Setup for the laboratory environment:

1. HARDWARE REQUIREMENTS

OVERVIEW

2x Raspberry Pi
1x BME280 sensor
5x Experimental cable connectors <female/male>
1x Breadboard
2x Ethernet cable
2x SD Card
10x Arduino WEMOS Module
8x Incremental sensor OR 8x Gyro sensor
2x Robot arms with servos and power supply
1x Personal computer (program Arduinos)

optional: (if not configured remotely with SSH)
1x HDMI cable
1x Monitor
Keyboard and Mouse

RASPBERRY A

Raspberry A acts as a MQTT broker. It will provide a Wifi Access Point with IP-tables as a firewall and the service mosquitto,

RASPBERRY B

Raspberry B acts as a publisher and/or as a subscriber. Appropriate python scripts and node-red flows will be created on it.

Sources:

Raspberry pi as a MQTT broker:
https://diyprojects.io/mqtt-mosquitto-communicating-connected-objects-iot/#.Wzt4tMJpG00

Raspberry pi as a Wifi Access Point:
https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

IP-tables firewall:
http://www.gtkdb.de/index_36_2167.html

Adafruit GPIO:
https://github.com/adafruit/Adafruit_Python_GPIO

Paho MQTT documentation:
https://pypi.org/project/paho-mqtt/

Arduino MQTT (PubSubClient) documentation:
https://pubsubclient.knolleary.net/api.html

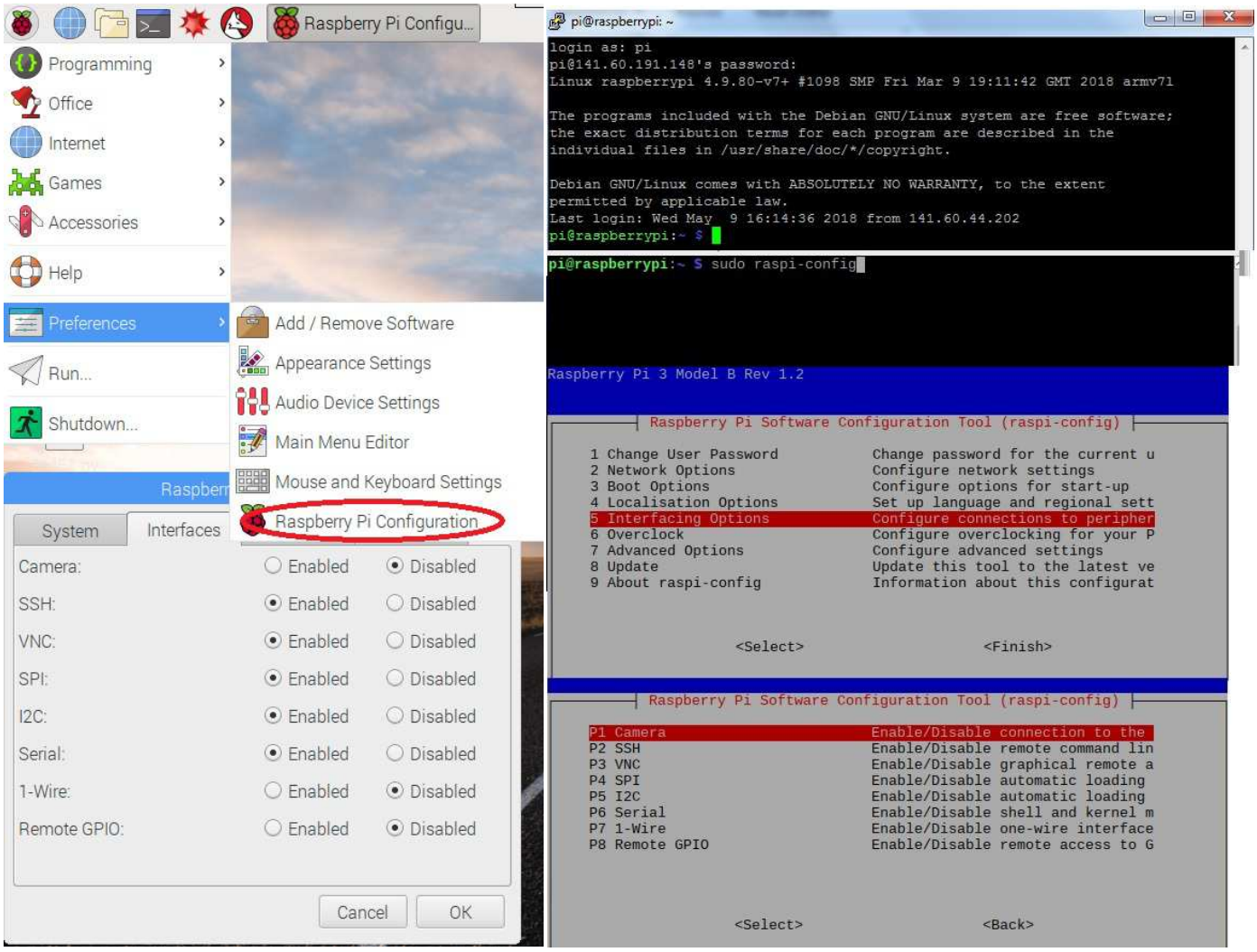
For further questions mailto:josef.joerg@stud.fh-rosenheim.de or mailto:joerg-josef@gmx.de

2. SOFTWARE REQUIREMENTS

RASPBERRY A&B

Install Raspbian (Stretch) on both Raspberries. Node-Red should be included with the initial installation of Raspbian. If not, install it by using sudo apt-get install nodered in the command line.

Enable the interfaces as shown in the pictures below. This can be done either with VNC using the Desktop environment or the shell command sudo raspi-config. For remote control from a windows system putty is required.



If you use the modified raspbian image then this is already enabled. Also, the password for the user pi is "masterproj"

3. MQTT Broker on Raspberry

Raspberry A

We use the package "mosquitto" to create a MQTT broker (server) which handles all incoming messages and forwards them to all subscribers. Only one broker is necessary for a entire network. The installation can be done with the command

```
sudo apt-get install mosquitto
```

(optional test)

We can use the command "sudo service mosquitto status" to check if the server is running correctly. It should look like the picture below. If it doesn´t, you should restart your system and check again. The service is supposed to start after the system boots by default.

```
pi@raspberrypi:~ $ sudo service mosquitto status
● mosquitto.service - LSB: mosquitto MQTT v3.1 me
   Loaded: loaded (/etc/init.d/mosquitto; generat
   Active: active (exited) since Tue 2018-05-08 1
     Docs: man:systemd-sysv-generator(8)
  Process: 345 ExecStart=/etc/init.d/mosquitto st
   CGroup: /system.slice/mosquitto.service
```

Configuration of the MQTT broker

In order to use authentication with the MQTT broker we need to configure it first with a password, username and the allow_anonymous false options.

Be aware that this is only for testing purposes. I would recommend to let the students configure the MQTT broker with a username and a password after they managed to change the "subscriber_easy.py" and "publisher_easy.py" in a manner that they can connect to a basic MQTT broker without authentication.

If you use the raspbian image then the next steps are preconfigured with user: auth_user and password: iotlab You only have to uncomment the line #allow_anonymous false in the configuration file to set authentication active.

Change the working directory `cd /etc/mosquitto`
Create a password file

```
pi@raspberrypi:/etc/mosquitto $ sudo mosquitto_passwd -c passwd user
Password:
Reenter password:
```

then open the file /etc/mosquitto/mosquitto.conf with `sudo nano mosquitto.conf` and add the lines below at the bottom of the file.

Filename

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

The changes only become effective if you RESTART the raspberry pi.

Authentication can be tested with the Raspberry B later. After testing it remove the password file and the two lines above from the mosquitto.conf file to let the students configure it.(Attention only ONE broker is running for all student groups - maybe do it together with them via the projector in the classroom after everyone connected to the basic version without authentication)

4. USING THE RASPBERRY AS A WIFI ACCESS POINT

If you use the modified raspbian image, the WIFI is already configured with the SSID: iotlab and password: iotwifipasswd (WPA2)

INSTALL REQUIRED PACKAGES

hostapd - This package helps to use the built in WiFi adapter as an access point
dnsmasq - This is a combined DHCP and DNS server. It is very easy to configure.

```
sudo apt-get install dnsmasq hostapd
```

Since the configuration isn´t complete yet, stop the services for now

```
sudo systemctl stop dnsmasq.service
sudo systemctl stop hostapd.service
```

CONFIGURATON OF A STATIC IP

For the configuration of a standalone network as a server, the Raspberry needs to have a static IP address assigned to the wireless port. We will assign the IP address 192.168.10.1 for the server. The wireless device used is wlan0. Any other address is also possible.

To configure the static IP address, edit the dhcpcd configuration file with:

```
sudo nano /etc/dhcpcd.conf
```

Go to the end of the file and add the following lines:

```
interface wlan0
static ip_address=192.168.10.1/24
```

save and close the file with ctrl+O - enter - and then ctrl+X.

Restart the dhcpcd and set up the new wlan0:

```
sudo service dhcpcd restart
```

CONFIGURATION OF THE DHCP-SERVER (DNSMASQ)

Rename this configuration file, and edit a new one:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

Copy the following lines into the dnsmasq configuration file and save it:
(we are going to provide IP addresses between 192.168.10.2 and 192.168.10.30, with a lease time of 24 hours)

```
interface=wlan0
dhcp-range=192.168.10.2,192.168.10.30,255.255.255.0,24h
```

CONFIGURING THE ACCESS POINT HOST SOFTWARE (HOSTAPD)

Edit the hostapd configuration file:
```
sudo nano /etc/hostapd/hostapd.conf
```

Add the following lines to the configuration file:

```
interface=wlan0
driver=nl80211
ssid=NameOfNetwork
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=<password>                    #at least 8 letters
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

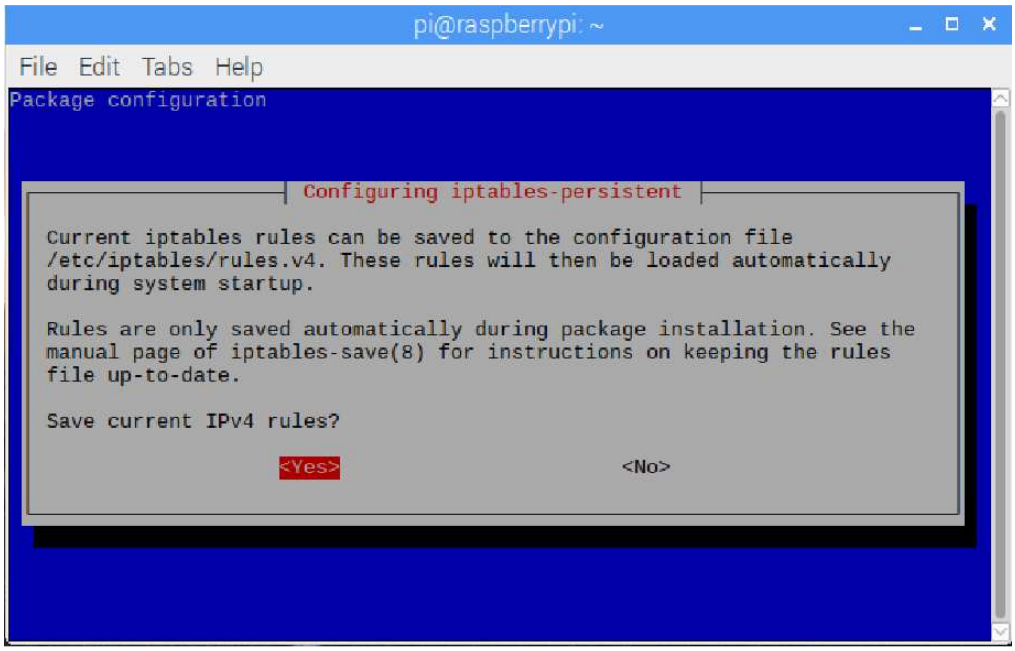
hostapd needs the location in order to find this configuration file,
```
sudo nano /etc/default/hostapd
```

Find the line with #DAEMON_CONF, and replace it with this:
```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Now, start the services again:
```
sudo service hostapd start
sudo service dnsmasq start
```

5. CONFIGURING IP-TABLES AS A FIREWALL

Auto-enable the iptables while booting:
`sudo apt-get install iptables-persistent`



If you see this message, confirm twice with "yes"

In order to avoid traffic between the wlan0 and the eth0 interface we need to change the rules:
`sudo iptables -A FORWARD -i eth0 -o wlan0 -j DROP`
`sudo iptables -A FORWARD -i wlan0 -o eth0 -j DROP`
`sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE`

Save the iptables rules:
`sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"`

The file /etc/iptables/rules.v4 should now contain the rulesin *filter and *nat category

For enabling IPv4-traffic forwarding(in general e.g. wlan0 <-> wlan0) open up the sysctl.conf file
`sudo nano /etc/sysctl.conf`

and uncomment this line
`net.ipv4.ip_forward=1`

Reboot the raspberry

(OPTIONAL TEST)

Connect the raspberry to a wired internet connection. Connect to its WIFI with a phone or laptop and try to google something. If it works, your rules are not effective.

If there is no connection, ping 192.168.10.1 to check if there is connection to the access point. If this works, everything is fine.

6. INSTALL TSHARK FOR RASPBERRY

In order to record the traffic in the lab exercise we need to install tshark
```
sudo apt-get install tshark
```

and allow other users than root to use tshark
```
sudo dpkg-reconfigure wireshark-common
```
click on "yes"

Allow user pi to work with tshark
```
sudo usermod -a -G wireshark pi
```

(OPTIONAL TEST)

In the command line use
```
sudo touch /home/pi/Desktop/testcapture.cap
```

to create a container for the records

and make it accessible for tshark with
```
sudo chmod o=rw /home/pi/Desktop/testcapture.cap
```

7. Publisher And Subscriber on Raspberry

Raspberry B

Login to the user "pi" at the raspberry and change your working directory with
`cd /home/pi/Desktop`

Use the PUB_SUB folder as the source with the scripts

publisher_easy.py publisher_enhanced.py subscriber_easy.py subscriber_enhanced.py

in it. The comments in the python scripts should explain the code.

If you work on the raspberry itself with an USB drive just copy the folder on the Desktop.
If you work remotely from a Windows system and you want to copy the files via SSH, use the Windows command line.
You should have installed putty with the pscp.exe first.(Included in the default installation of putty)
Open up the command line tool in Windows and use the command "pscp" with the option "-r" in order to copy folders:

Usage in this case: `pscp <Options> Source User@Destination:Destination-Path`

Example:

```
C:\Users\Sepp>pscp -r C:\Users\Musteruser\Desktop\PUB_SUB pi@141.60.191.148:/hom
e/pi/Desktop
```

Before we can use these scripts we have to install the python libraries for the Adafruit_BME280 sensor and the MQTT protocol.

Adafruit GPIO Library

Use these commands to install it:

```
sudo apt-get update
sudo apt-get install build-essential python-pip python-dev python-smbus git
git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
cd Adafruit_Python_GPIO
sudo python setup.py install
```

(optional test)

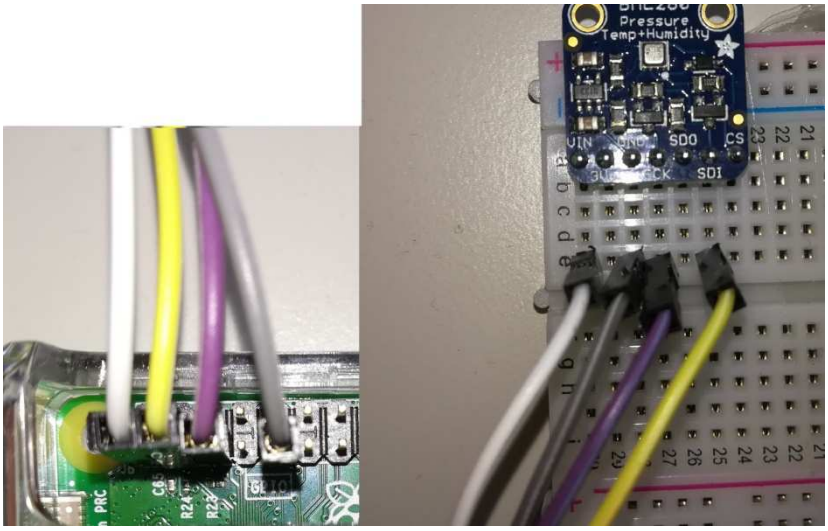
Change the working directory to the Desktop again. Use the command
`git clone https://github.com/adafruit/Adafruit_Python_BME280.git`

to clone (download) appropriate examples. With the following commands you can test whether the installation was correct or not.

```
cd Adafruit_Python_BME280
python Adafruit_BME280_Example.py
```

If you connected the BME280 sensor correctly, you should see something like this

```
Temp      = 23.022 deg C
Pressure  = 965.21 hPa
Humidity  = 46.33 %
```



PAHO-MQTT LIBRARY

Paho MQTT Library provides a client class which enable applications to connect to an MQTT broker to publish messages and to subscribe to topics and receive published messages.
It supports Python 2.7.9+ or 3.4+, with limited support for Python 2.7 before 2.7.9

Package python-pip is required. If you follow this tutorial from the beginning it is already installed.
`sudo apt-get install paho-mqtt`

If this step finishes successfully the system is good to go.

(OPTIONAL TEST)

The copied python scripts should work after you set the correct broker address. In order to do that open them and enter the IP address of your broker raspberry.

```
7   #define the mqtt broker adress (destination)
8   broker address = "141.60.191.41"
```

Depending on the port you set for the MQTT to listen on (default 1883) you have to edit some more lines. If you use the default port settings, skip this step.

```
38  #try to connect the client to the mqtt broker
39  try:
40      client.connect(broker_address,1883)
```

If you haven´t set up a password and username for your MQTT broker you can use the scripts "publisher_easy.py" and "subscriber_easy.py" as they are.
If you configured your MQTT broker on Raspberry A with an username and a password use the scripts "publisher_enhanced.py" or "subscriber_enhanced.py" and replace appropriate lines in the code.

```
26  #authentication via password and username
27  client.username_pw_set("thangz","hello")
```

Read more here: https://pypi.org/project/paho-mqtt/

8. SET UP STUDENT GROUPS:

Once this part of the environment is working we can set up multiple users for each group. Before doing this make sure your PUB_SUB folder with the contained python scripts is stored on the Desktop of the pi user. (/home/pi/Desktop/PUB_SUB)

Copy the file useradd.sh from your source folder to the pi user´s desktop and open the linux command line.

As user pi change your working directory with
`cd /home/pi/Desktop`

In order to make the script executable use the command
`sudo chmod +x useradd.sh`

You can use this script to create a default student user with all necessary rights to work through the whole lab exercise. The usage of the script is shown, if you just enter
`sudo ./useradd.sh`

The usage is "sudo useradd.sh <Groupnumber>".
Example "sudo ./useradd.sh 99" creates a user named grp99 with the password group99.

After execution it should look like this:

```
pi@raspberrypi:~/Desktop $ sudo ./useradd.sh 99
Adding user `grp99' ...
Adding new group `grp99' (1001) ...
Adding new user `grp99' (1001) with group `grp99' ...
Creating home directory `/home/grp99' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: Retype new UNIX password: passwd: password updated succe
ssfully
Changing the user information for grp99
Enter the new value, or press ENTER for the default
        Full Name []:   Room Number []:         Work Phone []:  Home Phone []:  O
ther []: Is the information correct? [Y/n]

---User added successfully---

Username: grp99
Password: group99

User was assigned to required groups (permissions) for the  MQTT lab



Project folder was copied to /home/grp99/PUB_SUB
```

Set up as many groups as you want for your students.

9. MULTIPLE NODE-RED RUNNING ON RASPBERRY PI

Assuming that the groups are already created using the script “useradd.sh” let us start installing the node red dashboard in each user and modify the port numbers according to their group numbers.

Make sure you are logged in as user pi and install npm (necessary for the dashboard installation)
`sudo apt-get install npm`

Go into each user and execute below commands to assign unique port numbers:

(*)

`sudo login grpX` and type in password groupX	#X stands for the group number
`node-red`	#Start node-red in order to create the config folder for grpX
`ctrl+c`	#End node-red
`logout`	#Logout from grpX (you should be user pi again)
`cd /home/grpX/.node-red`	#From pi user going into grp1 node red config directory
`sudo npm install node-red-dashboard`	#Install Dashboard
`sudo nano settings.js`	#Open settings.js file to modify port numbers

Modify the port number according to each groups:
For Group1 use port 18801 and so on ->GroupX uses port 1880X

```
// to make it available:
//var fs = require("fs");

module.exports = {
    // the tcp port that the Node-RED web server is listening on
    uiPort: process.env.PORT || 18801,

    // By default, the Node-RED UI accepts connections on all IPv4 interfaces.
```

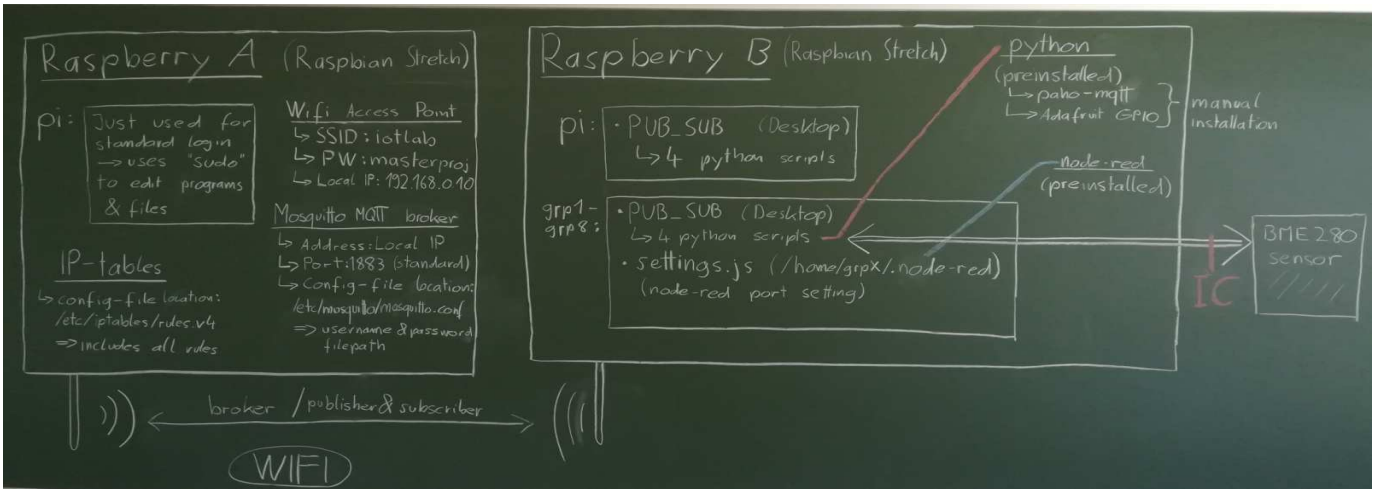
Attention: No port numbers greater than int-range (65535) and also not 1883 (that´s by default the MQTT broker)

(OPTIONAL TEST)

Login to GrpX again and start node-red:
`node-red`

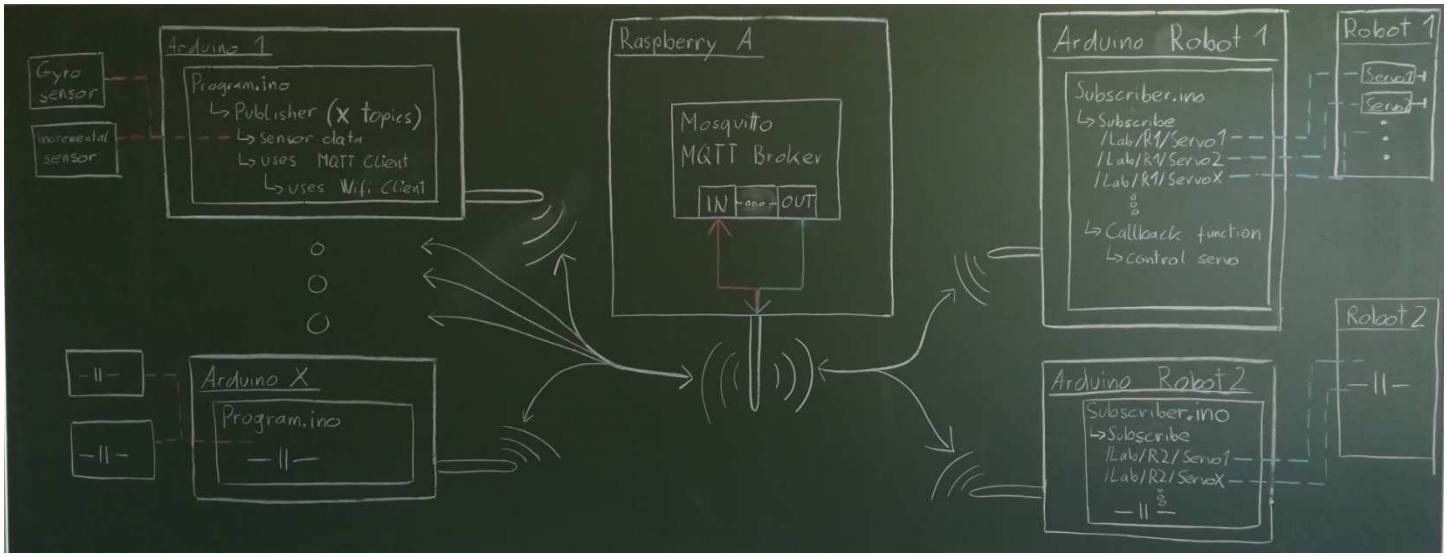
Open Browser and type in URL “localhost: 1880X” and check if the dashboard is available on the right top side.
Repeat the same from (*) for multiple users (Grp1 to Grp8)

If everything is done, the environment should look like this schematic:



10. ARDUINO SYSTEMS

We have the first part of the setup working now, so we extend the system with our Arduinos. After programming all Arduinos it should look like this:



The programs are located in the source folder -> subfolder Arduinos.

We have two programs for the robots. The difference is that each robot only subscribes to its topic.
For robot2 it is like that

```
void reconnect() {
  // Loop until we're reconnected
  while (!MQTTclient.connected())
  {
                    ...
    MQTTclient.subscribe("/lab/robot2/axgrp1");
    MQTTclient.subscribe("/lab/robot2/axgrp2");

                    ...
  }
}
```

For robot1 this part is programmed accordingly.

For the Gyros there is only one code. Depending on the button pressed at the power up or reset it can control the robots e.g.

Button 1	Button 2	Publisher topic
0	0	/lab/robot1/axgrp1
0	1	/lab/robot1/axgrp2
1	0	/lab/robot2/axgrp1
1	1	/lab/robot2/axgrp2

This table needn´t be true, it is only for the demonstration of the principle
Further documentation about the programs can be found as comments in the code or send me an Email.