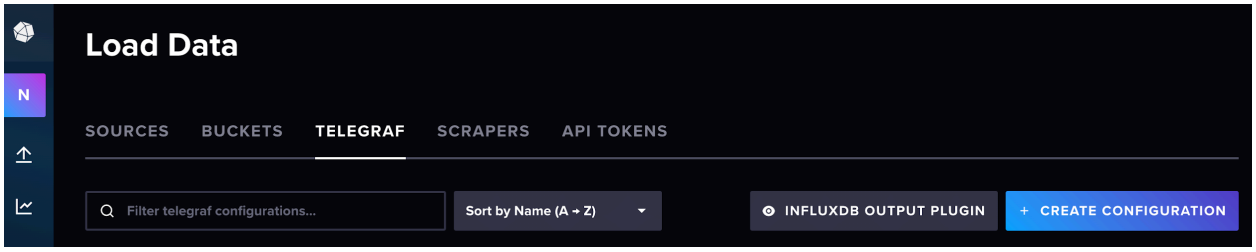


To generate Telegraf config file:

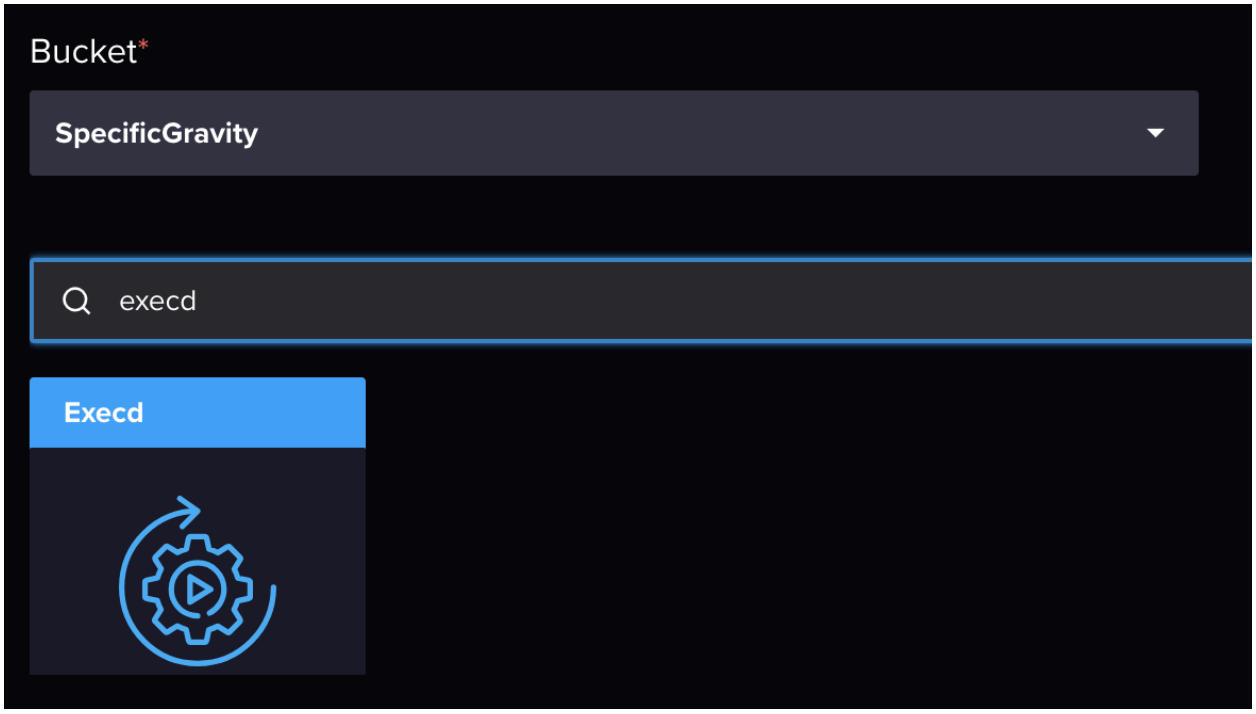
- This is a simplified explanation. Official Telegraf configuration file documentation can be found here: https://docs.influxdata.com/telegraf/v1/configuration/

- There is an example configuration file located in this repository, but here are steps to generate a new one. This is required as you need a unique key for each instance of data transfer on a host.

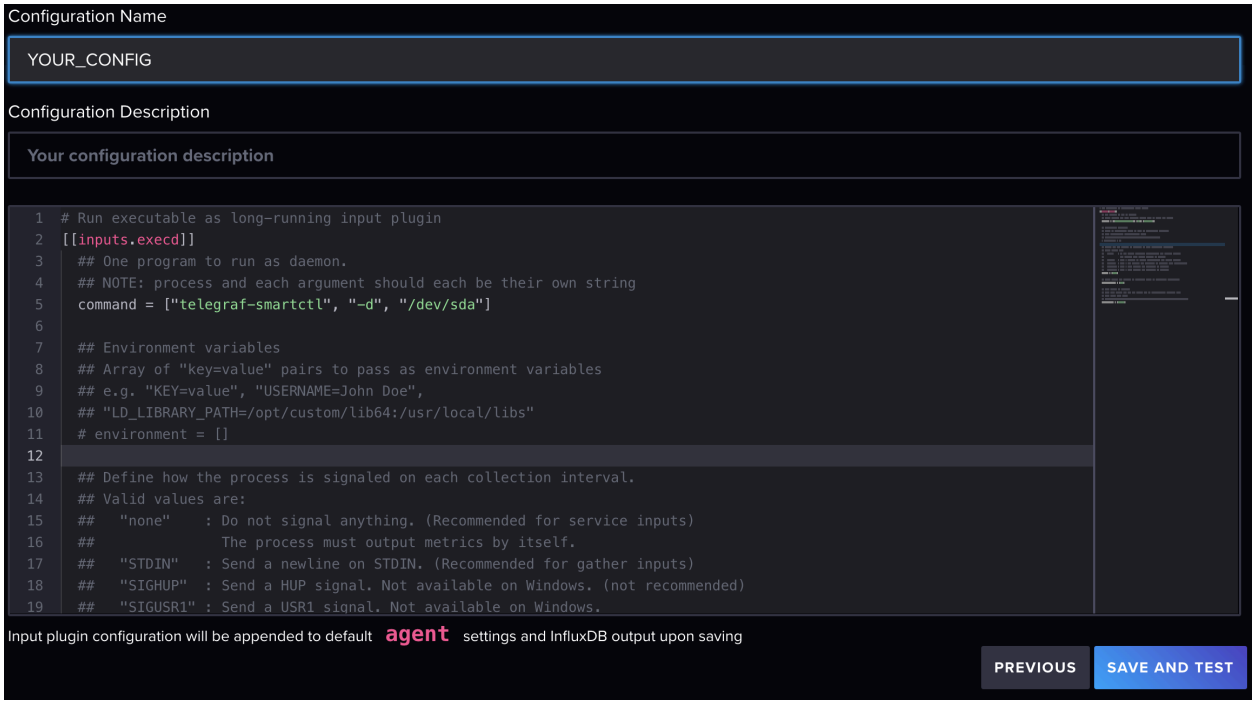
- Click create configuration:



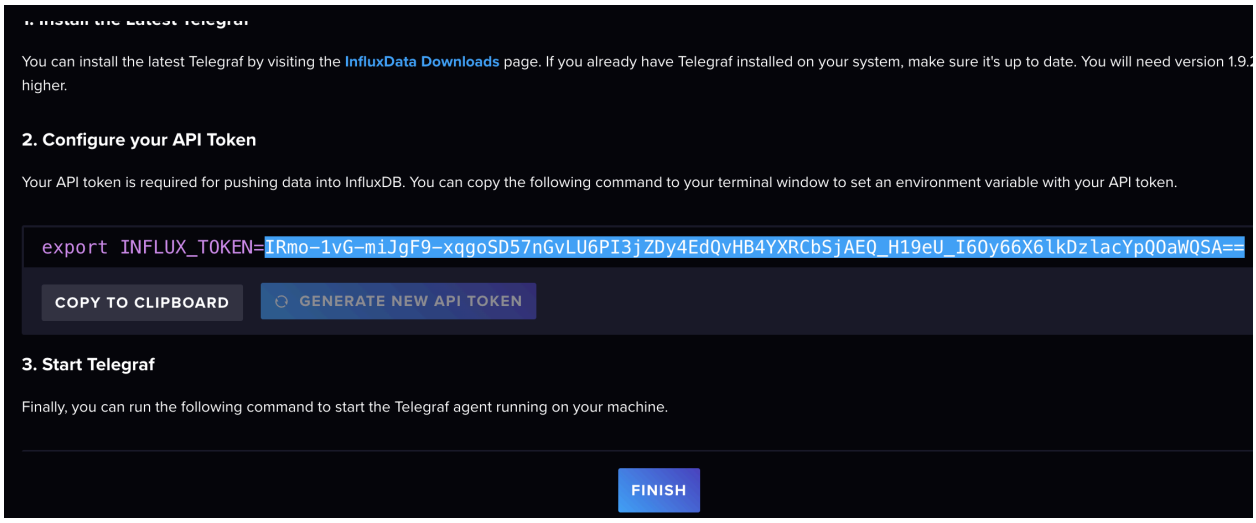
- Select the bucket and find execd source:



- Generate config file by clicking save and test:



- Copy generated token for later and click Finish:



- Erase auto generated config file in telegraf and paste in the example configuration found in this repository: https://github.com/MichaelChestnut/FermenTech/tree/outline-code

- Insert generated token into configuration file:

- Copy new config file to raspberry pi at /etc/telegraf/telegraf.d/CONFIG_NAME.conf:

Final Telegraf Set Up:

- Now that the configuration is done, Move working directory (containing measuring/executing code) into /opt:

    Sudo mv <directory> /opt

- Test telegraph user:

    sudo -u telegraf /opt/YOUR_DIRECTORY/YOUR_EXECUTABLE.py

- If error: serial.serialutil.SerialException: [Errno 13] could not open port /dev/ttyUSB0: [Errno 13] Permission denied: '/dev/ttyUSB0'

  - Use command:

    sudo usermod -aG dialout telegraf

  - To give telegraf i2c permissions:

    sudo groupadd i2c

    sudo chown :i2c /dev/i2c-1

    sudo chmod g+rw /dev/i2c-1

    sudo usermod -aG i2c telegraf

- Enable telegraf service:

sudo systemctl enable telegraf

- Start telegraf service:

sudo systemctl start telegraf

- Check telegraf status:

sudo systemctl status telegraf

Using Grafana to display data

NOTES:

- These instructions use ubuntu installation, for other OS, refer to https://grafana.com/docs/grafana/latest/setup-grafana/installation/

- These instructions assume you are installing grafana on the same device on which the database is located

STEPS:

- Install required packages

sudo apt-get install -y apt-transport-https software-properties-common wget

- Download the Grafana repository signing key

sudo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key

- Add a repository for stable releases:

echo "deb [signed-by=/usr/share/keyrings/grafana.key] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

- Update the list of available packages:

sudo apt-get update

- Install the latest OSS release:

sudo apt-get install grafana

- Enable the Grafana service to run on boot:

sudo systemctl enable grafana-server.service

- Start the Grafana service:

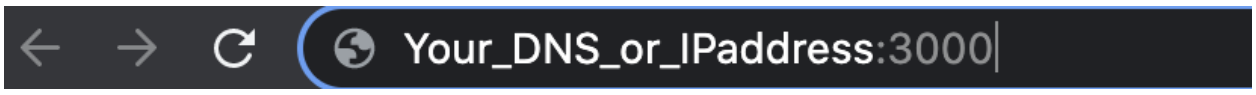
sudo systemctl start grafana-server.service

- Check the status of the Grafana service to ensure it is running:

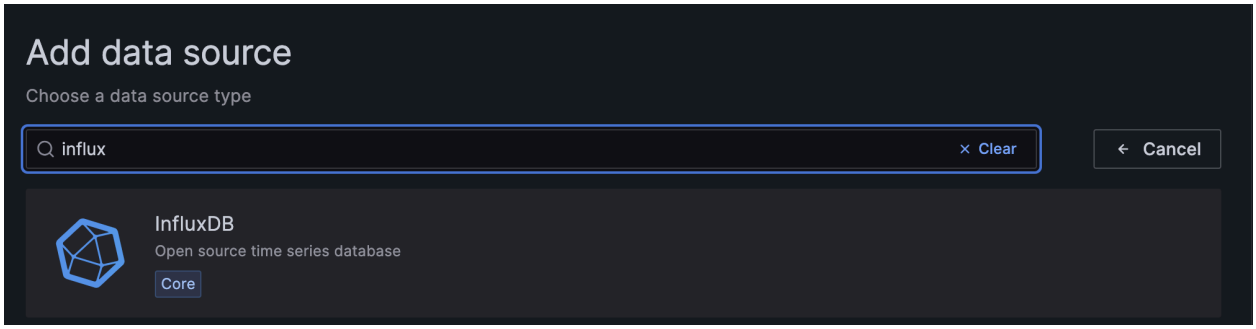
sudo systemctl status grafana-server.service

Listed below are steps to begin setting up a dashboard to display the data from the influxDB database. For official instructions, refer to the documenation here: https://grafana.com/docs/grafana/latest/

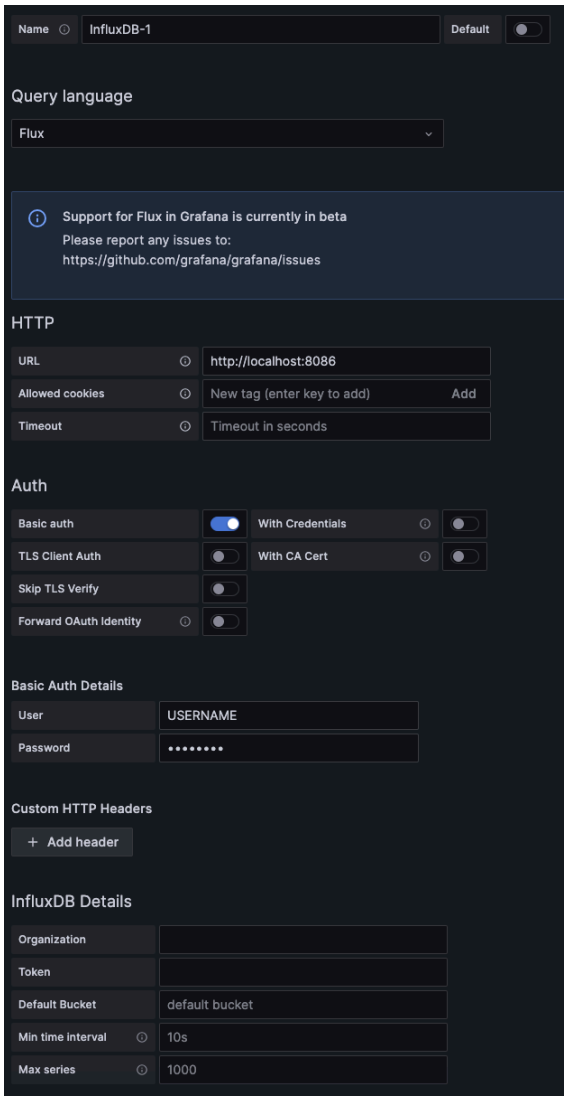
1. Open a browser and access port 3000 of the device that the database and Grafana instance are running on



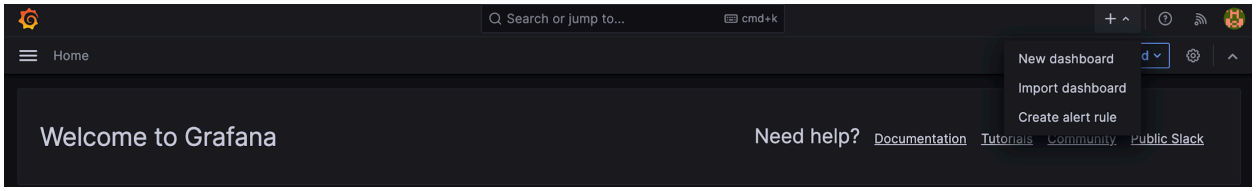
2. Navigate to the "Add Data Source" page and add InfluxDB



3. Fill out necessary fields, scroll to the bottom and click "Save and Test"



5. Back on the home page, click the plus and select "New dashboard" from the dropdown menu and select "New visualization"



6.

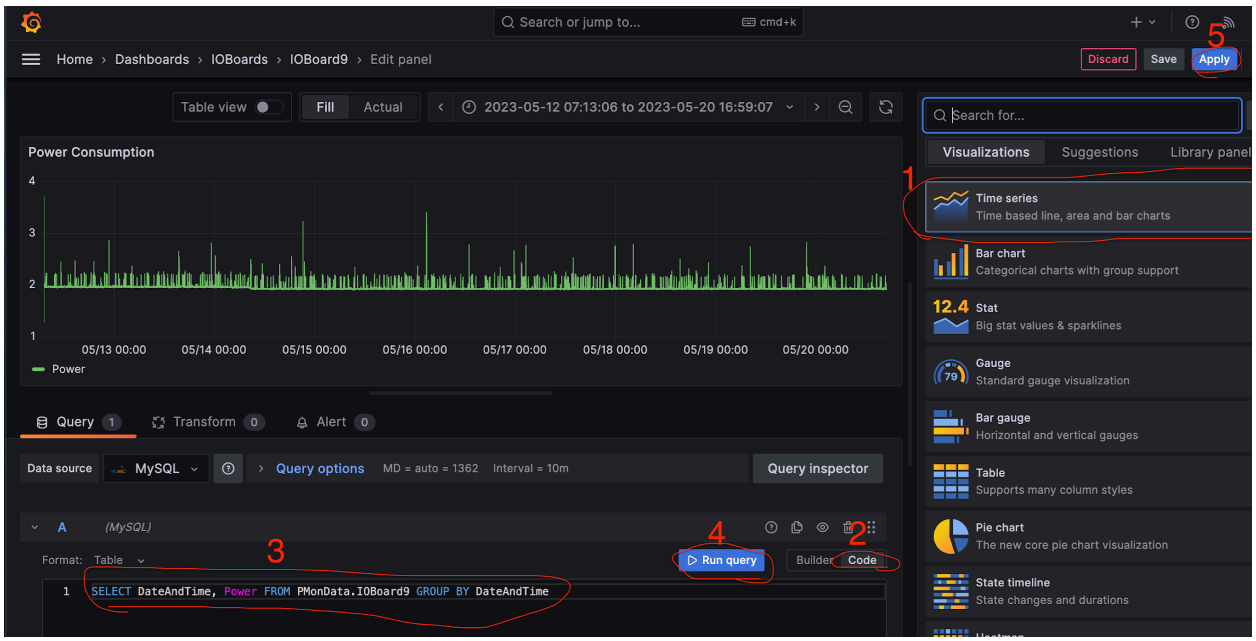
1) Select type of visualization. This example is comparing Power Consumption of a load against time using MySQL, but you will use flux instead.

2) Switch from builder to code under the query section.

3) Next, write your query (documentation for querying: https://docs.influxdata.com/flux/v0/query-data/influxdb/)

4) Next, hit run query.

5) Finally, if you are satisfied with the look of your graph, click Apply.



Google Sheets

Helpful Documentation:

- https://developers.google.com/sheets/api/guides/concepts

- https://developers.google.com/sheets/api/quickstart/python

- https://developers.google.com/sheets/api/guides/authorizing

- https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets.values/append

-http://www.whatimade.today/log-sensor-data-straight-to-google-sheets-from-a-raspberry-pi-zero-all-the-python-code/

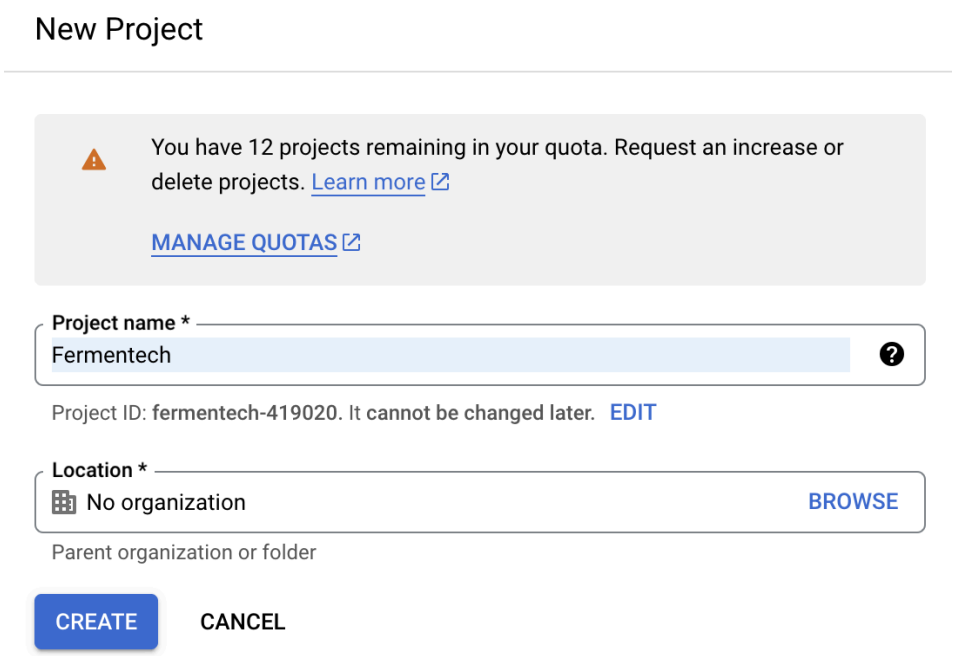
- https://erikrood.com/Posts/py_gsheets.html

This section will explain how to set up the google sheets api for cloud backup of data.

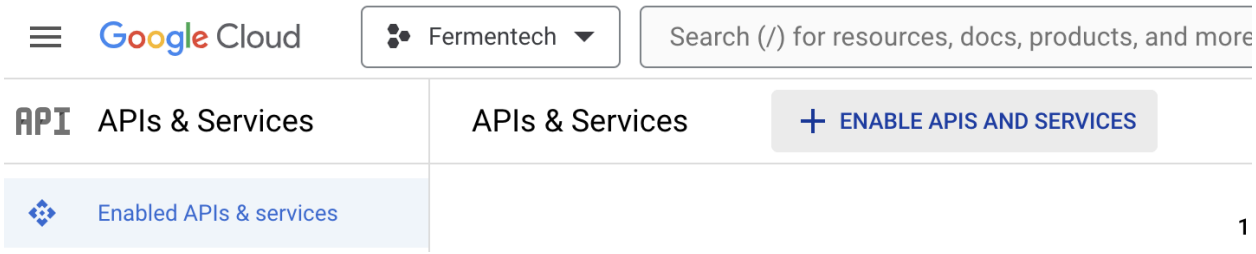
- Go to google cloud

    https://console.cloud.google.com/

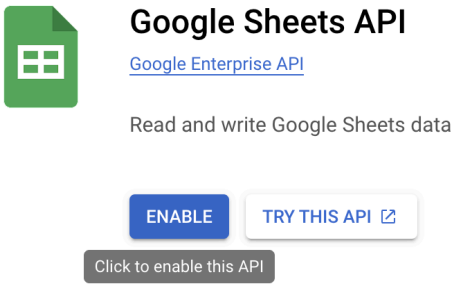
- Create and name new project:



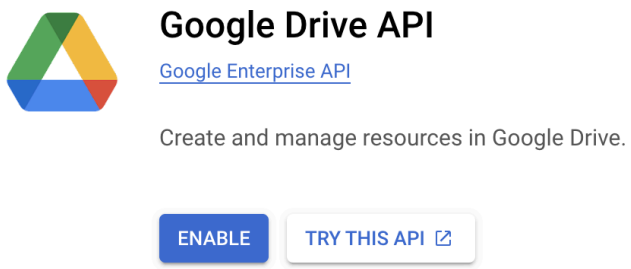
- Click Enable APIs and Services:



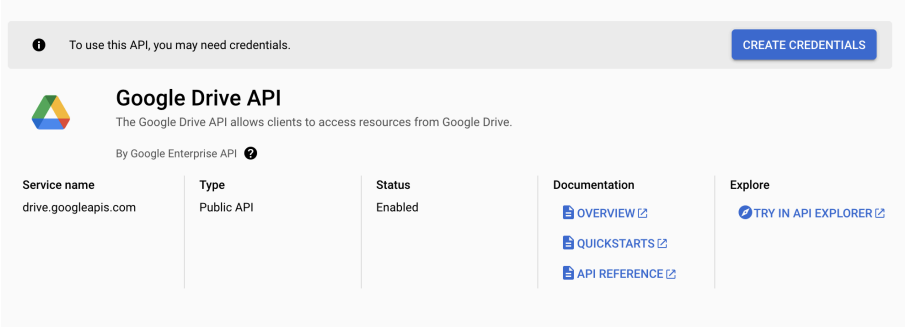
- Enable google sheets API:



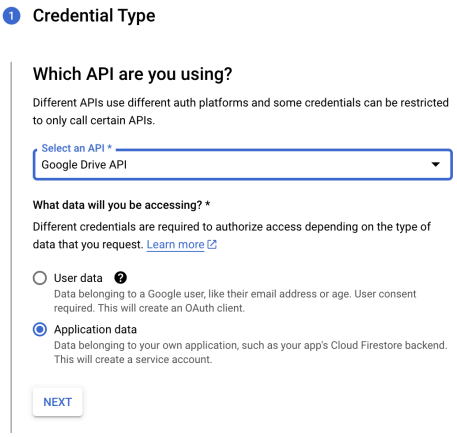
- Enable google drive API:



- Click create credentials after enabling google drive API:



- Select application data:



- Fill out service account details, i.e. service account name

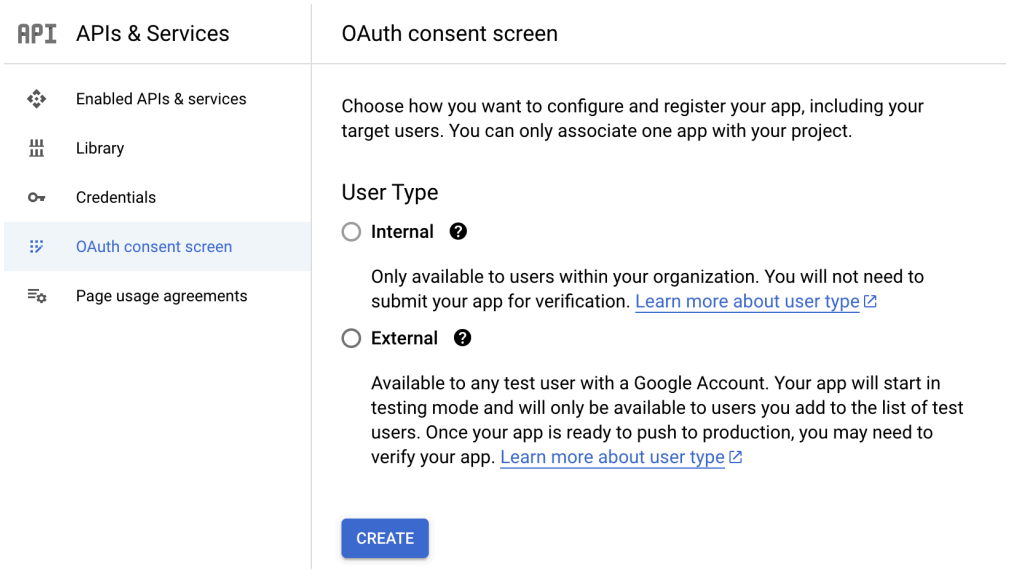
- Grant service account access to project

  - Select role as owner

- Go to OAuth consent screen, create new and fill out required fields.

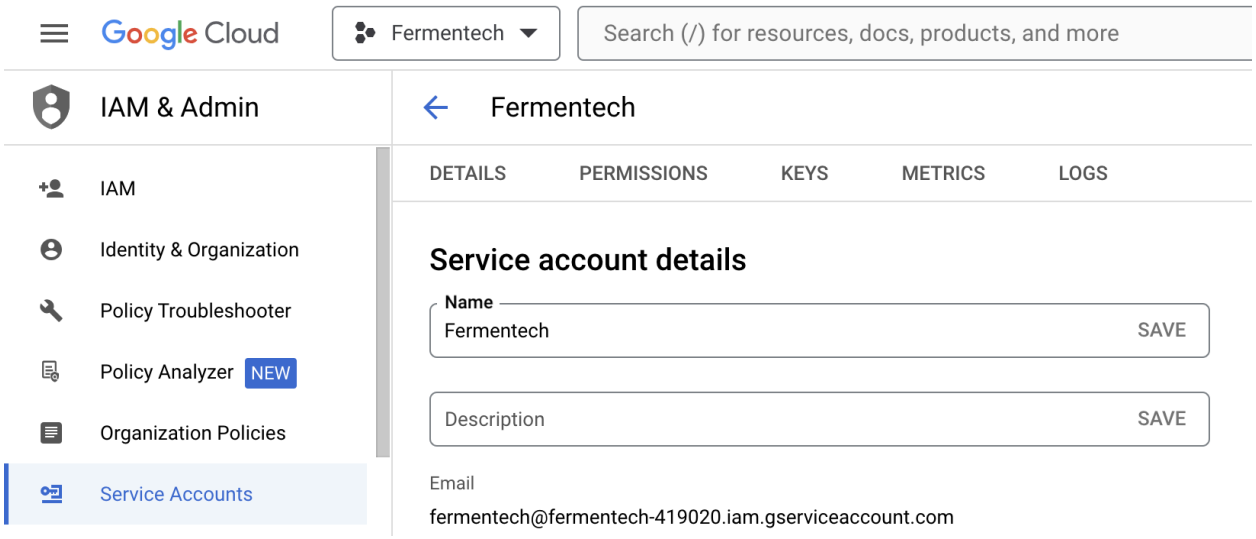
  - No scopes are required

  - Select External

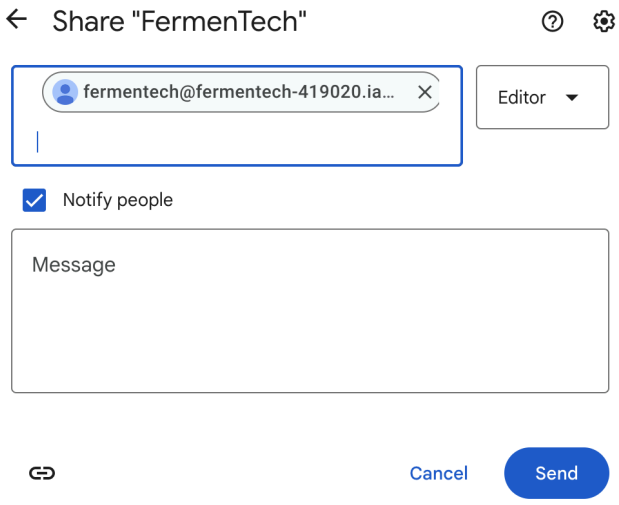


- Access Service Accounts under IAM and Admin

  - Copy the email listed in this tab:

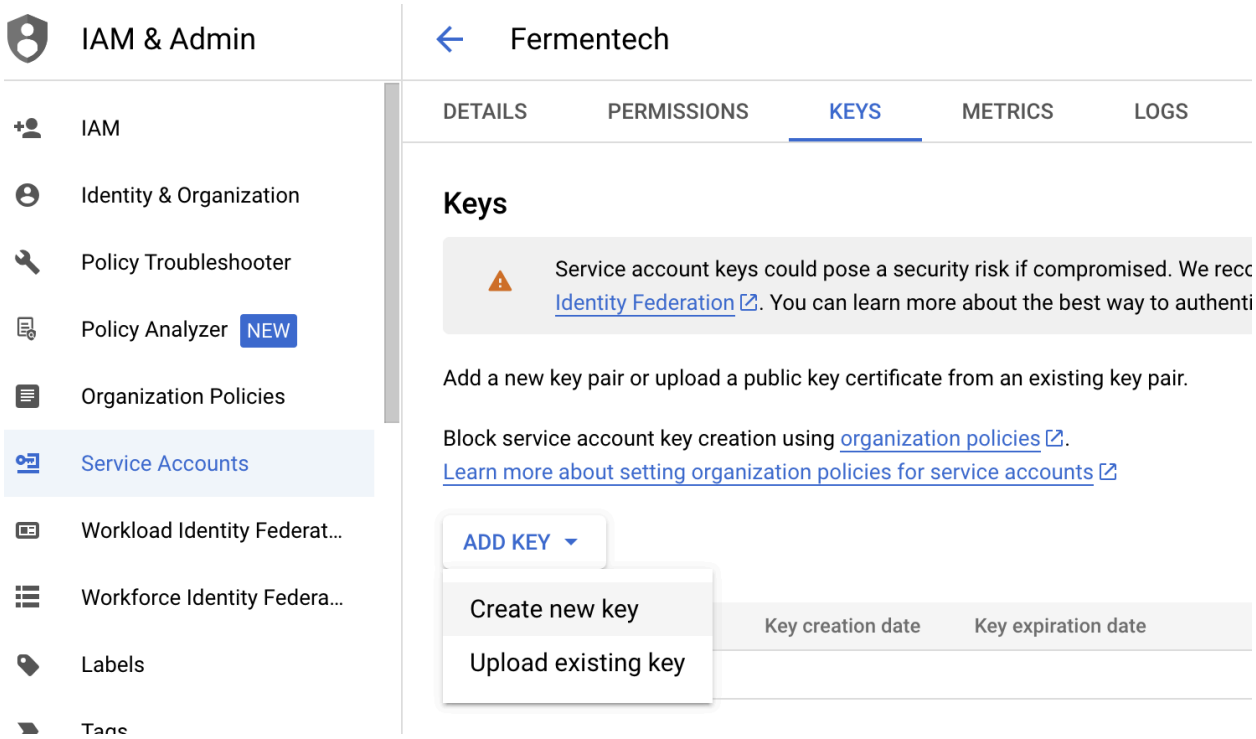


- Go to google sheet of interest and paste this email in the share option:



- Within Service Accounts, access the Keys tab and create a new key:

    - When prompted, select JSON



- Save the JSON keyfile to your computer and rename as FermentechKey.JSON

- Copy the keyfile to the raspberry pi into the same directory as the executable code

- Update spreadsheet ID on line 44 of Measure.py

- Update Sheetname (first field of upate_sheet function) to match the name of your google sheet on line 174

- Done!

Conclusions

In this Capstone project, during the first semester, we have explored various solutions: flex sensor, hall effect sensor and refractometer. Ultimately, we determined that using an ultrasonic sensor for measuring specific gravity was the final method to pursue. In this semester, we conducted a series of experiments with the ultrasonic sensor, ranging from validating its accuracy to assembling the entire product, and then integrating all the experimental components: data measurement, code debugging, and cloud data storage transmission. Finally, we conducted practical fermentation tests using yeast and sugar. The final test results showing 0.0028 g/ml were essentially in line with the client's requirement of 0.0025 g/ml, achieving the objectives of the experiment.

Throughout this project, we have not only acquired knowledge in various fields but also enhanced our skills and understanding significantly. Specifically:

- We learned about the brewing process and its associated units (Brix, Specific Gravity).
- We improved our coding skills in Python by writing the ultrasonic sensor’s measuring code.
- Our technical level has significantly advanced through exploring and applying the Raspberry Pi microcontroller and interfacing it with the sensor.
- We learned how to efficiently store and manage data using a database.
- We implemented cloud data backup to ensure the reliability of our data.
- Moreover, We used live data display to help users obtain real-time data in a simple and intuitive manner.

- Additionally, we became familiar with the RS485 communication protocol, utilized for the data transfer from the ultrasonic sensor.

For the future group, the plan is to implement the following steps to further refine the project:

- Develop a more user-friendly interface to enhance user experiences.
- Improve the product design to increase its structural rigidity, which is expected to enhance measurement accuracy.
- Implement a nutrient dispensing system for the yeast during the brewing process, aiming to improve the efficiency of the brew.
- Continue exploring the use of Hall effect sensors, as they may offer advantages in terms of cost and precision.
- Plan to explore using Brix as a measurement indicator instead of traditional specific gravity.
- Additionally, investigate the relationship between gas release rate and Brix/specific gravity, which involves the infrared sensor to detect escaping gas bubbles from the brewing container.

Bibliography

[1] N. Neibaron, "TiltTM wireless hydrometer and thermometer," Tilt Hydrometer. [Online]. Available: https://tilthydrometer.com/. [Accessed: Nov. 29, 2023].

[2] "Specific gravity of a liquid," in Density (Specific Gravity) - an overview | ScienceDirect Topics. [Online]. Available: https://www.sciencedirect.com/topics/materials-science/density-specific-gravity. [Accessed: Date when you accessed the article].