

Use TensorFlow for predictive analysis on ARTIK 710

In this tutorial, we will use TensorFlow for predictive analysis on ARTIK710. We will collect pressure sensor data from an ARTIK055s, stream the data to ARTIK710. On ARTIK 710, we have InfluxDB and Grafana running for data storage and visualization. We will use TensorFlow and a pre-trained Keras model to predict pressure sensor data based on real-time sensor readings.

We are trying to simulate a use case in a milk processing factory, where we need to apply alternating pressure to the containers for homogenization. To simplify the use case, we are providing a pre-trained Keras model which takes input of 4 pressure sensor readings, alternating between two low pressure values (in the range of 300 – 600) and two high pressure values (in the range of 1150 – 1450).

1. Installation

1.1 Download TensorFlow wheel files

<https://github.com/lhelontra/tensorflow-on-arm/releases>

You can download TensorFlow wheel files from the link above. You will find different versions of wheel files created for different processor architectures. Since the default Python version on 710 is v2.7 and its processor adopts aarch64 architecture, we will use the tensorflow-1.11.0-cp27-none-linux_aarch64.whl file for our development.

You can also set up a virtual Python 3 environment on your ARTIK 710, and use the tensorflow-1.11.0-cp35-none-linux_aarch64.whl.

1.2 Install InfluxDB, Grafana and TensorFlow

```
#apt-get update
```

```
//install influxdb  
#apt-get install -y influxdb
```

```
//install latest version of grafana  
# wget https://dl.grafana.com/oss/release/grafana_5.3.4_arm64.deb  
# dpkg -i grafana_5.3.2_arm64.deb  
# systemctl daemon-reload  
# systemctl start grafana-server  
# systemctl status grafana-server
```

```
#apt-get install -y python-pip python-dev python-scipy  
#pip install setuptools  
#pip install paho-mqtt  
#pip install influxdb
```

```
#apt-get install -y build-essential  
#apt-get install -y unzip  
#apt-get install -y cmake  
#apt-get install -y libblas-dev liblapack-dev libssl-dev libhdf5-dev  
#pip install keras
```

```
#pip install tensorflow-1.11.0-cp27-none-linux_aarch64.whl
```

Be patient, the installation takes about an hour on a 710 device

1.3 Some additional configuration

We are ready to test TensorFlow now, launch Python and load tensorflow

```
#python
>> import TensorFlow
```

You will run into error message “libstdc++.so.6:version ‘GLIBCXX_3.4.22’ not found. To resolve this issue, we need to go through the steps below:

```
#apt-get install -y software-properties-common python-software-properties
#add-apt-repository ppa:ubuntu-toolchain-r/test
#apt-get update
#apt-get upgrade
#apt-get install libstdc++6
```

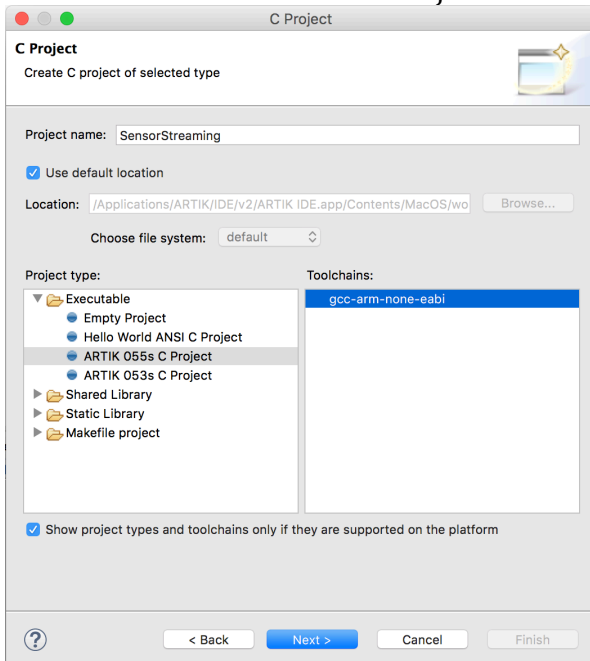
2. Stream telemetry data from ARTIK 055s to 710

In my environment, I am using an ARTIK 055s to collect sensor data and stream the telemetry data to 710 for visualization and prediction.

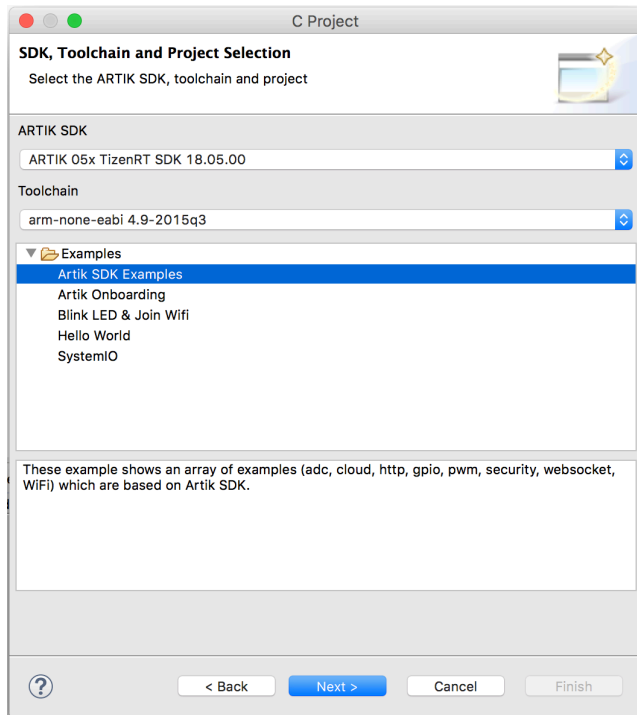
2.1 Create an ARTIK 055s application for sensor data collection and streaming

We will create an application by using ARTIK SDK/IDE for

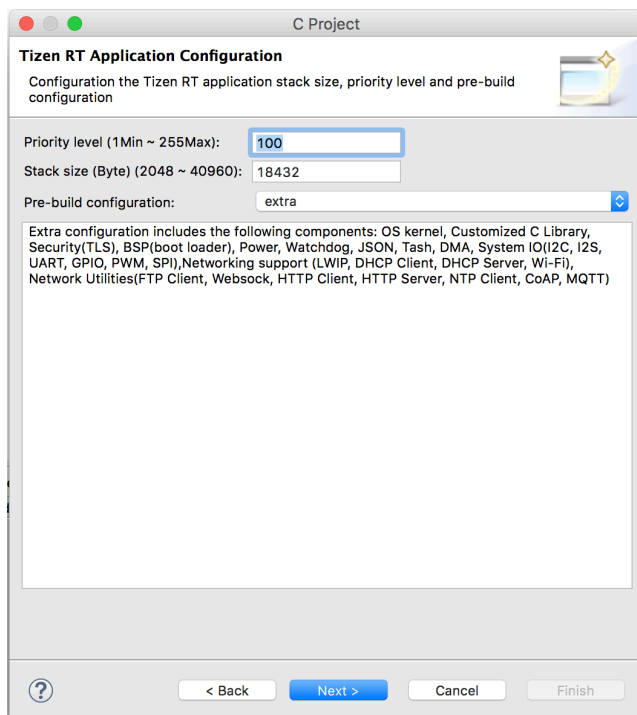
1. Launch your ARTIK IDE, go to File->New->C Project.
2. In the “C Project” dialog, select “ARTIK 055s C project”, and choose “gcc-arm-none-eabi” as the default toolchain. Name the Project name as “SensorStreaming”. Then, click Next.



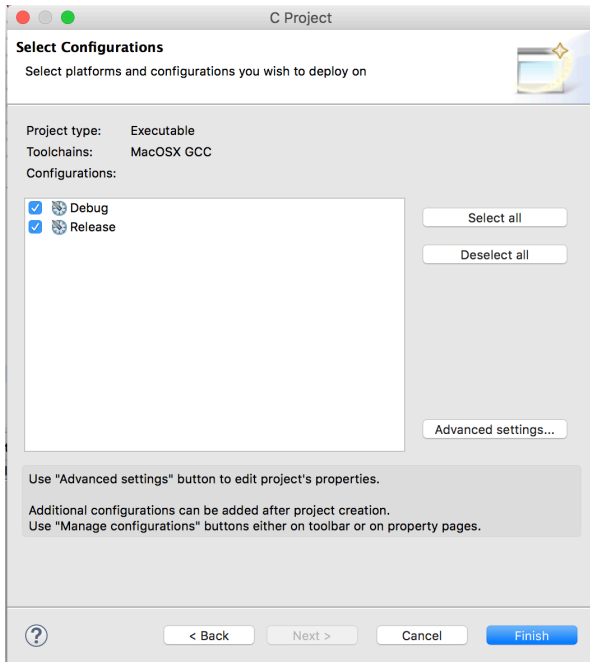
3. In “SDK, Toolchain and Project Selection” dialog, select “ARTIK 05x TizenRT SDK 18.05.00” as the target SDK version. Click on “Artik SDK Examples”, we will use this example as our template to generate an application for sensor data collecting. Then, click Next.



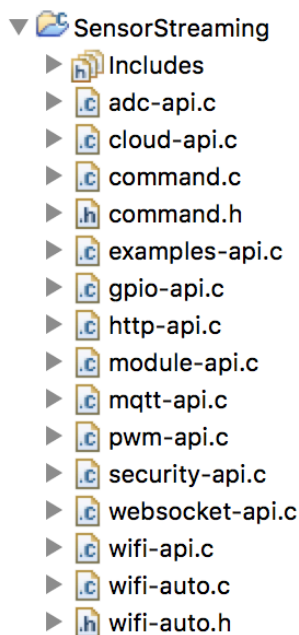
4. In “Tizen RT Application Configuration” dialog, use the default Priority level and Stack Size settings. Change the Pre-build configuration to “extra”, then click Next.



5. In “Select Configurations” dialog, keep the default Debug and Release builds settings, and click Finish.



6. Now you have an application 'SensorStreaming' created in your IDE Project Explorer.



2.2 Adapt the Application


1. Open wifi-auto.c, look for WIFI_SSID at the beginning of the file, and replace them by using the SSID/PWD of your own router.

```
#define WIFI_SSID "REPLACE_WITH_YOUR_ROUTER_SSID"  
#define WIFI_PASSPHRASE "REPLACE_WITH_YOUR_ROUTER_PASSWORD"
```

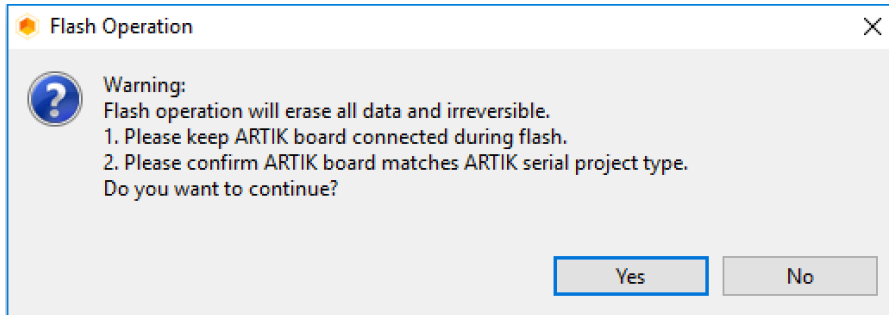
2. Drag and drop the attached mqtt-api.c file to SensorStreaming project. When you are prompted to overwrite existing file, click Yes to proceed.

In mqtt-api.c, we will publish sensor reading from pin ADC0 to a MQTT broker in a loop.

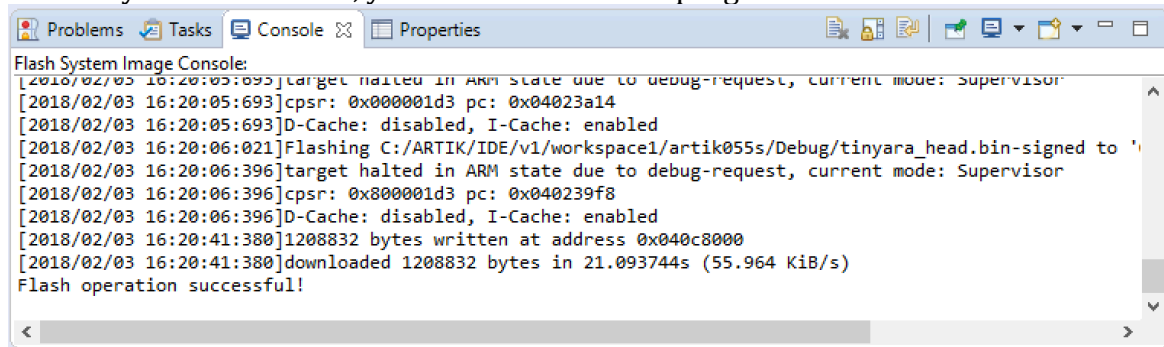
3. Build the application by selecting Project->Build Project. You can see the build log from the IDE Console tab.

4. Once you have a successful build, click the “Flash System Image” button  to flash the newly built image to the board.

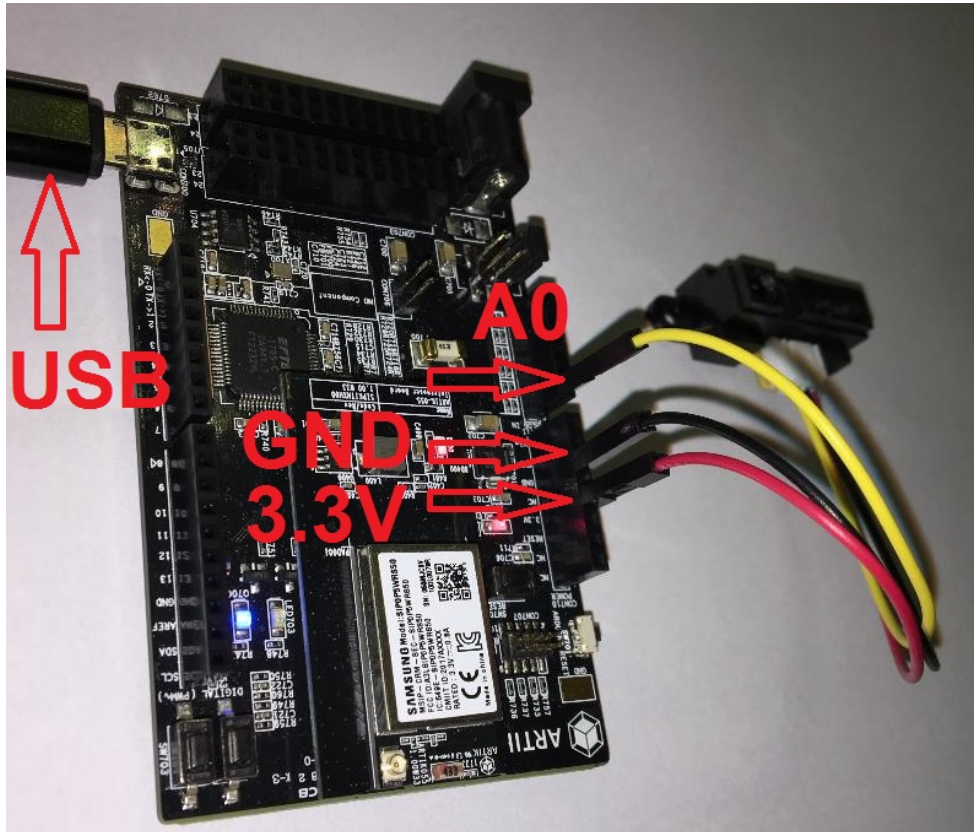
As a precaution, a warning dialog will pop up to confirm you want to flash the connected ARTIK device. Click “Yes” to proceed.



5. From your IDE console, you can view the flash progress.

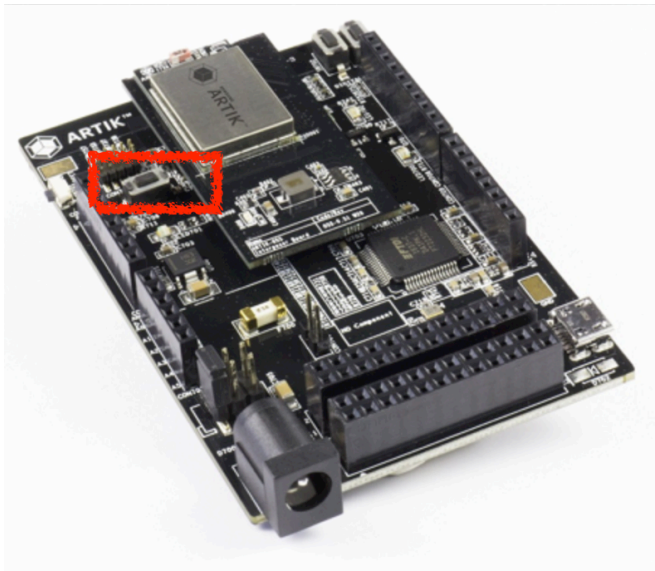


2.3 Connect the sensor to ARTIK 055s



2.4 Stream sensor data to ARTIK710 gateway device via MQTT

1. Push the Reset button.



2. From your serial console, you should be able to see the message below when the board boots

```
U-Boot 2017.01-00013-g814fa7d (Jan 08 2018 - 15:17:26 +0900)

CPU:   Exynos200 @ 320 MHz
Model: ARTIK-05x based on Exynos T20
DRAM:  1.3 MiB
WARNING: Caches not enabled
BL1 released at 2017-3-13 15:00
SSS released at 2017-09-12
WLAN released at 2017-12-21
Flash: 8 MiB
*** Warning - bad CRC, using default environment

In:     serial@80180000
Out:    serial@80180000
Err:    serial@80180000
Autoboot in 50 milliseconds
gpio: pin gpg16 (gpio 46) value is 1
## Starting application0 at 0x040C8020 ...
s5j_sflash_init: FLASH Quad Enabled
i2c_uio register: Registering /dev/i2c-0
i2c_uio register: Registering /dev/i2c-1
System Information:
    Board: ARTIK053S
    Version: 1.0.12
    Commit Hash: d1dfb56208b5897ac852a713b6f760ce4efdbd7a
    Build User: ARTIK@Samsung
    Build Time: 2018-07-17 15:58:54
    System Time: 01 Jan 2010, 00:00:00 [s] UTC Hardware RTC
Support
    ARTIK SDK version: 1.8.0
TASH>>
```

up.

3. From the ARTIK 055s TASH shell, connect to the MQTT broker running on ARTIK710. Use your ARTIK710's IP address to replace the one below. Press <Enter> to continue.

```
TASH>>mqtt connect <YOUR_ARTIK710_IP_ADDRESS>
Starting supplicant in foreground...
Connected to ARTIK
Client mosq/;L<pQ^K@gIIb3Xt1lu sending CONNECT
TASH>>Client mosq/;L<pQ^K@gIIb3Xt1lu received CONNACK
MQTT connection result: OK
```

4. Publish sensor telemetry data to ARTIK710 by running 'mqtt publish'.

3. Integrate with InfluxDB and Grafana on ARTIK710

```
TASH>>mqtt publish
ADC0=268
MQTT: publish 268 on Machine
Client mosq/W0MtF?rTGxm<u<TJ]v sending PUBLISH (d0, q0, r0, m1,
'Machine', ... (3 bytes))
TASH>>Assage published: 1
ADC0=1601
MQTT: publish 1601 on Machine
Client mosq/W0MtF?rTGxm<u<TJ]v sending PUBLISH (d0, q0, r0, m2,
'Machine', ... (4 bytes))
MQTT message published: 2
ADC0=1611
MQTT: publish 1611 on Machine
Client mosq/W0MtF?rTGxm<u<TJ]v sending PUBLISH (d0, q0, r0, m3,
'Machine', ... (4 bytes))
MQTT message published: 3
ADC0=582
MQTT: publish 582 on Machine
Client mosq/W0MtF?rTGxm<u<TJ]v sending PUBLISH (d0, q0, r0, m4,
'Machine', ... (3 bytes))
MQTT message published: 4
```

3.1 Configure InfluxDB

1. Verify InfluxDB instance is running.

```
[root@artik ~]# systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time se
ries database
   Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor
 preset:
   enabled)
   Active: active (running) since Tue 2018-10-16 18:55:56 UTC; 49min
 ago
     Docs: man:influxd(1)
  Main PID: 2558 (influxd)
    Tasks: 14 (limit: 815)
   CGroup: /system.slice/influxdb.service
           └─2558 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```


2. Make sure your host machine is on the same network as your target ARTIK 710 device. Launch a browser on your host machine and visit <YOUR_ARTIK710_IP_ADDRESS>:8083. Port 8083 is the default admin interface of your local InfluxDB.



3.2 Configure Grafana

Verify Grafana instance is running

```
[root@artik electronica]# systemctl status grafana
● grafana.service - Starts and stops a single grafana instance on this system
   Loaded: loaded (/lib/systemd/system/grafana.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Mon 2018-10-29 18:38:42 UTC; 19min ago
     Docs: http://docs.grafana.org
    Main PID: 2336 (grafana)
      CGroup: /system.slice/grafana.service
              └─2336 /usr/sbin/grafana --config=/etc/grafana/grafana.ini
   cfg:default
   t.paths.logs=/var/log/grafana  cfg:default.paths.data=/var/lib/grafana
```

3.3 Subscribe to incoming data and write it to InfluxDB

We can use the Python script below to listen to incoming MQTT messages from ARTIK055s and send it to InfluxDB for data storage. We will use the provided subscriber.py code for MQTT message subscription etc.

The functionalities of the subscriber.py can be divided into a few parts:

Set up an InfluxDB client and create InfluxDB database

InfluxDB uses port 8086 to listen for incoming queries. In the code snippet below, we connect to the port, and pull a list of existing databases. If database with name 'ARTIKML' does not exist, we will add it.

```
dbname = 'ARTIKML'
dbclient = InfluxDBClient('localhost', 8086, 'root',
                          'root', dbname)
dblist = dbclient.get_list_database()
db_found = False
for db in dblist:
    if db['name'] == dbname:
        db_found = True
if not(db_found):
    print('Database' + dbname + ' not found, trying to create')
    dbclient.create_database(dbname)
```

Listen for incoming messages from ARTIK 05x

In the code, we spin up an MQTT client to listen for incoming messages from ARTIK 05x. The MQTT topic we are subscribing to is "Machine".

```
import paho.mqtt.client as mqtt
import datetime
import time

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("Machine")

def on_message(client, userdata, msg):
    print("Received a message on topic: " + msg.topic)
    print("payload: " + msg.payload)

# Initialize the MQTT client that should connect to the
# Mosquitto broker
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
connOK=False
while(connOK == False):
    try:
        client.connect("localhost", 1883, 60)
        connOK = True
    except:
        connOK = False
        time.sleep(2)

# Blocking loop to the Mosquitto broker
client.loop_forever()
```

Extract incoming data and write it to InfluxDB

```
def on_message(client, userdata, msg):
    print("Received a message on topic: " + msg.topic)
    receiveTime=datetime.datetime.utcnow()
    message=msg.payload.decode("utf-8")
    val = int(message)
    json_body = [
        {
            "measurement": "artikfactory",
            "tags": {
                "host": "machine01",
                "region": "US"
            },
            "time": receiveTime,
            "fields": {
                "pressure": val,
            }
        }
    ]

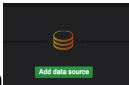
    dbclient.write_points(json_body)
```

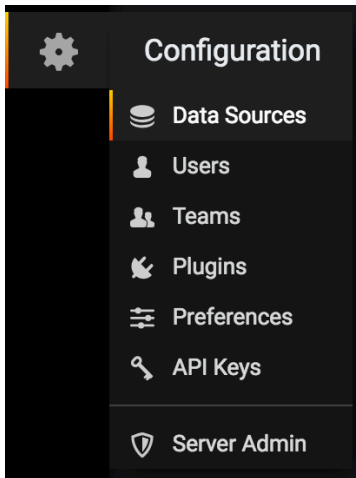
Once we extract the data from incoming message from 05x, we need to generate a JSON format message body as the input of InfluxDB. In our code, we create a measurement with the name "ARTIKfactory", we included tags info such as host name, region etc. In "fields", we only write one data field into the database, which is the pressure sensor data.

3.4 Configure Grafana for visualization

Launch a browser on your host machine, and access the 3000 port of your ARTIK 710 device, you should be able to see the Grafana portal. Use admin/admin to log in.

Configure Grafana data sources

Click on the "Add data source" button , or go to Configuration/Data Sources to configure Grafana data source,



click on the “Add data source” green button on the top right corner, and add configuration details like below:

- Name should be configured as your InfluxDB measurement name, which is “artikfactory”
- Type is InfluxDB
- URL links to the local HTTP InfluxDB REST API interface, which should be <http://localhost:8086>
- Database name is ARTIKML
- User/password is root/root

Data Sources / New
Type: InfluxDB

Settings

Name: ⓘ Default ☒

Type:

HTTP

URL: ⓘ

Access: Help ▶

Auth

Basic Auth ☐ With Credentials ⓘ ☐

TLS Client Auth ☐ With CA Cert ⓘ ☐

Skip TLS Verification (Insecure) ☐

Advanced HTTP Settings

Whitelisted Cookies ⓘ

InfluxDB Details

Database:

User: Password:

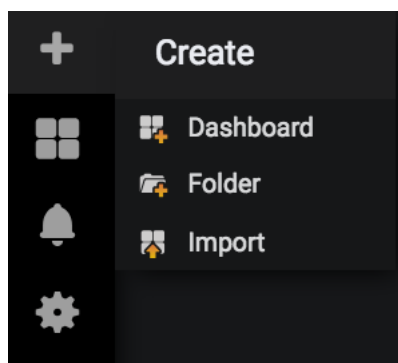
Click the “Save and Test” green button at the bottom to verify Grafana’s connection to the database is

✓ Data source is working

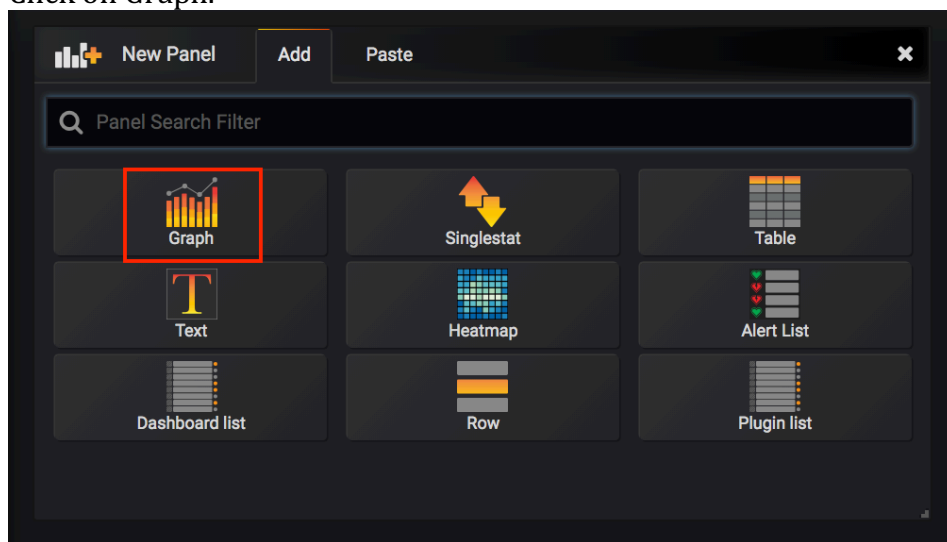
live. You should be able to see the message if everything is configured properly.

Visualize the real-time sensor reading

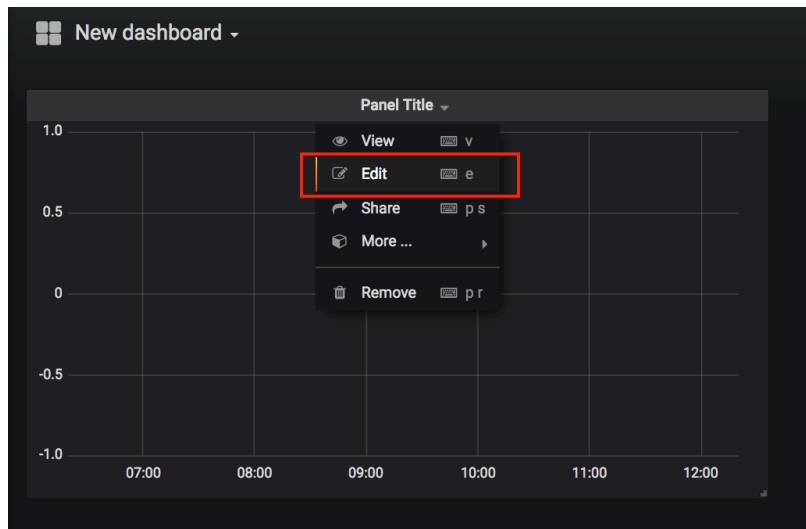
As the next step, we will add a dashboard for visualization. From the left pane of your Grafana portal, select +/Create Dashboard.



Click on Graph.



4. Click on the little arrow next to Panel Title, and select Edit.



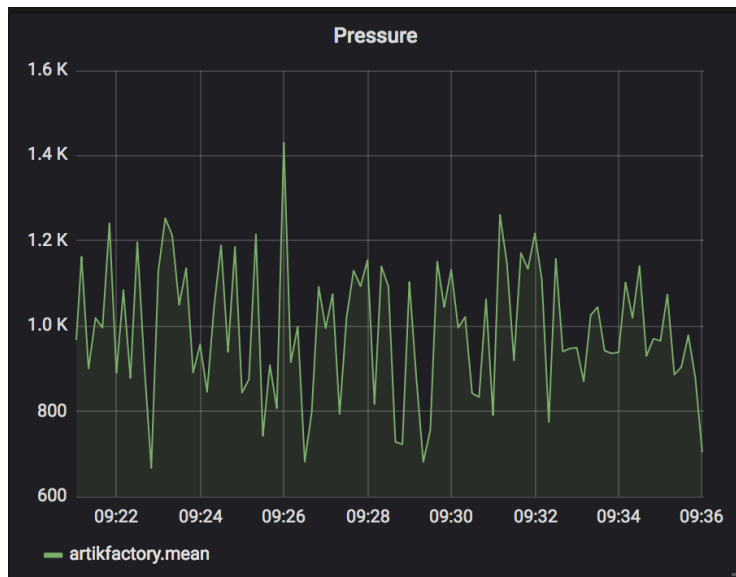
Under Metrics tab, configure Data Source as "artikfactory" and filter it by using criteria below.

A screenshot of the "Metrics" configuration tab in a dashboard. The "Data Source" is set to "default". The query is configured as follows: FROM default artikfactory WHERE; SELECT field (pressure) mean (); GROUP BY time (10s) fill (null); FORMAT AS Time series; ALIAS BY Naming pattern. The "artikfactory" data source, "field (pressure)" field, "time (10s)" group by, and "Time series" format are highlighted with red boxes.

Then, you will be able to see the pressure data in the dashboard. By default, the dashboard shows the data collected in the last 6 hours. You can click the "Last 6 hours" on top right corner, and make it show the data in the last 15 minutes, and change the refresh rate to every 10s.

A screenshot of the time range and refresh rate configuration interface. The "Custom range" section shows "From: now-15m" and "To: now". The "Refreshing every:" is set to "10s". The "Quick ranges" section shows various time ranges, with "Last 15 minutes" highlighted by a red box. The "Apply" button is visible.

You will be able to see the graph as below.



4. Use TensorFlow for Predictive Analysis

Train the time-series prediction model

For Time Series Prediction, we developed a Long Short-Term Memory network(LSTM network) in Python using Keras deep learning library. Pre-trained model can be found [here](#).

Use pre-trained model for ML Inference

In subscriber.py, we retrieve the last 3 pressure sensor readings from InfluxDB, append the latest sensor reading to it, and feed these data to our TensorFlow model for prediction.

```
results = dbclient.query("SELECT pressure FROM ARTIKfactory ORDER BY time DESC  
LIMIT 3")  
points = results.get_points()  
items = []  
X = []  
Items.insert(0, val)  
for item in points:  
    items.insert(0, item['pressure'])  
X.append(items)  
X=np.array(X)  
print("Historical Data")  
print(X)  
classes = model.predict(X)  
predicted_value = int(classes[0][0])  
print("Predicted Data")  
print(predicted_value)  
print("\n")
```

Launch the python script

```
[root@artik ~]# python subscriber.py
Using TensorFlow backend.
Keras model loaded
Connected with result code 0
Received 348.0 at 2018-09-10 04:07:51.404227
Historical Data
[[1059. 1633. 1634. 348.]]
Predicted Data
845

Received 1202.0 at 2018-09-10 04:07:56.402263
Historical Data
[[1633. 1634. 348. 1202.]]
Predicted Data
1629

Received 1635.0 at 2018-09-10 04:08:01.404101
Historical Data
[[1634. 348. 1202. 1635.]]
Predicted Data
1646
```

Visualize the predicted data

From your Grafana portal, click the 'Panel Title', and select Edit button.

You should be able to see the view as shown below. Click 'Metrics', then Query button to add a new query to the graph.

Select *artikfactory* measurement, select *predicted* field and update it every 10 secs.

Click Back to dashboard.

The screenshot shows the Grafana query editor interface. At the top, the 'Data Source' is set to 'artikfactory'. Below this, there are two query panels, A and B. Panel A is configured with 'FROM' default, 'artikfactory' as the data source, 'WHERE' clause, 'SELECT' field (pressure), 'mean ()' aggregation, 'GROUP BY' time (10s), 'fill (null)' fill, and 'FORMAT AS' Time series. Panel B is configured with 'FROM' default, 'artikfactory' as the data source, 'WHERE' clause, 'SELECT' field (predicted), 'mean ()' aggregation, 'GROUP BY' time (10s), 'fill (null)' fill, and 'FORMAT AS' Time series. The 'predicted' field in the 'SELECT' clause of Panel B is highlighted with a red box.

Overtime, you will be able to see the charts of predicted data and actual data. In the graph below, the green line represents the actual sensor data, and yellow line is the predicted data.

