# Installing InfluxDB and Grafana

## Instructions

We first will need to install the latest Debian packages of InfluxDB.

Note: this is very important. If you skip this step you will install an older version of InfluxDB on Ubuntu which is out of data and causes a lot of problems. In that case you will need to fully remove InfluxDB and reinstall it.

First go too.

<https://portal.influxdata.com/downloads/>

We will need to install the InfluxDB Time-Series Data Storage files. In my case this was the v1.8.0.

**Note:** You can install the beta version if it is available. But I recommend installing v1.8.0 or the latest completed in your case.

By clicking on the link, you will get a lot of ways to install the packages. Because we are using Ubuntu 16.04 Xenial Xlde on a Raspberry Pi 4 we will not use the “Ubuntu & Debian” as this uses amd64.deb files which cannot be installed on the Raspberry Pi.

### Installing InfluxDB v1.8.0

We will therefore use the “Linux Binaries (ARM)” install [1]

**$ wget https://dl.influxdata.com/influxdb/releases/influxdb-1.8.0\_linux\_armhf.tar.gz**

**$ tar xvfz influxdb-1.8.0\_linux\_armhf.tar.gz**

After that is done, we can begin installing InfluxDB OSS using the following link.

<https://docs.influxdata.com/influxdb/v1.8/introduction/install/>

You can follow the tutorial in this link if you want to do it yourself. But I will place the commands I used below with the information needed.

**Note:** These commands are for the version 1.8 and there may be a newer version out. In that case I recommend you follow those instructions.

For Ubuntu users, add the InfluxData repository with the following commands: [2]

**$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -**

**$ source /etc/lsb-release**

**$ echo "deb https://repos.influxdata.com/${DISTRIB\_ID,,} ${DISTRIB\_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.list**

After these commands are finished, we can begin install and start the InfluxDB service. [2]

**$ sudo apt-get update && sudo apt-get install influxdb**

**$ sudo systemctl unmask influxdb.service**

**$ sudo systemctl start influxdb**

We can check the status of InfluxDB with the following commands.

**$ systemctl status influx**

**$ journalctl -u influxdb**

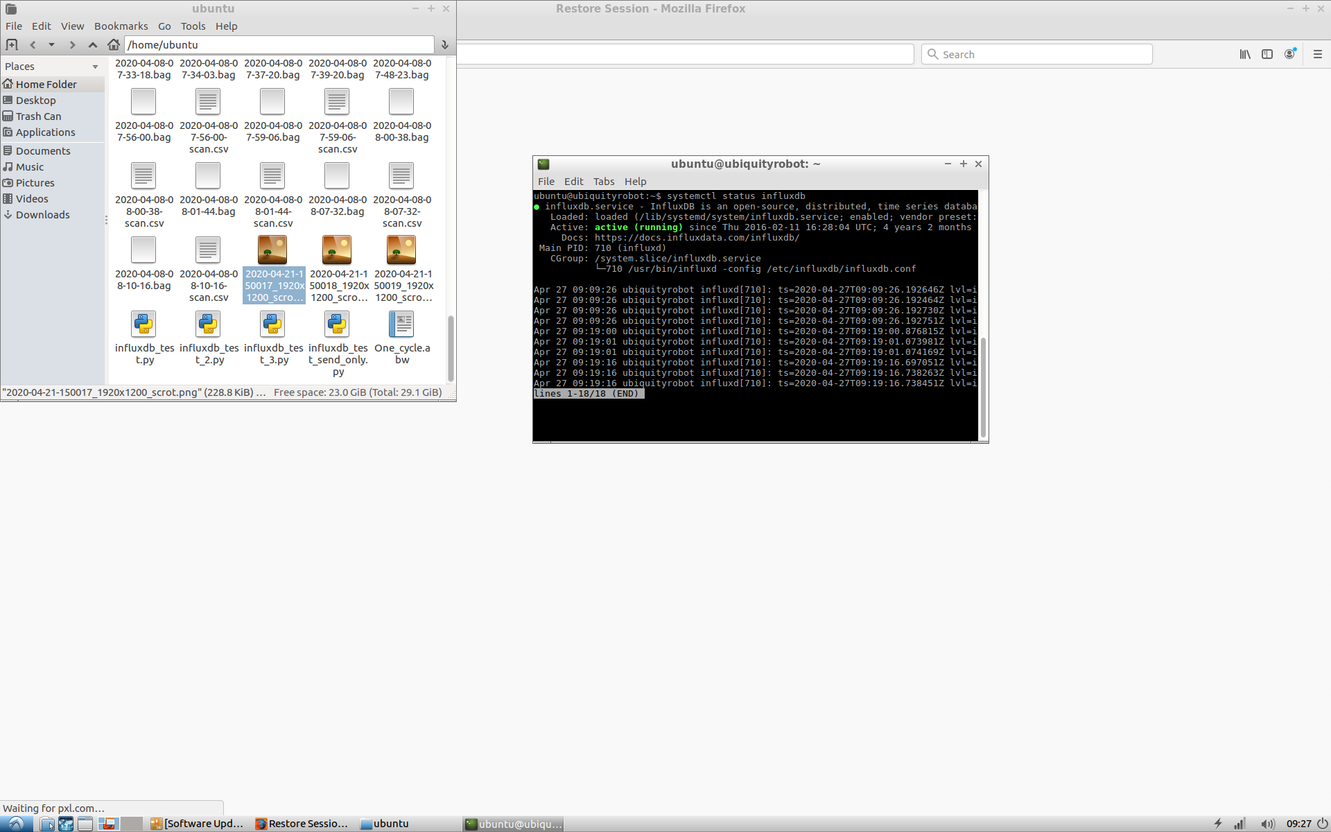


Figure 1: status InfluxDB

### Getting started with InfluxDB

Let us now do some small tests with InfluxDB to get acquainted.

First let us connect to the InfluxDB shell [3]

**$ influx**

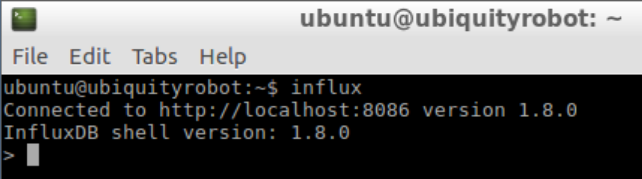


Figure 2: connecting InfluxDB shell

Let us create a database named “statsdemo”. [3]

**> CREATE DATABASE statsdemo**

The following command shows you all the database that available on your InfluxDB server. [3]

**> SHOW DATABASES**

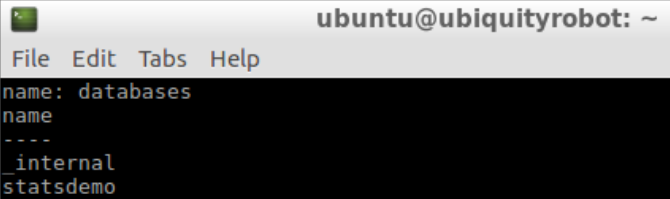


Figure 3: showing databases

Select the newly created database: [3]

**> USE statsdemo**

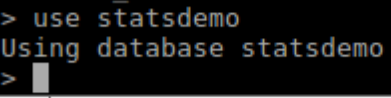


Figure 4: using database

We can insert some test data using the following command. [3]

**> INSERT cpu,host=serverA value=0.64**

The following link gives more information on inserting date into a database: https://docs.influxdata.com/influxdb/v0.12/guides/writing\_data/ [3]

Let us do a query of the data base to see what data it contains. [3]

**> SELECT \* from cpu**

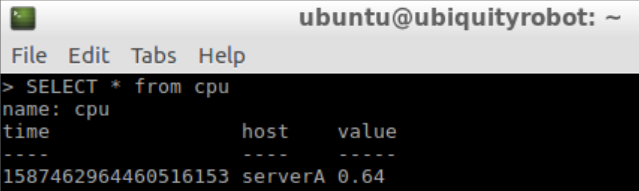


Figure 5: showing data in database

Use “exit” to leave the InfluxDB shell. [3]

**> exit**

### Writing test data

Another way to send data to the “statsdemo” database is with the following command. [3]

**$ curl -i -XPOST 'http://localhost:8086/write?db=statsdemo' --data-binary 'cpu,host=serverA value=0.90'**

You can verify if data was sent to the InfluxDB database by using the influx shell and running a query. [3]

**$ influx**

**> USE statsdemo**

**> SELECT \* FROM cpu**

### Install latest Grafana OSS release:

Now we will install Grafana OSS. Grafana is an application that will help us better display the database data in Influx. [4]

**$ sudo apt-get install -y apt-transport-https**

**$ sudo apt-get install -y software-properties-common wget**

**$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –**

**$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list**

**sudo apt-get update**

**sudo apt-get install grafana**

#### Start the server with systemctl

To start the service and verify that the service has started: [4]

**$ sudo systemctl daemon-reload**

**$ sudo systemctl start grafana-server**

**$ sudo systemctl status grafana-server**

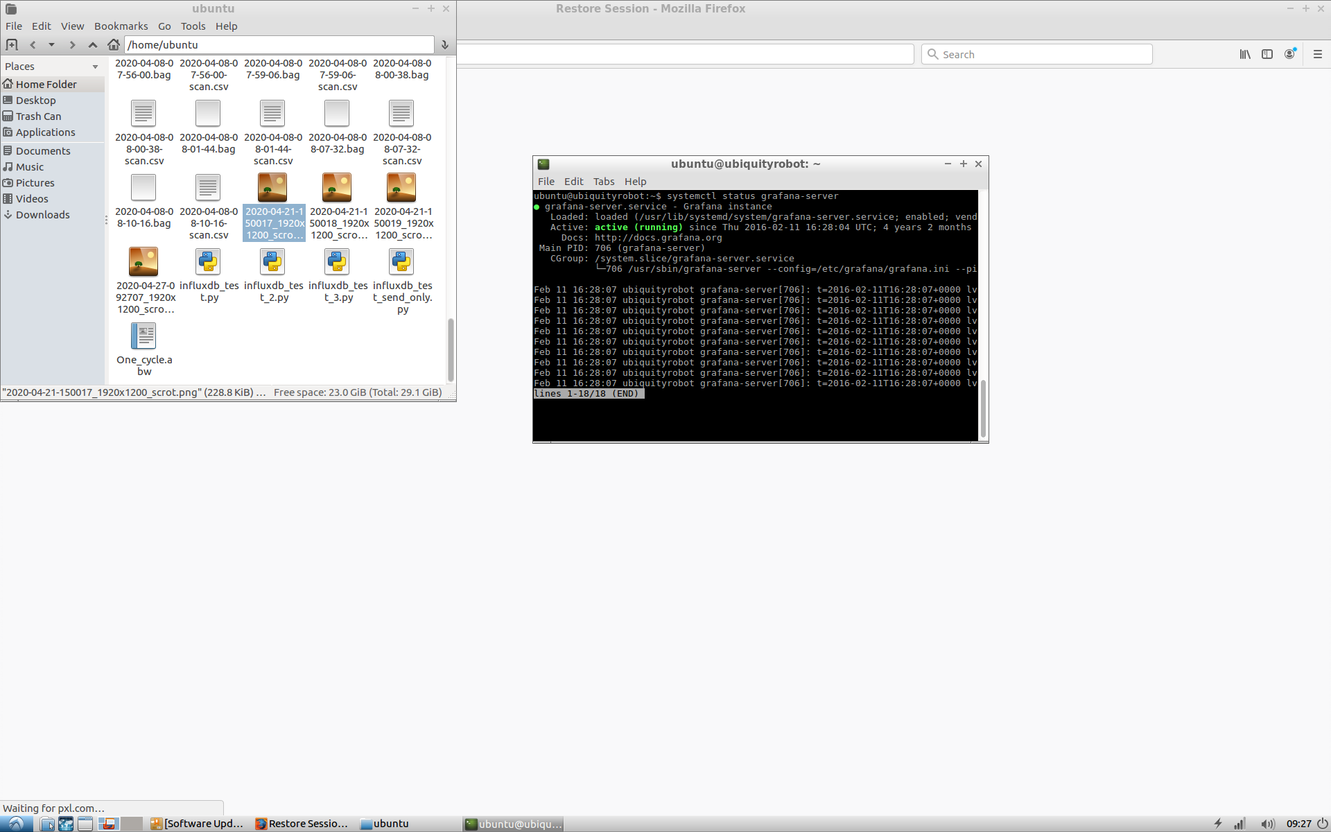


Figure 6: status Grafana-server

Now let us configure the Grafana server to start at boot: [4]

**$ sudo systemctl enable grafana-server.service**

#### Updating Firefox on Ubuntu

The following step is to update our web browser which in my case is Firefox. It was found/determined that the web browser of the Ubuntu system that we are using is not up to date. Which will cause problems when trying to login to Grafana. [5]

**$ sudo apt-det update && sudo apt-get install firefox**

### Logging into Grafana

Let us now log into Grafana.

Open a web browser and go to.

<http://localhost:3000>

And if you want to login form another device on the same network use.

http://\*\*IP-address of the Ubuntu device\*\*:3000

**Note:** replace the part between asterixis with the IP-address of the Ubuntu device. You also need to be on the same network as the Ubuntu device in order for this the work.

You will be prompted with a login page.

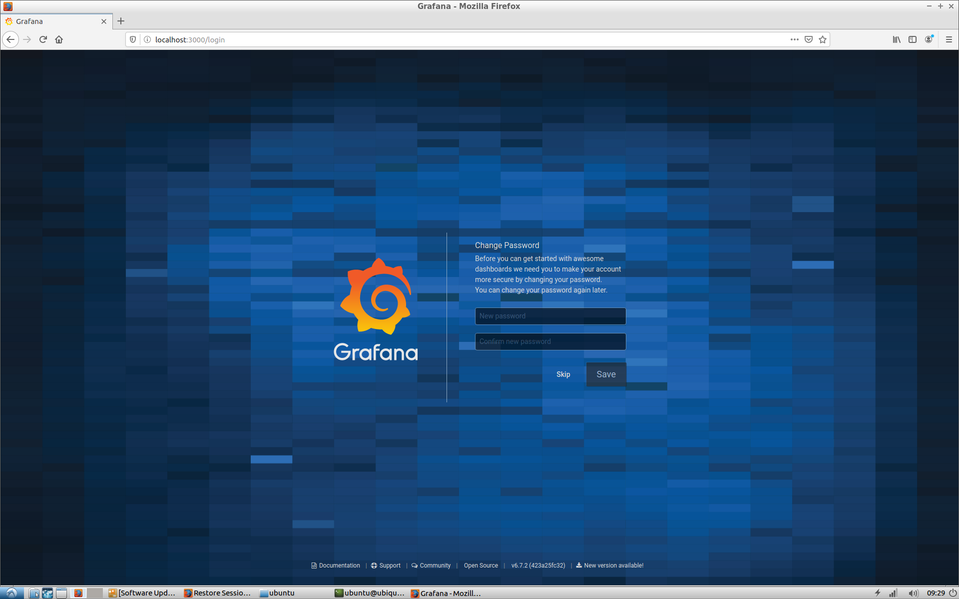


Figure 7: login Grafana

The password and login are bout “admin”.

After pressing enter you will be prompted to change the password. We will skip this for now by pressing the “Skip” button.

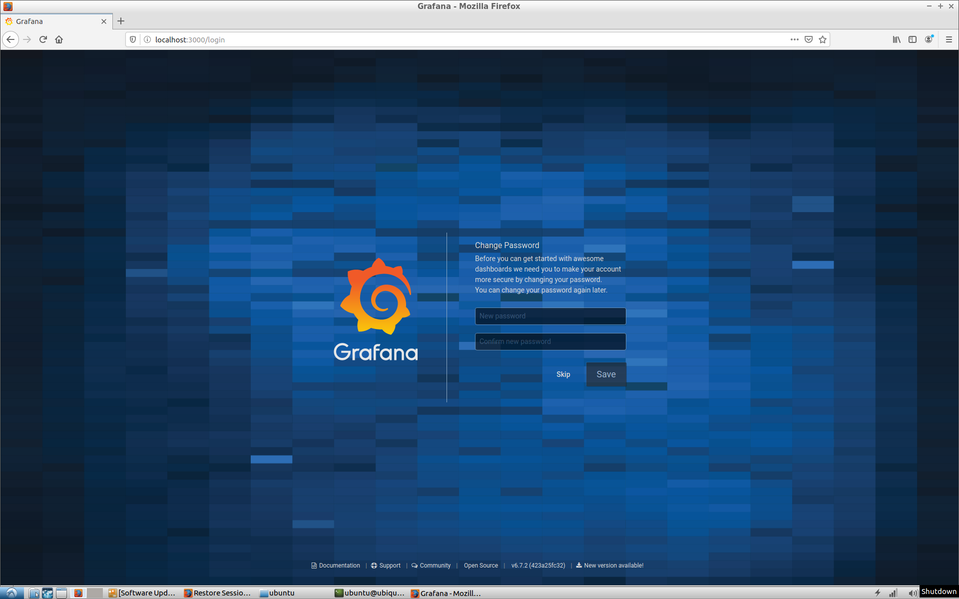


Figure 8: changing password Grafana

Now we can begin working in Grafana.

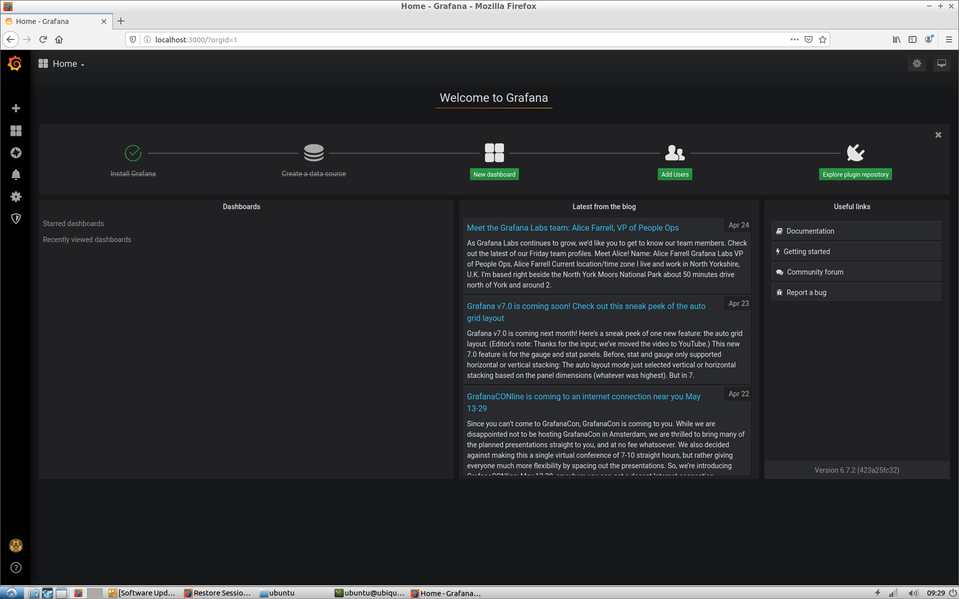


Figure 9: Grafana home screen

### How to use Grafana with InfluxDB

The following video helps explain how to like your InfluxDB database to Grafana and display some data.

<https://www.youtube.com/watch?v=BBcj-ZoufMw>

### Installing Python packages

First let us check the different versions of Python we have.

**$ python –V**

**$ python2 –V**

**$ python3 –V**

There are 2 packages that we need to install which are argparse and influxdb.

The following command are used to install argparse and influxdb for Python 2 and Python 3. [6] [7]

**$ pip3 install argparse**

**$ pip install argparse**

**$ python3 –m pip install influxdb**

**$ python2 –m pip install influxdb**

**$ python –m pip install influxdb**

The last command is just to make sure the InfluxDB Python packages are installed for Python 2.7.

**Note:** the following steps are only if you had any problems installing the previous 2 packages.

|  |
| --- |
| Let us update pip  **$ pip install --upgrade pip**  It can be that there may be a problem with the upgrade because it still uses Python 2.7.  You can use the following commands to solve the problem.  **$ sudo apt-get install python3**  **$ sudo apt-get install python3-pip**  **$ pip3 install --upgrade pip**  Now let us install argparse.  **$ pip3 install argparse**  **$ pip install argparse**  Now let us also install the InfluxDB python packages.  **$ python3 –m pip install influxdb**  **$ python2 –m pip install influxdb** |

### Trying Python code

The next thing we will do is test a Python code that connects to InfluxDB, creates a database and sends some data to the newly created database. [8]

|  |
| --- |
| *# -\*- coding: utf-8 -\*-*  *"""Tutorial on using the InfluxDB client."""*  **import** **argparse**  **from** **influxdb** **import** InfluxDBClient  **def** main(host='localhost', port=8086):  *"""Instantiate a connection to the InfluxDB."""*  user = 'root'  password = 'root'  dbname = 'example'  dbuser = 'smly'  dbuser\_password = 'my\_secret\_password'  query = 'select Float\_value from cpu\_load\_short;'  query\_where = 'select Int\_value from cpu\_load\_short where host=$host;'  bind\_params = {'host': 'server01'}  json\_body = [  {  "measurement": "cpu\_load\_short",  "tags": {  "host": "server01",  "region": "us-west"  },  "time": "2009-11-10T23:00:00Z",  "fields": {  "Float\_value": 0.64,  "Int\_value": 3,  "String\_value": "Text",  "Bool\_value": True  }  }  ]  client = InfluxDBClient(host, port, user, password, dbname)  **print**("Create database: " + dbname)  client.create\_database(dbname)  **print**("Create a retention policy")  client.create\_retention\_policy('awesome\_policy', '3d', 3, default=True)  **print**("Switch user: " + dbuser)  client.switch\_user(dbuser, dbuser\_password)  **print**("Write points: {0}".format(json\_body))  client.write\_points(json\_body)  **print**("Querying data: " + query)  result = client.query(query)  **print**("Result: {0}".format(result))  **print**("Querying data: " + query\_where)  result = client.query(query\_where, bind\_params=bind\_params)  **print**("Result: {0}".format(result))  **print**("Switch user: " + user)  client.switch\_user(user, password)  **print**("Drop database: " + dbname)  client.drop\_database(dbname)  **def** parse\_args():  *"""Parse the args."""*  parser = argparse.ArgumentParser(  description='example code to play with InfluxDB')  parser.add\_argument('--host', type=str, required=False,  default='localhost',  help='hostname of InfluxDB http API')  parser.add\_argument('--port', type=int, required=False, default=8086,  help='port of InfluxDB http API')  **return** parser.parse\_args()  **if** \_\_name\_\_ == '\_\_main\_\_':  args = parse\_args()  main(host=args.host, port=args.port) |

The code above can be found with the following link.

<https://influxdb-python.readthedocs.io/en/latest/examples.html>

The code above is a tutorial code to get acquainted with some of the commands and get started.

You can execute the code either with Python 2 or 3, but we are mostly interested in Python 2 as ROS uses Python 2.7.

**Python 2.7**

**$ python \*\*name\_file\*\*.py**

**Python 3**

**$ python3 \*\*name\_file\*\*.py**

**Note:** replace the \*\*name\_file\*\* with the name of the python file you want to run. In this case the file with the InfluxDB code.

### Creating ROS package

Now let us create a new ROS package.

First go to the src map in the ROS workspace.

**$ ~/catkin\_ws/src**

Now let us create a new ROS package. We also need some extra maps and build the package afterwards. [9]

**Note:** the rospy at the end is an extra package that we are including. If you do not place the package at the and you have to add it yourself later on.

**$ catkin\_create\_pkg laser\_values rospy**

**$ cd laser\_vales**

**$ mkdir launch**

**$ mkdir src**

**$ cd ../..**

**$ catkin\_make**

When the command is finished, we can add the Python program and launch file.

The Python code has to be placed in the src map of the laser\_values map and the launch file in the launch map.

Let us take a look at the Python-ROS program and the ROS launch file of that program that we will be using.

**Python-ROS program (Influx.py)** [8] [9]

|  |
| --- |
| #! /usr/bin/env python  import rospy  from sensor\_msgs.msg import LaserScan  import argparse  from influxdb import InfluxDBClient  import time  # Name of the Database  # Username and password  user = 'root'  password = 'root'  dbname = 'RPLiDAR\_DB'  def callback(msg):  # print len(msg.ranges)  # print msg.time\_increment  getal = list(msg.ranges)  x = 0  minimum = 0.0  maximum = 0.0  # Gets the minimum detected range  for x in range(0, 1440):  if getal[x] < minimum:  minimum = getal[x]  # Converts all 'inf' values of ranges to 0.0  for x in range(0, 1440):  if str(getal[x]) == "inf":  #print "getal"  getal[x] = 0.0  # Gets the maximum detected range  for x in range(0, 1440):  if getal[x] > maximum:  maximum = getal[x]  # Time the sensor data was collected  date\_time = time.strftime("%d/%m/%y") + "T" +time.strftime("%I:%M:%S") + "Z" + str(msg.header.stamp.nsecs) + "N"  # Data sent to the Influx database  json\_body = [  {  "measurement": "RPLiDAR\_sensor\_values",  "tags": {  "Sensor": "RPLiDAR A3M1"  },  "seq": msg.header.seq,  "frame\_id": msg.header.frame\_id,  "time\_frame": date\_time,  "fields": {  "angle\_min": msg.angle\_min,  "angle\_max": msg.angle\_max,  "angle\_increment": msg.angle\_increment,  "time\_increment": msg.time\_increment,  "scan\_time": msg.scan\_time,  "datasheet\_min": msg.range\_min,  "datasheet\_max": msg.range\_max,  "range\_min": minimum,  "range\_max": maximum,  "range\_0": getal[0],  "range\_89": getal[89],  "range\_179": getal[179],  "range\_269": getal[269],  "range\_359": getal[359],  "range\_539": getal[539],  "range\_719": getal[719],  "range\_899": getal[899],  "range\_1079": getal[1079],  "range\_1259": getal[1259],  "range\_1439": getal[1439]  }  }  ]  # print("Write points: {0}".format(json\_body))  # Writes data to the database  client.write\_points(json\_body)  print("Data send")  def main():  print("Create database: " + dbname)  # Creates a new database to save data in.  client.create\_database(dbname)  # In ROS, nodes are uniquely named. If two nodes with the same  # name is launched, the previous one is kicked off  # The anonymous=True flag means that rospy will choose a unique  # name for our 'listener' node so that multiple listeners can  # run simultaneously  rospy.init\_node('Influx\_sensor\_values', anonymous=True)  # The function subscribes to the '/scan' topic.  # When a new message is sent over this topic the function callback will be called.  # 'LaserScan' is the variable type of the message that will be received.  sub = rospy.Subscriber('/scan', LaserScan, callback)  # spin() simply keeps python from exiting until this node is stopped  rospy.spin()  if \_\_name\_\_ == '\_\_main\_\_':  client = InfluxDBClient('localhost', 8086, user, password, dbname)  main() |

**ROS launch file (Influx.launch)** [9]

|  |
| --- |
| <launch>  <node pkg="laser\_values" type="Influx.py" name="Influx\_sensor\_values" output="screen">    </node>  </launch> |

It must be noted that the “name” variable in the launch file must be the same as the value in the .init\_node() command. If these variables are not the same it can result in problems, include the program not working.

When the Python code is placed in the src map we have to run the following code. [9]

**$ ~/catkin\_ws/src/laser\_values**

**$ chmod +x src/Influx.py**

If you do not run this last command for the Python file, it cannot be used by ROS.

Now we can try and run our code.

Let us first start the RPLiDAR

**$ roslaunch rplidar\_ros view\_rplidar\_a3.launch**

**OR**

**$ roslaunch rplidar\_ros rplidar\_a3.launch**

Now that the RPLiDAR is running and is sending data over the /scan topic. we can now run our code to send this data to a new InfluxDB database that we create with.

**$ roslaunch laser\_values Influx.launch**

# Reference

|  |  |
| --- | --- |
| [1] | „Download,” Influxdata, [Online]. Available: https://portal.influxdata.com/downloads/. [Geopend 21 April 2020]. |
| [2] | „Install InfluxDB OSS,” Influxdata, [Online]. Available: https://docs.influxdata.com/influxdb/v1.8/introduction/install/. [Geopend 20 April 2020]. |
| [3] | Andre, „GRAFANA AND INFLUXDB QUICKSTART ON UBUNTU,” andremiller.net, 4 May 2016. [Online]. Available: http://www.andremiller.net/content/grafana-and-influxdb-quickstart-on-ubuntu. [Geopend 21 April 2020]. |
| [4] | „Install on Debian or Ubuntu,” Grafana Labs, [Online]. Available: https://grafana.com/docs/grafana/latest/installation/debian/. [Geopend 21 April 2020]. |
| [5] | Panther, „How can I update Firefox on Ubuntu? [duplicate],” AskUbuntu, 4 October 2015. [Online]. Available: https://askubuntu.com/questions/681312/how-can-i-update-firefox-on-ubuntu. [Geopend 22 April 2020]. |
| [6] | „argparse 1.4.0,” Pypi, 12 September 2015. [Online]. Available: https://pypi.org/project/argparse/. [Geopend 21 April 2020]. |
| [7] | N. Crowley, „Getting Started with Python and InfluxDB,” Influxdata, 30 March 2018. [Online]. Available: https://www.influxdata.com/blog/getting-started-python-influxdb/. [Geopend 22 April 2020]. |
| [8] | „InfluxDB Python Examples,” InfluxDB, [Online]. Available: https://influxdb-python.readthedocs.io/en/latest/examples.html. [Geopend 22 April 2020]. |
| [9] | A. Ezquerro, „[ROS Q&A] 031 – How to read LaserScan data (ROS python),” The Construct, 26 September 2017. [Online]. Available: https://www.theconstructsim.com/read-laserscan-data/. [Geopend 22 April 2020]. |