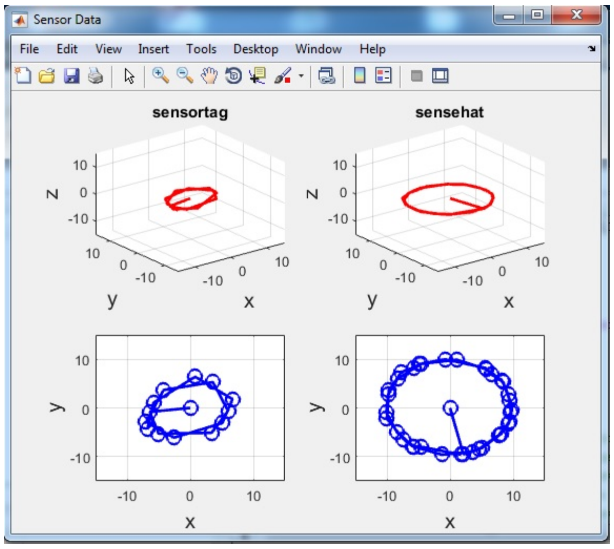
Processing sensor's data on Matlab over MQTT

Thesis · July 2017

DOI: 10.13140/RG.2.2.27998.46407

Björn Ziegler
Fachhochschule Kärnten
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Mahmoud Mansour
Technische Hochschule Mittelhessen
6 PUBLICATIONS **1** CITATION
SEE PROFILE



SenseHat and SensorTag Data Processing

IoT-Seminar SS2017

Authors:

Frank Lühring
email: frank.luehring@ei.thm.de
Mahmoud Mansur
email: mahmoud.mansour@iem.thm.de
Björn Ziegler
email: bjoern.ziegler@ei.thm.de

July 26, 2017

Contents

1 Introduction

This document is the result of groups 4 work in the Internt of Things-Seminar by Prof. Dr.-Ing Birkel in the summer-semester of 2017. We had to gather in groups of two to three and work on different topics all in regards to the Internet of Things. We choose the topic:

SenseHat and SensorTag Data Processing

We got as further information that processing of the data have to be done in Matlab or Octave and the protocol which has to be used is the MQTT-protocol. So final task was to gather data from different sensor sources, timestamp it, and send these data with the MQTT-protocol to an MQTT-Broker. From there we import the data into Matlab and visualize it. The sensors are attached to a turning wheel to simulate the movement(see figure 1).We will explain the hard- and soft-ware we used, also the problems and solutions we discovered. This document will be more a manual for students than a technical report.

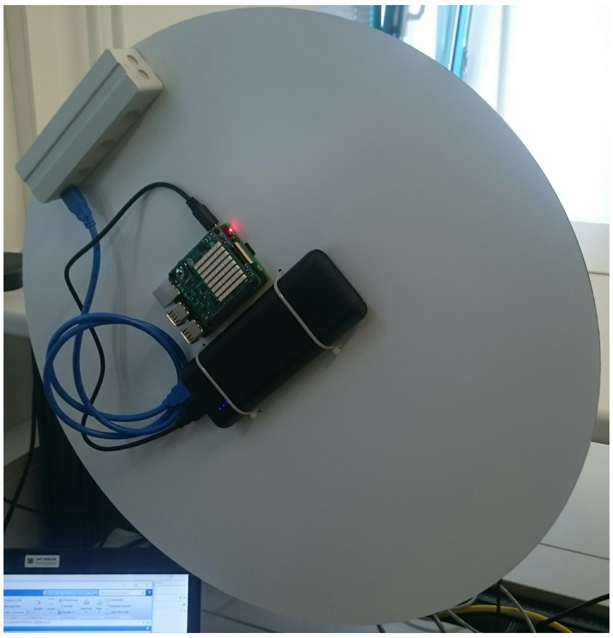


Figure 1: Wheel

2 Fundamentals

In this part we are going to give you an overview over the hard- and soft-ware we used in this project. The hard- and software were placed at our disposal by the THM. Raspbian was already installed on the raspberry pi, so we didn't have to search for a suitable OS. Every student could get a set consisting of a Raspberry Pi 3 and Sense-HAT to work at home with. Just for work at home we had to use a private license of Matlab or had to change software to octave.

2.1 Hardware

2.1.1 Raspberry Pi

Raspberry Pi 3 Model B released in February 2016, is bundled with on-board WiFi, Bluetooth and USB boot capabilities. As of January 2017, Raspberry Pi 3 Model B is the newest mainline Raspberry Pi. We use the Raspberry as processing unit for several sensors which we send to an MQTT-Broker.

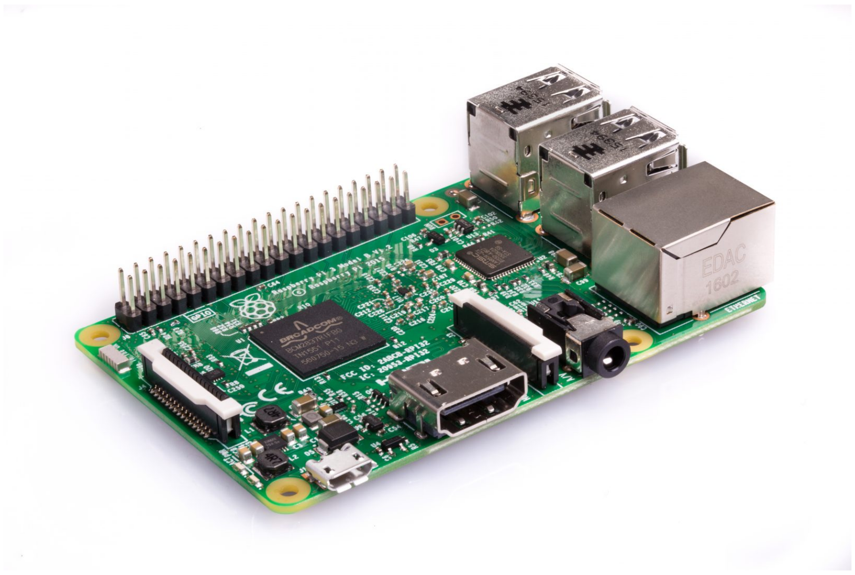


Figure 2: Raspberry Pi 3

- SoC: Broadcom BCM2837
- CPU: 4 x ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD

- GPIO: 40-pin header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4 x USB 2.0, Ethernet, Camera Serial
- Interface (CSI), Display Serial Interface (DSI)

2.1.2 Sense Hat

The Raspberry Pi Sense HAT is attached on top of the Raspberry Pi via the 40 GPIO pins (which provids the data and power interface) to create an "Astro Pi". For this project we use the gyroscope.

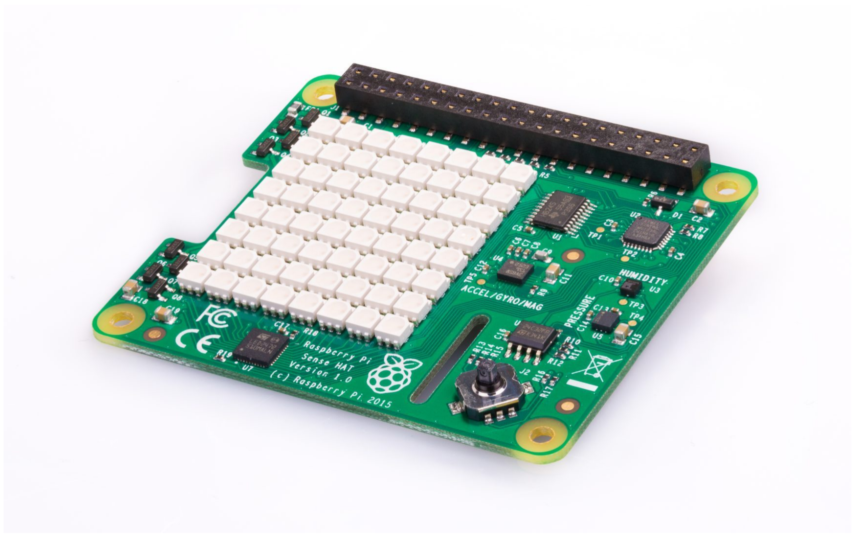


Figure 3: Sense-HAT

- Gyroscope - angular rate sensor: $\pm$245/500/2000dps
- Accelerometer - Linear acceleration sensor: 2/4/8/16 g
- Magnetometer - Magnetic Sensor: 4/8/12/16 gauss
- Barometer: 260 - 1260 hPa absolute range (accuracy depends on the temperature and pressure, $\pm$0.1 hPa under normal conditions)
- Temperature sensor (Temperature accurate to $\pm$ 2$°C$ in the 0-65$°C$ range)
- 8x8 LED matrix display
- Small 5 button joystick

2.1.3 Sensor Tag CC2650

The CC2650 SimpleLink Multistandard Wireless MCU device is a wireless MCU targeting Bluetooth, ZigBee and 6LoWPAN, and ZigBee RF4CE remote control applications. The device is a member of the CC26xx family of cost-effective, ultralow power, 2.4-GHz RF devices. Very low active RF and MCU current and low-power mode current consumption provide excellent battery lifetime and allow for operation on small coin cell batteries and in energy-harvesting applications.

Figure 4: Sensor Tag

- Normal Operation voltage: 1.8 to 3.8 V
- 10 sensors including support for light, digital microphone, magnetic sensor, humidity, pressure, accelerometer, gyroscope, magnetometer, object temperature, and ambient temperature

2.1.4 The Wheel

The raspberry pi with mounted Sense Hat and the Sensor Tag CC2650 are attached to a wheel like you see in figure 5. The wheel itself is vertically attached to a small DC Motor. By powering on the power source and changing the values we can adjust the rotation speed. However, the power source and motor aren't of further interest for this project.

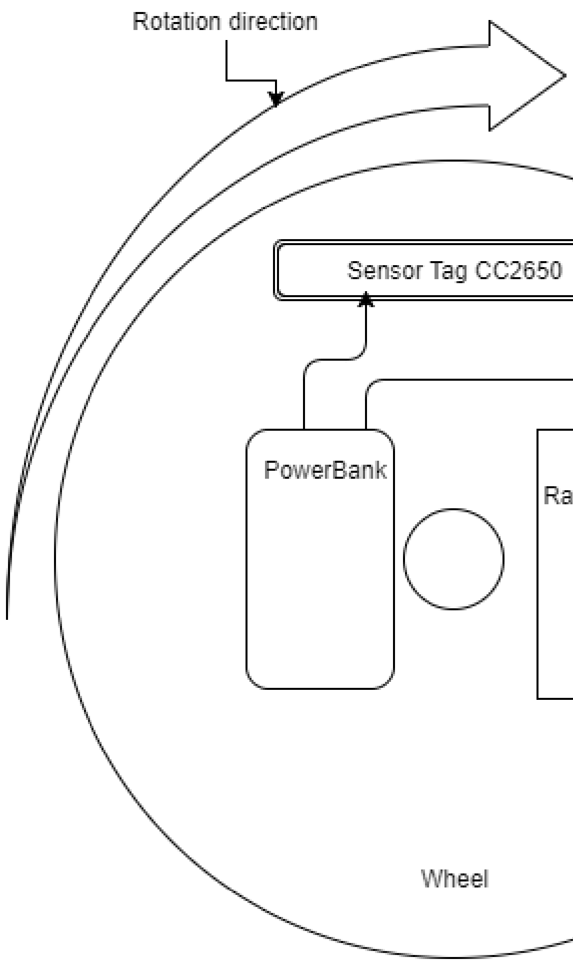


Figure 5: Wheel

2.2 Software

We will give you a short overview over the Software we used in this project. This should give you shallow knowledge and a base for further investigation.

2.2.1 Raspbian

Raspbian[1] is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi. The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

2.2.2 Node-RED

Node-RED is a programming tool for wiring hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.We will explain more in chapter 3.1 and 3.2.

2.2.3 Matlab

MATLAB[2] (MAtrix LABoratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python. In this project we use Java to interface with. Also we chose Matlab instead of Octave because of, in our experience, better documentation and bigger community.

2.2.4 Java

Java[3] is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

[1]https://www.raspbian.org/
[2]https://www.mathworks.com/
[3]https://www.oracle.com/java/index.html

2.2.5 MQTT

MQTT[4] stands for MQ Telemetry Transport, it's an simple and light messaging protocol invented in 1999 by IBM. It's designed for low-bandwidth, high latency or unreliable networks. The protocol uses a publish/subscribe architecture consisting of clients and a broker(see figure 6). Messages in MQTT are published on topics. Topics are treated as a hierarchy, using a slash (/) as a separator. This provides a logical structure of common themes to be created. So, for example, multiple sensors can publish their position information on the following single topic, by inserting their unique sensor name: sensors/sensor_name/position.
Multiple clients can then receive messages by creating a subscription to this specific topic. However, there are two wildcards available + or #. The + character can be used as a wildcard for a single level of hierarchy.
The # character can be used as a wildcard for all remaining levels of hierarchy, which means that it must be the final character in a subscription in order to match.

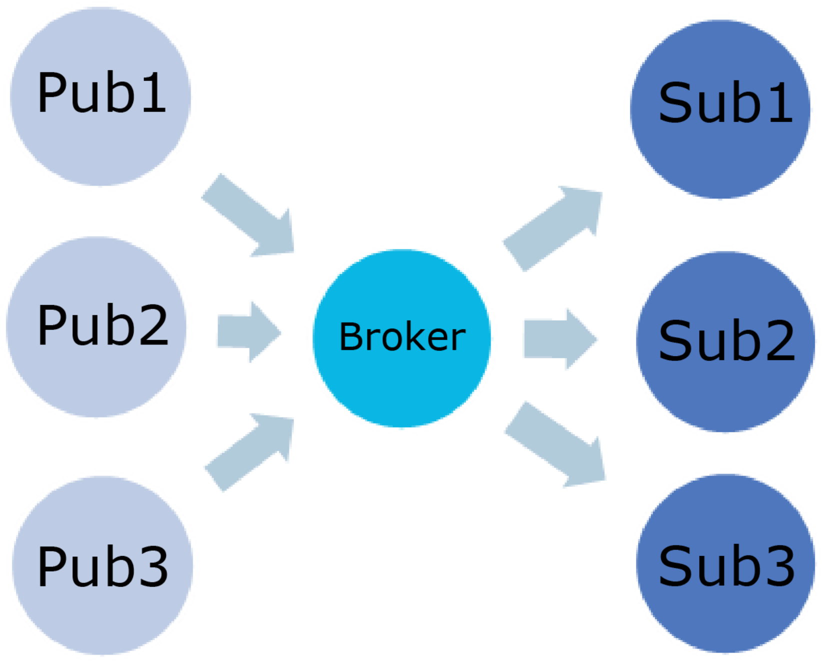


Figure 6: MQTT

[4] http://mqtt.org/

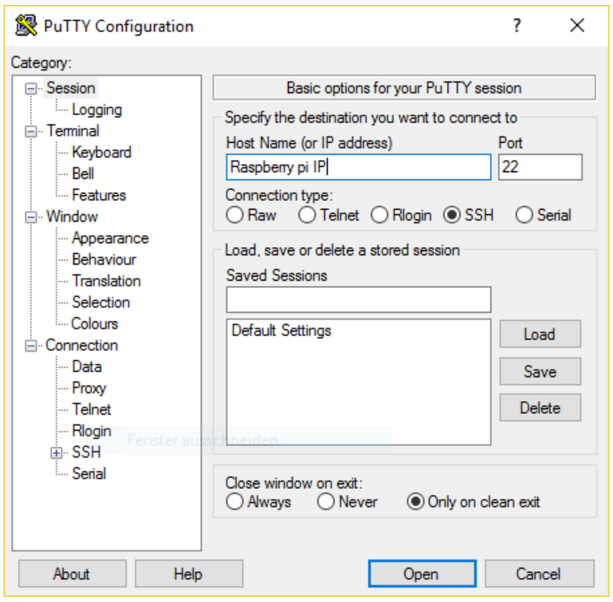
3 Implementation

In this part we explain how we put the programs and hardware into operation. All code we use that aren't in this section can be found in the attachment. It will start with the programming tool node-red which runs on the raspberry. After that we will explain the Matlab code an it's functionality. More info about how to run Java-code in Matlab can be found in the attachment5.

3.1 How to start node-red

To start node-red on the raspberry pi you have to download a program like PuTTY [5] and connect to the raspberry pi on port 22 (See figure 7: PuTTY config). Login with:

- user: pi
- password: raspberry



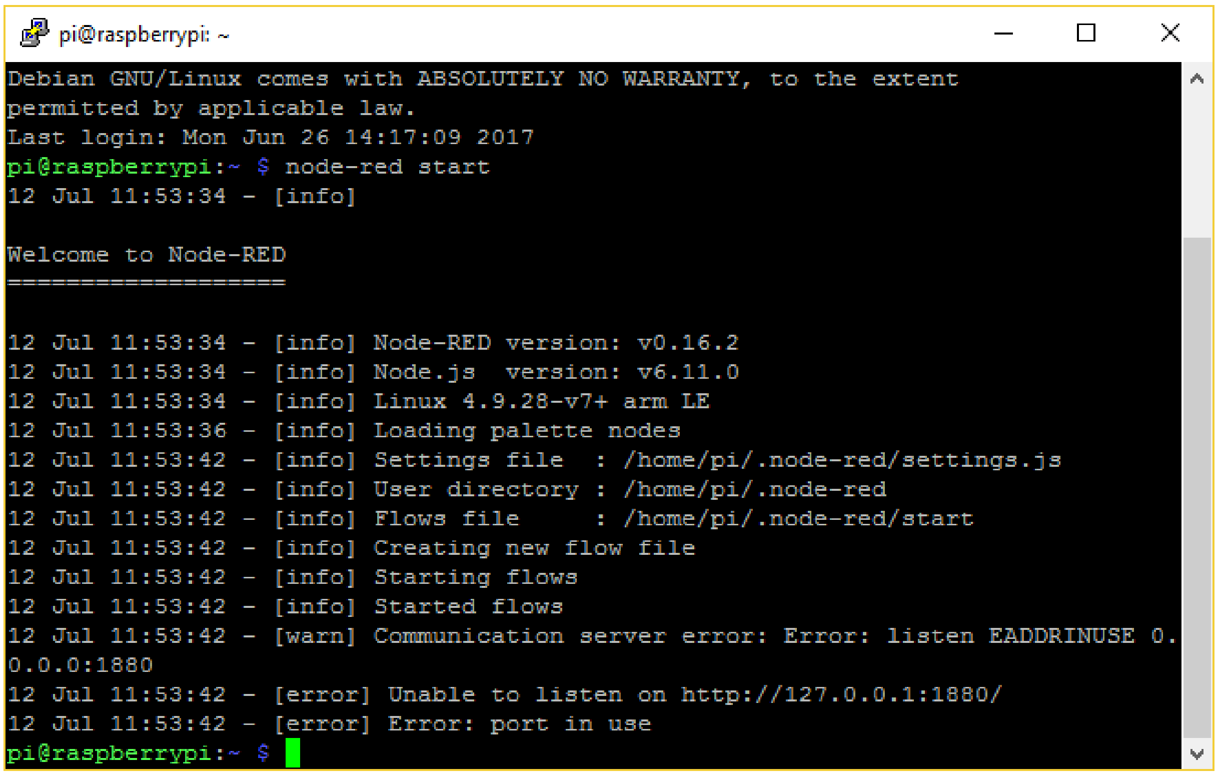
Specify the destination you want to connect to
Host Name (or IP address): Raspberry pi IP
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Figure 7: PuTTY config

You have to change the default password if you don't work in a safe environment!
To Upgrade Node-Red to up-to-date version enter following commands into the command line(See figure 8: Start nodered).

- update-nodejs-and-nodered
- sudo systemctl enable nodered.service
- node-red-start

[5] http://www.putty.org/



pi@raspberrypi: ~

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 26 14:17:09 2017
pi@raspberrypi:~ $ node-red start
12 Jul 11:53:34 - [info]

Welcome to Node-RED
===================

12 Jul 11:53:34 - [info] Node-RED version: v0.16.2
12 Jul 11:53:34 - [info] Node.js  version: v6.11.0
12 Jul 11:53:34 - [info] Linux 4.9.28-v7+ arm LE
12 Jul 11:53:36 - [info] Loading palette nodes
12 Jul 11:53:42 - [info] Settings file  : /home/pi/.node-red/settings.js
12 Jul 11:53:42 - [info] User directory : /home/pi/.node-red
12 Jul 11:53:42 - [info] Flows file     : /home/pi/.node-red/start
12 Jul 11:53:42 - [info] Creating new flow file
12 Jul 11:53:42 - [info] Starting flows
12 Jul 11:53:42 - [info] Started flows
12 Jul 11:53:42 - [warn] Communication server error: Error: listen EADDRINUSE 0.
0.0.0:1880
12 Jul 11:53:42 - [error] Unable to listen on http://127.0.0.1:1880/
12 Jul 11:53:42 - [error] Error: port in use
pi@raspberrypi:~ $
```

Figure 8: Start nodered

Check if node-red is running by entering -ip of your raspberry pi:1880- into the adress bar of your browser. Be sure to enter the port number, otherwise it won't work. If you get displayed a website like shown in figure 9 you have successful started node-red. You are now able to compile a program! (See chapter 3.2)

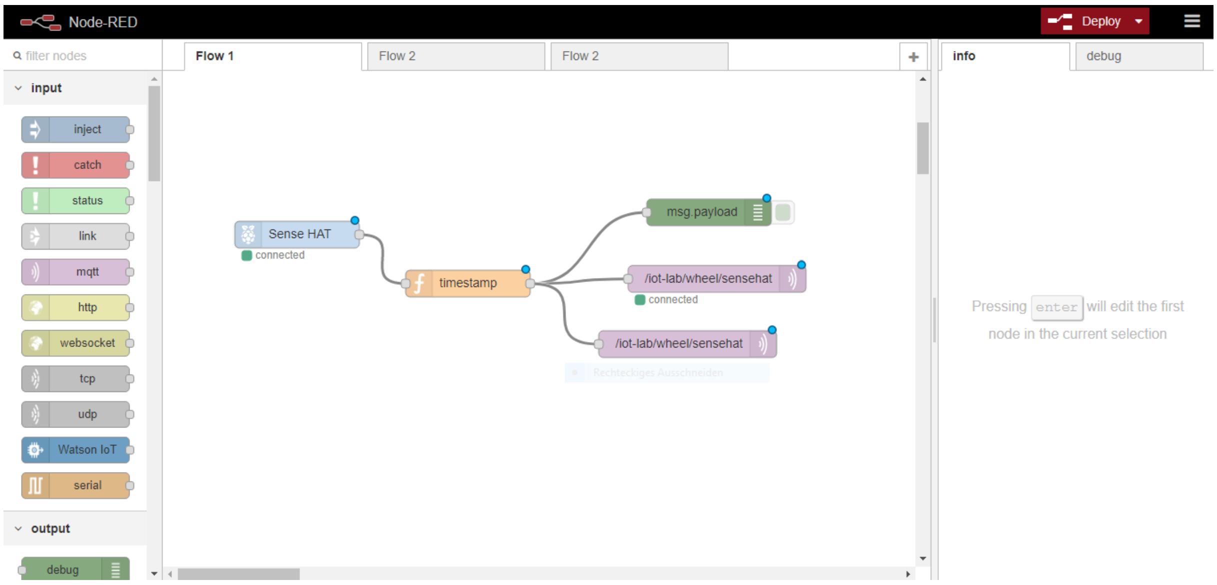


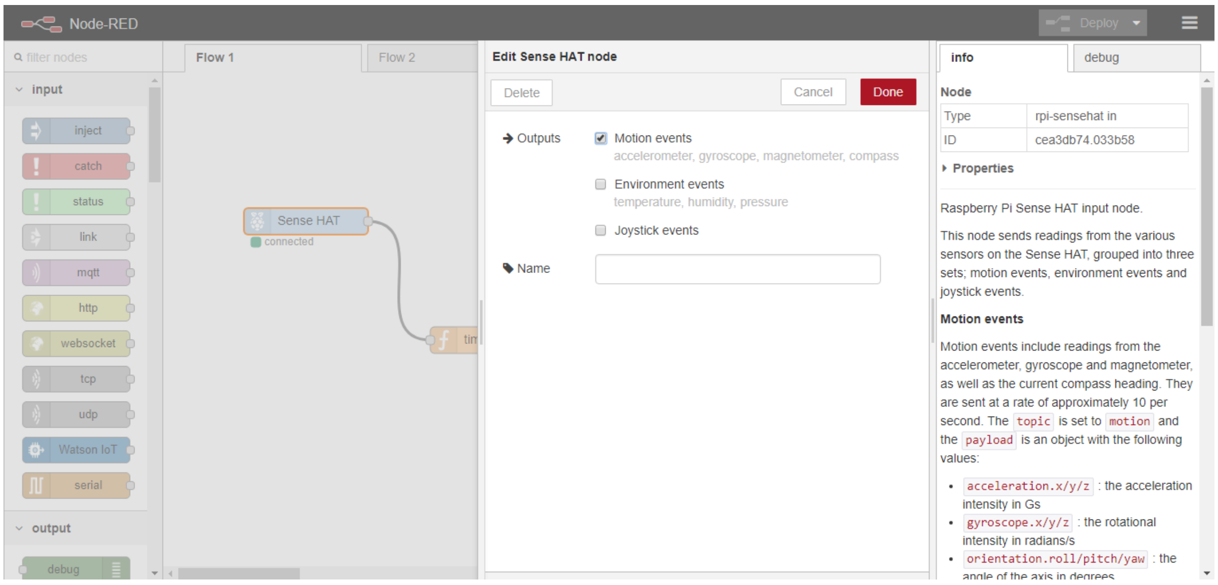
Figure 9: Node RED

3.2 Node-RED Program

Node-RED [6] works with functional blocks which you can add, delete and connect by Drag'n'Drop. In figure 9 you see our program to get the data from the Sense HAT which is connected directly to the raspberry pi, timestamp it, and send it out to two MQTT-Brokers and a debug node. One click on deploy in the upper right corner compiles your program. It will appear gray after the deploy, and will turn back to red if any changes are done to the program. The Node-red code can be imported with the source code attached in 5.1

3.2.1 Sense HAT node

The first node you see in figure 9 is the Sense HAT node. This is a premade node wich supports the Sense-HAT. Here we choose the data we need, in our case these are the motion events. Environment and joystick events aren't necessary for our purpose, so we uncheck these two points, to decrease the data traffic.



Outputs
[✓] Motion events
accelerometer, gyroscope, magnetometer, compass
[ ] Environment events
temperature, humidity, pressure
[ ] Joystick events

Name

Node

Type	rpi-sensehat in
ID	cea3db74.033b58

Properties

Raspberry Pi Sense HAT input node.

This node sends readings from the various sensors on the Sense HAT, grouped into three sets; motion events, environment events and joystick events.

Motion events

Motion events include readings from the accelerometer, gyroscope and magnetometer, as well as the current compass heading. They are sent at a rate of approximately 10 per second. The `topic` is set to `motion` and the `payload` is an object with the following values:

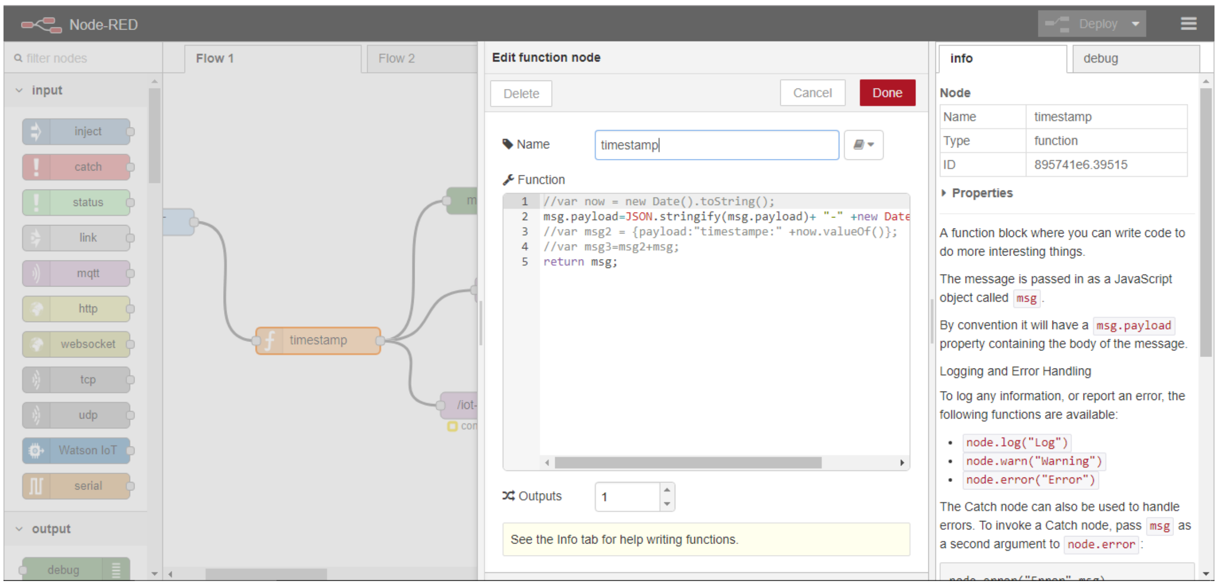
- `acceleration.x/y/z` : the acceleration intensity in Gs
- `gyroscope.x/y/z` : the rotational intensity in radians/s
- `orientation.roll/pitch/yaw` : the angle of the axis in degrees

Figure 10: Sense HAT node

3.2.2 Function node

For our purpose in Matlab we need an timestamp, to display and process the data correctly which we get from the sensors. Because we haven't found a suitable node, we had to write our own node. Therefor we used the function block, named it timestamp and entered the code in JavaScript, See figure 11.

[6] https://nodered.org/



Name: timestamp

Function

```
//var now = new Date().toString();
msg.payload=JSON.stringify(msg.payload)+ "-" +new Date
//var msg2 = {payload:"timestampe:" +now.valueOf()};
//var msg3=msg2+msg;
return msg;
```

Outputs: 1

See the Info tab for help writing functions.

Node

Name	timestamp
Type	function
ID	895741e6.39515

Properties

A function block where you can write code to do more interesting things.

The message is passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

Logging and Error Handling

To log any information, or report an error, the following functions are available:

- `node.log("Log")`
- `node.warn("Warning")`
- `node.error("Error")`

The Catch node can also be used to handle errors. To invoke a Catch node, pass `msg` as a second argument to `node.error`:

Figure 11: Timestamp node

3.2.3 MQTT node

The MQTT out node is provided by Node RED. It's a node to send data with the MQTT protocol.

- Server
  Here you enter the IP or URL of your desired MQTT-Broker. For both port 1883 is used. Here is a small list of online-broker we used:

  10.24.3.66:1883 (THM local broker)

  iot.eclipse.org

  broker.dashboard.com

  test.mosquitto.org

  broker.hivemq.com

  test.mosca.io

- Topic
  Here you can enter a topic under which you can subscribe again to get data from. We choose for simplification /iot-lab/wheel/sensehat

- QoS
  You have the possibility to choose between three kinds of Qualitiy of Service:

  QoS 0 - at most once

  Level zero guarantees a best effort of delivery. A message won't be

  acknowledged or stored by an Broker. Often called "fire and forget"

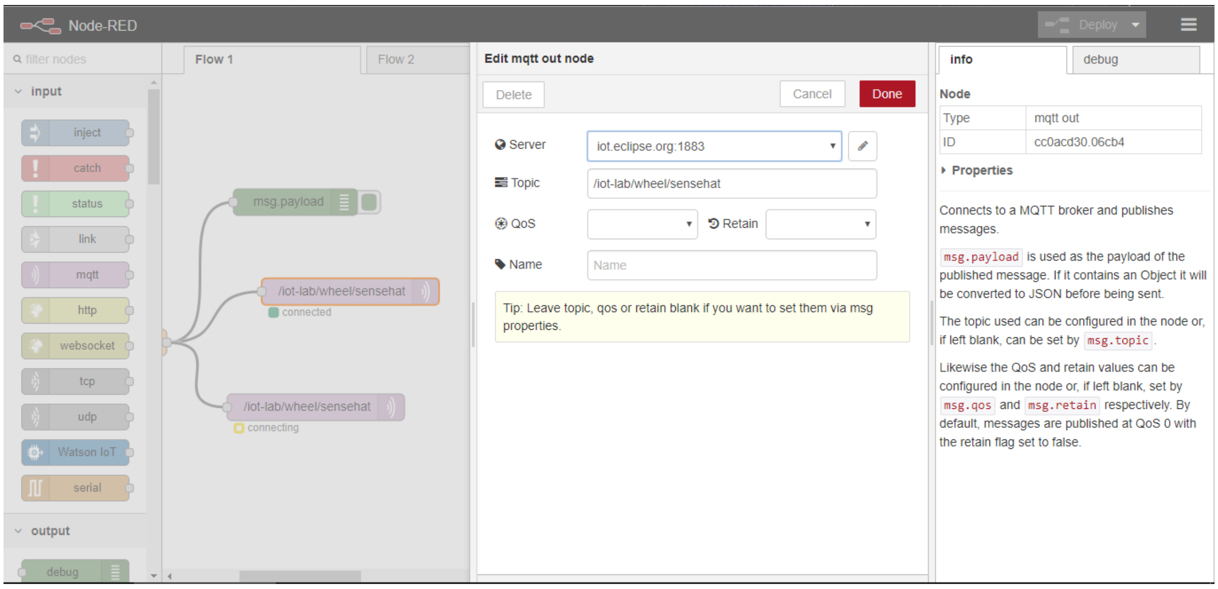
  QoS 1 - at least once

Level one guarantees that the message will be delivered at least once to the Broker. The sender will wait for an acknowledgement from the receiver.

QoS 2 - once

The highest QoS is 2, it guarantees that each message is received only once by the counterpart. It is the safest and also the slowest quality of service level.

- Retain
  A retained message is a normal MQTT message with the retained flag set to true. The broker will store the last retained message and the corresponding QoS for that topic. Each client that subscribes to a topic pattern, which matches the topic of the retained message, will receive the message immediately after subscribing.



Server: iot.eclipse.org:1883
Topic: /iot-lab/wheel/sensehat
QoS:
Retain:
Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Node
Type: mqtt out
ID: cc0acd30.06cb4

Properties

Connects to a MQTT broker and publishes messages.

msg.payload is used as the payload of the published message. If it contains an Object it will be converted to JSON before being sent.

The topic used can be configured in the node or, if left blank, can be set by msg.topic.

Likewise the QoS and retain values can be configured in the node or, if left blank, set by msg.qos and msg.retain respectively. By default, messages are published at QoS 0 with the retain flag set to false.

Figure 12: MQTT node

3.3 Matlab

In the previous part we explained how we get the data from the Raspberry pi to the MQTT-Broker. In this part we will illuminate how we import the data into matlab and visualize it.

3.3.1 Important commands for Matlab

We will explain some important commands we used and found for Matlab which are not imperative coherent to our project and are not explained anywhere else in this document, but may be of use for future projects.
javaclasspath('-dynamic')

Displays the current dynamic path

t = Matlab_MQTT_Client_Main_Class;

Creates a new java instance.

t.Subscribe(u);

This command is not valid in Matlab without creating a Java instance "t". This is how to call a Java method in Matlab, using a Java syntax. "t" could be a Java object or the name of the class. "Subscribe(u)" is our method in Java to subscribe to a topic.

u = '/iot-lab/wheel/sensehat','2';

Creates a cell array with two columns, one for the topic and one for the QoS.

r = reshape(n,1,[]);[7]

From its name, this command will reshape the dimensions of your array. If you have for instance an array with 10 rows and 1 column (10x1 = 10 elements), you could reshape it to 5 rows and 2 columns (5x2 = 10 elements). In our example the array n carries the message payload and it has always 1 column and number of its rows differ in each iterate, so we would like change it to 1 row and we write these square brackets to let Matlab choose the number of its columns

a = na-
tive2unicode(r);[8] / r = unicode2native (a);[9]

These two command convert your array from ASCII code to Unicode character and vice versa.

w = strsplit (a, '/') ;[10]

Matlab will split the string (a) into substrings when it finds the delimiter (/).

[theta2 , rho2 , z2] = c a r t 2 p o l (x2 , y2 , z2) ;[11]

Converting from the Cartesian coordination to polar.

[7] https://de.mathworks.com/help/matlab/ref/reshape.html
[8] https://de.mathworks.com/help/matlab/ref/native2unicode.html
[9] https://de.mathworks.com/help/matlab/ref/unicode2native.html
[10] https://de.mathworks.com/help/matlab/ref/strsplit.html
[11] https://de.mathworks.com/help/matlab/ref/cart2pol.html

n= t.get_message_(u);

This command are not valid in Matlab without creating your Java instance "t". This is how to call a Java method in Matlab using a Java syntax."t" could be a Java object or the name of the class. "get_message_(u)" is our method in Java to fetch the messages coming under a topic defined in the argument "u". If you are not subscribing to any topic, the method will return an empty array. This method will return the last received message, in format of a byte array (int8 column vector). In our example, we are send a string of characters with the Sense-HAT node which we receive as a byte array. Each byte represents one character in ASCII format: If the message at node-red was like that:
"acceleration":"x":-0.0015,"y":0.0269,"z":0.9831,"gyroscope":"x":-0.0024,"y":-0.0016,"z":0.0002,"orientation":"roll":1.6198,"pitch":0.0462,"yaw":245.0466, "compass":245
You will get a an ASCII format in Matlab as a vertical array (216 X 1). Like the following: [123, 34, 97, 99, 99, 101, 108, 101, 114, 97, 116, 105, 111, 110, 34, 58, 123, 34, etc In the ASCII table[a] each number of the previous ones meets a character. Numbers from 0 - 9 , the dot , and the minus symbol could be what you are looking for. Here are the values of the characters:

- ascii code 45 - (Hyphen)
- ascii code 46 . (Full stop , dot)
- ascii code 48 0 (number zero)
- ascii code 49 1 (number one)
- ascii code 50 2 (number two)
- ascii code 51 3 (number three)
- ascii code 52 4 (number four)
- ascii code 53 5 (number five)
- ascii code 54 6 (number six)
- ascii code 55 7 (number seven)
- ascii code 56 8 (number eight)
- ascii code 57 9 (number nine)

[a] www.theasciicode.com.ar

3.3.2 Startup

This is the Startup Script. It starts the GUI[12], like you see in figure 13 to connect to the MQTT-Broker. You will find the source code for the GUI under point 5.5. The command javaaddpath must point to the java file. Insert the Server address and choose the configuration then press connect to connect to the broker. The Matlab-code for Startup:

clear[13]

Clears all the variables stored in the workspace.

clear java;

Removes any dynamic entry to the dynamic path.

javaaddpath*'C:/IoT/wheel/code_V1.4/java/dist/testmatlab5_1_1_2.jar'*;[14]

Adds a java .class / .jar file to the environment.

wheelGUI = Matlab_MQTT_Client_Main_Class;

Calls the graphical user interface

javaMethod('main',wheelGUI,'');[15]

After creating your java instance, you have to call the main Method to run your java application. Actually, this command works for any method you want to call. It is matlab syntax.

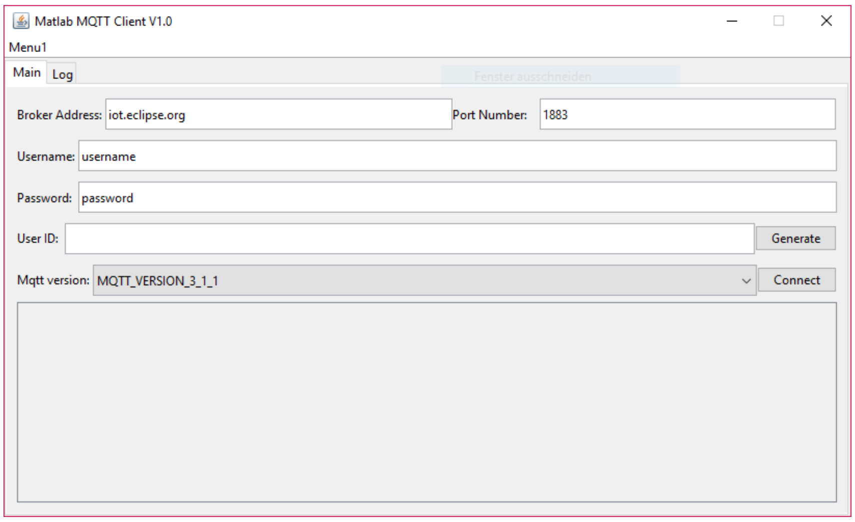

Matlab MQTT Client V1.0
Menu1
Main Log
Broker Address: iot.eclipse.org Port Number: 1883
Username: username
Password: password
User ID:
Mqtt version: MQTT_VERSION_3_1_1

Figure 13: GUI

Note: Don't close the GUI, it will close Matlab aswell.

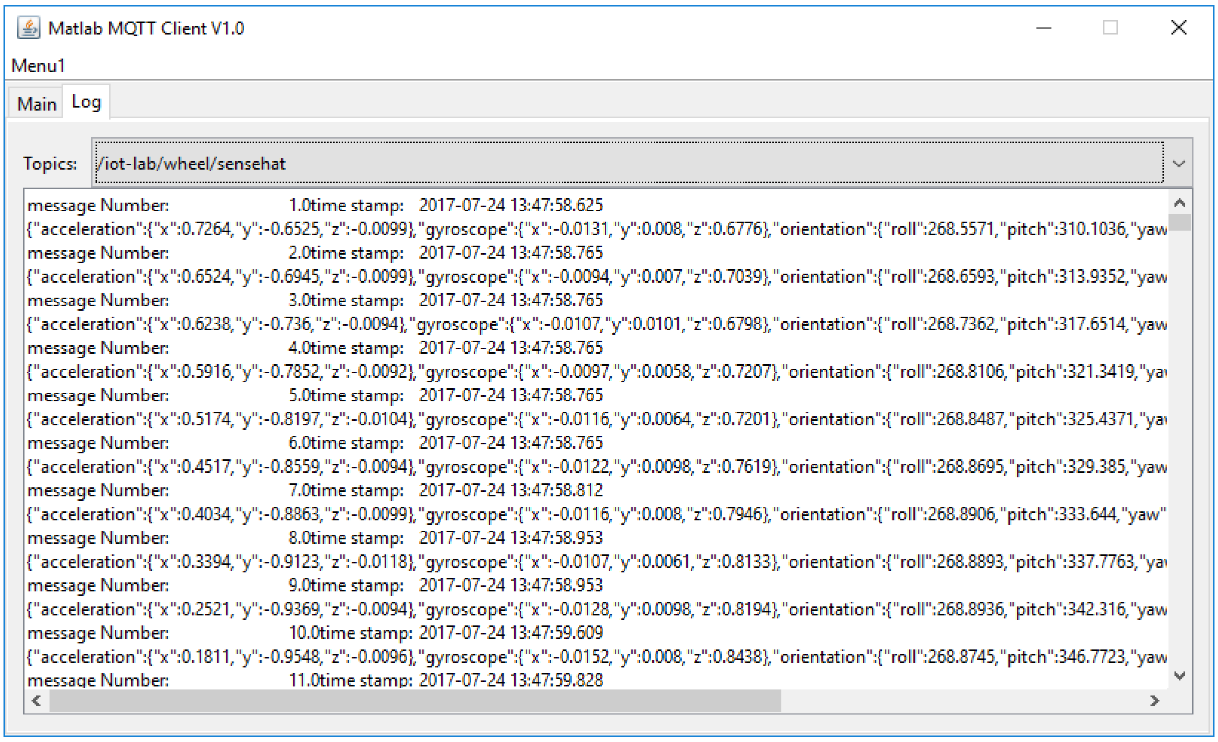
3.3.3 Subscribe

This is the Matlab code for subscribing to one or more brokers. You can change the number of strings you would connect to. You are free to add or delete connections if you need different ones. You can see now the subscribed Topics in the GUI by selecting the Log tab.

[12]graphical user interface
[13]https://de.mathworks.com/help/matlab/ref/clear.html
[14]https://de.mathworks.com/help/matlab/ref/javaaddpath.html
[15]https://de.mathworks.com/help/matlab/ref/javamethod.html

You can select the Topics and see what data you get from the sensors.(See figure 14)
The wheel_subscribe.m-code:

```
subscribes=["/iot-lab/wheel2/sensehat2","2";
                    "iot_lab/wheel/sensortag","2"];
wheelGUI.Subscribe(subscribes(1,:));
wheelGUI.Subscribe(subscribes(2,:));
numberOfSubscribes=numel(subscribes)/2;
```



```
message Number:            1.0time stamp:   2017-07-24 13:47:58.625
{"acceleration":{"x":0.7264,"y":-0.6525,"z":-0.0099},"gyroscope":{"x":-0.0131,"y":0.008,"z":0.6776},"orientation":{"roll":268.5571,"pitch":310.1036,"yaw
message Number:            2.0time stamp:   2017-07-24 13:47:58.765
{"acceleration":{"x":0.6524,"y":-0.6945,"z":-0.0099},"gyroscope":{"x":-0.0094,"y":0.007,"z":0.7039},"orientation":{"roll":268.6593,"pitch":313.9352,"yaw
message Number:            3.0time stamp:   2017-07-24 13:47:58.765
{"acceleration":{"x":0.6238,"y":-0.736,"z":-0.0094},"gyroscope":{"x":-0.0107,"y":0.0101,"z":0.6798},"orientation":{"roll":268.7362,"pitch":317.6514,"yaw
message Number:            4.0time stamp:   2017-07-24 13:47:58.765
{"acceleration":{"x":0.5916,"y":-0.7852,"z":-0.0092},"gyroscope":{"x":-0.0097,"y":0.0058,"z":0.7207},"orientation":{"roll":268.8106,"pitch":321.3419,"ya
message Number:            5.0time stamp:   2017-07-24 13:47:58.765
{"acceleration":{"x":0.5174,"y":-0.8197,"z":-0.0104},"gyroscope":{"x":-0.0116,"y":0.0064,"z":0.7201},"orientation":{"roll":268.8487,"pitch":325.4371,"ya
message Number:            6.0time stamp:   2017-07-24 13:47:58.765
{"acceleration":{"x":0.4517,"y":-0.8559,"z":-0.0094},"gyroscope":{"x":-0.0122,"y":0.0098,"z":0.7619},"orientation":{"roll":268.8695,"pitch":329.385,"yaw
message Number:            7.0time stamp:   2017-07-24 13:47:58.812
{"acceleration":{"x":0.4034,"y":-0.8863,"z":-0.0099},"gyroscope":{"x":-0.0116,"y":0.008,"z":0.7946},"orientation":{"roll":268.8906,"pitch":333.644,"yaw"
message Number:            8.0time stamp:   2017-07-24 13:47:58.953
{"acceleration":{"x":0.3394,"y":-0.9123,"z":-0.0118},"gyroscope":{"x":-0.0107,"y":0.0061,"z":0.8133},"orientation":{"roll":268.8893,"pitch":337.7763,"ya
message Number:            9.0time stamp:   2017-07-24 13:47:58.953
{"acceleration":{"x":0.2521,"y":-0.9369,"z":-0.0094},"gyroscope":{"x":-0.0128,"y":0.0098,"z":0.8194},"orientation":{"roll":268.8936,"pitch":342.316,"yaw
message Number:            10.0time stamp: 2017-07-24 13:47:59.609
{"acceleration":{"x":0.1811,"y":-0.9548,"z":-0.0096},"gyroscope":{"x":-0.0152,"y":0.008,"z":0.8438},"orientation":{"roll":268.8745,"pitch":346.7723,"yaw
message Number:            11.0time stamp: 2017-07-24 13:47:59.828
```

Figure 14: GUI-Log

3.3.4 Plot

The Plot-Code is the main-code in our data processing. Therefore we are going to explain the code more exact. In the following you will see the code itself, with explanation in the boxes.

MAIN-CODE:

```
xvec=[];
yvec=[];
zvec=[];
figure('Name','Sensor_Data','NumberTitle','off');
line_wid        = 2;
FontSize        = 15;
marker_size     = 10;
buffer_length   = 100;
```

The following examples demonstrate all the possible Methods and Fields in java code that you could use in matlab: t.get_message_(...)

- receiving a MQTT message. The argument is your MQTT topic.

```
while true
 coordmat           = zeros(buffer_length,3);
 coordmat2          = zeros(buffer_length,3);
 for buff_index     = 1:buffer_length
         coord_vec      = [];
         coord_vec2     = [];
         pause (0.05);

         if wheelGUI.get_message_(subscribes(2,:))~=0
          m = wheelGUI.get_message_(subscribes(2,:));

          e = reshape(m,1,[]);
          a = native2unicode(e);
          c = strsplit(a,'␣');
          x=strrep(c(2),';','');
          x=str2double(x)/5-7.5;
          x=x*-1;
          y=strrep(c(4),';','');
          y=str2double(y)/5-5;
          z=strrep(c(6),';','');
          z=str2double(z)/5-7.5;
```

We multiply the value we get by the sensor so we can visualize it in a proper way.

```
         else
          x=0;
          y=0;
          z=0;
```

If no sensor data is available, the values are set to zero to prevent an error.

```
         end
          x = round(x,1, 'decimals');
          y = round(y,1, 'decimals');
          z = round(z,1, 'decimals');
```

The value is rounded to two decimal figures and converted into Cartesian and polar coordinates.

```
          coord_vec          = [x,y,z];
          [theta,rho,z]      = cart2pol(x,y,z);
```

coordmat is a matrix representing the buffer where the data is stored.

```
          coordmat(buff_index,1:3) = coord_vec;
```

Sensortag-Plot

```
subplot(2,2,1)

 plot3(coordmat(:,1),coordmat(:,2),coordmat(:,3),'r','
    linewidth',line_wid);
 grid on;
 xlim([-15 15]);ylim([-15 15]);zlim([-15 15]);
 Xlab= xlabel('x'); Ylab=ylabel('y');Zlab=zlabel('z');
 set(Xlab, 'FontSize', FontSize);
 set(Ylab, 'FontSize', FontSize);
 set(Zlab, 'FontSize', FontSize);
 title("sensortag")
```

Above we see the commands for the 3D-Plot with X,Y and Z direction

```
subplot(2,2,3)

 plot(coordmat(:,1),coordmat(:,2),'b','markersize',
    marker_size,'marker','o','linewidth',line_wid);
 grid on;
 xlim([-15 15]);ylim([-15 15]);
 Xlab= xlabel('x'); Ylab=ylabel('y');
 set(Xlab, 'FontSize', FontSize);
 set(Ylab, 'FontSize', FontSize);
```

Above we see commands the for the 2D-Plot with X and Y direction, the following part below is nearly the same as before so we will not explain again.

```
if wheelGUI.get_message_(subscribes(1,:))~=0
 n= wheelGUI.get_message_(subscribes(1,:));

 e2 = reshape(n,1,[]);
 v2 = native2unicode(e2);
 r2=strrep(v2,':','');
 c2 = strsplit(r2,'"');
 if strcmp(c2(2),'acceleration')
 x2=str2double(c2(5))*10;
 y2=str2double(c2(7))*10;
 z2=c2(9);
 z2 = strrep(z2,'},','');
 z2=str2double(z2)*10;
else
 x=0;
 y=0;
 z=0;
end
```

```
        end
         x2 = round(x2,1, 'decimals');
         y2 = round(y2,1, 'decimals');
         z2 = round(z2,1, 'decimals');
        coord_vec2       = [x2,y2,z2];
        [theta2,rho2,z2]  = cart2pol(x2,y2,z2);
        coordmat2(buff_index,1:3) = coord_vec2;
```

Sense-HAT-Plot

```
        subplot(2,2,2)

         plot3(coordmat2(:,1),coordmat2(:,2),coordmat2(:,3),'r','
            linewidth',line_wid);
         grid on;
         xlim([-15 15]);ylim([-15 15]);zlim([-15 15]);
         Xlab= xlabel('x'); Ylab=ylabel('y');Zlab=zlabel('z');
         set(Xlab, 'FontSize', FontSize);
         set(Ylab, 'FontSize', FontSize);
         set(Zlab, 'FontSize', FontSize);
         title("sensehat")

        subplot(2,2,4)

         plot(coordmat2(:,1),coordmat2(:,2),'b','markersize',
            marker_size,'marker','o','linewidth',line_wid);
         grid on;
         xlim([-15 15]);ylim([-15 15]);
         Xlab= xlabel('x'); Ylab=ylabel('y');
         set(Xlab, 'FontSize', FontSize);
         set(Ylab, 'FontSize', FontSize);
        end
end
```

So if you successfully started and subscribed, the result should look like shown in figure 15. In the top-area we have the the 3D-plots of the two sensors we are using. Right beneath we have the plots in X and Y direction, consequently the 2D-plot. This matlab script is just working for these exactly sensors. If you like to change the sensors, or the values you want to process, this part has to be changed!

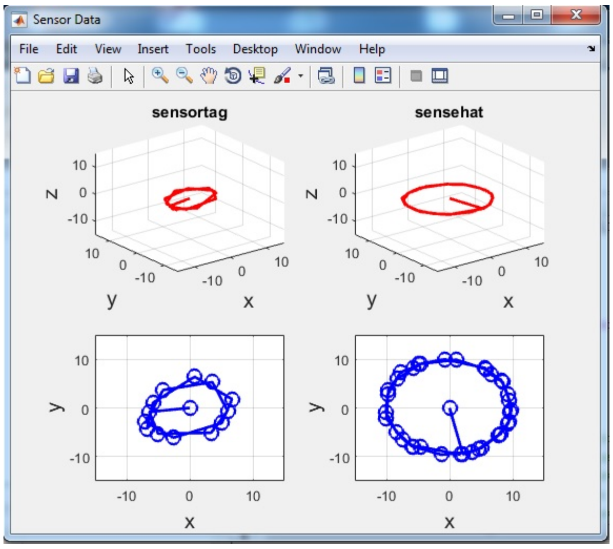


Figure 15: Plot

4 Summary and Conclusion

4.1 Summary

This report presents a possibility to transport data with the MQTT-Protocol from different sensors, in our case the Sense-HAT and the TI SensorTag, to a MQTT-Broker and import these data into Matlab by subscribing to the Broker. First of all we look for a solution to the implementation of the MQTT-Client into Matlab. For this purpose we choose the eclipse paho open-source client implementatin. It supports MQTT and MQTT-SN messaging protocols aimed for the Internet of Things. The libraries for the implementation are available for different programming languages like C, C++, Java, JavaScript and several more. In regards to our knowledge we choose Java as a fitting language - even tough it took us some time to get Java running in Matlab. The major issues and how to treat them are explained in the attachment (see: 5.3 and 5.4). When the transmission of the data with the MQTT-Protocol and our Client in Matlab succeeds, we proceed with the recondition of the data we get from the sensors. We reduce the data-string we get from the sensor to the necessary. In result we receive cartesian values. Additionally to the cartesian values we commute them into the polar form. With both of these values we create a visualization by using the cartesian values for the 2D-plot in x and y direction and the polar values for the 3D-plot in x, y and z direction. For better operability we add also a GUI. This GUI allows to easily connect to different brokers. Also it contains a log where you can see the incomming data of the topic you subscribed to.

4.2 Conclusion

In regards to the work order we got in the start of this project, some changes had happend during the process. First of all the data processing was reduced from 3 sensors to 2 sensors. The third sensor was based on a XBEE connection. There were a other group of students

working on, and we didn't get the sensor in time, to implant it into our solution. The MQTT-Protocol, SenseHAT and SensorTAG have proven to be reliable.
Regarding the sensors there is still an issue with the TI SensorTAG and our data processing. First, the sampling rate of the SensorTAG is way to low, it is about 1 value/second. With this speed the visualization is rather rectangular than a circle. But there is a solution to flash the sensor and get a higher sampling rate. In our case it didn't work.
The second issue is the string we get from the SensorTAG. The values are mixed. So we get messages from acceleration, gyroscope and magnetometer mixed in such a way that we can not discern the value for acceleration correctly.
One of the most important points that we have to take into account in the future is the jitter or the variance of our MQTT packets. It could be in some cases that we could receive (at a certain point of time) a very late MQTT message. This message is not helpful for us to visualize the rotating wheel. A good solution could be to introduce a jitter buffer.

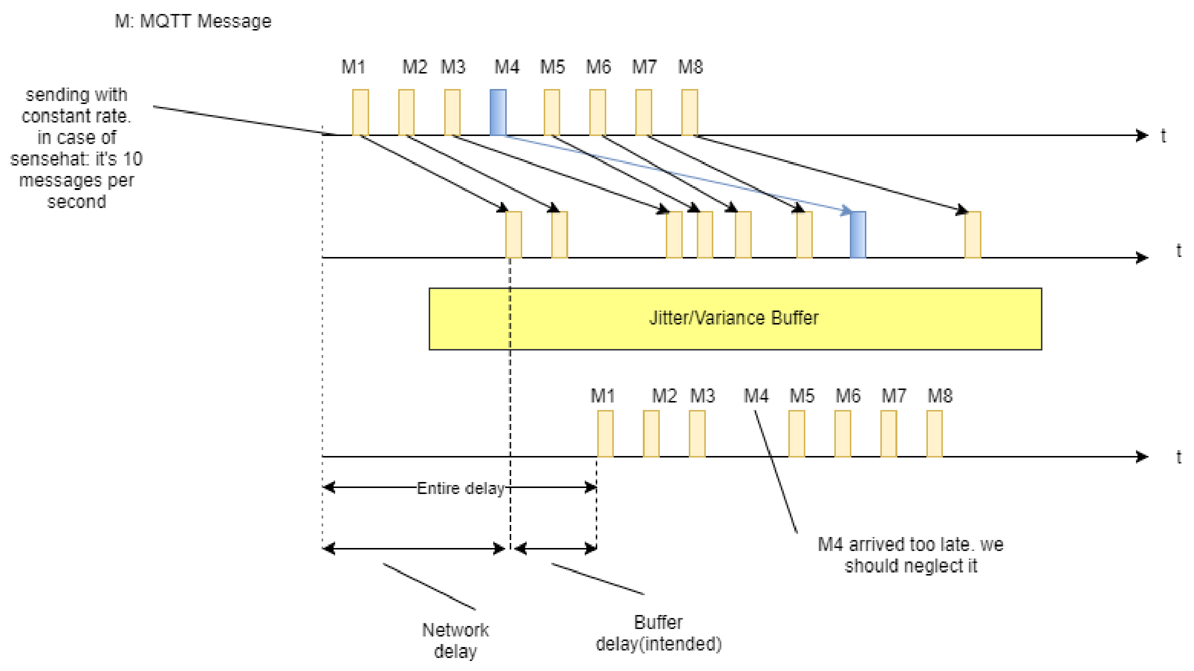


Figure 16: Jitter buffer

Another issue, in our code is that for each message we receive we have a time stamp (time stamp at the destination) and we add also a message number. But the trick is that we have noticed the time stamp in the message payload (the source time stamp) are in some cases earlier than the time stamp that we have created at the receiving point.
A good solution would be to sync your clock at the sender and the receiver when you start your connection. We have to look at a solution that could operate at milisecond accuracy. ITP (internet time protocol) could be a solution but we are not sure.
A furthermore problem is how we could improve the visualization of the wheel. It could be better in future work to use Simulink[16] and especial the library (3D animation). We are not

[16]https://de.mathworks.com/videos/series/getting-started-with-simulink-3d-animation-94454.html

sure if it's possible to do it with the current code. Any way, you could manipulate the java code.
The last point we have to announce is Octave[17]. This open source program is a possible alternative to Matlab that has similar capabilities and a similar programming language syntax. Because of this, Matlab code often can be executed using Octave. We could not guarantee that our program could work in octave and we didn't tested it there.

[17] https://www.gnu.org/software/octave/

5 Attachment

5.1 Source Code Node-red

```
[
    {
        "id": "a4f27947.c4f0b8",
        "type": "sensorTag",
        "z": "c690242a.15d058",
        "name": "",
        "topic": "sensorTag",
        "uuid": "",
        "temperature": false,
        "humidity": false,
        "pressure": false,
        "magnetometer": true,
        "accelerometer": true,
        "gyroscope": true,
        "keys": false,
        "luxometer": false,
        "x": 189,
        "y": 191,
        "wires": [
            [
                "8c12001a.4cddf"
            ]
        ]
    },
    {
        "id": "3a465523.35244a",
        "type": "debug",
        "z": "c690242a.15d058",
        "name": "",
        "active": true,
        "console": "false",
        "complete": "true",
        "x": 559,
        "y": 163,
        "wires": []
    },
    {
        "id": "8c12001a.4cddf",
        "type": "function",
        "z": "c690242a.15d058",
        "name": "Timestamping",
        "func": "function␣addZero(x,␣n)␣{\n␣␣␣␣while␣(x.toString().
            length␣<␣n)␣{\n␣␣␣␣␣␣␣␣x␣=␣\"0\"␣+␣x;\n␣␣␣␣}\n␣␣␣␣return␣x
            ;\n}\n\nvar␣d␣=␣new␣Date();\nvar␣h␣=␣addZero(d.getHours(),
            ␣2);\nvar␣m␣=␣addZero(d.getMinutes(),␣2);\nvar␣s␣=␣addZero
            (d.getSeconds(),␣2);\nvar␣ms␣=␣addZero(d.getMilliseconds()
            ,␣3);\nmsg.payload␣=␣␣␣\"x:␣\"+␣msg.payload.x\n␣␣␣␣␣␣␣␣␣␣␣␣
            ␣+␣\";␣y:␣\"+␣msg.payload.y\n␣␣␣␣␣␣␣␣␣␣␣␣␣+␣\";␣z:␣\"+␣msg.
            payload.z\n␣␣␣␣␣␣␣␣␣␣␣␣␣+␣\";␣Time:␣\"␣+␣h␣+␣\":\"\n␣␣␣␣␣␣␣␣
            ␣␣␣␣␣+␣m␣+␣\":\"␣+␣s␣+␣\":\"␣+␣ms;\n\nreturn␣msg;\n\n\n",
        "outputs": 1,
        "noerr": 0,
        "x": 385,
        "y": 191,
        "wires": [
            [
                "3a465523.35244a",
                "1e0b149e.a5495b",
                "b779960b.0d7d98",
                "37ed6c37.f42e54"
            ]
        ]
    },
    {
        "id": "1e0b149e.a5495b",
        "type": "mqtt␣out",
        "z": "c690242a.15d058",
        "name": "",
        "topic": "sensortag",
        "qos": "",
        "retain": "",
        "broker": "da700779.10d088",
        "x": 568,
        "y": 219,
        "wires": []
    },
    {
        "id": "b779960b.0d7d98",
        "type": "mqtt␣out",
        "z": "c690242a.15d058",
        "name": "",
        "topic": "iot_lab/wheel/sensortag",
        "qos": "",
        "retain": "",
        "broker": "da700779.10d088",
        "x": 631,
        "y": 288,
        "wires": []
    },
    {
        "id": "37ed6c37.f42e54",
        "type": "mqtt␣out",
        "z": "c690242a.15d058",
        "name": "",
        "topic": "iot_lab/wheel/sensortag",
        "qos": "",
        "retain": "",
        "broker": "da700779.10d088",
        "x": 644,
        "y": 380,
        "wires": []
    },
    {
        "id": "da700779.10d088",
        "type": "mqtt-broker",
        "z": "",
        "broker": "broker.hivemq.com",
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": ""
    }
]
```

5.2 How to write a simple MQTT Client in Java with NetBeans

At first we searched for a way how we can get our data over MQTT into Matlab. We have found Paho[18] as a suitable solution. The Eclipse Paho project provides open-source client implementations of MQTT and MQTT-SN messaging protocols aimed at new, existing, and emerging applications for the Internet of Things (IoT). So it was nearly perfect for our purpose. In the following we explain the steps it takes to run an MQTT Client an its main method in Matlab:

Download Paho eclipse MQTT Library (as a .jar file) from its eclipse paho repository:

- Create a new java application project in your IDE such as eclipse.
- Add the .jar library to your project

```
our_project>properties>libraries>add a .jar file
```

Create a new class with name: 'SimpleMqttClient' inside the main method and create a new mqtt client:

```
MqttClient client = new MqttClient( "tcp://broker.mqttdashboard.com
    :1883", MqttClient.generateClientId() , new MemoryPersistence() );
```

Here you pass an ip or url for the broker, in this case: broker.mqttdashboard.com

```
Set the call back method:
client.setCallback(new MqttCallback() {
@Override
public void connectionLost(Throwable cause) {}
```

Called when the client lost the connection to the broker

```
@Override
public void messageArrived(String topic, MqttMessage message) throws
    Exception {mess = new String (message.getPayload());}
@Override
public void deliveryComplete(IMqttDeliveryToken token) {}
});
```

Called when a outgoing publish is complete

Also you have to declare a static string variable (mess) Create a get-method called: getmessage() to call it in Matlab and get the messages. Connect to the broker:

```
client.connect();
```

Subscribe for a topic and pass the QoS(Quality of Service):

```
client.subscribe("Student_first_topic", 1);
```

[18] http://www.eclipse.org/paho/

Do not forget to import any required class. Compile the project and find the .jar file in your project directory. Add the file to dynamic path of matlab as we have described in the previous steps.

```
javaaddpath ('C:\Users\Student\Desktop\testmatlab1\testmatlab1.jarâĂŹ)
```

Now create a new instance:

```
Client= SimpleMqttClient
```

Call the main method:

```
javaMethod('main', Client,' ')
```

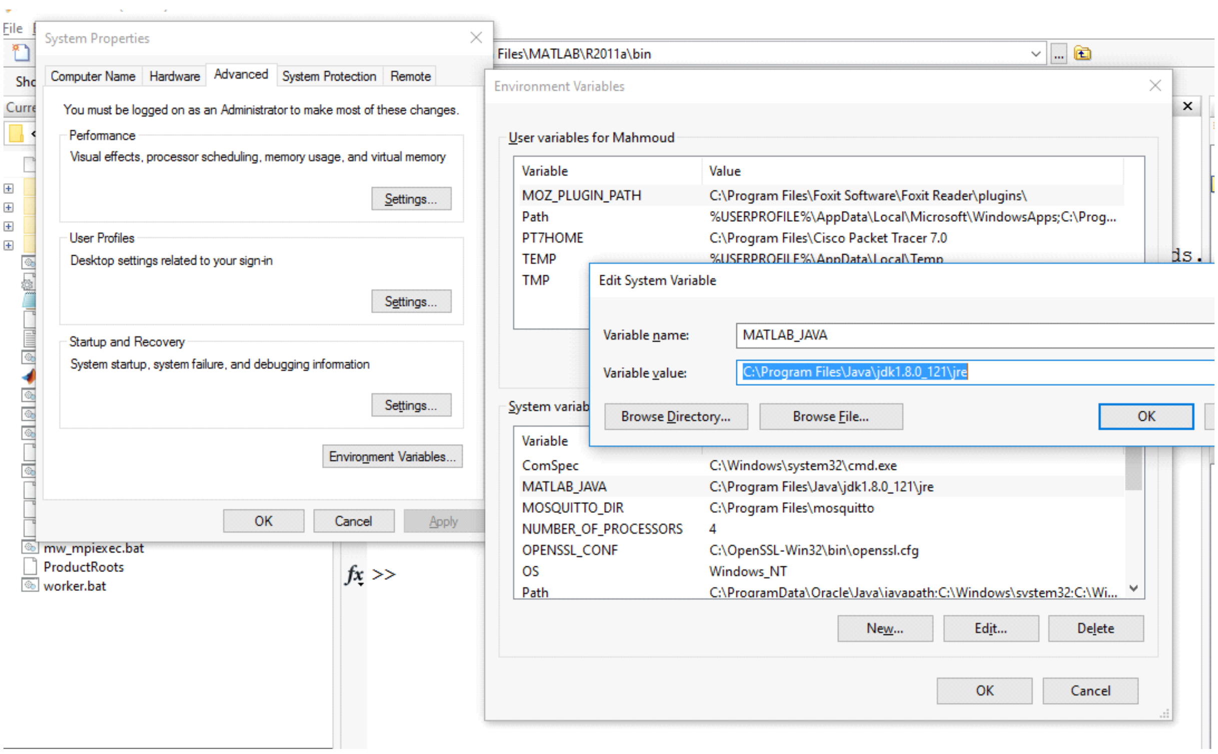
The last command in Matlab is to invoke any method in Matlab. You pass the method-name as Chat vector in the first argument the object you have created as second argument (or the class name if the method was static) and the method arguments in the next fields. Notice that if your method donâĂŹt has arguments you have to pass an empty array (zero dimension).

5.3 How to setup Java in Matlab

For our solution we choose Java to run the MQTT-Protocol with Matlab. To be able to run Java code in Matlab, you have to check Java-Version of your Computer and Java-Version of Matlab. Here is how it's done:

- Start Matlab to determine if and which Java-Version is running
- Write following command into the Matlab console: version -java
- To check Java-Verison on your machine start command interface(cmd)and enter: java -version
- If this command is not recognized this means that you have to install (Java development kit) JDK onto your machine.(We worked with Version 1.8.0-31)[19]
- If the output of previous commands gives you the same Java-Version, then skip this part and go straight to 5.4. Otherwise take the next steps.
- If not, or Matlab is giving you an error, you have to change the JVM inside Matlab. To do that, go through the following steps:
  - Write into the console of Matlab the following command:
    getenv MATLAB_JAVA
  - If the value is empty(âĂŸ âĂŸ) then you have to create a new MATLAB_JAVA environment variable in your system properties. Therefore go to:
    - System Properties
    - Press Environment Variables...
    - Press New...
    - add Variable name, and Variable value as shown in figurea 17
    - The system variable should be your JRE path.
    - add with ok and restart your PC

[19] http://www.oracle.com/technetwork/java/javase/downloads/index.html



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

Files\MATLAB\R2011a\bin

mw_mpiexec.bat
ProductRoots
worker.bat

Environment Variables

User variables for Mahmoud

Variable	Value
MOZ_PLUGIN_PATH	C:\Program Files\Foxit Software\Foxit Reader\plugins\
Path	%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;C:\Prog...
PT7HOME	C:\Program Files\Cisco Packet Tracer 7.0
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	

System variab

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
MATLAB_JAVA	C:\Program Files\Java\jdk1.8.0_121\jre
MOSQUITTO_DIR	C:\Program Files\mosquitto
NUMBER_OF_PROCESSORS	4
OPENSSL_CONF	C:\OpenSSL-Win32\bin\openssl.cfg
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Wi...

Edit System Variable

Variable name: MATLAB_JAVA
Variable value: C:\Program Files\Java\jdk1.8.0_121\jre

Figure 17: Set Java path

- Check the value of MATLAB_JAVA by writing the previous command in Matlab. Now check the Java-Version of your machine. It should be identical.

5.4 How to add a Java class to Matlab

Imagine you would like to use a Java file called test.java. First of all you have to write and compile this code by using either an IDE (eclipse or netbeans) or using the command line entering:

```
javac test.java
```

After compiling you get the file test.class. We are interested in this .class file. Now you have to add this .class file to your Matlab class path. There are two ways:

- Add this file to the static class path
  - Write into the console of Matlab <which classpath.txt>
  - Now you get the directory where you could append your file to the static class file
  - Just open the file as administrator and write at the end of the text the directory of your test.class file, e.g.:

    ```
    C:\Users\Student\Desktop\testmatlab1
    ```

  - Now you have to save the file and relaunch Matlab again.
  - Declare in Matlab an instance of your class:

```
t = test
```

- if you get something like that:

```
test@18c1ddd
```

- then you have successfully added a static class path

- Adding your .class file to the dynamic path
  - In this way you do not need system administrative privileges and you do not have to relaunch Matlab. It's also the solution we use in our program.
  - Just write the following command:

    ```
    Javaaddpath ("write_here_your.class path")
    ```

  - Example:

    ```
    javaaddpath ('C:\Users\Student\Desktop\testmatlab1\testmatlab1.class')
    ```

    You also can add .jar files like this:

    ```
    javaaddpath ('C:\Users\Student\Desktop\testmatlab1\testmatlab1.jar')
    ```

5.5 Source Code Java - Main class

```
import java.awt.Color;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Arrays;
import javax.swing.*;

/*
```

```
 * To change this license header, choose License Headers in Project
    Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author Mahmoud Mansour
 */
public class Matlab_MQTT_Client_Main_Class {

    public static MqttClient client;
    static String client_id;
    static int Mqtt_version = 4;
    static JFrame frame;
    static JMenuBar MenuBar;
    static JMenu MenuFile;
    static JMenuItem Item1, Item2;
    static JTabbedPane mainpanel;
    static JPanel tab1, tab2;
    static JPanel panel_up;
    static JPanel panel1;
    static JLabel label1_1;
    static JTextField textfield1_1;
    static JLabel label1_2;
    static JTextField textfield1_2;
    static JPanel panel2;
    static JLabel label2;
    static JTextField textfield2;
    static JPanel panel3;
    static JLabel label3;
    static JTextField textfield3;
    static JPanel panel4;
    static JLabel label4;
    static JTextField textfield4;
    static JButton button4;
    static JPanel panel5;
    static JButton button5;
    static JComboBox list5;
    static JLabel label5;
    static JPanel panel6;
    static JLabel label6;
    static JComboBox list6;
    static JTextArea textarea;
static JScrollPane scroller2;
static JPanel panel7;
static JScrollPane scroller;
static JTextArea textarea_down;
static ArrayList<topic> list = new ArrayList<topic>();
static String[] QOS = {Integer.toString(topic.QOS_ZERO), Integer.
   toString(topic.QOS_ONE), Integer.toString(topic.QOS_TWO)};

public static void main(String args[]) {
     // trying to get the System look and Feel. depends on the
        operating system.
     try {
         UIManager.setLookAndFeel(UIManager.
             getSystemLookAndFeelClassName());
     } catch (ClassNotFoundException | IllegalAccessException |
        InstantiationException | UnsupportedLookAndFeelException e
        ) {
     }
     //**************************

     // create a frame with size 830 * 500 pixels.
     frame = new JFrame();
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setSize(830, 500);
     frame.setVisible(true);
     frame.setResizable(false);
     frame.setLocationRelativeTo(null);
     frame.setTitle("Matlab MQTT Client V1.0");

     // create a menu Ban and Menus.
     MenuBar = new JMenuBar();
     MenuFile = new JMenu("Menu1");
     Item1 = new JMenuItem("Item1");
     Item2 = new JMenuItem("Item2");
     //Item2.addActionListener(new AboutListener());
     MenuFile.add(Item1);
     MenuFile.add(Item2);
     MenuBar.add(MenuFile);
     frame.setJMenuBar(MenuBar);

     // adding all the components to the main panel
     mainpanel = new JTabbedPane();

     // tab1:
```

```
tab1 = new JPanel();
tab1.setName("Main");
tab1.setLayout(new GridLayout(2, 1, 4, 4));
tab1.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
    10));

panel_up = new JPanel();
panel_up.setLayout(new GridLayout(5, 2, 10, 10));

panel1 = new JPanel();
panel1.setLayout(new BoxLayout(panel1, BoxLayout.X_AXIS));
label1_1 = new JLabel("Broker␣Address:␣");
textfield1_1 = new JTextField("iot.eclipse.org");
label1_2 = new JLabel("Port␣Number:␣␣␣␣");
textfield1_2 = new JTextField("1883");
panel1.add(label1_1);
panel1.add(textfield1_1);
panel1.add(label1_2);
panel1.add(textfield1_2);

panel2 = new JPanel();
panel2.setLayout(new BoxLayout(panel2, BoxLayout.X_AXIS));
label2 = new JLabel("Username:␣");
textfield2 = new JTextField("username");
panel2.add(label2);
panel2.add(textfield2);

panel3 = new JPanel();
panel3.setLayout(new BoxLayout(panel3, BoxLayout.X_AXIS));
label3 = new JLabel("Password:␣␣");
textfield3 = new JTextField("password");
panel3.add(label3);
panel3.add(textfield3);

panel4 = new JPanel();
panel4.setLayout(new BoxLayout(panel4, BoxLayout.X_AXIS));
label4 = new JLabel("User␣ID:␣␣");
textfield4 = new JTextField(5);
button4 = new JButton("Generate");
button4.addActionListener(new generate_id_button_listener());
panel4.add(label4);
panel4.add(textfield4);
panel4.add(button4);
```

```
panel5 = new JPanel();
panel5.setLayout(new BoxLayout(panel5, BoxLayout.X_AXIS));
button5 = new JButton("Connect");
button5.addActionListener(new connection_button_listerner());

list5 = new JComboBox();
list5.addItem("MQTT_VERSION_3_1_1");
list5.addItem("MQTT_VERSION_3_1");
list5.addItem("MQTT_VERSION_DEFAULT");
list5.addItemListener(new version_listener());
label5 = new JLabel("Mqtt␣version:␣");
panel5.add(label5);
panel5.add(list5);
panel5.add(button5);

panel_up.add(panel1);
panel_up.add(panel2);
panel_up.add(panel3);
panel_up.add(panel4);
panel_up.add(panel5);

tab1.add(panel_up);
//////////////////////////////////////////////////
textarea_down = new JTextArea();
textarea_down.setEditable(false);
textarea_down.setBackground(new Color(240, 240, 240));
scroller = new JScrollPane(textarea_down);

tab1.add(scroller);

//tab2:
tab2 = new JPanel();
tab2.setName("Log");
tab2.setLayout(new BoxLayout(tab2, BoxLayout.Y_AXIS));
tab2.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
   10));

panel6 = new JPanel();
panel6.setLayout(new BoxLayout(panel6, BoxLayout.X_AXIS));
label6 = new JLabel("Topics:␣␣␣");
list6 = new JComboBox();
//list6.addItemListener(new topic_listener());
panel6.add(label6);
panel6.add(list6);
```

```
        panel7 = new JPanel();
        textarea = new JTextArea(20, 50);
        scroller2 = new JScrollPane(textarea);

        tab2.add(panel6);
        tab2.add(scroller2);

        mainpanel.add(tab1);
        mainpanel.add(tab2);
        frame.setContentPane(mainpanel);
        frame.revalidate();


    }

    //this inner class to react when the user press Connect/
        disconnect button.
    //create an MQTT client and have a connection with the specified
        broker by the user
    static class connection_button_listerner implements
        ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {

            // In case the user wants to Connect:
            if (button5.getText().equals("Connect")) {
                client_id = textfield4.getText();
                if (client_id == null) {
                    client_id = MqttClient.generateClientId();
                }
                try {
                    client = new MqttClient("tcp://" + textfield1_1.
                        getText() + ":" + textfield1_2.getText(),
                        client_id, new MemoryPersistence());
                } catch (MqttException ex) {
                    Logger.getLogger(Matlab_MQTT_Client_Main_Class.
                        class.getName()).log(Level.SEVERE, null, ex);
                }
                MqttConnectOptions options = new MqttConnectOptions()
                    ;
                options.setUserName(textfield2.getText());
                options.setPassword(textfield3.getText().toCharArray
                ());
            options.setMqttVersion(Mqtt_version);
            try {
                client.connect(options);

                if (client.isConnected()) {
                    System.out.println("connected");
                    textarea_down.setText("Client_ID:\t" +
                        client_id + "\thas␣successfuly␣connected␣
                        to:\n"
                                + "broker:\t" + "tcp://" +
                                    textfield1_1.getText() + ":" +
                                    textfield1_2.getText() + "\n"
                                + "with␣credentials:\tUsername:\t" +
                                    options.getUserName() + "\
                                    tPassword:\t" + Arrays.toString(
                                    options.getPassword()) + "\n"
                                + "Mqtt␣version:\t" + list5.
                                    getSelectedItem());
                }
                client.setCallback(new callback());
            } catch (MqttException ex) {
                Logger.getLogger(Matlab_MQTT_Client_Main_Class.
                    class.getName()).log(Level.SEVERE, null, ex);

            }

            if (client.isConnected()) {
                button5.setText("Disconnect");
            }

        } // In case the user wants to disconnect:
        else {
            try {
                client.disconnect();
                System.out.println("Disconnected");
                textarea_down.setText("Client_ID:\t" + client_id
                    + "\thas␣successfuly␣disconnected␣from:\n"
                            + "broker:\t" + "tcp://" + textfield1_1.
                                getText() + ":" + textfield1_2.getText
                                () + "\n");
            } catch (MqttException ex) {
                Logger.getLogger(Matlab_MQTT_Client_Main_Class.
                    class.getName()).log(Level.SEVERE, null, ex);
                }
                button5.setText("Connect");
            }
        }

    }
/////////// when a message has been arrived:
    static class callback implements MqttCallback {

        @Override
        public void connectionLost(Throwable thrwbl) {
            throw new UnsupportedOperationException("Not␣supported␣
               yet."); //To change body of generated methods, choose
               Tools | Templates.ss
        }

        @Override
        public void messageArrived(String string, MqttMessage mm)
           throws Exception {
            //System.out.println(Arrays.toString(mm.getPayload()));
            double message_number = 0;
            String message = new String();
            Timestamp time_stamp = new Timestamp(System.
               currentTimeMillis());
            for (int i = 0; i < list.size(); i++) {
                if (list.get(i).get_topic().equals(string)) {
                list.get(i).set_message(mm.getPayload());
                time_stamp.setTime(System.currentTimeMillis());
                list.get(i).set_message_time_stamp(System.
                   currentTimeMillis());
                message_number = (list.get(i).get_message_number()) +
                    1;
                list.get(i).set_message_number(message_number);
                if ((list.get(i).get_topic()).equals(list6.
                   getSelectedItem())) {
                    message = new String( mm.getPayload());
                textarea.append("message␣Number:\t" + message_number
                  + "time␣stamp:\t"
                        + list.get(i).get_message_time_stamp() + "\n"
                        + message + "\n");
                }


            }
        }
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken imdt) {
            throw new UnsupportedOperationException("Not␣supported␣
               yet."); //To change body of generated methods, choose
               Tools | Templates.
        }

    }
/////////////if the user pressed ("Generat-ID) button, to get an ID
   for his Client
    static class generate_id_button_listener implements
       ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            String a = MqttClient.generateClientId();
            textfield4.setText(a);
            client_id = a;
        }

    }

    //// when the user selects a verion for MQTT Connection

    static class version_listener implements ItemListener {

        @Override
        public void itemStateChanged(ItemEvent e) {
            if (e.getItem().equals("MQTT_VERSION_3_1_1")) {
                Mqtt_version = 4;
            } else if (e.getItem().equals("MQTT_VERSION_3_1")) {
                Mqtt_version = 3;
            } else {
                Mqtt_version = 0;
            }
        }

    }
///////when the user select different topic to display in the second
   tab
```

```
    static class topic_listener implements ItemListener {

        @Override
        public void itemStateChanged(ItemEvent e) {
            textarea.setText("");

        }

    }
//// this Method Used inside Matlab to subscribe for a topic:

    /// it works as follow:
    // In Matlab:
    //1-add the *.jar file to the javapath:
    //  ex:  javaaddpath('C:\Users\Mahmoud Mansour\Desktop\090720147\
       testmatlab5_1_1_2.jar');
    //2- Create an array for your topic: u = {'/iot-lab/wheel/
       sensehat','2'};
    //3- creat a 'Matlab_MQTT_CLient_Class':  t =
       Matlab_MQTT_Client_Main_Class;
    //4- call the main Method: javaMethod('main',t,'')
    //5- Now: a Client with GUI should appears.
    //5-1 enter all the required fields
    // ex: broker add: iot.eclipse.org , port number: 1883
    //      Usernaem: "username", Password: "password"
    //      User ID: insert one or press generate to generate a uniqe
       ID.
    //      select the MQTT version.
    //
    public boolean Subscribe(String[] args) {
        if (client.isConnected()) {
            int qos;
            switch (args[1]) {
                case ("" + 0):
                    qos = 0;
                    break;
                case ("" + 1):
                    qos = 1;
                    break;
                case ("" + 2):
                    qos = 2;
                    break;
                default:
                    qos = 2;
            }
            topic topic = new topic(args[0], qos, 0);

            try {
                client.subscribe(args[0], qos);
                list.add(topic);
                list6.addItem(topic.get_topic());
                for (topic i : list) {
                    System.out.println("you are subscribing to the
                       following topics:\n");
                    System.out.println(i);
                }
            } catch (MqttException ex) {
                Logger.getLogger(Matlab_MQTT_Client_Main_Class.class.
                   getName()).log(Level.SEVERE, null, ex);
            }

            return true;
        } else {
            return false;
        }

    }

    public boolean unsubscribe(String args[]) {
        if (client.isConnected()) {
            if (list.isEmpty()) {
                return false;
            } else {
                for (int i = 0; i < list.size(); i++) {
                    if (list.get(i).get_topic().equals(args[0]) &&
                       list.get(i).get_QOS() == Integer.parseInt(args
                       [1])) {
                        list6.removeItem(list.get(i));
                        list.remove(i);
                        try {
                            client.unsubscribe(args[0]);
                        } catch (MqttException ex) {
                            Logger.getLogger(
                               Matlab_MQTT_Client_Main_Class.class.
                               getName()).log(Level.SEVERE, null, ex)
                               ;
                        }
                    }
                }

            }
        } else {
            return false;
        }
        return true;
    }

    public byte[] get_message_(String args[]) {

        if (list.isEmpty()) {
            System.out.println("you␣have␣not␣subscribed␣to␣any␣topic
                !!␣zero␣byte␣array␣has␣been␣returned");
            return (new byte[]{0b0});
        } else {
            for (int i = 0; i < list.size(); i++) {
                if ((list.get(i).get_topic().equals(args[0])) && (
                    list.get(i).get_QOS() == Integer.parseInt(args[1])
                    )) {
                     System.out.println(Arrays.toString(list.get(i).
                         get_message()));
                     return list.get(i).get_message();

                }

            }

            System.out.println("else␣ended␣here");
            return (new byte[]{0b0});
        }

    }

    //i am returning the time not the time stamp. take care here
    public long get_message_time_(String args[]) {
        if (list.isEmpty()) {
            return 0;
        } else {
            for (int i = 0; i < list.size(); i++) {
                if ((list.get(i).get_topic().equals(args[0])) && (
                    list.get(i).get_QOS() == Integer.parseInt(args[1])
                    )) {
```

```
                    return list.get(i).get_message_time_stamp().
                        getTime();

                }

            }
        }
        return 0;

    }

    public double get_message_number_(String args[]) {
        if (list.isEmpty()) {
            return 0;
        } else {
            for (int i = 0; i < list.size(); i++) {
                if ((list.get(i).get_topic().equals(args[0])) && (
                    list.get(i).get_QOS() == Integer.parseInt(args[1])
                    )) {
                    return list.get(i).get_message_number();

                }

            }
        }
        return 0;

    }

    public boolean publish(String args[]) {
        if (client.isConnected()) {
            int qos;
            switch (args[2]) {
                case ("" + 0):
                    qos = 0;
                    break;
                case ("" + 1):
                    qos = 1;
                    break;
                case ("" + 2):
                    qos = 2;
                    break;
                default:
                    qos = 2;
                }
                //topic topic = new topic(args[0], qos, 1);
                //list.add(topic);
                //list6.addItem(topic);
                try {
                    client.publish(args[0], args[1].getBytes(), qos,
                        false);
                } catch (MqttException ex) {
                    Logger.getLogger(Matlab_MQTT_Client_Main_Class.class.
                        getName()).log(Level.SEVERE, null, ex);
                }

                return true;
            } else {
                return false;
            }
        }

}
```

5.6 Source Code Java - Topic class

```
import java.sql.Timestamp;
public class topic {
    static final int QOS_ZERO = 0 ;
    static final int QOS_ONE = 1 ;
    static final int QOS_TWO = 2;
    static final int SUBSCRIBE = 0;
    static final int PUBLISH = 1;

    private double message_number ;
    private Timestamp message_time_stamp  ;
    private int QOS ;
    private String topic ;
    private int topic_type;
    private byte[] message = {0b0};
    topic (String topic, int QOS,int topic_type)
    {
        this.QOS=QOS;
        this.topic=topic;
        this.topic_type=topic_type;
        message_number = 0;
        message_time_stamp = new Timestamp (System.currentTimeMillis
           ());

    }

    public void set_topic (String topic)
    {
        this.topic=topic;
    }
    public String get_topic ()
    {
        return topic;
    }
    public void set_QOS (int QOS)
    {
        switch(QOS){
            case QOS_ZERO :
                this.QOS=QOS_ZERO;
                break;
            case QOS_ONE :
                this.QOS=QOS_ONE;
                break;
```

```
          case QOS_TWO:
               this.QOS=QOS_TWO;
               break;
          default  :
               this.QOS=QOS_TWO;

                    }
}
public int get_QOS ()
{
     return QOS;
}
public int get_topic_type()
{
     return topic_type;
}
public void set_topic_type(int topic_type)
{
     switch (topic_type)
     {
          case SUBSCRIBE  :
               this.topic_type=SUBSCRIBE;
               break;
          case PUBLISH:
               this.topic_type=PUBLISH;
               break;
          default:
               this.topic_type=SUBSCRIBE;
     }
}

public byte[] get_message ()
{
     return this.message;
}



public void set_message (byte[] message)
{
     this.message= message;
}

public void set_message_number (double message_number)
```

```
    {
       this.message_number = message_number;
    }

    public double get_message_number ()
    {
        return this.message_number;
    }

    public void set_message_time_stamp (long time)
    {
        this.message_time_stamp.setTime(time);
    }
    public Timestamp get_message_time_stamp ()
            {
                return this.message_time_stamp;
            }

}
```