A two step approach to get started sending data to HiveEyes Grafana dashboard

ONE

Start from this page
https://hiveeyes.org/docs/system/acquisition/index.html#daq-mqtt

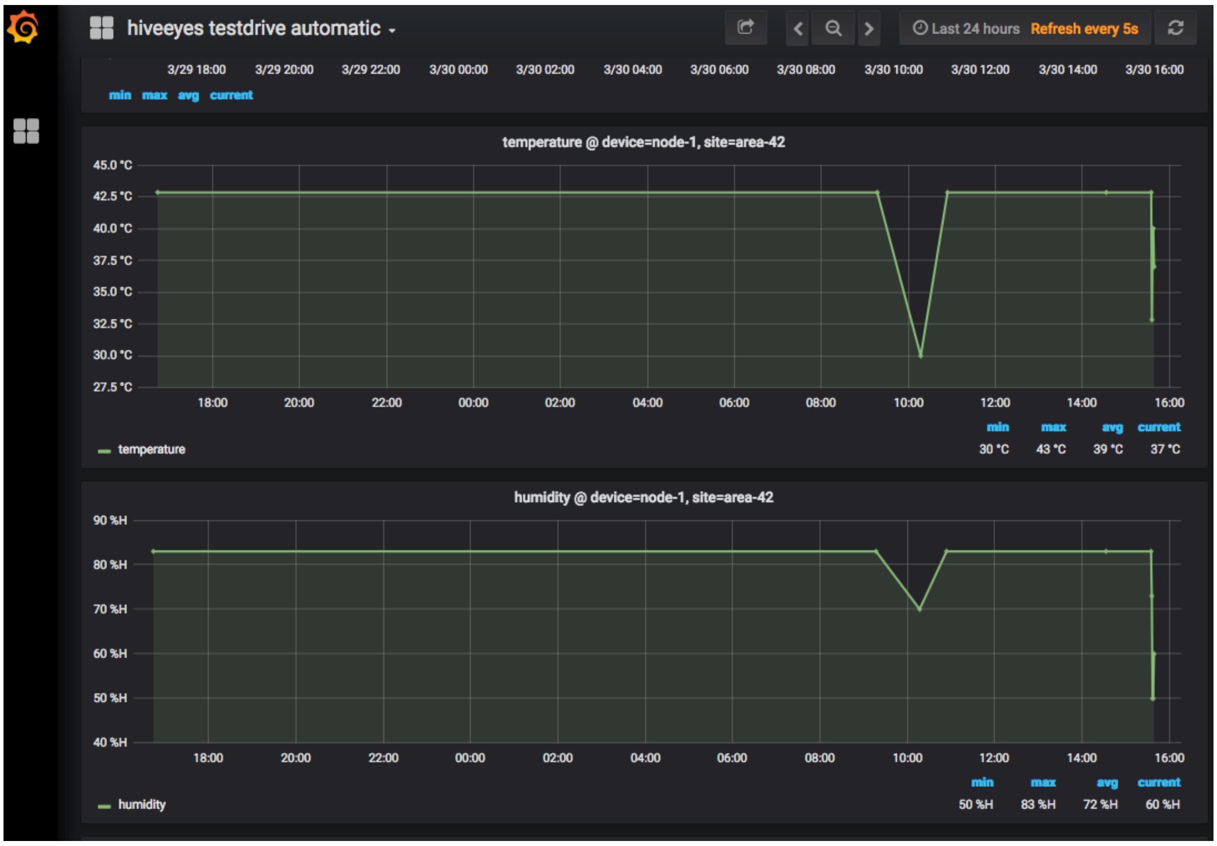
```
# Get hold of a MQTT client of your choice
aptitude install mosquitto-clients

# Define the target server
export BROKER=swarm.hiveeyes.org
# Define the channel as realm/network/gateway/node
# use the available "testdrive" network that you will find
# in the Graphana list of dashboards as "hiveeyes testdrive automatic"
# "area-42" and "node-1" are not mandatory keywords, you can use your own ones
# and they'll be automatically created on the dashboard
export CHANNEL=hiveeyes/testdrive/area-42/node-1


# Publish a single measurement sample
echo '{"temperature": 42.84, "humidity": 83}' | mosquitto_pub -h $BROKER -t
$CHANNEL/data.json -l

# Check that the server receives your publication
mosquitto_sub -h swarm.hiveeyes.org -p 1883 -t 'hiveeyes/testdrive/#' -v

#Data should appear automatically in Grafana
```



TWO

When you're ready with this step, move to the python code that you'll find in In GitHub simply updating lines 31 to 36 with your set of realm/network/gateway/node described previously.
Suggestion: Before connecting any sensor, try to send static data by simply comment the line retrieving data from the DTH sensor and setting constant values for a check.

```python
# -*- coding: utf-8 -*-
# (c) 2015-2016 Andreas Motl, Hiveeyes <andreas@hiveeyes.org>
# (c) 2015-2016 Richard Pobering, Hiveeyes <richard@hiveeyes.org>
# License: GNU LGPL, see https://www.gnu.org/licenses/lgpl-3.0.txt
"""
Example program to send measurement data to the MQTT software bus to
store it persistently in a time series database, draw beautiful graphs,
and optionally get alerted for important events (e.g. "Schwarmalarm").
Suitable even for many beekeepers having multiple hives at different sites.
"""
import os
import json
import logging
import paho.mqtt.client as mqtt
import sys
import Adafruit_DHT as dht
import time

logger = logging.getLogger(__name__)

# Data capture and upload interval in seconds. Less interval will eventually hang the DHT22.
INTERVAL=2

sensor_data = {'temperature': 0, 'humidity': 0}


def send_dummy_measurement():

    next_reading = time.time()

    # --------------------------------
    # A. Where to send measurements to
    # --------------------------------

    # The MQTT host
    mqtt_host = 'swarm.hiveeyes.org'

    # The MQTT topic
    # See also: https://hiveeyes.org/docs/system/vendor/hiveeyes-one/topology.html#rationale
    mqtt_topic = u'{realm}/{network}/{gateway}/{node}/data.json'.format(
        realm   = 'hiveeyes',               # Kollektiv
        network = 'testdrive',                  # Imker
        gateway = 'area-42',            # Standort
        node    = 'node-1'              # Beute
    )

    # ------------------------------------------------------------
    # C. Publish measurement data to software bus / send to backend
    # ------------------------------------------------------------

    # Serialize data as JSON
    payload = json.dumps(sensor_data)

    # Publish to MQTT
    pid = os.getpid()
    client_id = '{}:{}'.format('hiveeyes', str(pid))
    backend = mqtt.Client(client_id=client_id, clean_session=True)
    backend.connect(mqtt_host)

    #backend.publish(mqtt_topic, payload)
    backend.loop_start()

    try:
        while True:
    #           humidity,temperature = dht.read_retry(dht.DHT22, 4)
    #           humidity = round(humidity, 2)
    #           temperature = round(temperature, 2)
                humidity = 60.0
                temperature = 37.0
                print(u"Temperature: {:g}\u00b0C, Humidity: {:g}%".format(temperature, humidity))
                sensor_data['temperature'] = temperature
                sensor_data['humidity'] = humidity

                # Sending humidity and temperature data to ThingsBoard
                backend.publish(mqtt_topic, json.dumps(sensor_data))

                next_reading += INTERVAL
                sleep_time = next_reading-time.time()
                if sleep_time > 0:
                    time.sleep(sleep_time)
    except KeyboardInterrupt:
        pass

    backend.loop_stop()

    backend.disconnect()


if __name__ == '__main__':
    send_dummy_measurement()
```

AND FURTHER

When this is working, you can ask HiveEyes for a personal dashboard with your personal identification

You have just to add

```
username = "XXXXXXX"
password = "XXXXXXX"
backend.username_pw_set(username, password)
```

between

```
 backend = mqtt.Client(client_id=client_id, clean_session=True)
```

and

```
backend.connect(mqtt_host)
```

Update the network and data should appear automatically in your personal Grafana dashboard