Bakery eLogistics

Abstract

Sophie's Bakery (owned and operated by Sophi Westfah and her family), located in Neu Ulm, Germany is looking for an e-Logistics solution assist with the packaging, sorting and transporting of her products (cakes) from a placing an order to delivery. Furthermore, the bakery would like the following requirements in place:

1. All product(s) to be monitored for temperature and humidity at various points in the solution, as the products cannot be over or under a certain temperature and humidity otherwise it will be regarded as spoilt.
2. They would like to offer the customer a tracking solution, with a web or mobile application throughout the cycle.
3. An alert system that would notify the bakery when a product is recognized as spoilt so that the order can be reissued and the customer informed via email or visually on a monitoring system.

Business Value

- The business value is to provide our customers with fresh products, thus improving our market value as a bakery that guarantees freshness, as our solution measures humidity and temperature throughout the cycle.
- We provide our customers with visual and email monitoring throughout the solution using RFID technologies, which links a customer to order.
- Our system will be fully integrated and autonomous with the sorting and delivery system, to provide efficient service to our customers.
- This solution would provide our staff with some relief, as currently baking staff is used for packaging and sorting, instead of concentrating on their normal daily duties of cake production.
- The solution will provide a manner in which mistakes are mitigated, as the autonomous system will deal with packaging and sorting, which will be more efficient and less error-prone.

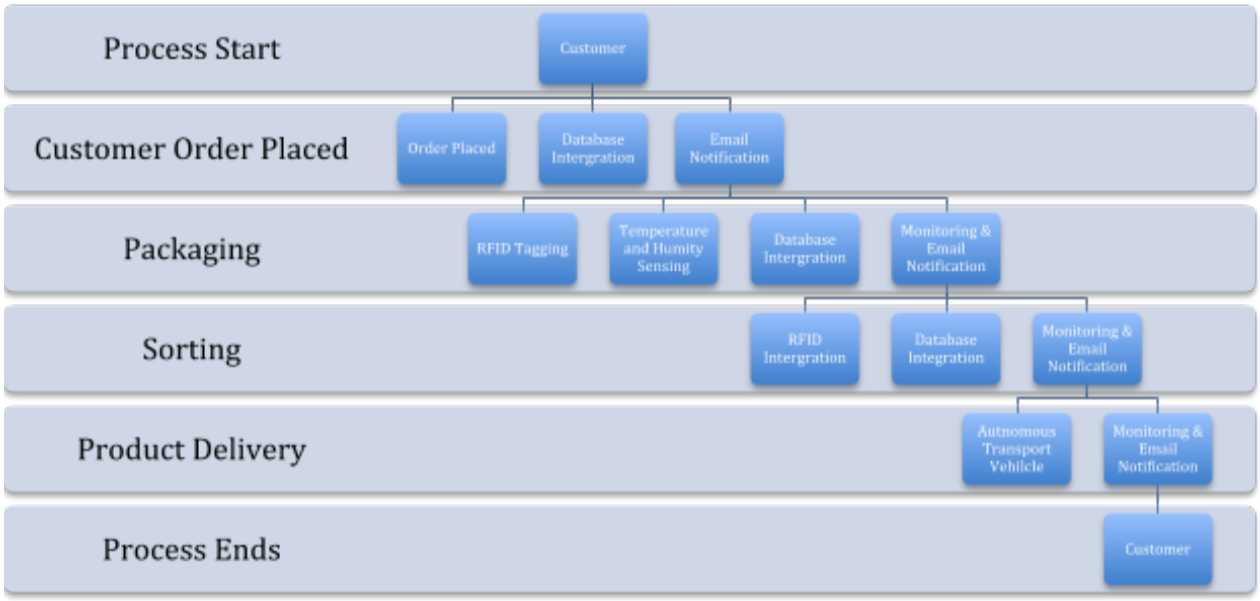
Business process

Business process goals

Implement a section where the cake is packaged.
Implement a sorting section, where the cake is sorted according to the delivery address.
Deliver the product to the customer.

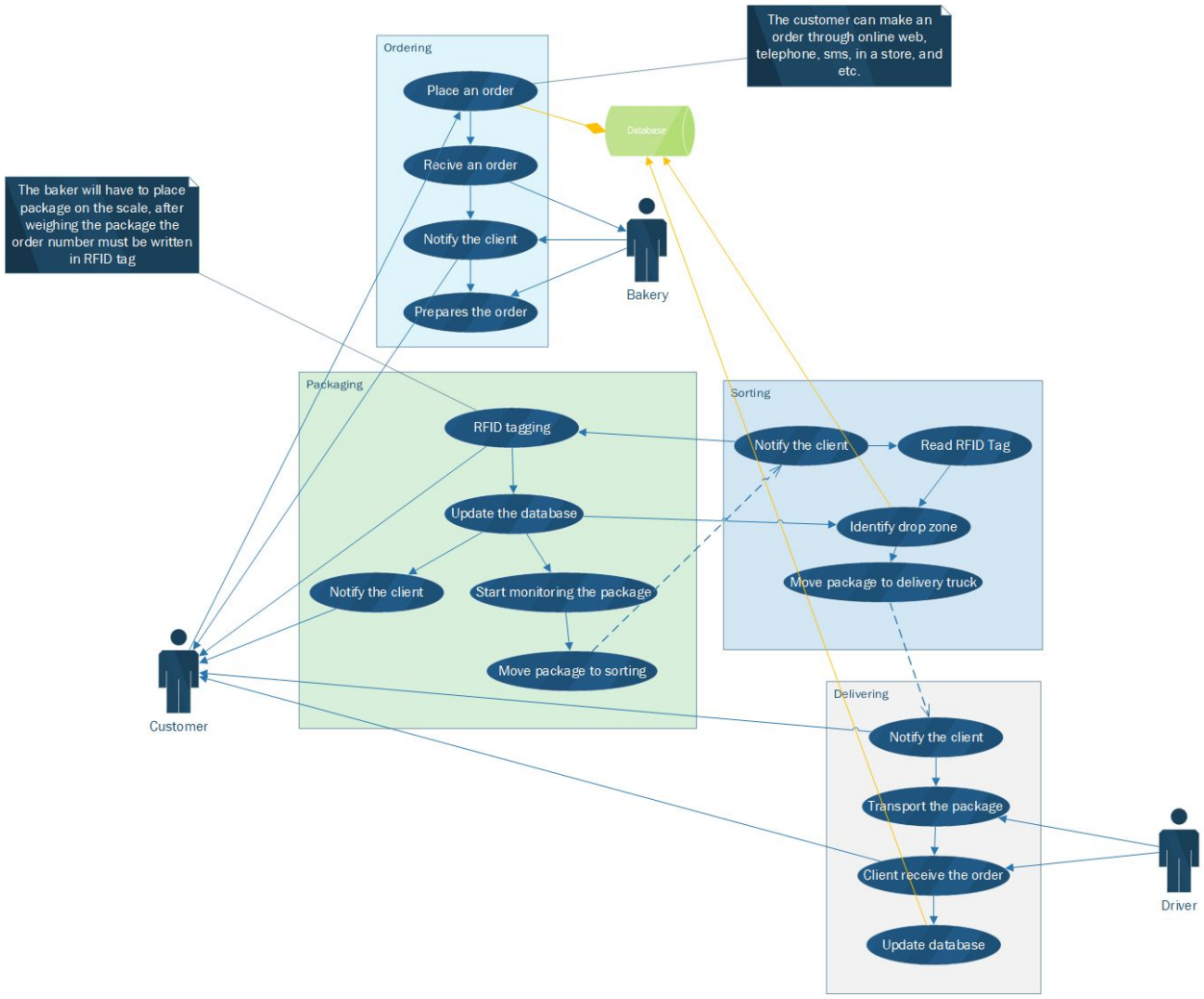
Goals to be realized at a later stage:
Implement a platform for customers to purchase goods.
Implement efficiency

Business process model



IT / UML

High Level Design



Hardware and Software Used

Stage	Hardware	Software
Process Start	Web Server	Apache, Tkinter, Flask, SMTP
Customer Order Placed	Web Server	Apache, Tkinter, Flask, SMTP
Packaging System	Raspberry Pi 3+, DHT11 Temperature and Humidity Sensor RC522 RFID Module RFID Tags Weight Sensor	Python, MySQL, SMTP
Sorting System	Raspberry Pi 3+ BrickPI Lego Mindstorms Large Motors Lego Mindstorms Infrared Sensor Lego Blocks and Gears(Constructions) RC522 RFID Module RFID Tags	Python, MySQL, SMTP
Delivery System	Lego Mindstorms EV3 Brick Lego Mindstorms Colour Sensor Lego Mindstorms Touch Sensor	Lego Mindstorms Education EV3 Development Suite
Process Ends	Web Server	SMTP

Running the Solution

Github Solution

GITHub Solution: https://github.com/IoTLabHNU/i4SC

Packaging

In order to run the packaging solution, you have to download a GitHub repository from the link above. Then, you will have to, i) install all the mentioned Python packages in Packaging class diagram using pip [e.g. for Unix: python-pip install mysql.connector], ii) connect the weighing scale and RFID module, and iii) you have to run “PackagingGUI_2.py” in Packaging folder of the downloaded repository. The image below depicts the program when is ran. The proposed solution assumes that, there is an order in the database. The user will have to enter the order number into the entry field.

Counting

Navigate into the Counter folder and run the Counter.py (python3 Counter.py). The program will display “Counter Program now running”. Press Ctrl + C to stop the counting program.

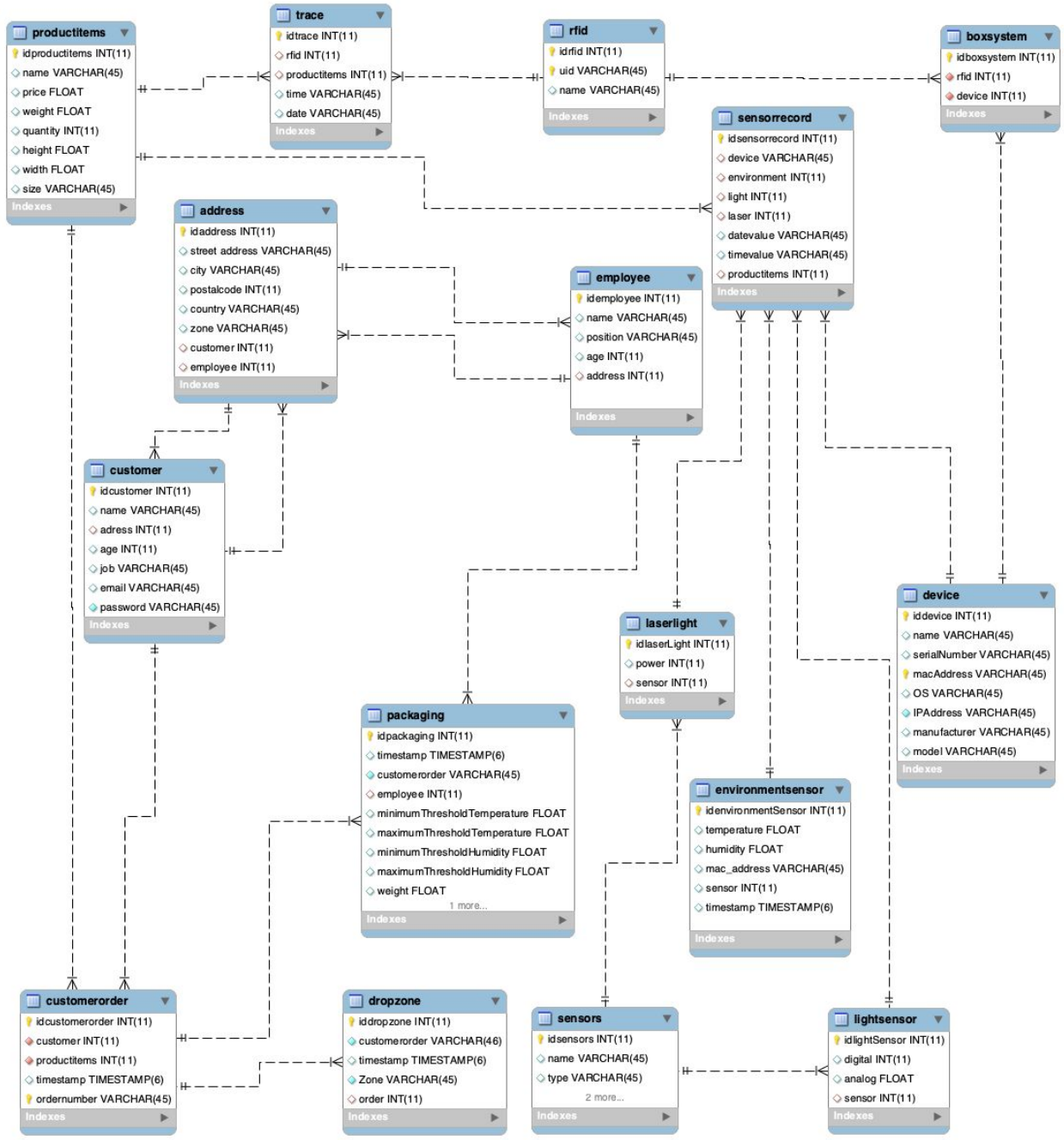
Sorting

For the sorting system you have to first start the mindstorm software and select the hackathon program on it to power the conveyor belt . After that when you want to start the sorting station you will need to start up the pi and then navigate to the Desktop/Shipping/BakingCakes directory with the terminal. In said directory you will find that there is a RunSortingSystem.py file. Now type “python3 RunSortingSystem.py” and the program will automatically start. To end the program press control - c. Alternatively you can ssh into the pi and follow the same instructions above.

Delivery

Retrieve the i4SC_touch_blackV3_stop.ev3 file from the Delivery section of the Github repository. Import the file into the Lego Mindstorms Education EV3 Development Suite. Connect the ATV via USB, then copy the aforementioned file to the ATV. The program can be run from the development suite when connected via USB, Bluetooth or WiFI, alternatively execute the program from the ATV (Brick) by selecting the aforementioned program.

Database Schema



Architecture

Counting System

Many times in the industrial processes it is important to count the number of items that pass through a certain point. The smart counter systems use a laser emitter and a light detector to count the number of items that obstruct the laser beam.

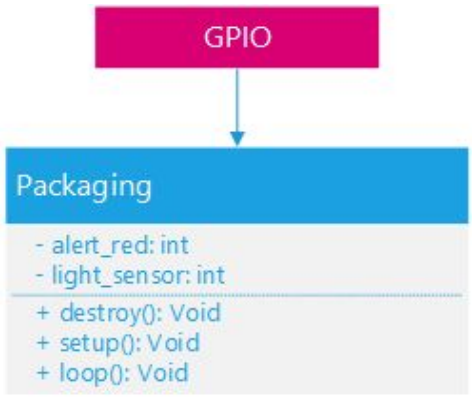
Hardware

- Raspberry Pi - To connect the sensors.
- Breadboard - To extend power, ground and GPIO pins.
- LM393 Dual Comperator and laser emitter modules (HW-483)
- LED light - Alert if there is an object obstructing the laser light beam.

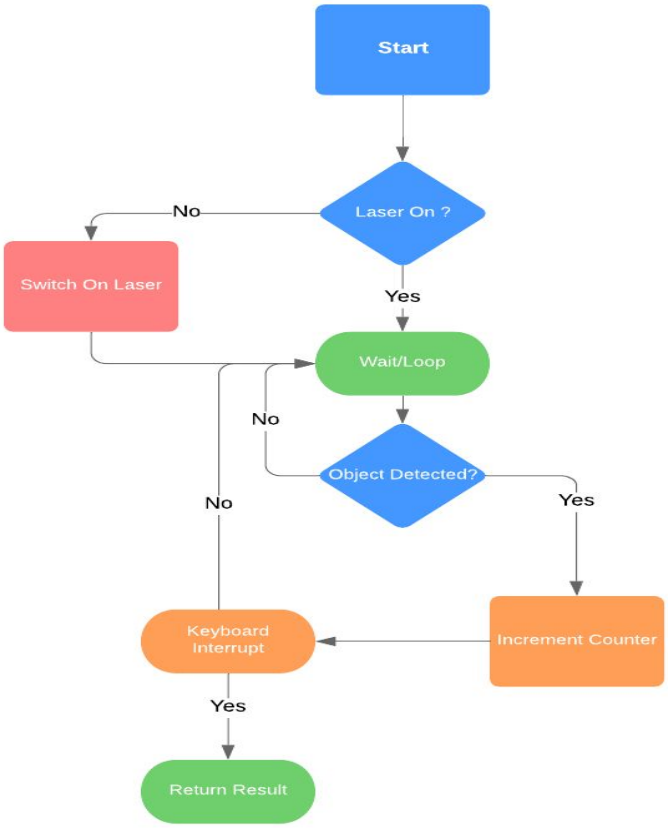
Software

- MySQL - To implement a relational database that stores a timestamp and the number of objects that have passed through the laser system.
- Grafana Server - Data Visualization.
- MySQL.Connector – A package used for connecting a client to a MySQL server.
- Python Programming Language:
    - GPIO – A package to define and select which pins are being used for what.

Class Diagram

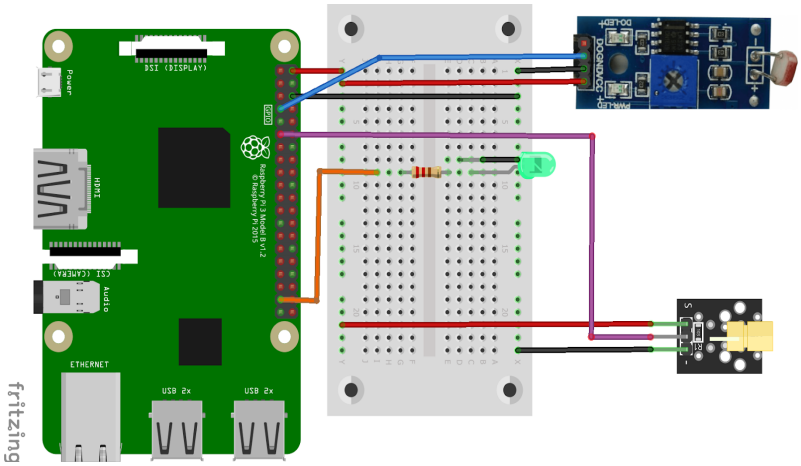


System Flowchart



Counting System Flow Diagram

System Schematic



Counting System Schematic

Packaging System

From the business case above, one can attach sensors to their packaging containers. The sensor will sense humidity and temperature, then upload these values to the server, where they can be available for report analysis with Tableau. We propose that we use the following equipment:

Hardware

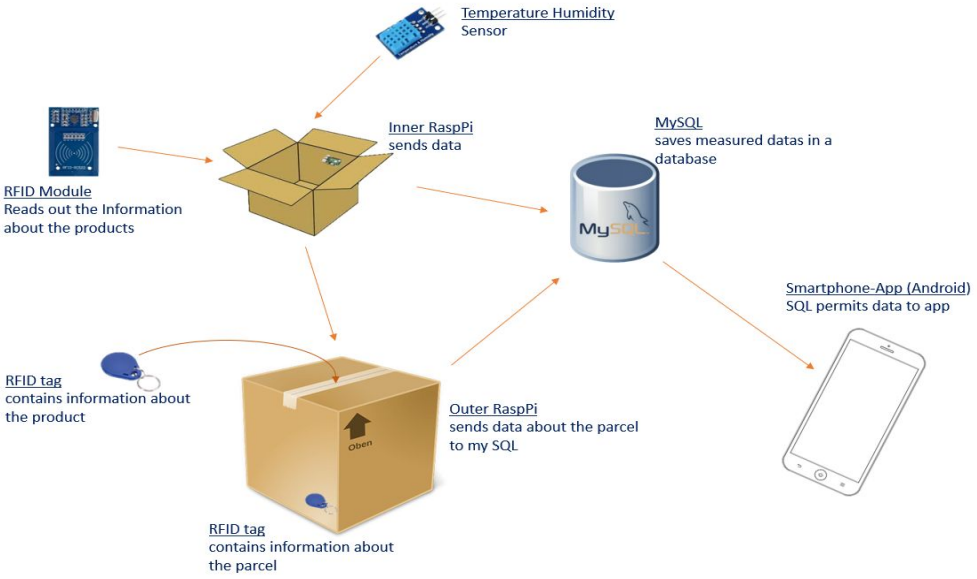
- Raspberry Pie – To connect the sensors and RFID module.
- RFID Module – To write and read the information on the RFID tag.
- RFID Tag – To store order information after packaging.
- Temperature and Humidity Sensor – To read the environment conditions of the package.
- Box – Where sensors and RFID are installed.
- Android Smartphone – To run a monitoring application.

Software

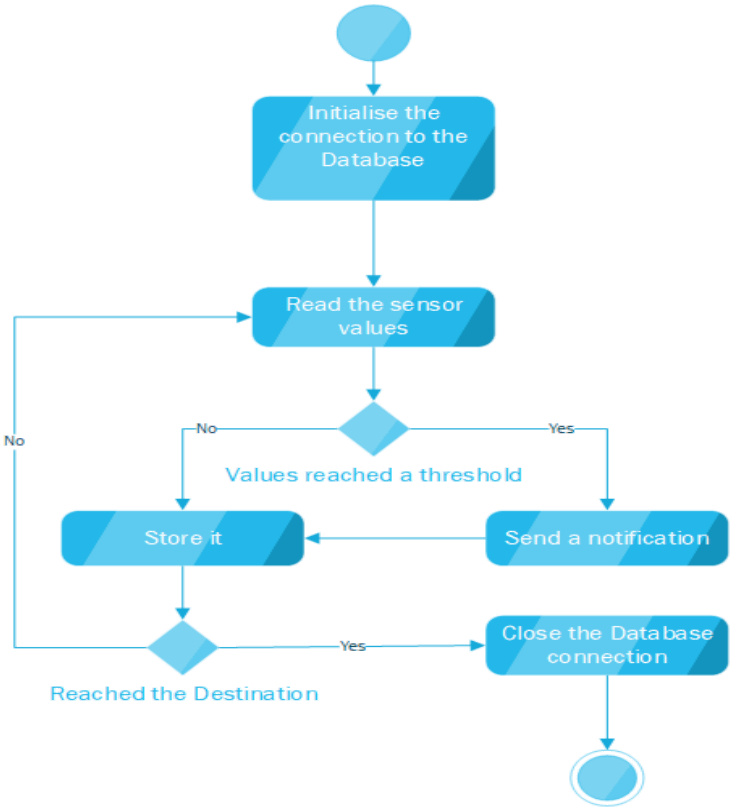
- MIT App Inventor – For creating a monitoring application.
- MySQL – To implement a relational database that will store customer, product, and order information.
- Python Programming Language – Below are package can be used for implementing the proposed solution:
    - GPIO – A package to define and select which pins are being used for what.
    - Adafruit_DHT – A package for reading temperature and humidity on the package.
    - MySQL.Connector – A package used for connecting a client to MySQL server.
    - Pyrebase – A Google time series database to store package conditions in real time.
    - MFRC522 – A package for commanding the RFID module to read or write information on the tag.
    - HX711 – A package to calibrate the scale and weigh the package.
    - Tkinter – A package used for implementing Graphical User Interface.

Our solution is divided into three sections namely, i) Packaging – where the package is weighed and receive an order number that will be using in the Delivery System, ii) Monitoring – a section for monitoring temperature and humidity conditions in the package, and iii) Application – this is where design and functional components of the application are discussed. Furthermore, the sections below discuss the design of the proposed solution.

System Flowchart

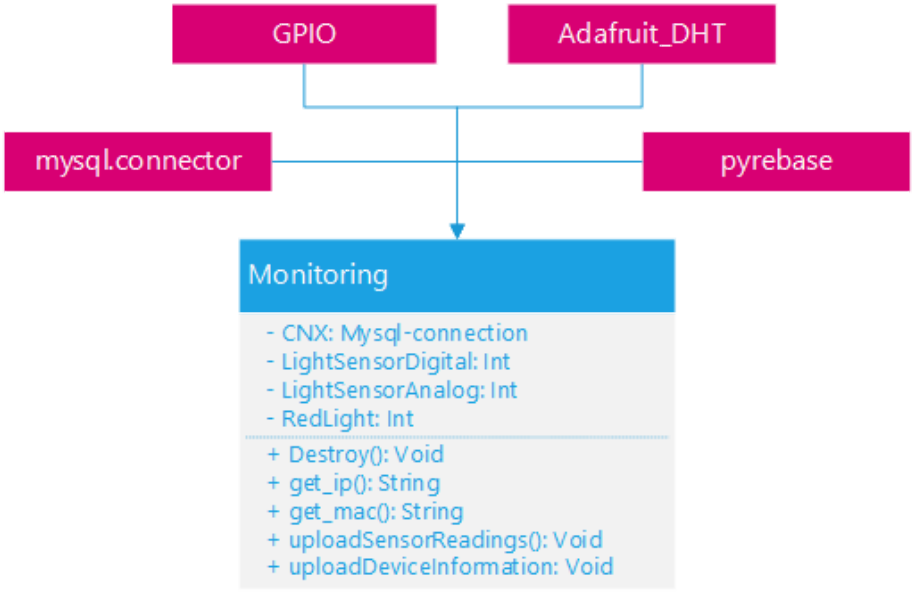


Packaging System Approach



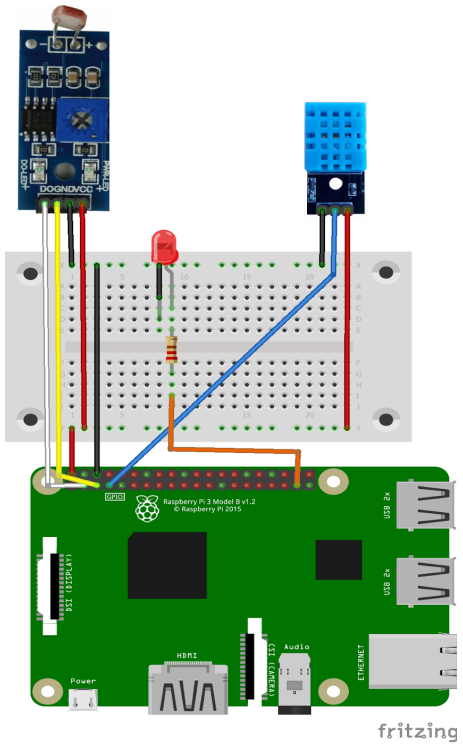
Packaging System Monitoring Flow

Class Diagram



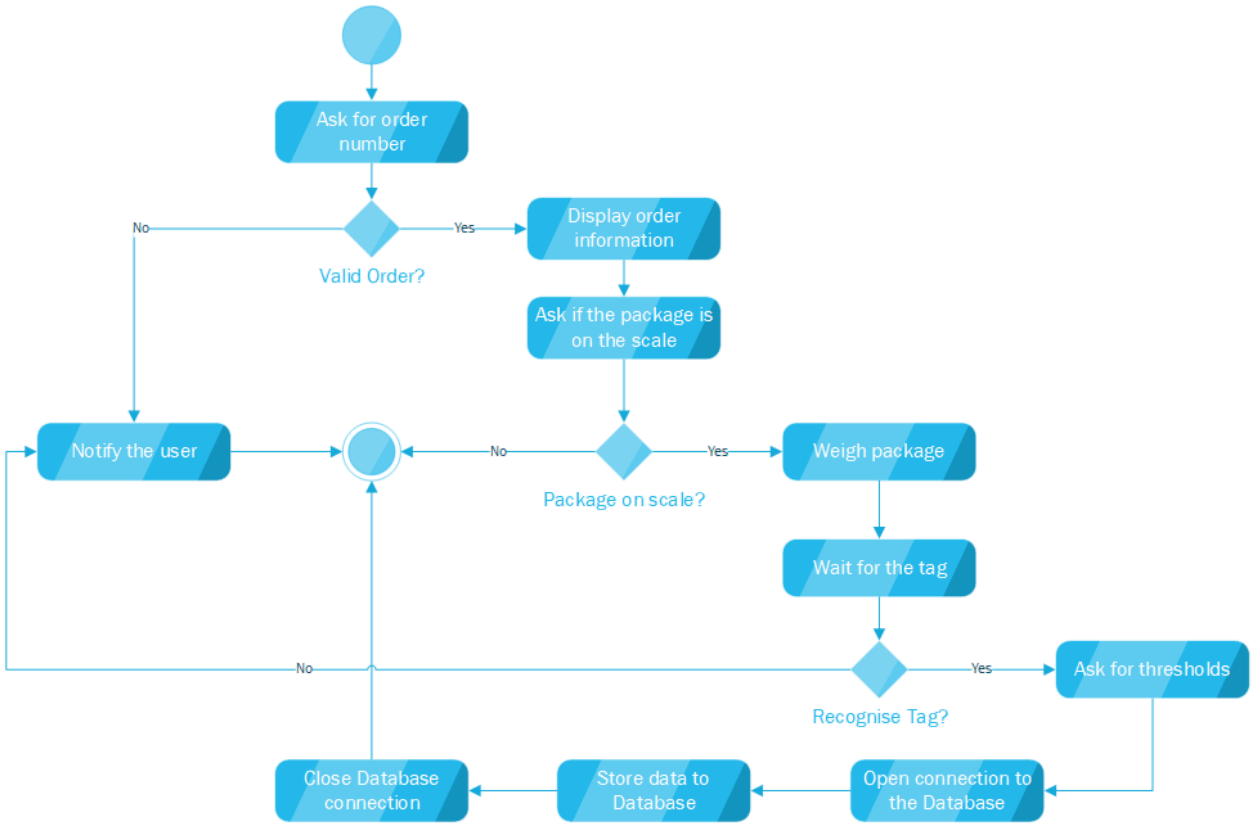
Monitoring Class.

System Schematic

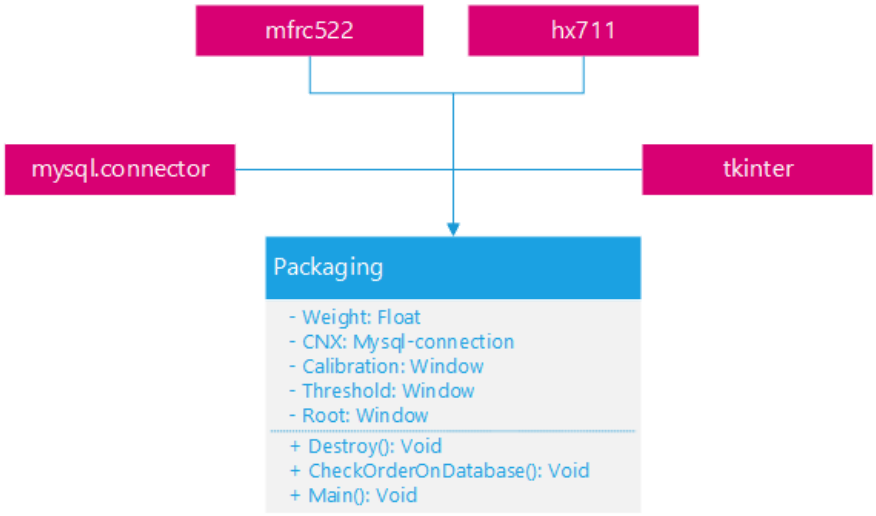


Light, Temperature, and Humidity Monitoring Schematic.

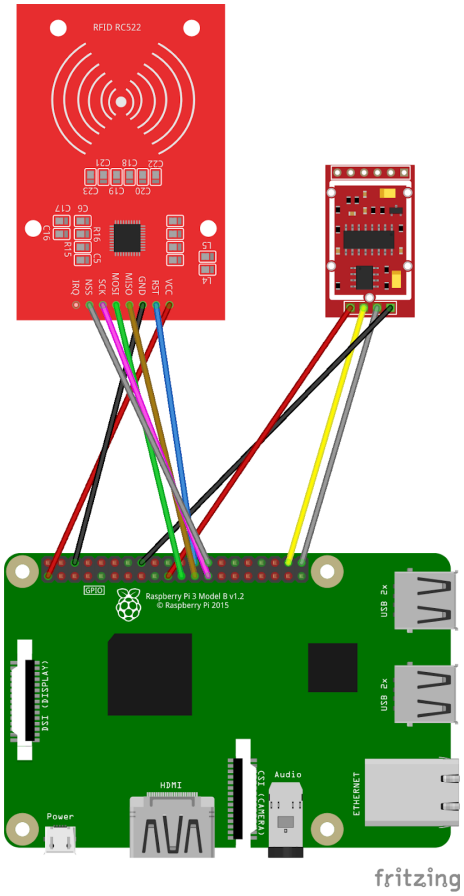
Monitoring



Packaging System Flow Diagram



Packaging Class.

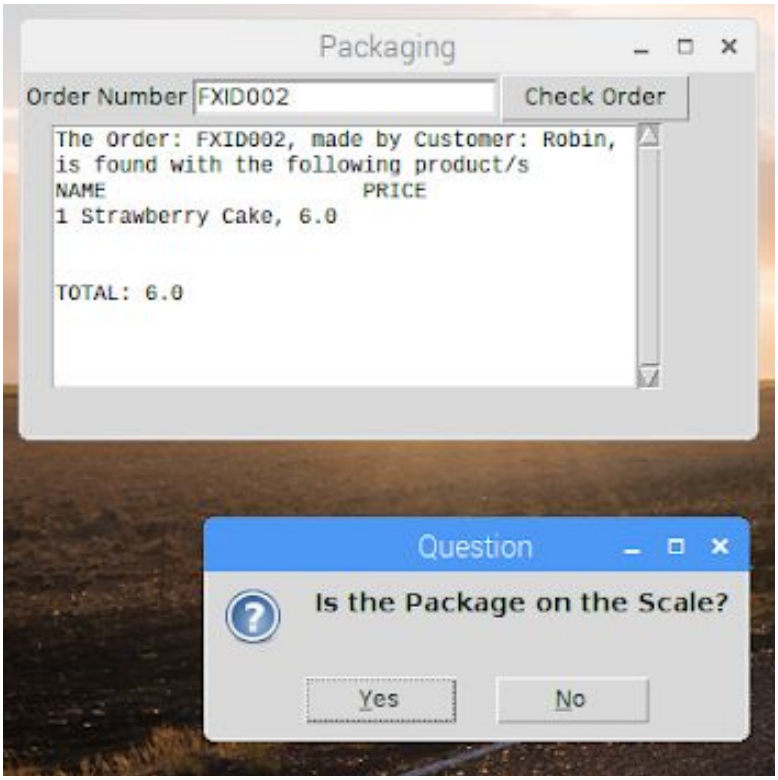


RFID Module and Weighing Scale Schematic.

Running the Solution.

GITHub Solution: https://github.com/IoTLabHNU/i4SC

In order to run the solution, you have to download a GitHub repository from the link above. Then, you will have to, i) install all the mentioned Python packages in Packaging class diagram using pip [e.g. for Unix: python-pip install mysql.connector], ii) connect the weighing scale and RFID module, and iii) you have to run “PackagingGUI_2.py” in Packaging folder of the downloaded repository. The image below depicts the program when is ran. The proposed solution assumes that, there is an order in the database. The user will have to enter the order number into the entry field.



Python Application.

Sorting System

When packaging is done the packages needs to be sorted to go to their predetermined destinations. The sorting machine developed is able to drop off packages at 3 different destinations. The destination is determined by the customers zip code. The main idea is to have multiple vehicles at the zones to drop off and deliver the said packages. A package will arrive on a conveyor belt and placed on the sorting platform. Thereafter a RFID scanner will retrieve the package information and use this information to derive the destination. The sorting program will then check if the package has ever exceeded its temperature and humidity barriers , if it has it will move the sorting arm backwards and dispose of the compromised package. If the destination is decided the sorting arm moves the package forward onto the transport platform which would drop the package at the given destination.

Hardware

- Raspberry Pi - Used for connecting the Brick-pi on top of it.
- Brick-Pi - used to connect all the sensory equipment and motors. Also it extends the GPIO ports from the pi to connect the RFID scanner and the infrared scanner.
- RFID Module – To read the information on the RFID tag.
- RFID Tag – Data stored here to be read by the RFID sensor
- 4 motors , one to run the platform from the docking station to the zone , another for the movement of the belt to move the object off of the platform to the given zone. The last one is to power the rotor that moves the package onto the platform or remove it from the sorting system if it has been compromised.
- Lego - To build the model sorting system
- Ultrasonic sensor - detects if a package is placed on the belt and will activate the belt to move the package to the sorting system.

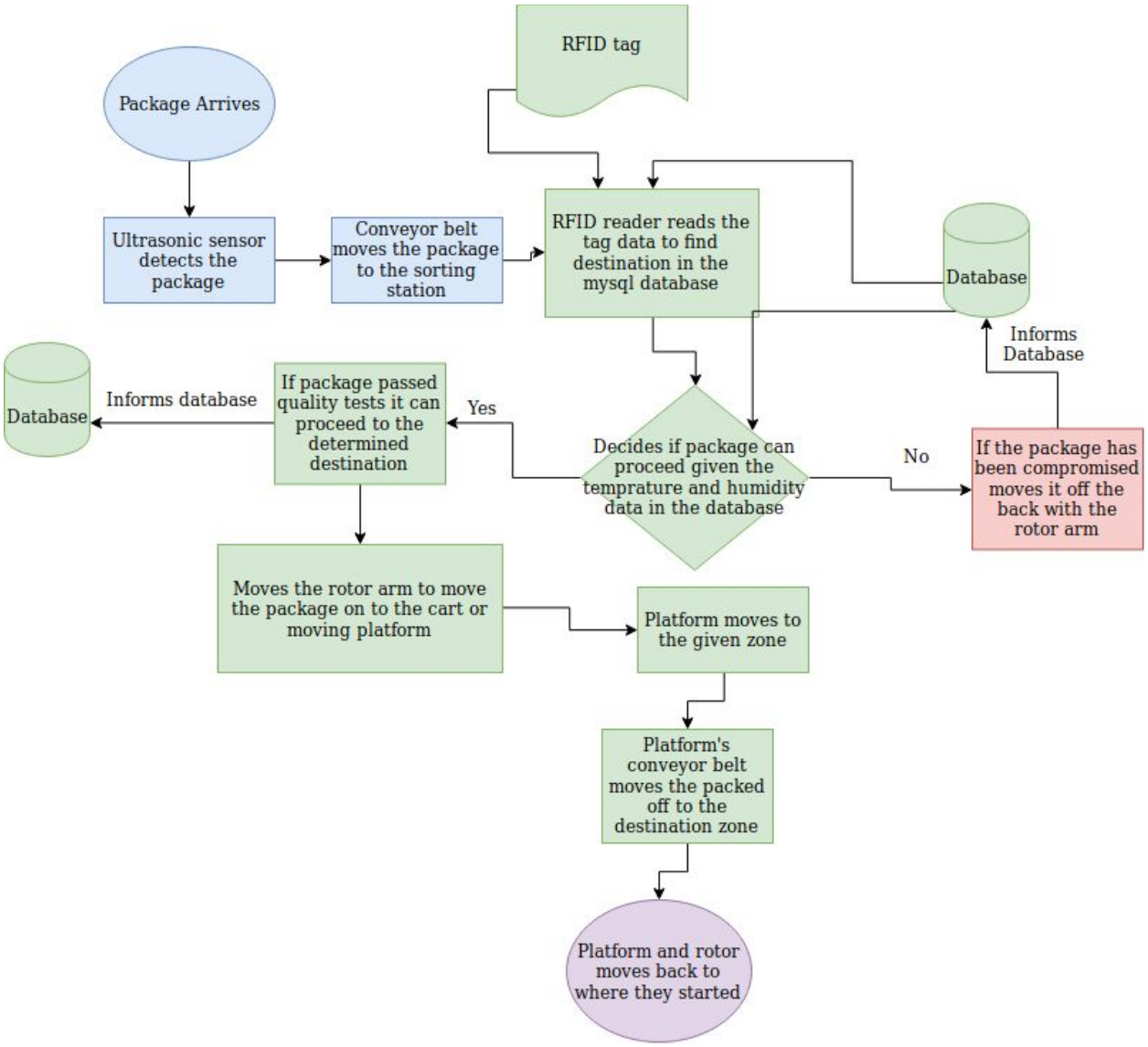
Software

- MySQL – To implement a relational database that will store customer, product, and order information.
- Mindstorm - used to program the conveyor belt.
- Python Programming Language – Below are package can be used for implementing the proposed solution:
    - GPIO – A package to define and select which pins are being used for what.
    - MySQL.Connector – A package used for connecting a client to MySQL server.
    - MFRC522 – A package for commanding the RFID module to read or write information on the tag.
    - BrickPi - a packages with modules concerning the motors labeled A-D

This system is broken up into 2 systems at this point , a conveyor system to deliver the package to the sorting system. The conveyor system is programed with the mindstorm software and is not currently connected to the sorting platform , they are separate systems. While the package sorting transaction is running , the script does query the mysql database for information and also push data to it. This Data will be used to track the order and in the graphical interface of graffana.

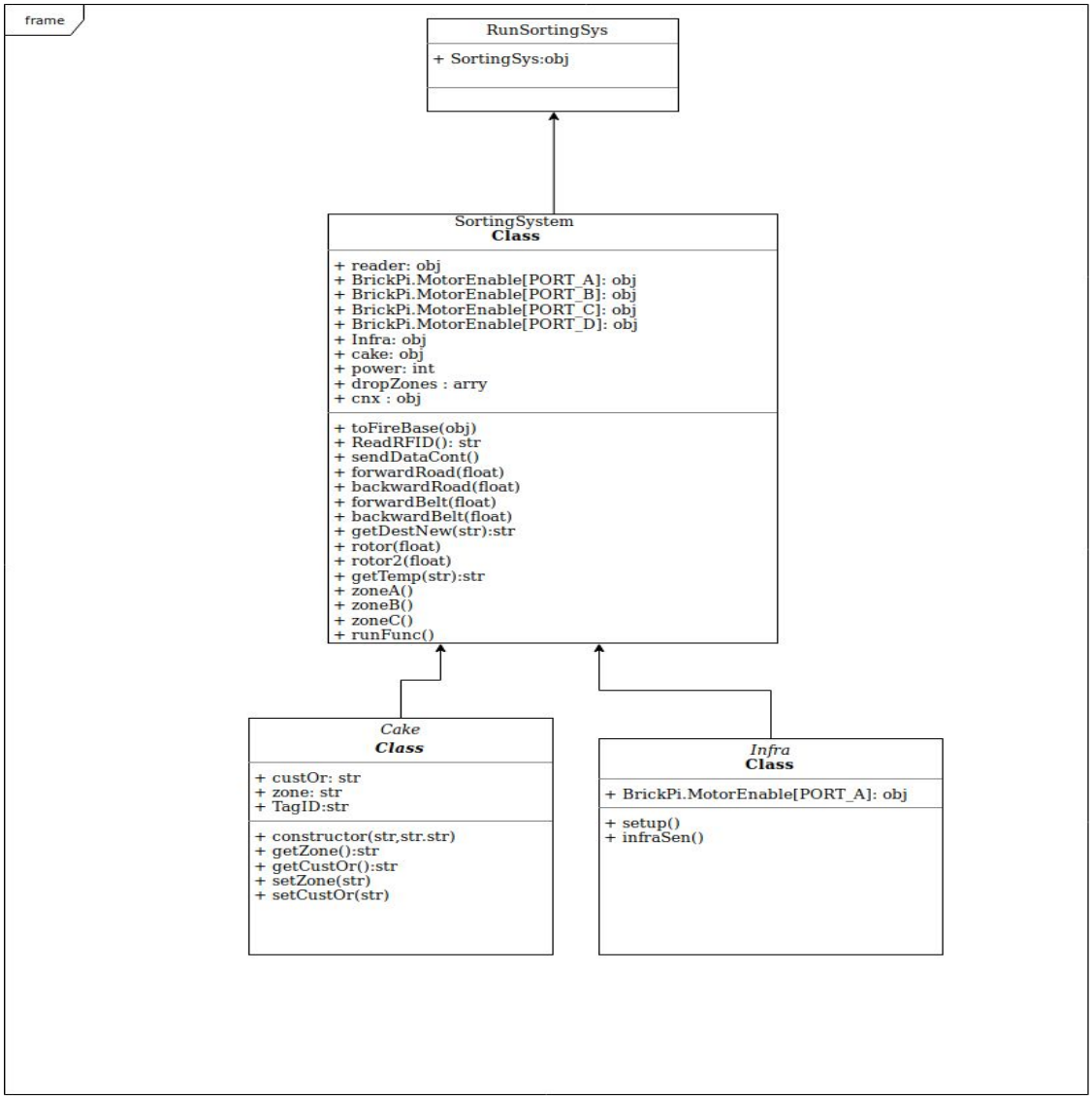
The below figure will show the flow packages through the sorting system.

System Flowchart



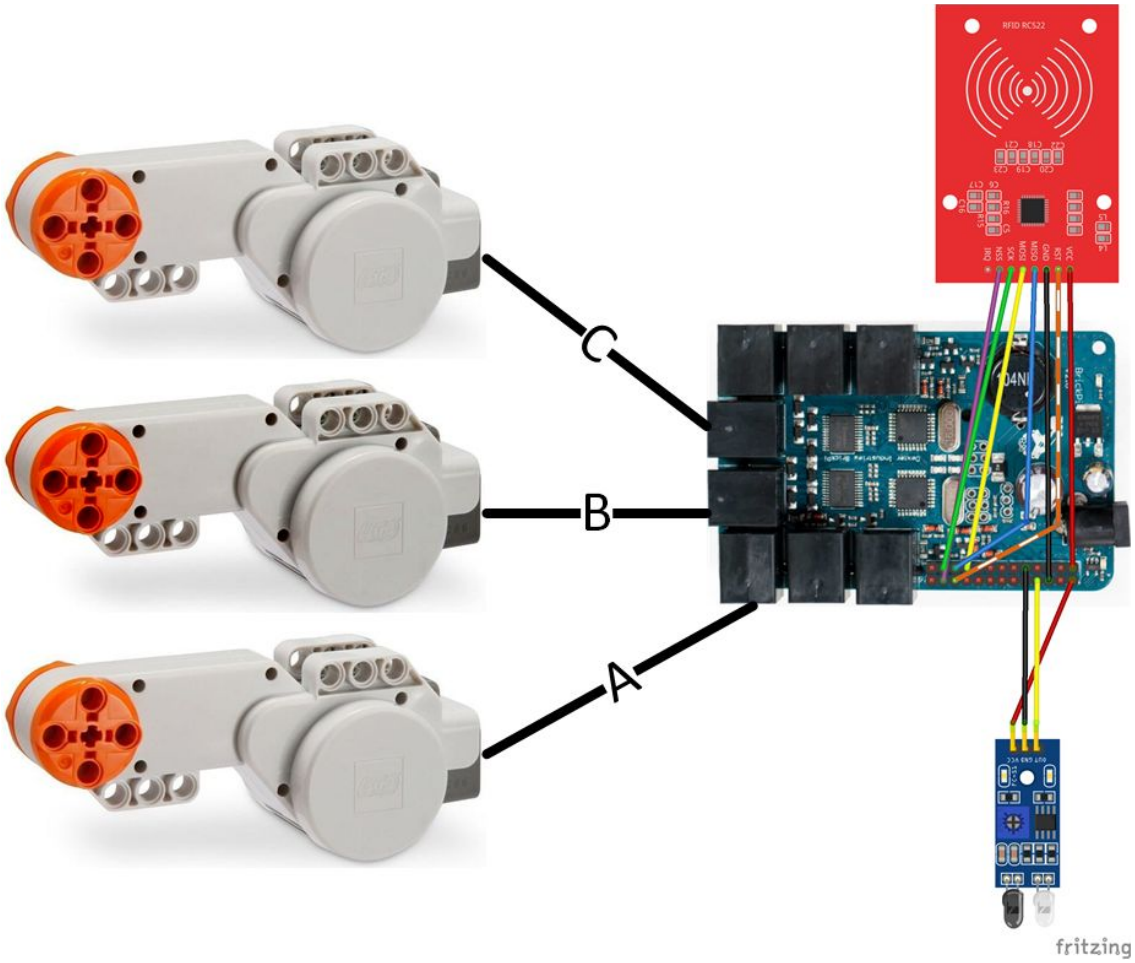
Packaging System Flow Diagram

Class Diagram



Class Diagram for Sorting System

System Schematic

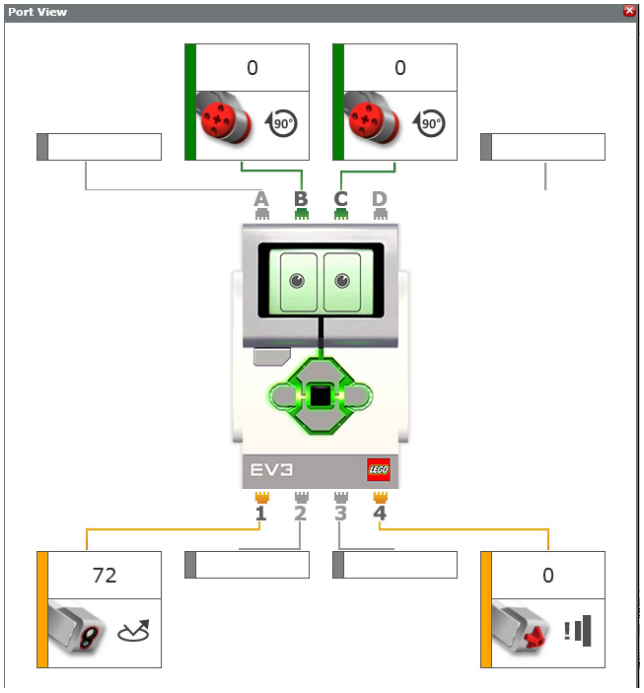


Sorting System Schematic.

Delivery System

The Lego Mindstorms EV3 brick was used to construct the Autonomous Transport Vehicle (ATV), using two motors for movement, a light sensor to follow the line to the storage location and a touch sensor to sense when the ATV is loaded. See below how the motors and sensor are connected to the EV3.

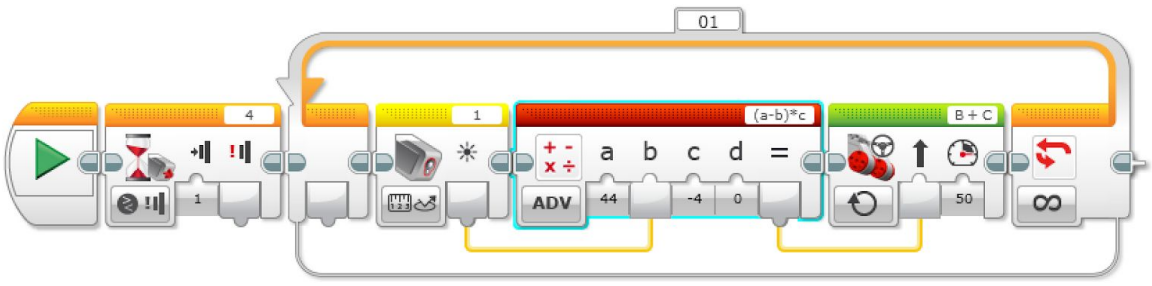
System Schematic



Mindstorms Brick and Motor and Sensor Schematic

The ATV code was developed using the Lego Mindstorms Education EV3 Student edition, using building blocks similar to scratch. The code blocks are shown below:

Code Block

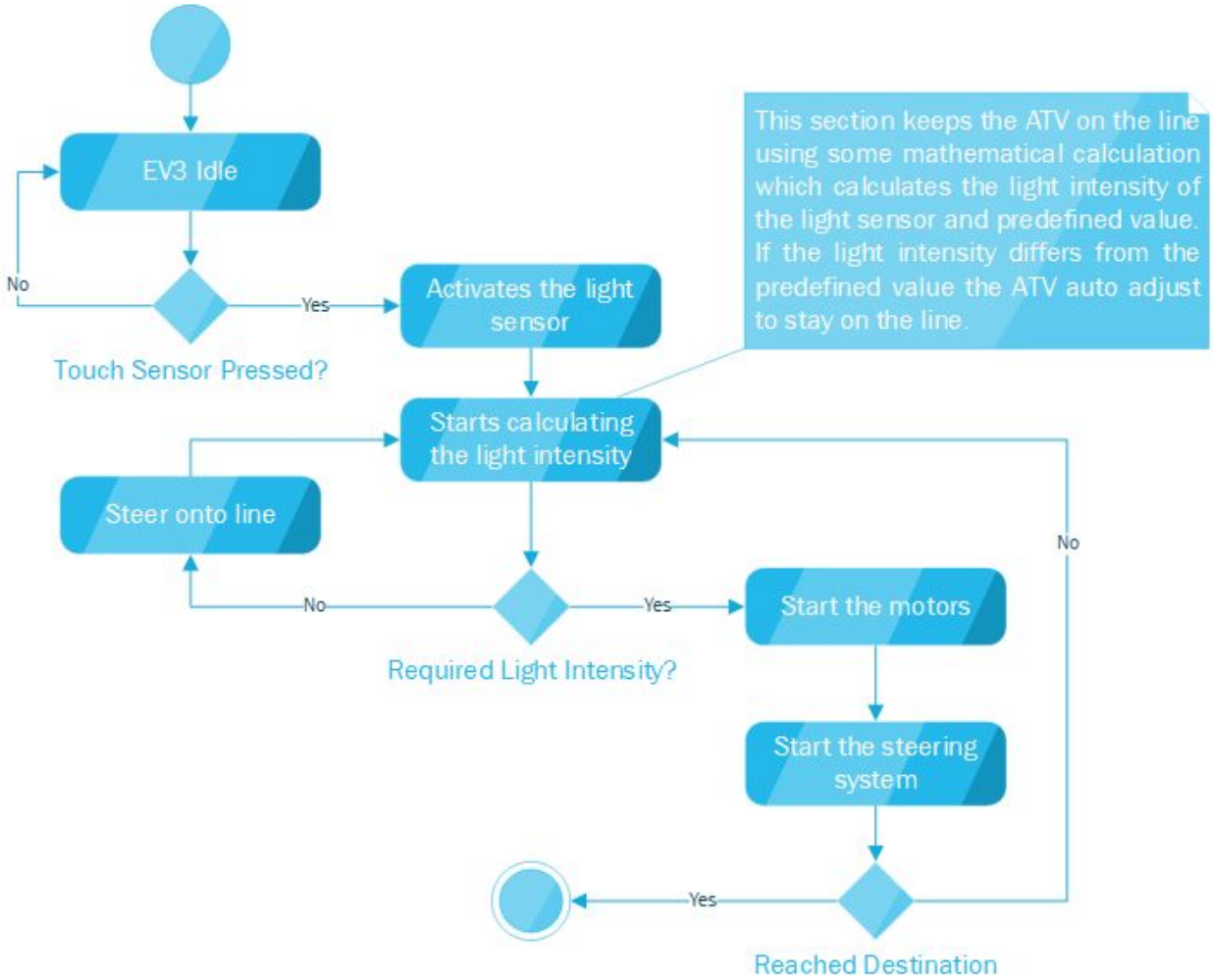


Mindstorms Code Block for ATV Line Follower

The code structure above has 6 (six) sections:

Section 1: Start the EV3
Section 2: Activates the touch sensor and wait for input, “button pressed”
Section 3: Activates the light sensor and starts calculating the light intensity
Section 4: This section keeps the ASR on the line using some mathematical calculation which calculates the light intensity of the light sensor and predefined value. If the light intensity differs from the predefined value the ASR auto adjust to stay on the line.
Section 5: Start the motors and steering system, this only happens when the touch sensor is pressed.
Section 6: Section 2 - 4 are enclosed in a loop, which allows for the ASR to deliver the production until

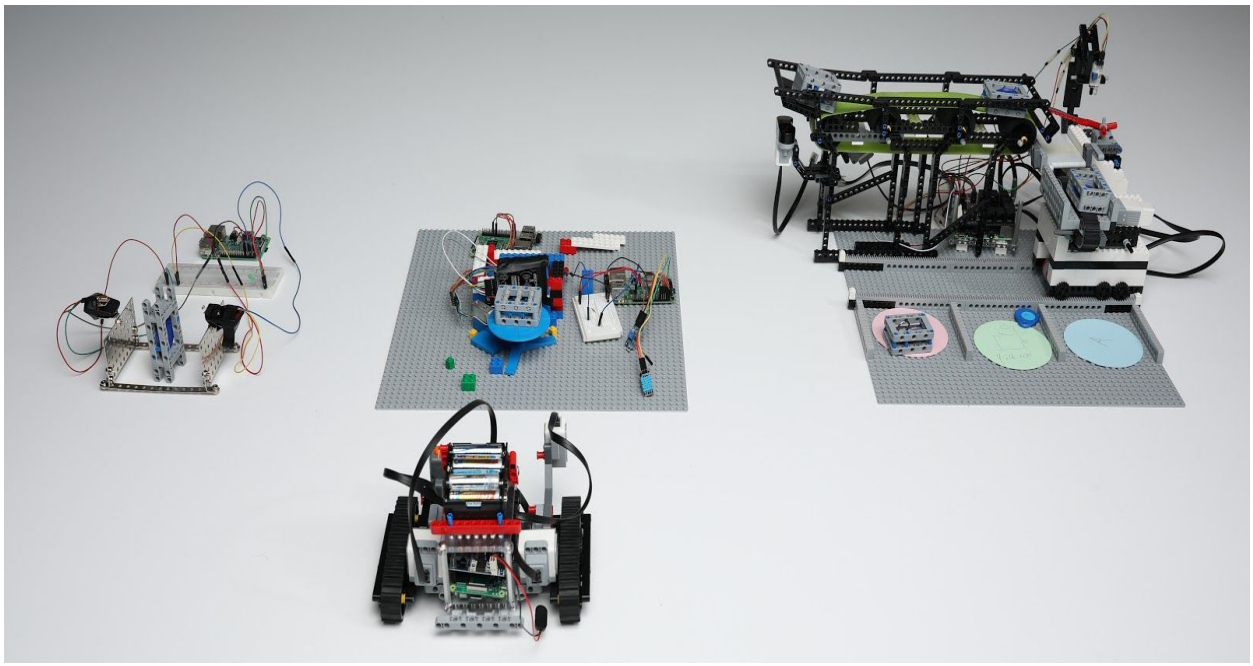
System Flowchart



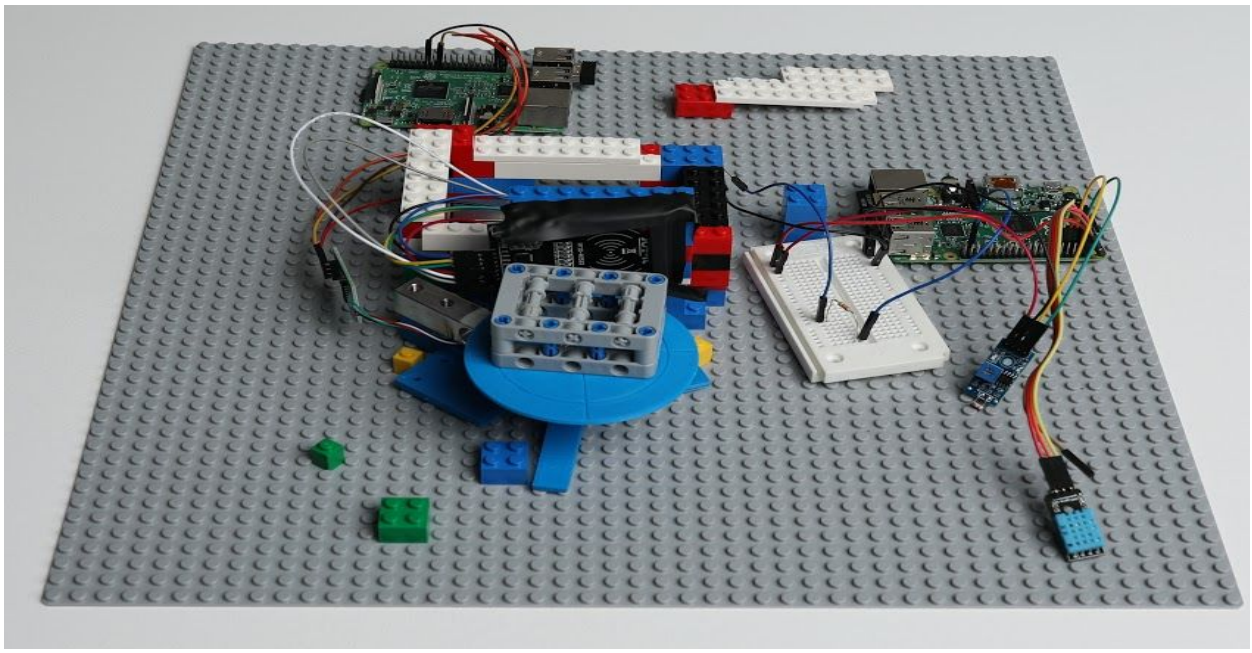
ATV Line Follower Flow Diagram

Pictures / Videos

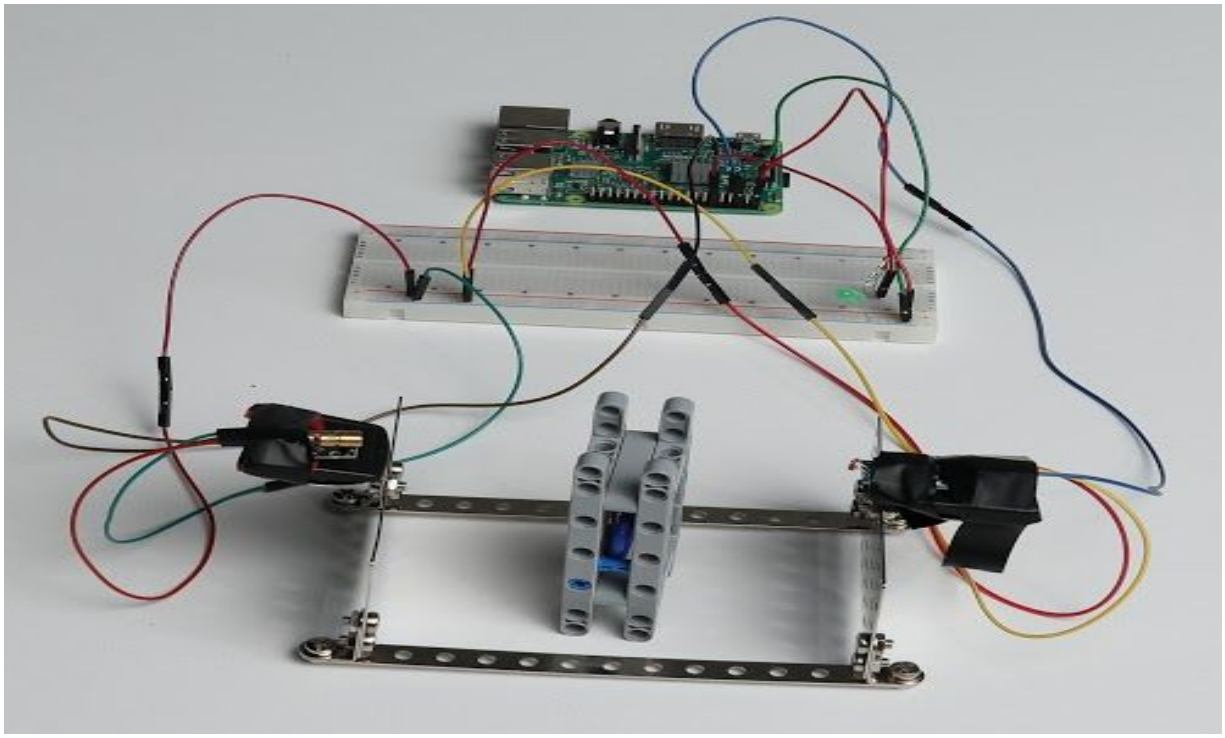
Complete solution:



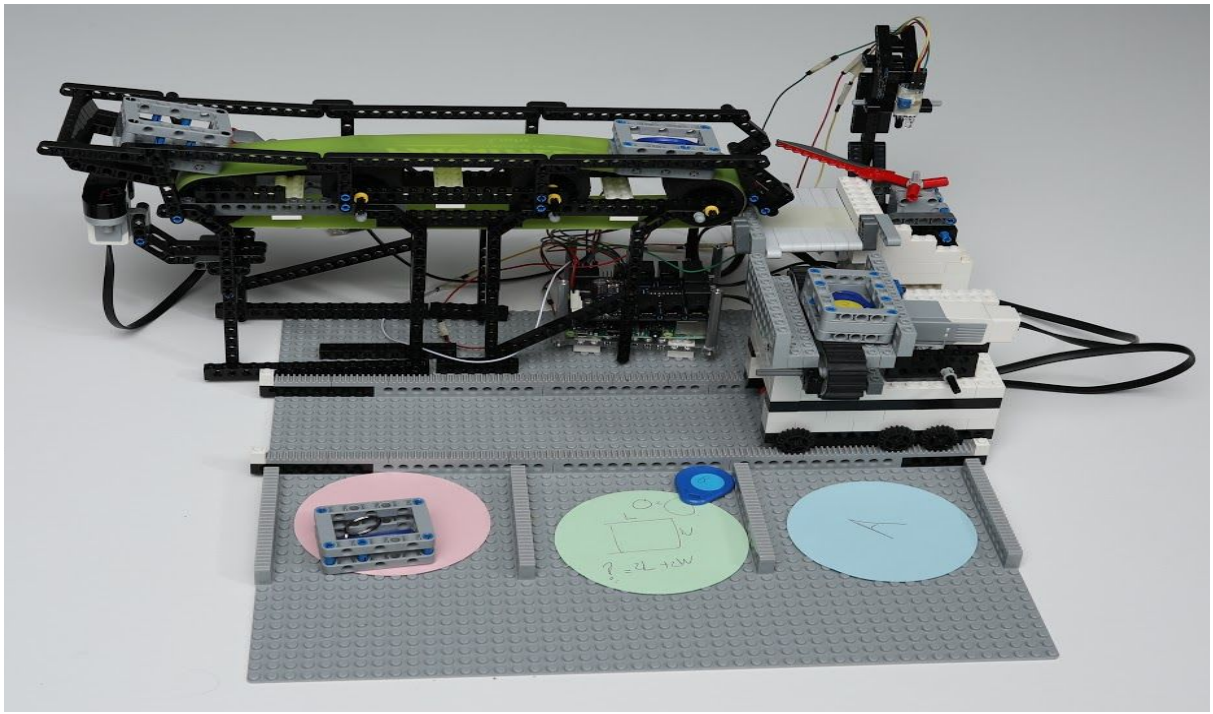
Packaging



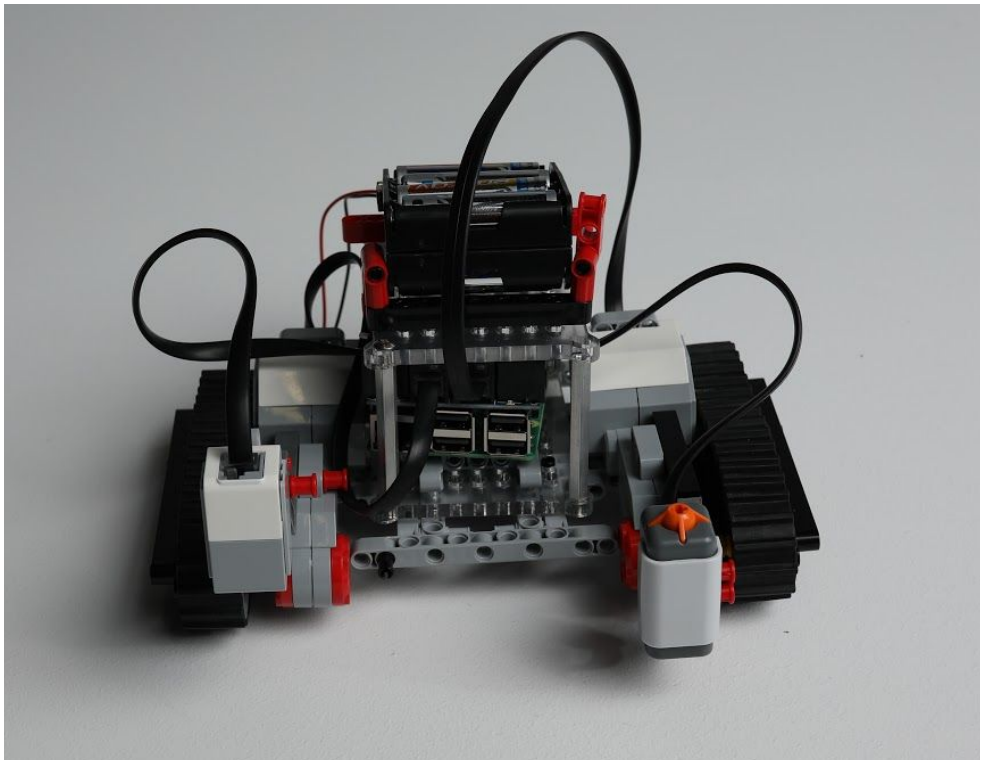
Counting



Sorting



Delivery



Future Enhancements

Packaging

1. Add RFID tag inside the package to verify and count what is inside.
2. Add more sensors (accelerometer, gyroscope, moisture, smoke, and GPS) to extend the packing container to allow for pharmaceutical and other sensitive products.
3. To make the bakery fully autonomous, a robot could be used to collect the cakes from a predefined shelf and place it onto the packaging station.
4. Implement a security monitoring system to verify who opens the package.
5. Improve the analysis script to check if the package is deviating from the planned route.

Counting

1. Add one more laser to determine the direction in which the object will be moving.
2. Use a more stable structure to align the laser emitter and light sensor (LM393 Dual Comperator and laser emitter modules (HW-483)).

Sorting

1. Add a RFID sensor and tag combo to identify drop zones instead of predefining their position.
2. Add a infrared sensor to the moving platform to detect when it needs to stop moving to its docking station.
3. Combine the both the sorting platform and the delivery conveyor belt

Delivery

1. Develop code for automatic stopping mechanism.
2. Construct a forklift mechanism and developed operational code to be used in the packaging system to collect cakes from predefined shelves.
3. Return to the sorting machine and wait for the next parcel.
4. Implement the whole delivery solution using a Raspberry Pi with a BrickPi module to control the motors and read the sensors.
5. Adding a GPS module, which allows for delivery based on GPS coordinates.
6. Add an alert system (SMS or email) which allows the ATV to inform the bakery that delivery has been successfully completed.

References

Packaging

1. https://www.raspberrypi.org/documentation/usage/gpio/
2. https://www.raspberrypi.org/forums/viewtopic.php?p=531442

Counting

1. https://pinout.xyz/
2. https://raspi.tv/2013/rpi-gpio-basics-3-how-to-exit-gpio-programs-cleanly-avoid-warnings-and-protect-your-pi

Sorting

1. https://www.raspberrypi.org/documentation/usage/gpio/
2. https://www.dexterindustries.com/brickpi/
3. https://firebase.google.com/docs/database/android/start
4. https://medium.com/coinmonks/for-beginners-how-to-set-up-a-raspberry-pi-rfid-rc522-reader-and-record-data-on-iota-865f67843a2d
5. https://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python

Delivery

1. https://www.flickr.com/photos/95054900@N08/sets/72157638050074174/
2. https://gist.github.com/moriarty/3944238
3. https://gist.github.com/CS2098/ecb3a078ed502c6a7d6e8d17dc095b48
4. https://www.handsontech.co.za/lego-education-mindstorms-ev3.html
5. https://www.lego.com/en-gb/mindstorms/build-a-robot/bobb3e
6. https://www.youtube.com/watch?v=_YRwKHP4v8A
7. https://www.flickr.com/photos/95054900@N08/sets/72157638050074174/
8. https://www.youtube.com/watch?v=5NVTbAigKWk
9. https://www.lego.com/en-us/service/buildinginstructions

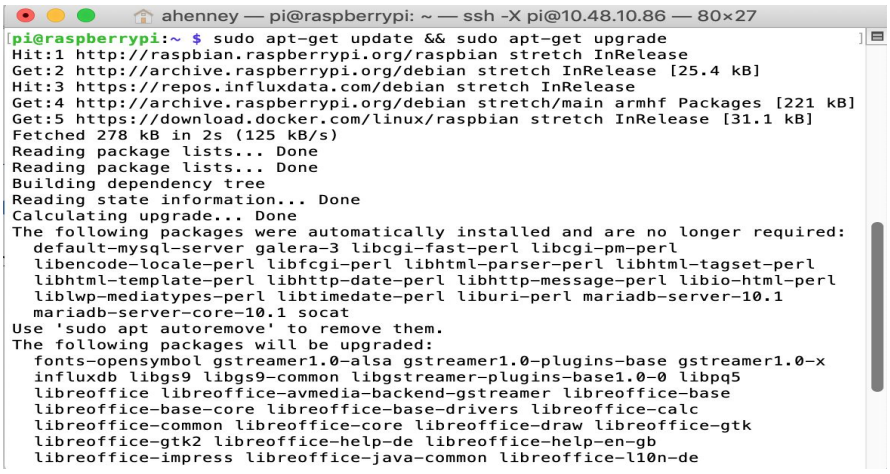
Appendix

MySQL Operations on Raspberry Pi

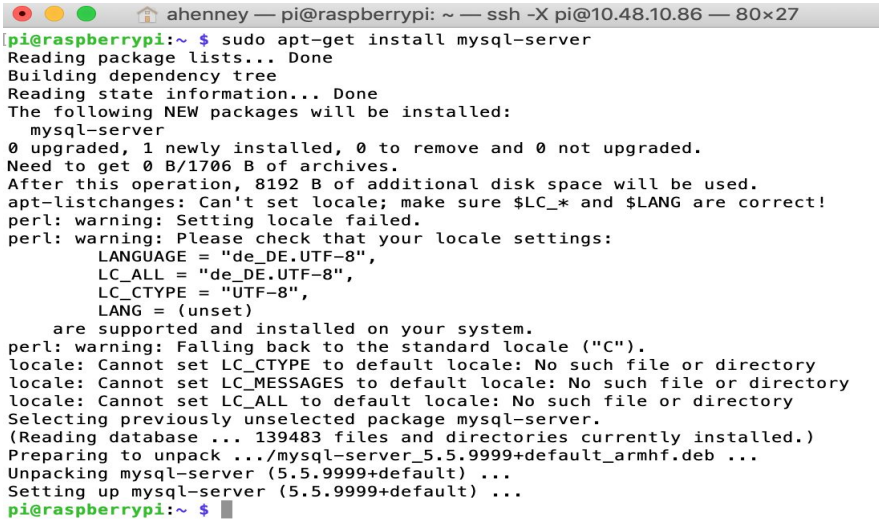
MySQL installation on Rasberry Pi

Type following commands from the command line:

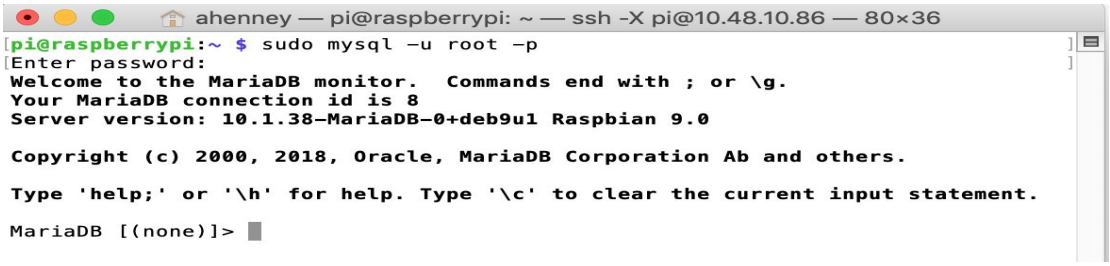
1. First, make sure everything is up top date.
   - Run the following command: **sudo apt-get update && sudo apt-get upgrade**



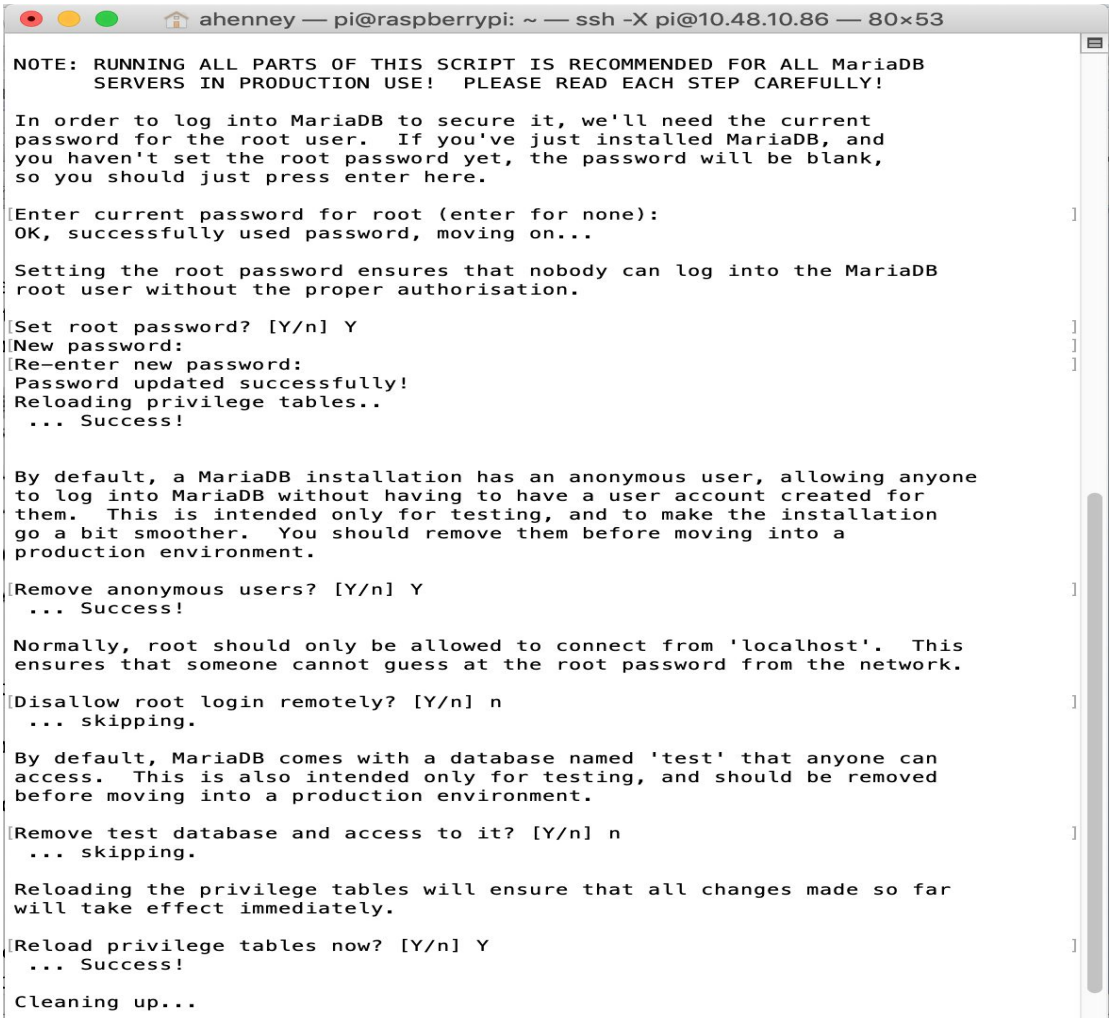
2. Install the MySQL server onto the Raspberry Pi.
   - Run the following command: **sudo apt-get install mysql-server**



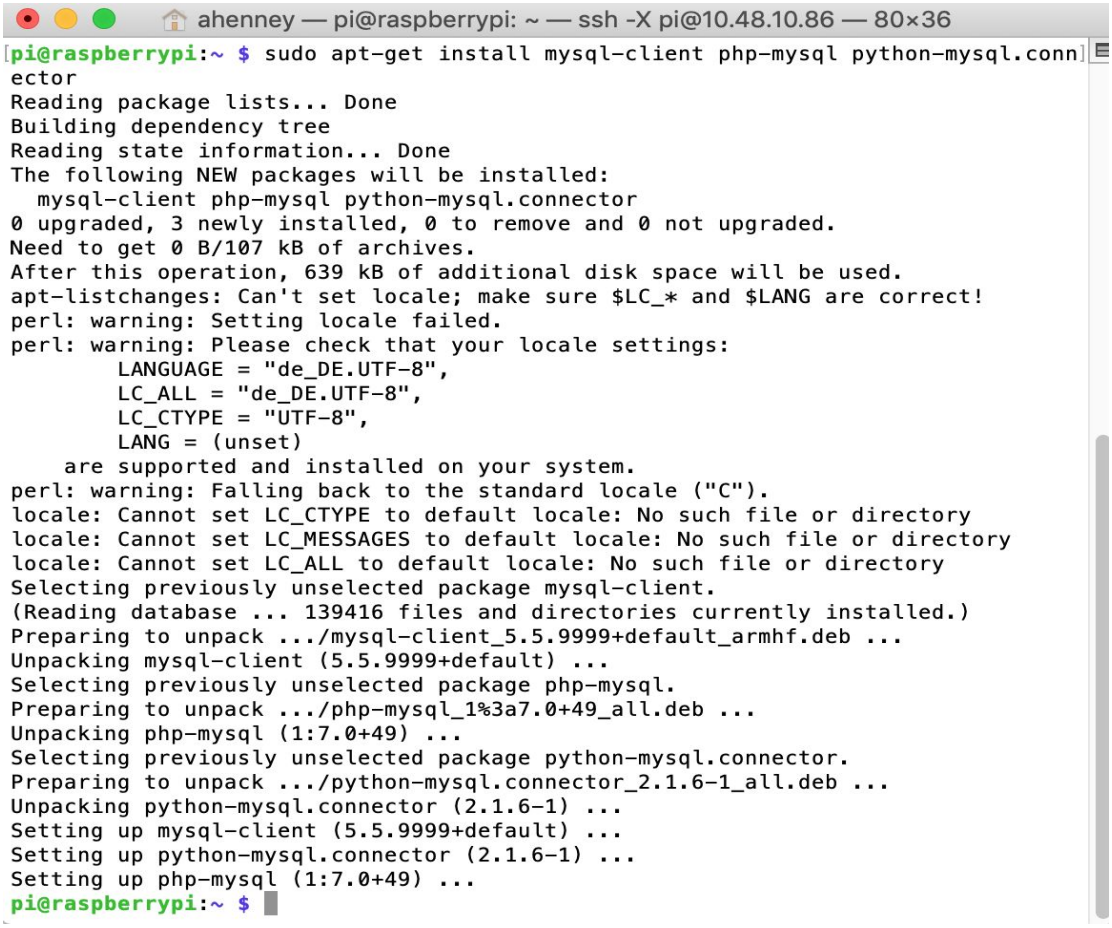
3. If you're not prompted to enter a password, the default password will be an empty field when you run the following command with sudo:
   - Run the following command: **sudo mysql –u root**



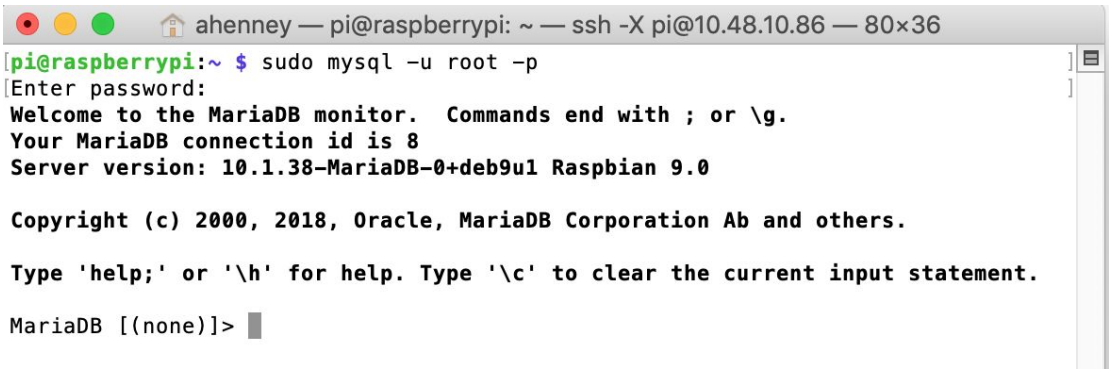
4. To secure the roo login with the password, you will need to enter the following command.
   - Run the following command: **sudo mysql_secure_installation**
     Answer all the questions so it will be set up to match your security requirements.



5. Install a couple of extra packages, php5-mysql, python-connector, mysql-client. The php5-mysql package allows connections to be made to MySQL Server through PHP, the python-mysql.connector package allows connections to be made to MySQL Server through Python and mysql-client allows us to connect to our local MySQL server through the CLI:
   - Run the following command: **sudo apt-get install mysql-client php-mysql python-mysql.connector**

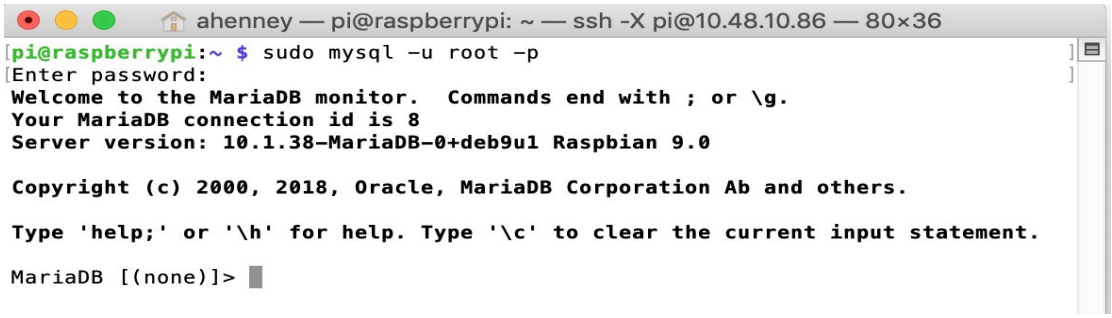


6. Test your login to the MySQL database using root with password.
   Run the following command: **sudo mysql –u root –p**

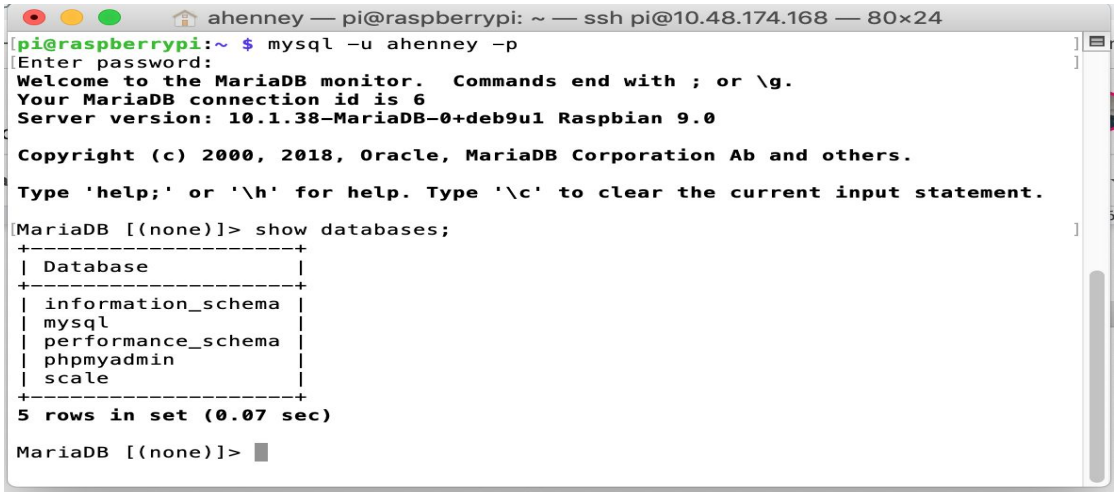


How to view, create and select a MySQL databases

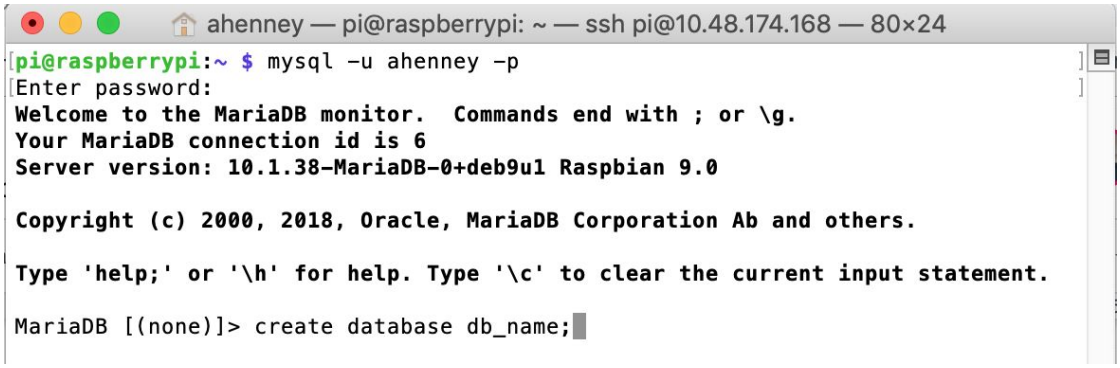
1. Login into your MySQL database.
   Run the following command: **sudo mysql –u username –p**



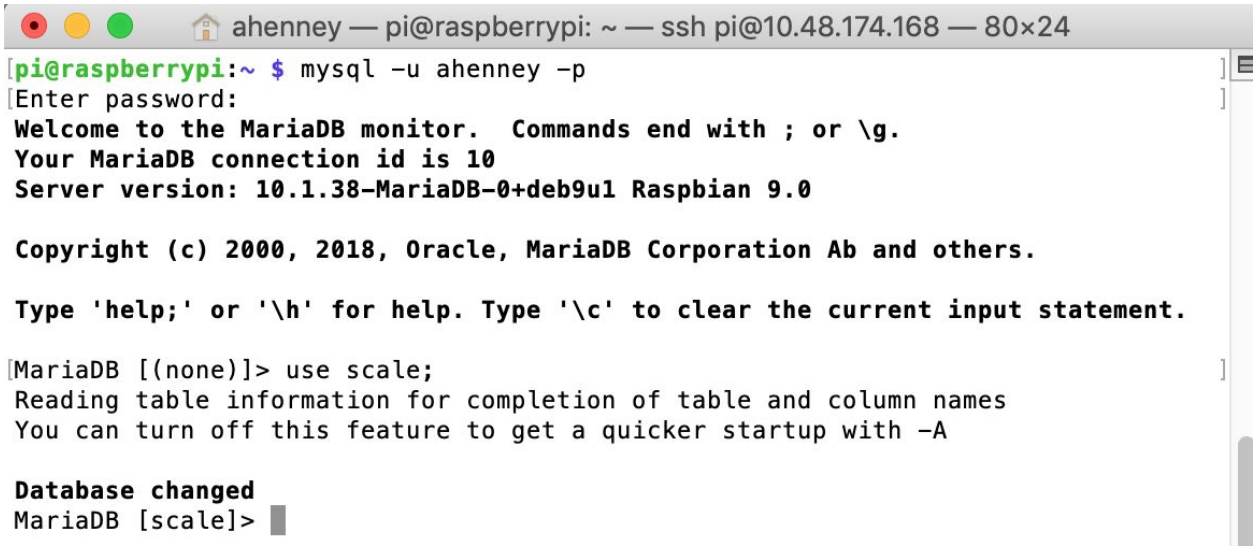
2. To view all MySQL databases:
   Run the following command: **show databases;**



3. To create a MySQL database.
   Run the following command: **create database db_name;**



4. To select a specific database :
   Run the following command: **use db_name;**

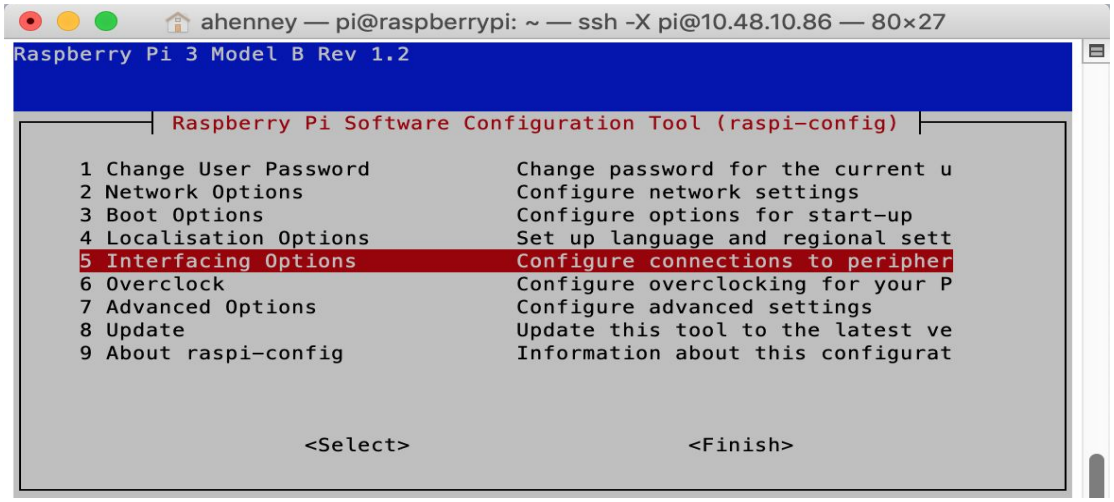


Installing and enabling SSH on a Raspberry PI

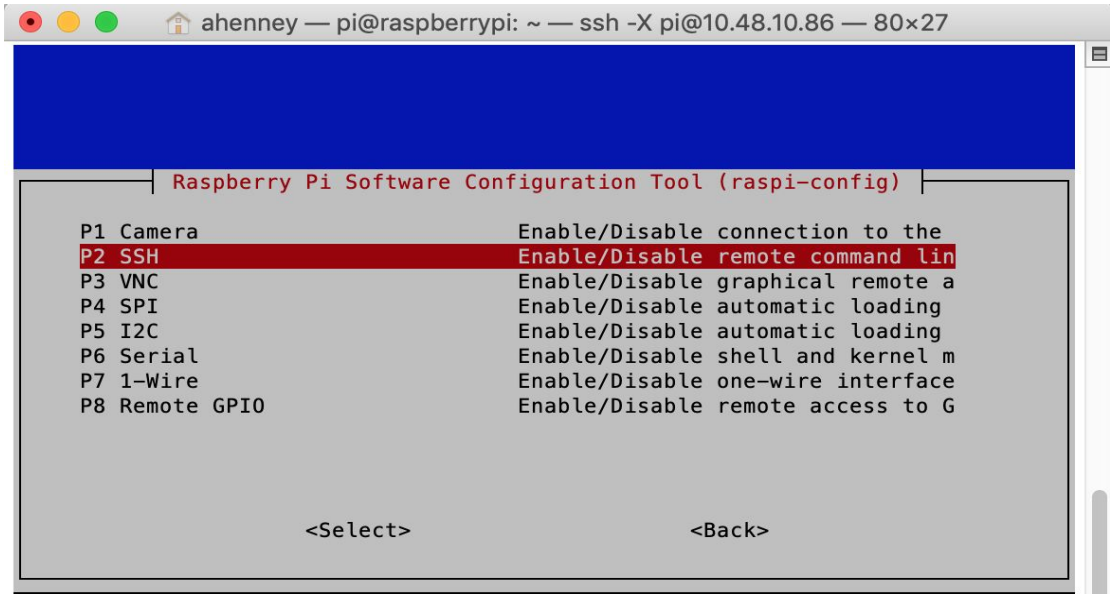
Launch Raspberry Pi Configuration from the command line:

Enter sudo raspi-config in a terminal window

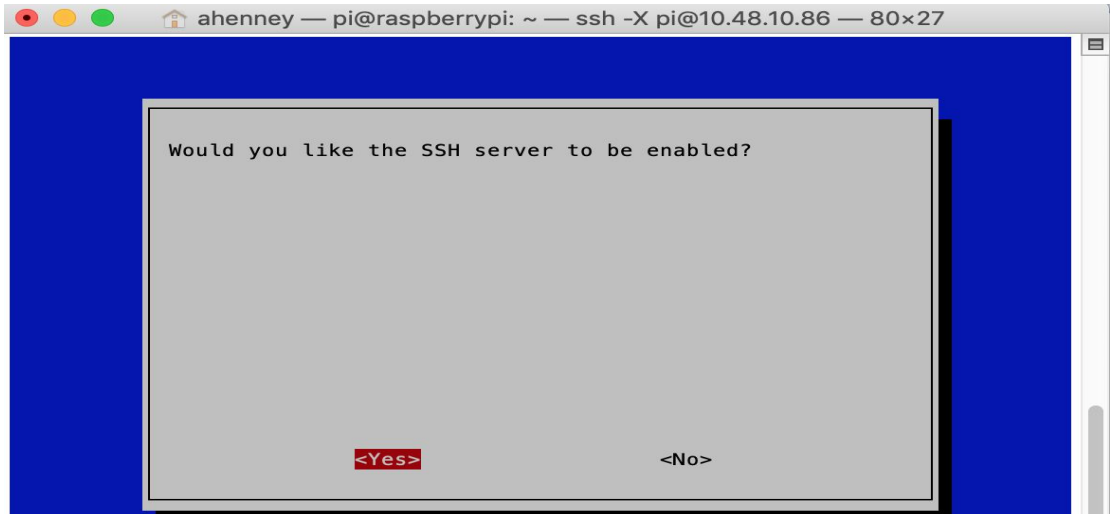
Select Interfacing Options



Navigate to and select SSH



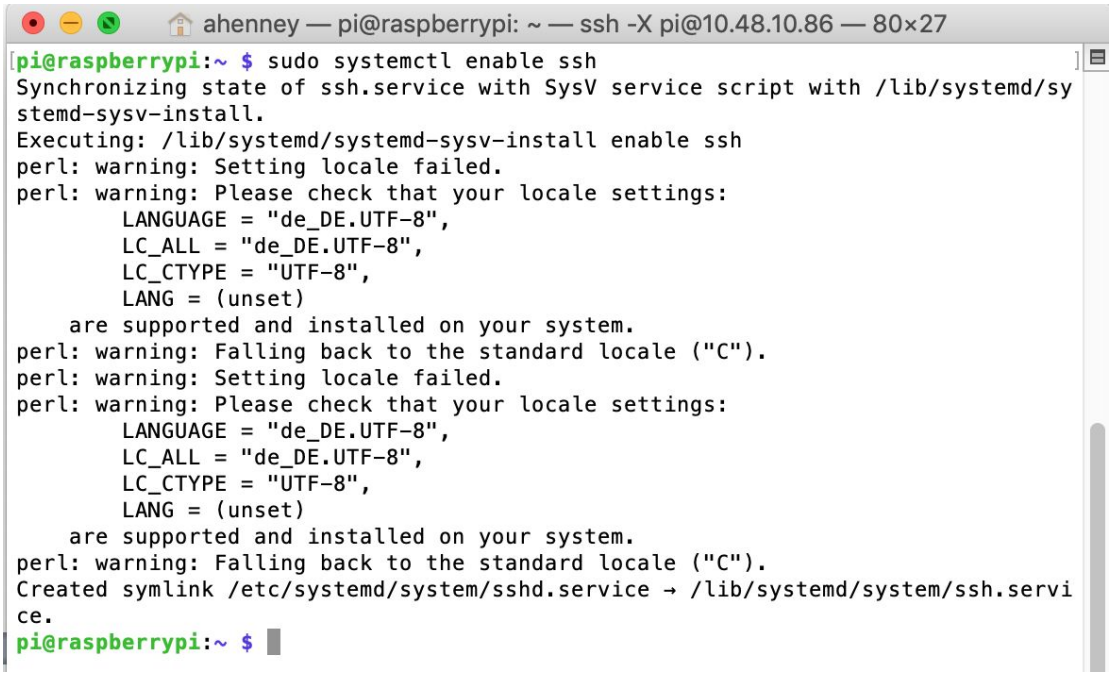
Choose Yes



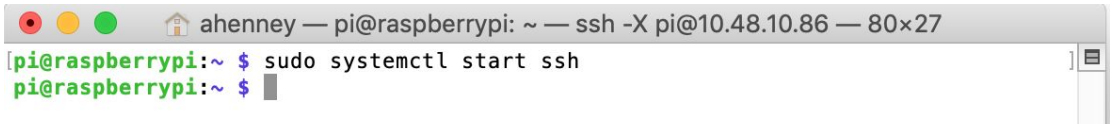
Select Ok

Alternatively, use systemctl to start the service

1. sudo systemctl enable ssh



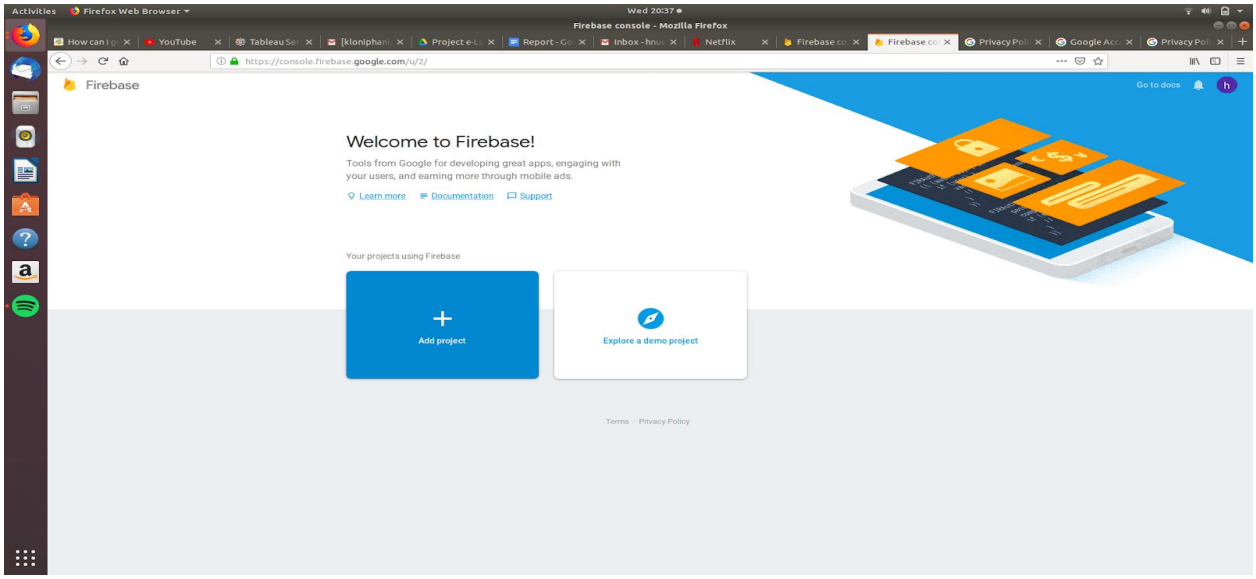
2. sudo systemctl start ssh



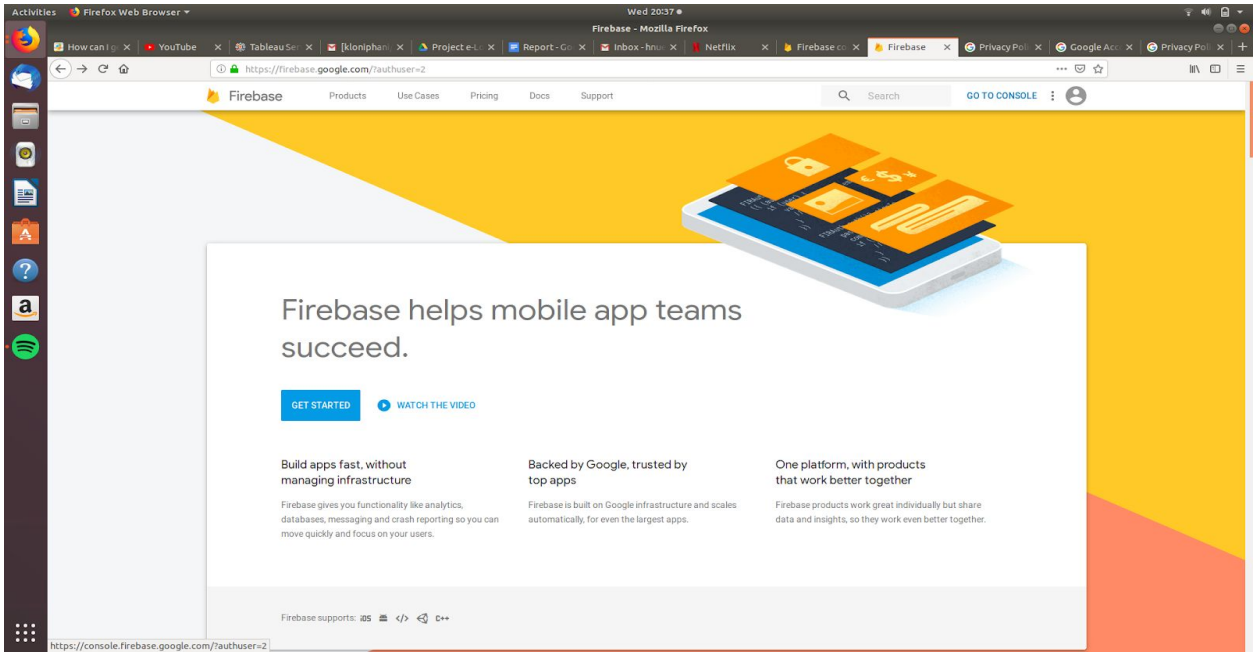
How to Install Firebase and Grafana

Creating the real-time NoSQL Firebase database

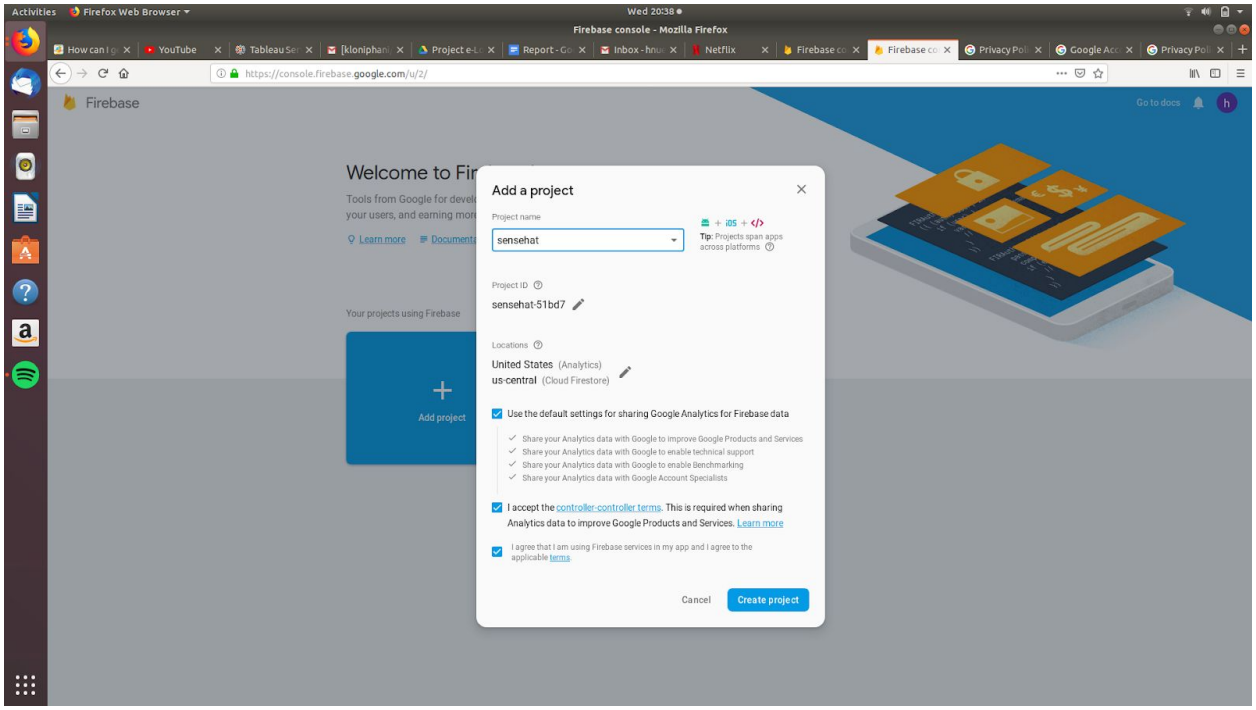
Step 1 : Go to the Firebase homepage and click on the **Go to console** link on the top right corner and login with your Google account details if your not logged in.



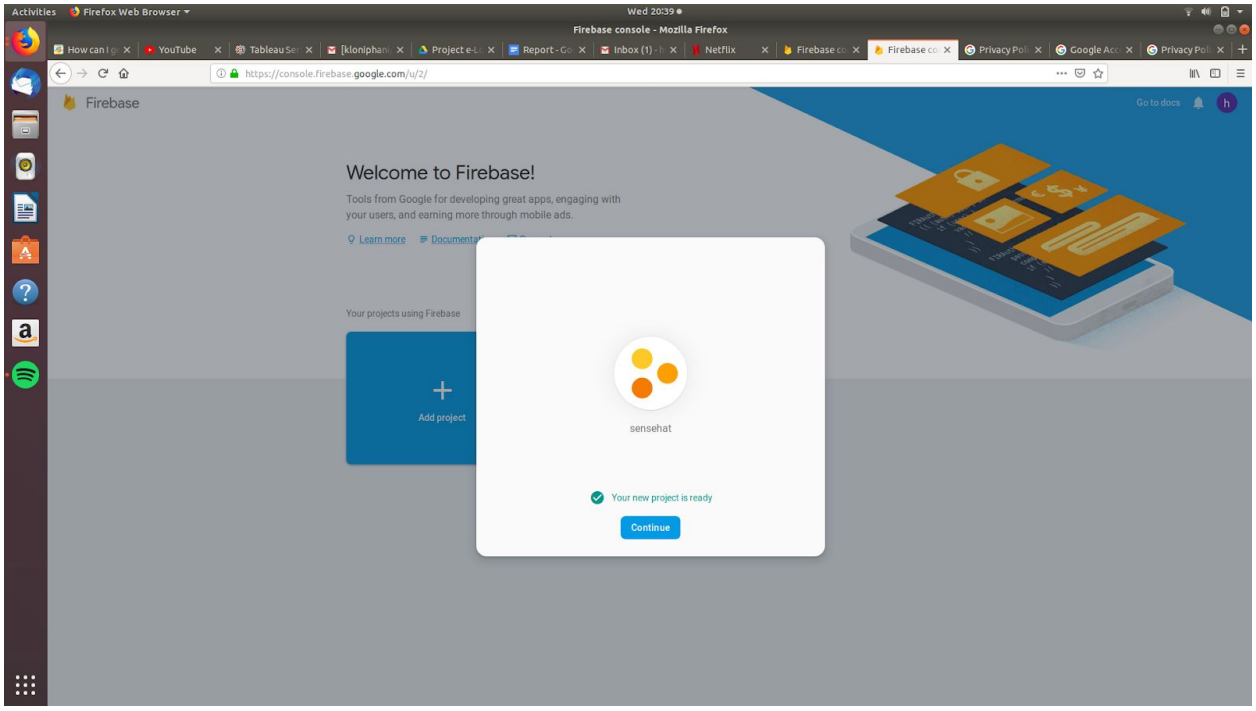
Step 2: To create a new project click on **Add project**.



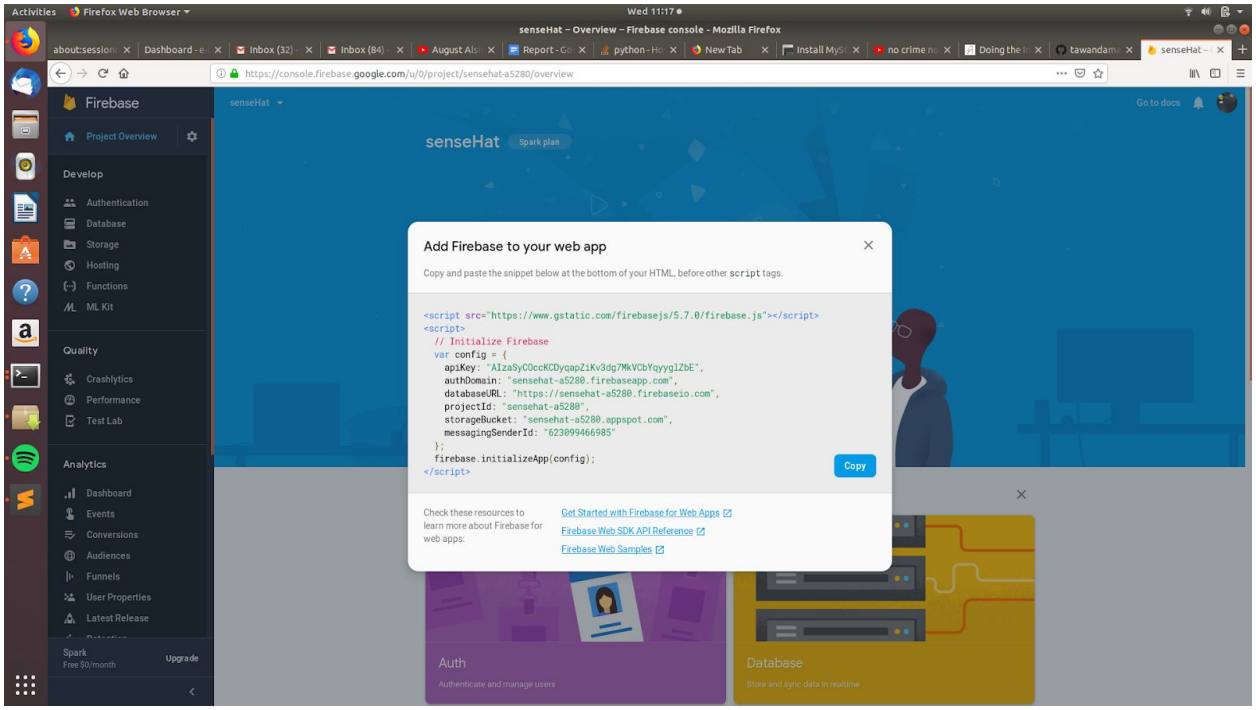
Step 3: A dialog will pop up, enter the **Project name** and your **Location** and agree to the Firebase terms and conditions and click on the **Create Project** button.



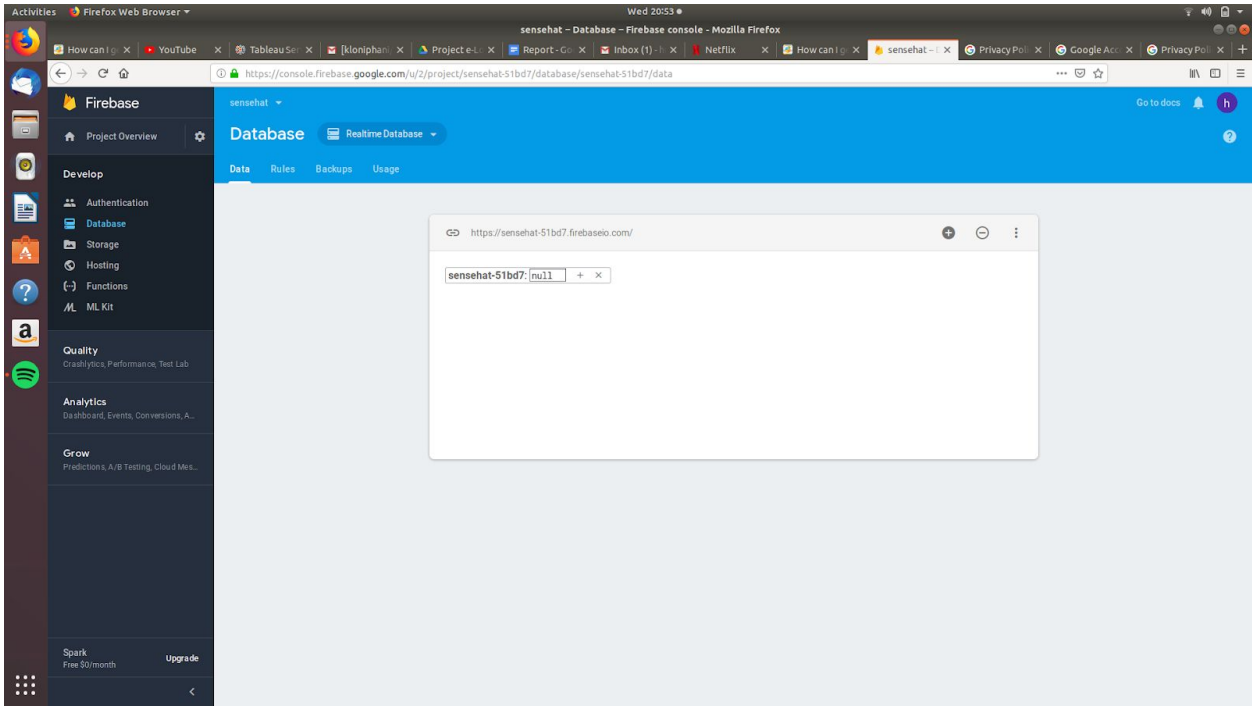
Step 4: Click on the **Continue** button.



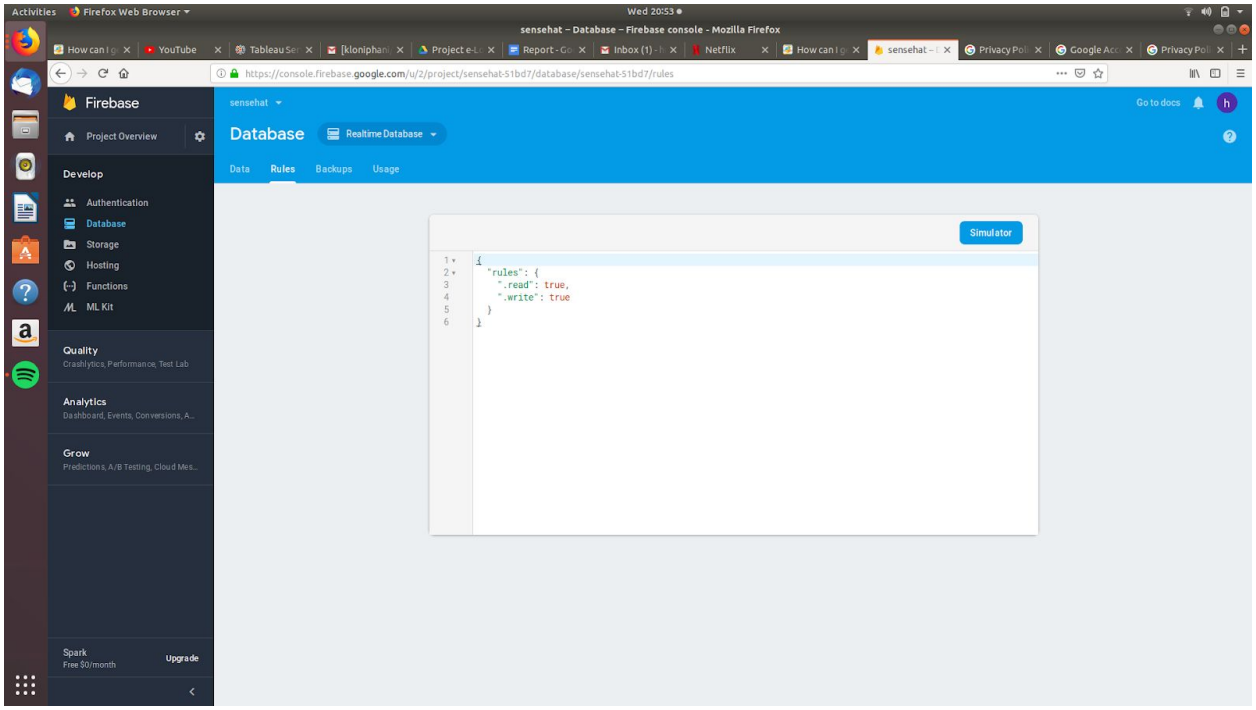
Step 5: Click on the **</>** link and save the Firebase database API keys and details that appear.



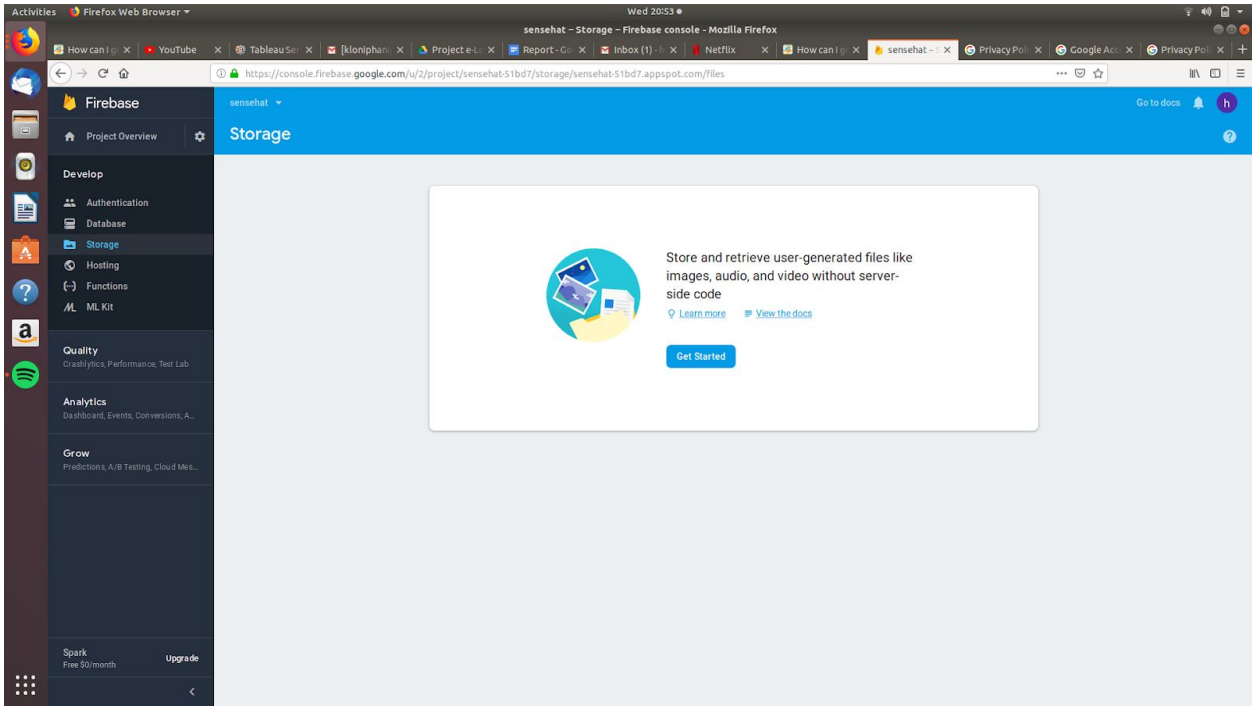
Step 6: Click on the **Databases** tab on the left.



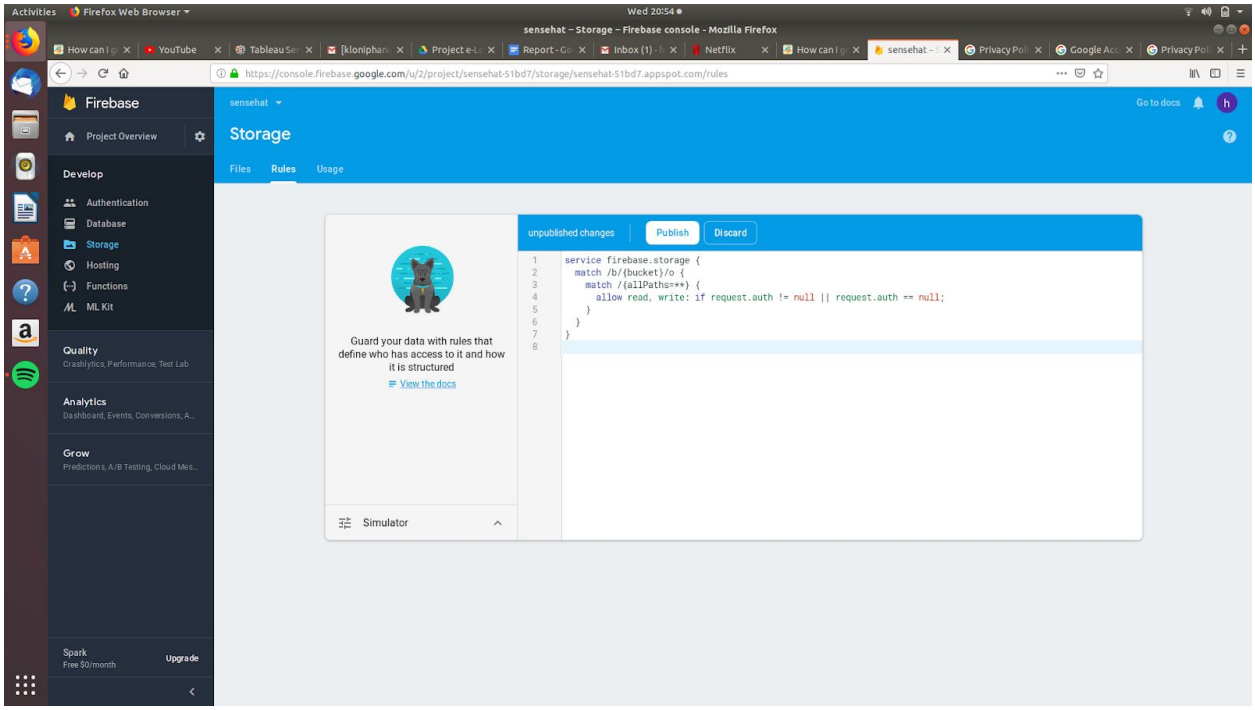
Step 7: Click on the **Rules** and set them to what is shown in the image below.



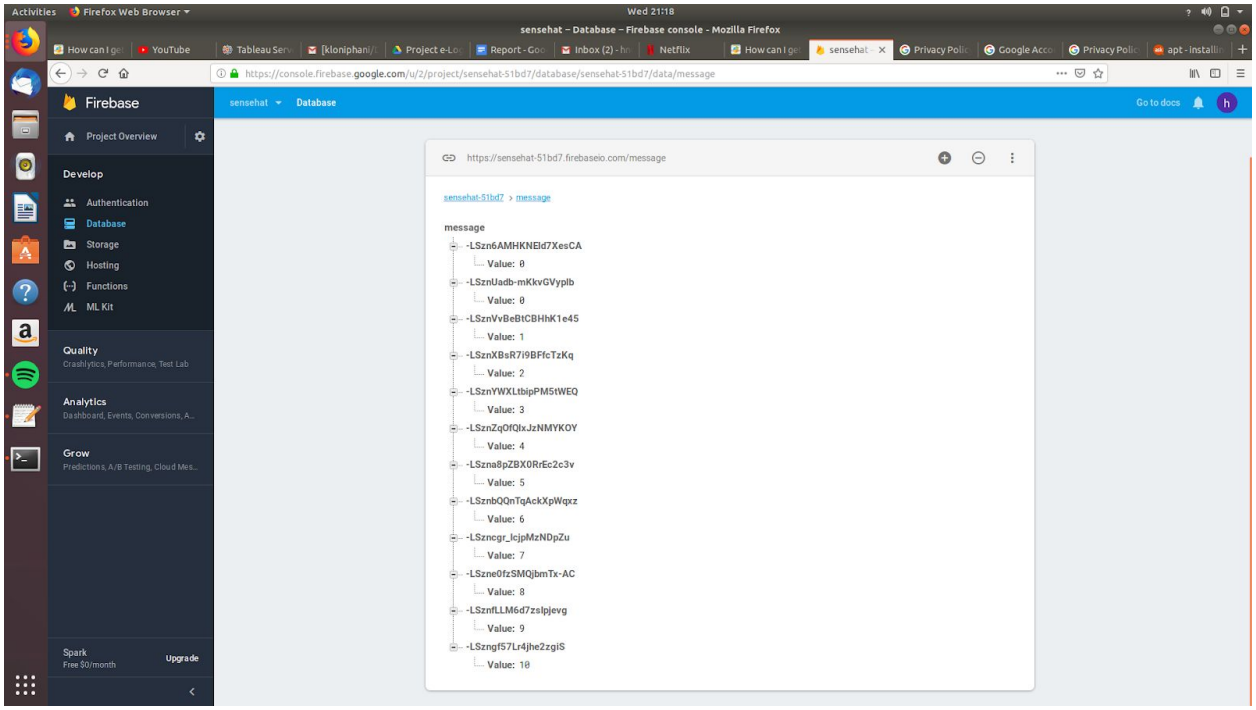
Step 8: Click on **Storage** on the left tab.



Step 9: Change the rules to what is shown in the picture. Add the “ || request.auth ==null ” on line 4 as shown in the picture.



Step 10: After linking your app with your database and pushing data to it. Click on **Databases** on the left tab to view the data.



Storing data in InfluxDB and visualizing with Grafana

Installing InfluxDB on Raspberry PI

- Adding repositories

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -source /etc/os-release
test $VERSION_ID = "7" && echo "deb https://repos.influxdata.com/debian wheezy stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
test $VERSION_ID = "8" && echo "deb https://repos.influxdata.com/debian jessie stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
test $VERSION_ID = "8" && echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

- Install InfluxDB from the added repository

```
sudo apt-get update && sudo apt-get install influxdb
```

- Start the InfluxDB service

```
sudo service influxdb start
```

- Test the installation by clicking the link below.

```
localhost:8086
```

If the installation was successful, the page will display “**404 page not found**”.

- Try to configure InfluxDB if it doesn’t work. Uncomment all the lines in the **[Admin]** section

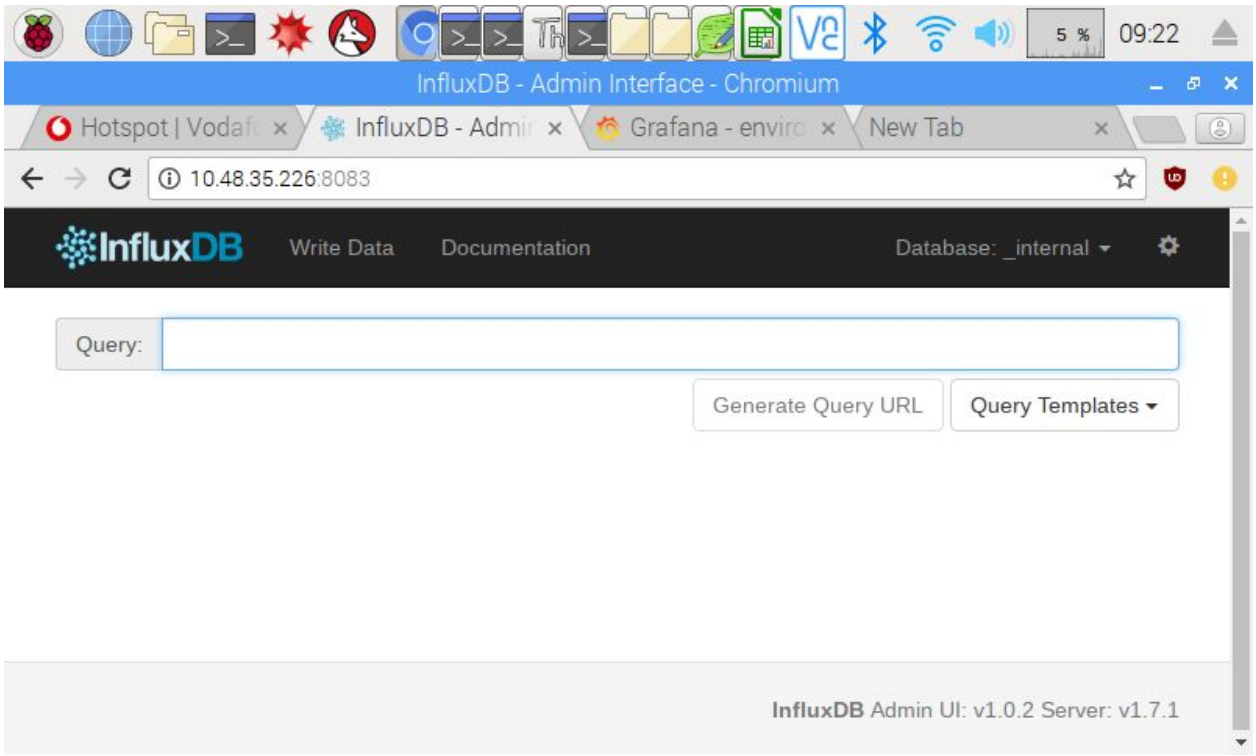
by removing all the “#” .

```
sudo nano /etc/influxdb/influxdb.conf
```

- Test the Web UI by clicking on the link below.

```
localhost:8083
```

The page below is supposed to show :



Installing Grafana on a Raspberry Pi

- Install some dependency packages

```
sudo apt-get install apt-transport-https curl
```

- Add the key

```
curl https://bintray.com/user/downloadSubjectPublicKey?username=bintray | sudo apt-key add -
```

- Add Grafana to the update list

```
echo "deb https://dl.bintray.com/fg2it/deb stretch main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

- Install Grafana

```
sudo apt-get install grafana
```

- Start the Grafana server service

```
sudo systemctl start grafana-server
```

- Set Grafana to run on startup

```
sudo systemctl enable grafana-server.service
```

- Click the link below to login to Grafana

```
localhost:3000
```

Creating a database in InfluxDB and visualizing the data using Grafana

- To create a database in InfluxDB type the following on the terminal. This will open the influxDB shell which accepts database queries (Influx Query Language).

```
$ influx -precision rfc3339

Connected to http://localhost:8086 version 1.2.x
InfluxDB shell 1.2.x
>
```

- Type the following command to create a database named **sense_Hat.**

```
> CREATE DATABASE senseHatDB
>
```

- To check if the database was created successfully type the following query.

```
> SHOW DATABASES

name: databases
---------------
name
_internal
senseHatDB
```

- Install the InfluxDB Python module.

```
sudo pip install influxdb
```

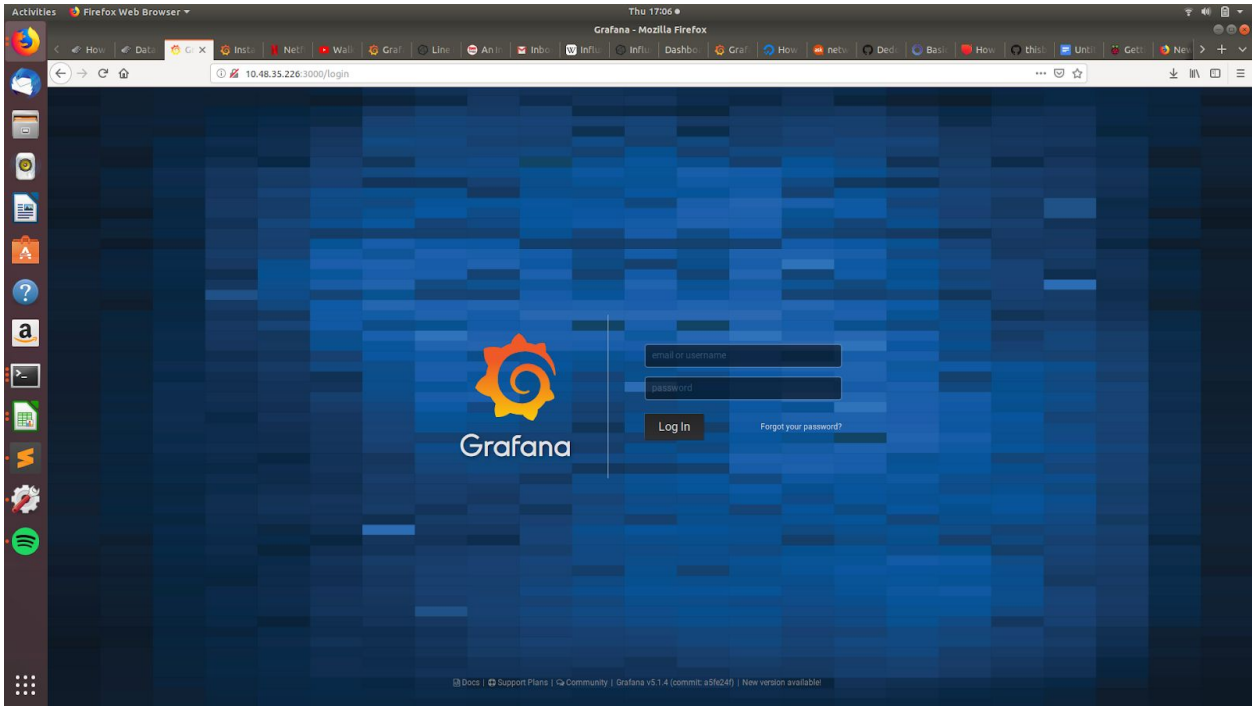
- Run the python script which reads in sensor readings and pushes them to the database created by typing the following on the Raspberry Pi terminal.

```
python sense_influxDB.py -db=dedomena2 -sn=test1
```

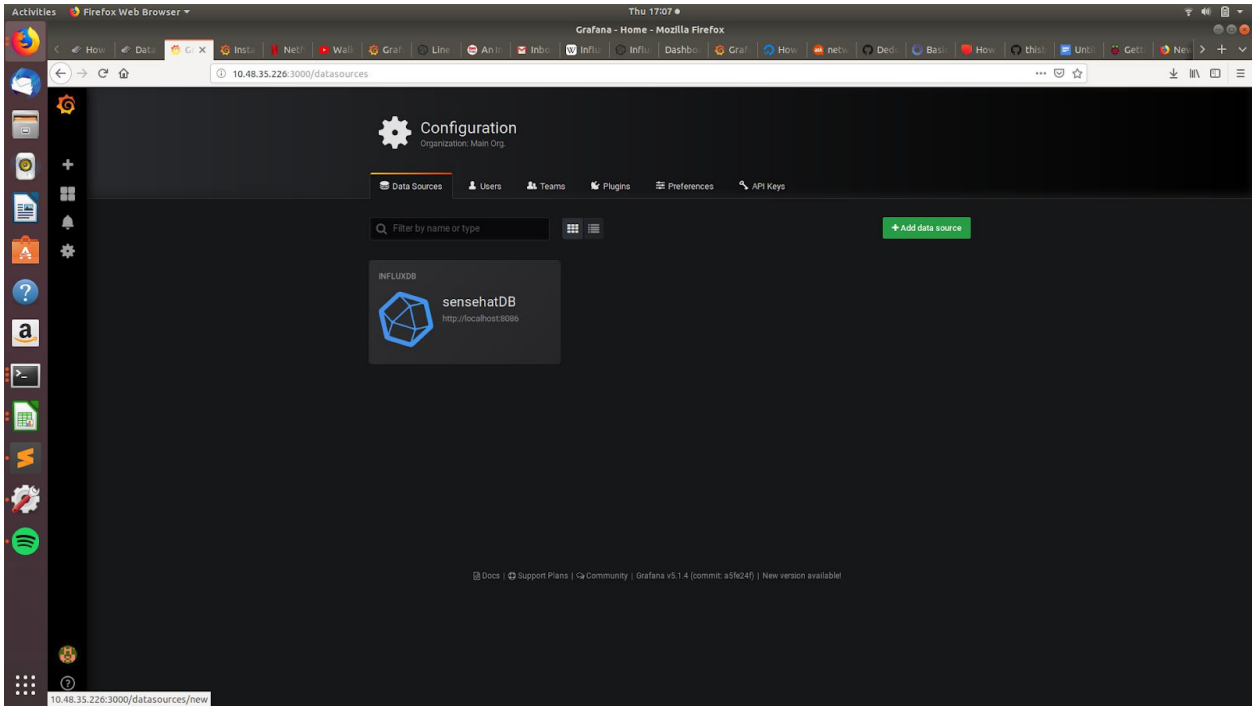
The python script is on GitHub in the Dedomena/Sensor folder, link :
https://github.com/kloniphani/Dedomena/blob/master/Sensor/sense_influxDB.py

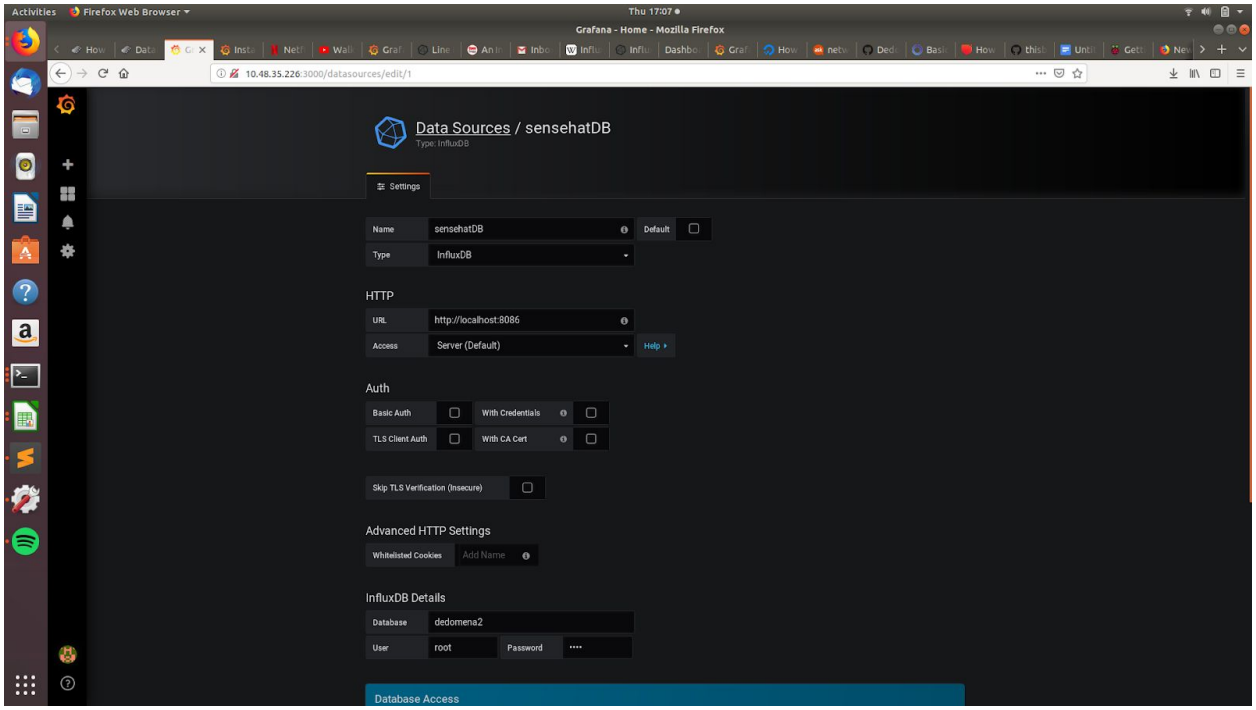
- Login to Grafana at localhost:3000

(username= “admin”, password=”admin”)



- Navigate to Datasource->Add New and fill the form as below

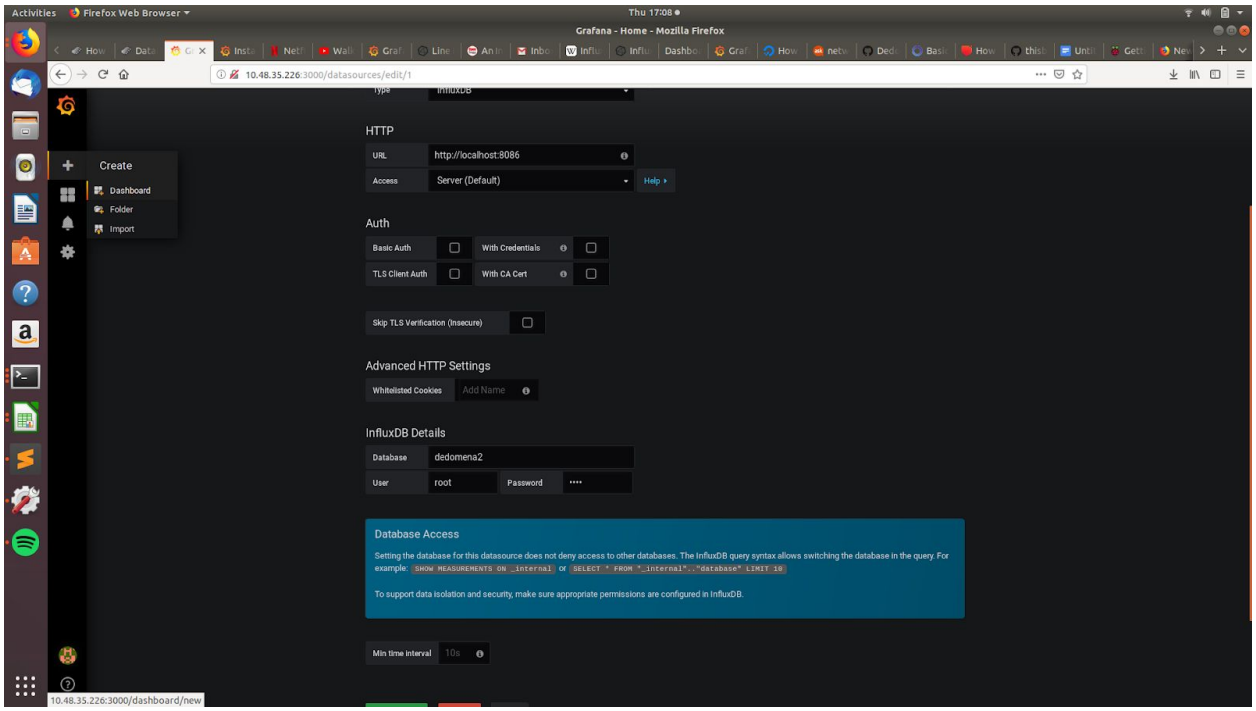


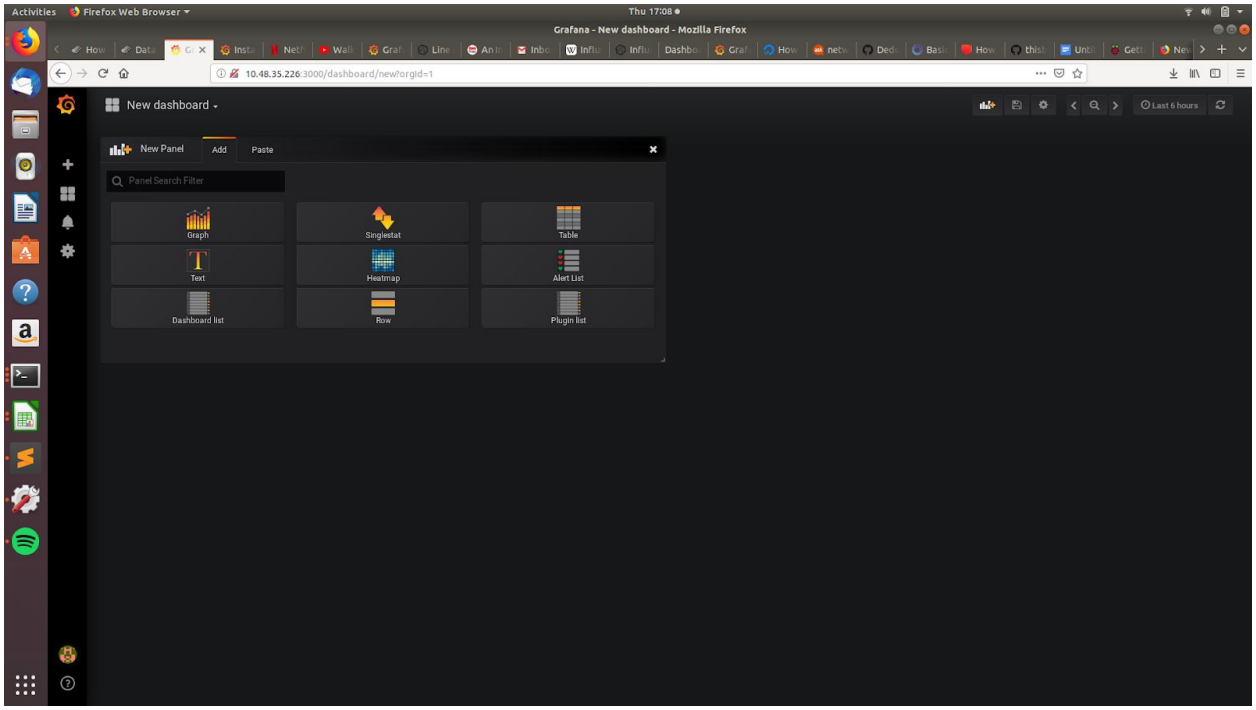


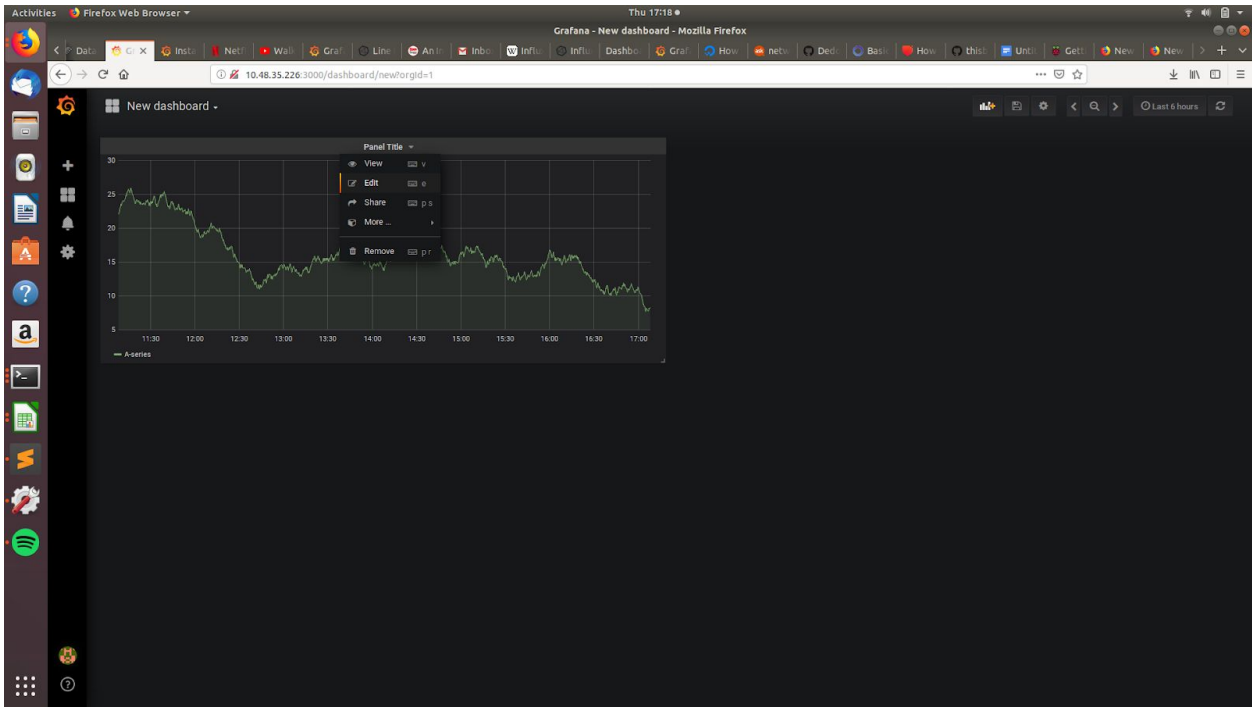
The influxDB details of the database we created are :

User: root
Password: root

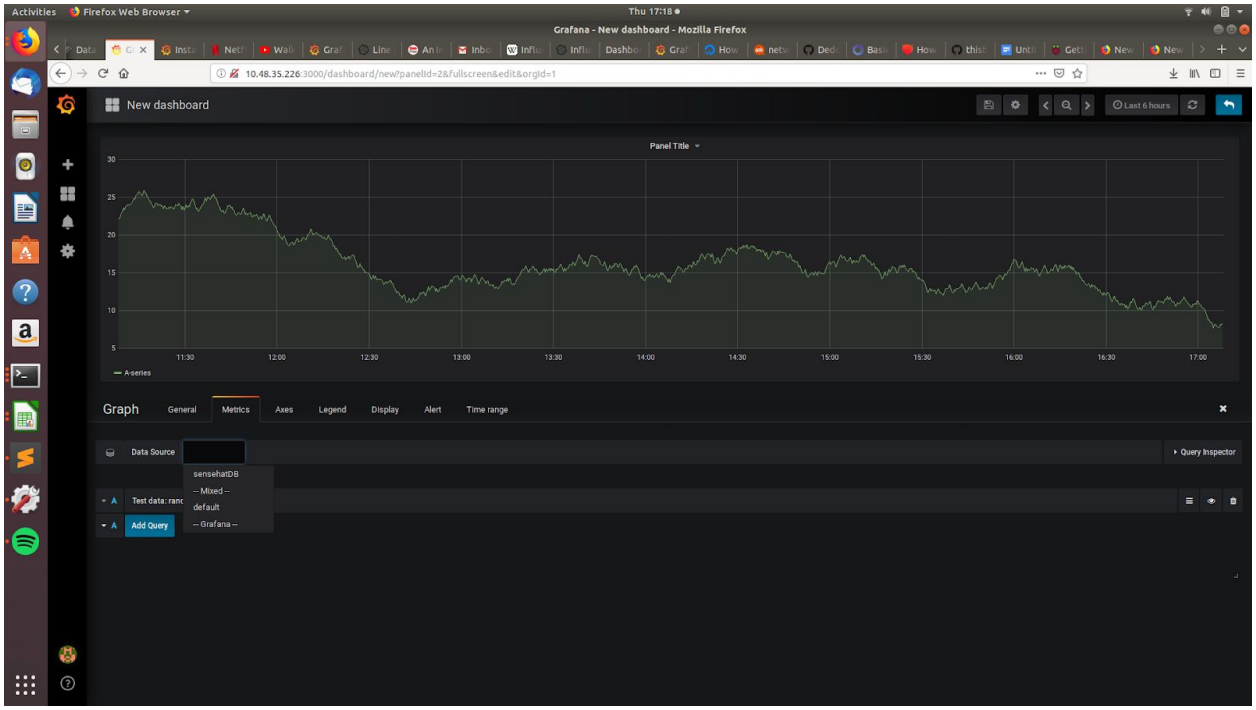
- Create a new dashboard and add graphs to it as shown below



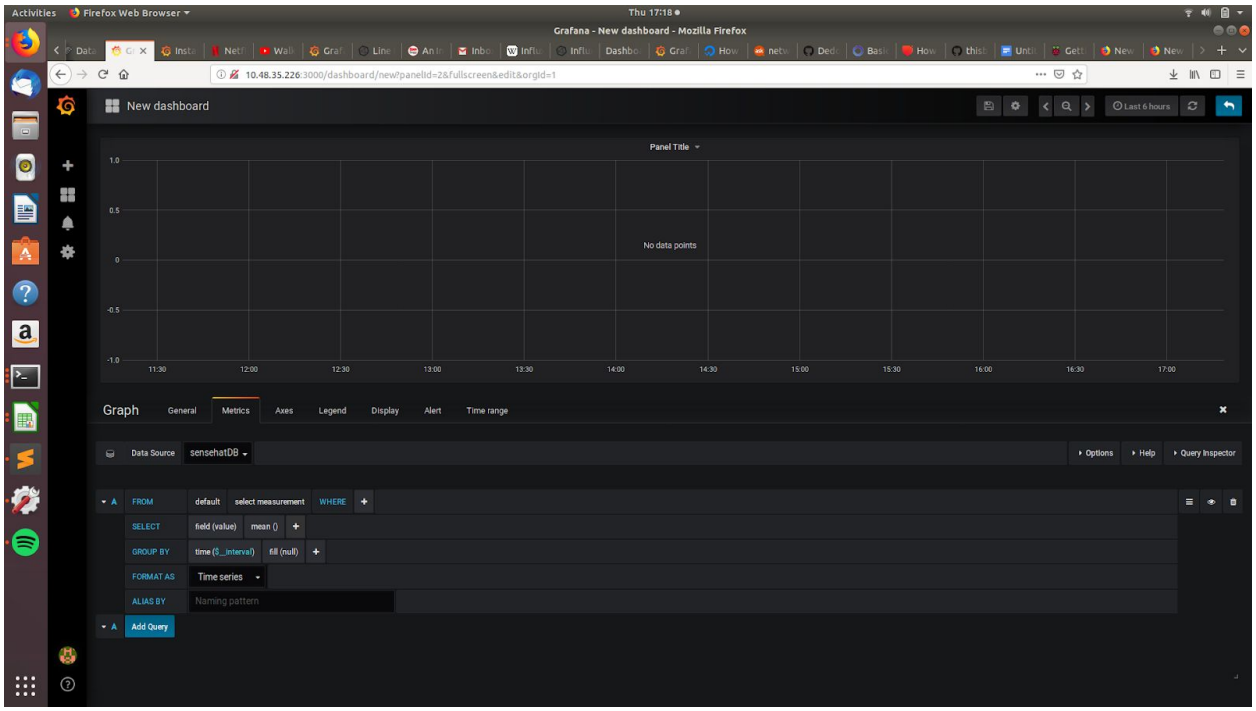




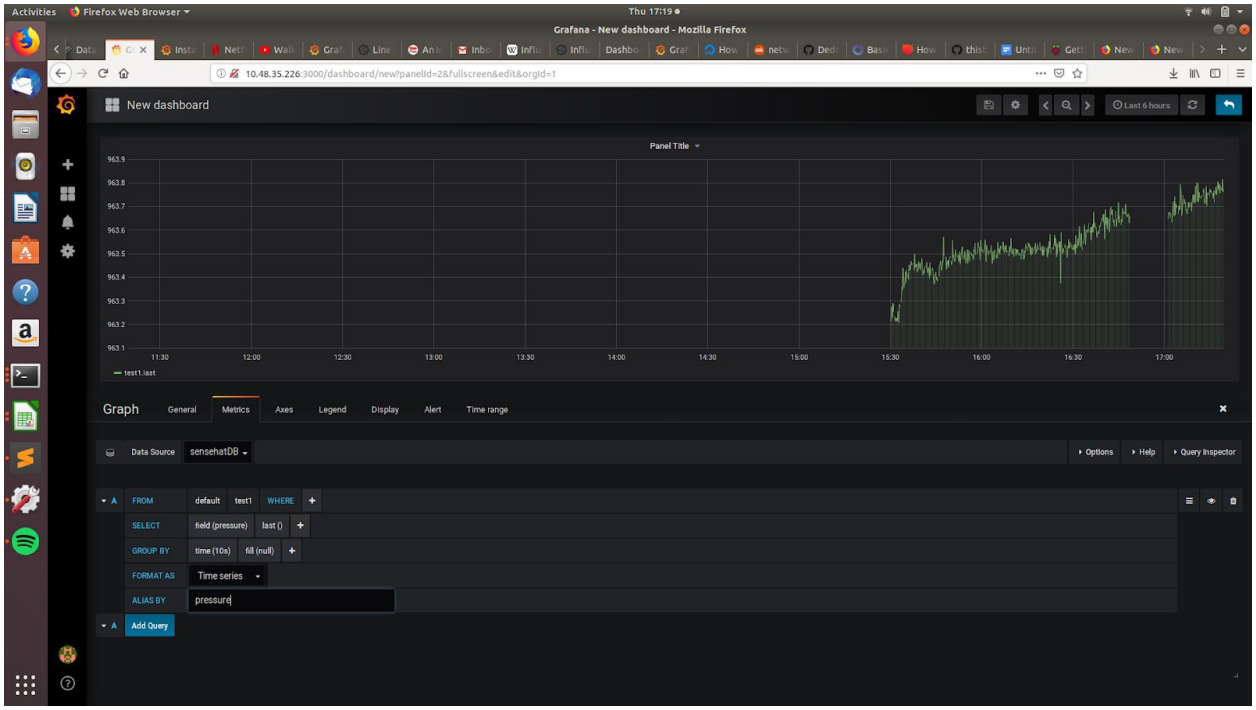
Click on the **edit** option



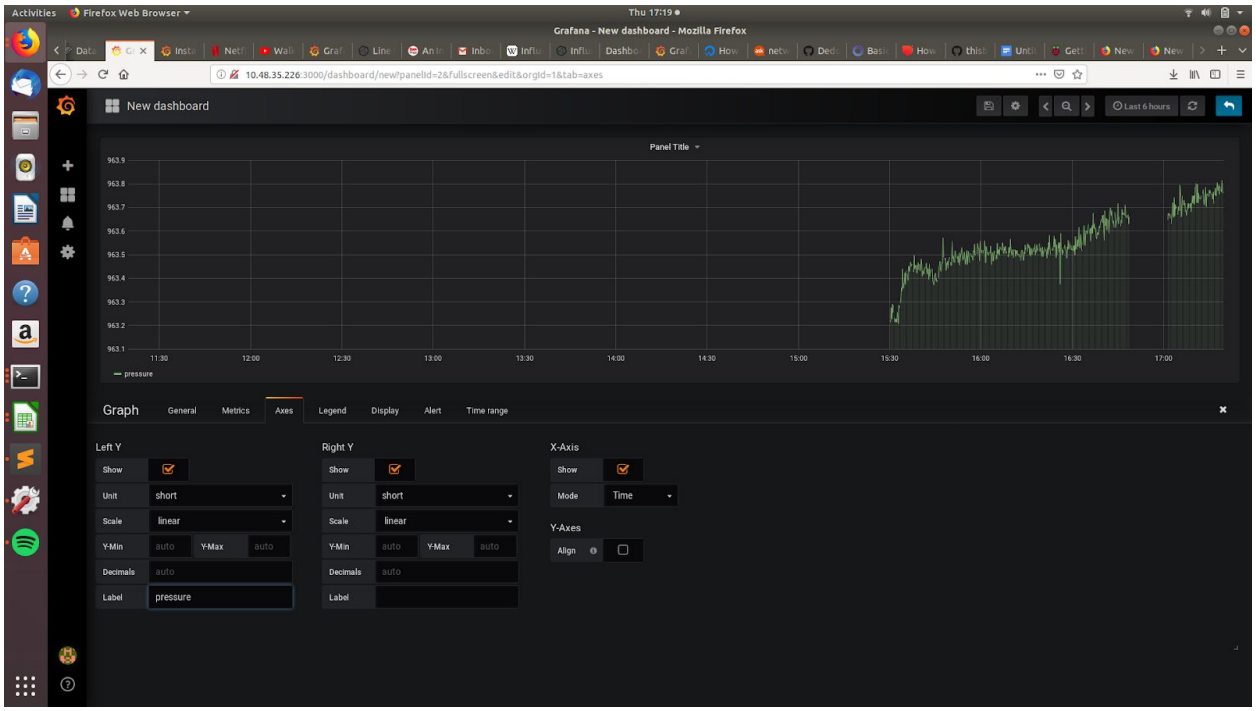
Select the data source as **senseHatDB**

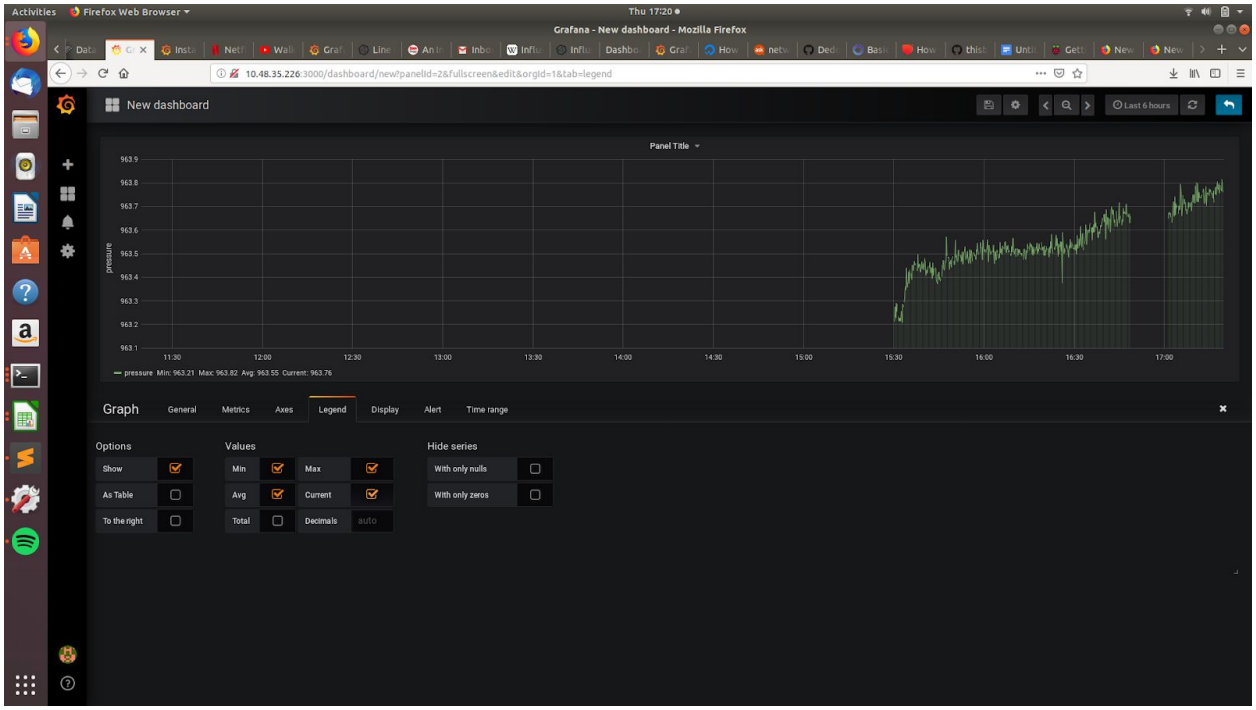


Create a query which will show data on the graph as shown below

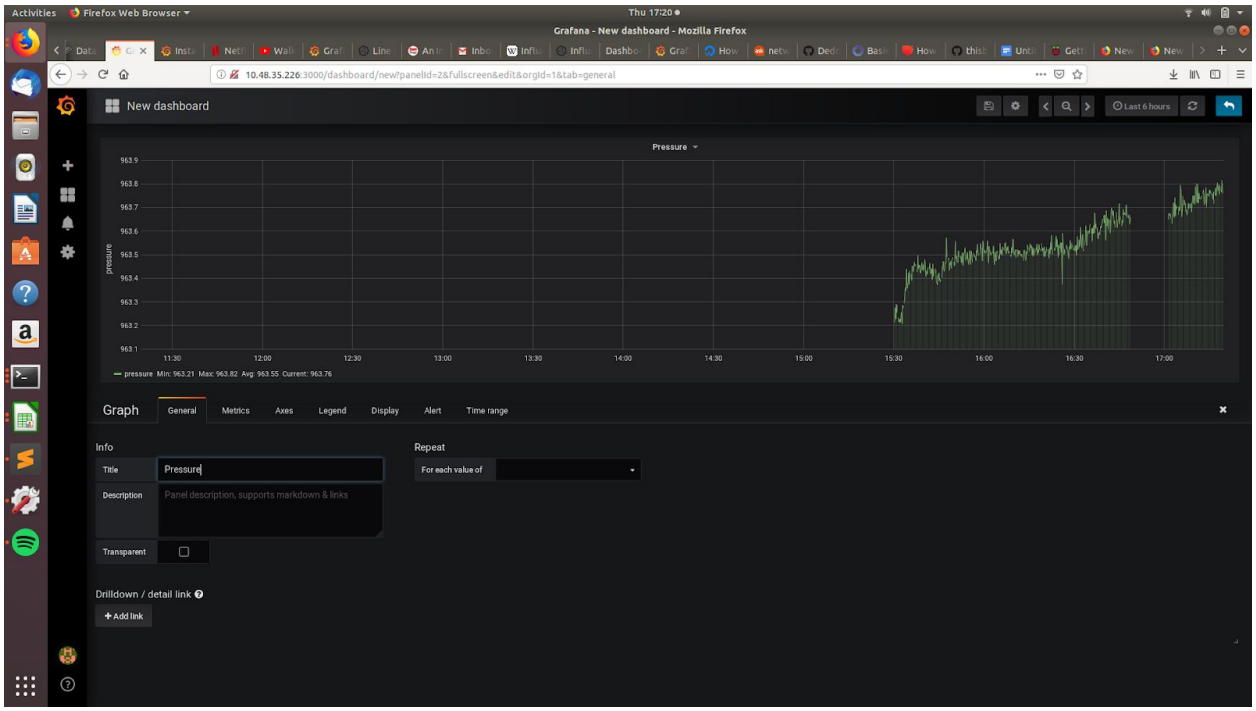


You can now visualize the data on the graph, you can now label the axes and add legends as shown below

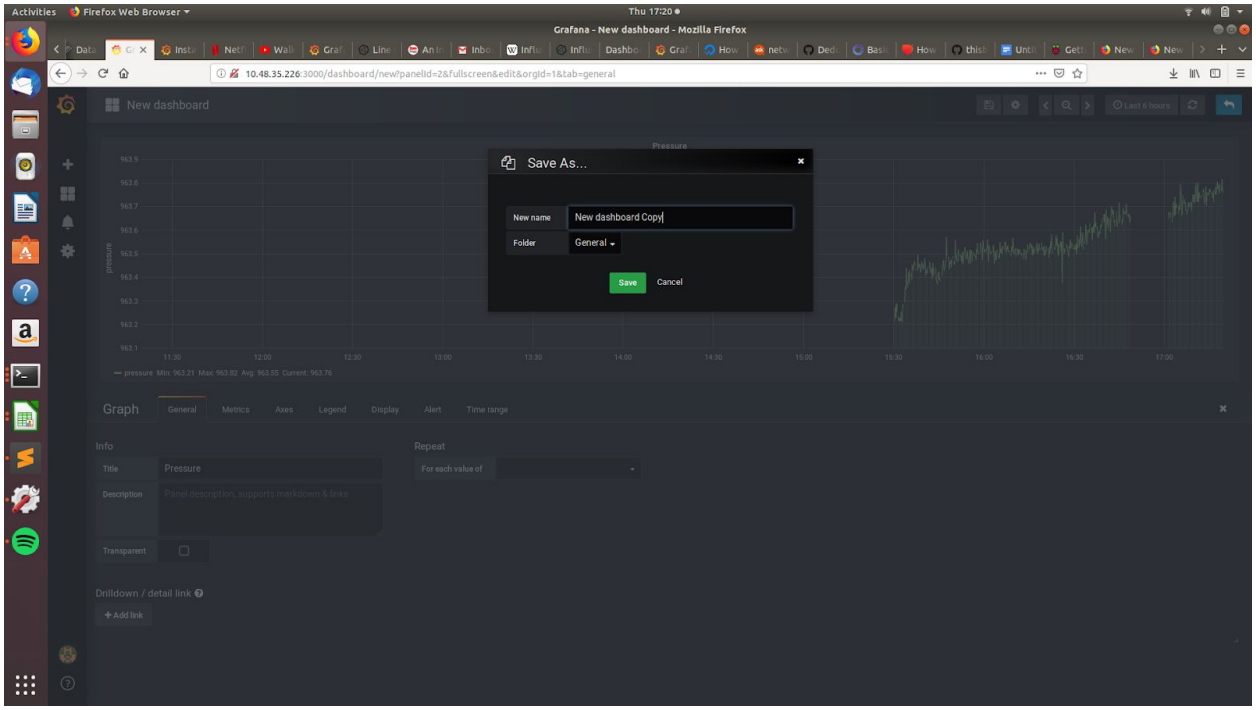




Add the max, min, avg and cur to your legends as shown above.

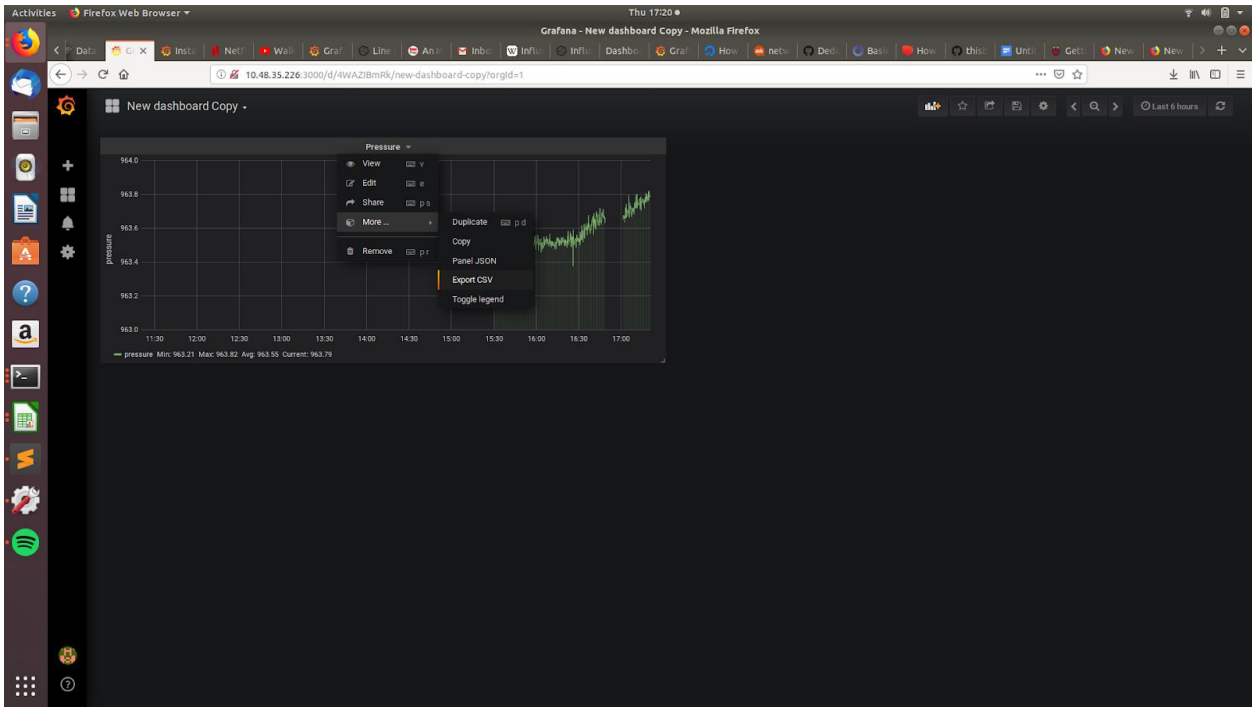


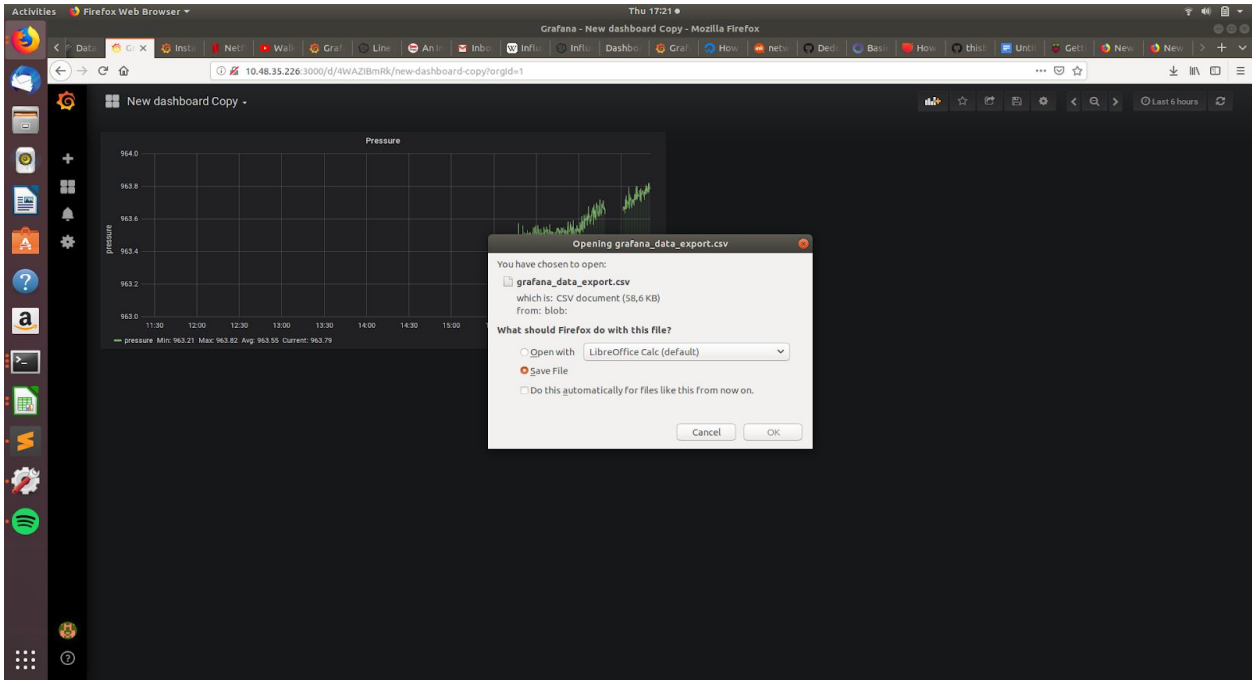
Add a **Title** to the graph as shown above.



Save the **Dashboard** when you're done.

Exporting the data as a CSV file





Save the CSV file.

Enabling Email Alerts and Setting up an email server on Grafana

- Navigate on the terminal to /etc/grafana
- Type “sudo nano grafana.ini” to edit the Grafana config file
- Edit the following sections as shown in the pictures below:

```
#################################### Alerting ############################
[alerting]
# Disable alerting engine & UI features
enabled = true
# Makes it possible to turn off alert rule execution but alerting UI is visible
execute_alerts = true
```

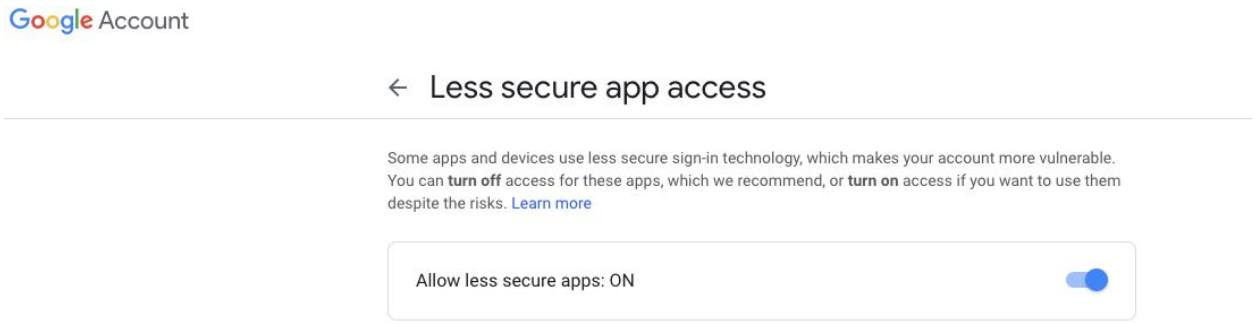
```
#################################### SMTP / Emailing ##########################
[smtp]
enabled = true
host =smtp.gmail.com:465
user =accnt4research@gmail.com
# If the password contains # or ; you have to wrap it with trippel quotes. Ex """#password;"""
password ="[redacted]"
cert_file =
key_file =
skip_verify = false
from_address = accnt4research@gmail.com
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
ehlo_identity = dashboard.example.com

[emails]
welcome_email_on_sign_up = false
templates_pattern = emails/*.html

#################################### Logging ##########################
```

Replace the user email address, from address and password with a valid email address. Change the email address settings to be less secure (it's a good idea to create a new email address for this task)

Setting up a less secure Gmail Account

- You might want to create a "playground" email account for this.
- Log in to the email account you just created.
- Go to this link: https://myaccount.google.com/security
- Scroll down to **Less Secure Apps**

- Click on **Turn on access**



- Turn on the toggle button