Smarthome Hackathon

Prerequisites

If you are on **Windows** you can either install the package manager ***Chocolatey*** first, or install ***NodeJS*** and ***Python3*** by hand.

https://chocolatey.org/docs/installation

NodeJS

- Go to https://nodejs.org/en/download
- Select your OS version installer

Alternatively use package manager:

Ubuntu

- `sudo apt-get install nodejs`
- `sudo apt-get install npm`

MacOS X

- `brew install nodejs`

Windows

- `choco install nodejs`

Python 3

Any python3 version will work

- Visit the python3.9 download website at https://www.python.org/downloads/release/python-390/
- Scroll down and select your OS version installer
- Run installation and select `Add Python 3.9 to PATH`

Alternatively use package manager:

Ubuntu

- `sudo apt-get install python3.9`

MacOS X

- `brew install python`

Windows

- `choco install python3 --pre`

Mosquitto

- Visit the mosquitto download website at https://mosquitto.org/download
- Select your OS version installer
- Add mosquitto folder to environment path if necessary

Alternatively use package manager:

Ubuntu

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

MacOS X

- `brew install mosquitto`
- `(optional) add /usr/local/sbin to path`

Version 2.X

If you have Version > 2.0 you need to add following information to your ***mosquitto.conf*** file

```
listener 1883
allow_anonymous true
```

Path to conf:

- Linux: `/etc/mosquitto/mosquitto.conf`
- MacOS: `/usr/local/etc/mosquitto/mosquitto.conf` or `/opt/homebrew/etc/mosquitto/mosquitto.conf`

Stop auto start:

Ubuntu

- `sudo systemctl stop mosquitto.service`

Windows

- `net stop mosquitto`

MacOS X

- `launchctl stop homebrew.mxcl.mosquitto`

ESPHome

- Install using python package manager pip
- Open a terminal as administrator and enter
  - `# pip3 install esphome`

Node-Red

- Install using node package manager npm
- Open a terminal as administrator and enter
  - `# npm install -g node-red`

Windows Drivers

- Make sure you have the necessary usb driver installed and your computer can detect the nodeMCU board
- Driver Options:
  - https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers
  - https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers
  - https://medium.com/@cilliemalan/installing-nodemcu-drivers-on-windows-d9bffdbad52

Cable Switch

ESPHome setup

- Make sure you installed ***ESPHome*** successfully, for reference check the prerequisites section
- Create a new folder and open a terminal to start the ESPHome setup wizard
  - `$ esphome cable_switch.yaml wizard` -- if you get an Errno -13 try to run esphome as admin
- Enter the following information, when prompted:
  STEP 1 CORE/NODE: cable_switch -> This will be the name in the yaml file
  STEP 2 ESP (platform): ESP8266
  STEP 2 ESP (board): nodemcuv2
  STEP 3 WIFI (ssid): YOUR_WIFI_SSID

STEP 3 WIFI (psk): YOUR_WIFI_PASSWORD

STEP 4 OTA (password): press enter (no password)

- Inspect the ***cable_switch.yaml*** file
- Use a micro usb cable to connect your NodeMCU to your computer
- Flash the above created firmware onto your NodeMCU with the following command
  - `$ esphome cable_switch.yaml run`
- After compilation enter `1` to select ***USB Serial*** to upload the firmware
- You should now see that your NodeMCU connects to your Wi-Fi

Output:

```
[12:12:44][C][wifi:303]:   SSID: 'WIFI_NAME'
[12:12:44][C][wifi:304]:   IP Address: IP_ADDRESS
```

Binary Sensor Component

- Open your ***cable_switch.yaml*** file and add following information

```
binary_sensor:
  - platform: gpio
    pin:
      number: D1
      mode: INPUT_PULLUP
      inverted: True
    name: "My first Binary Sensor"
```

- Flash the firmware onto your NodeMCU with the command
  - `$ esphome cable_switch.yaml run`
- Use a cable to connect the ***D1*** pin to a ***G*** pin on your NodeMCU
- You should see output similar to the following when connecting and disconnecting the two pins

Output:

```
[10:51:42][D][binary_sensor:036]: 'My first Binary Sensor': Sending state ON
[10:51:43][D][binary_sensor:036]: 'My first Binary Sensor': Sending state OFF
```

MQTT

- Open a new terminal and start the mqtt broker ***mosquitto*** with the command `$ mosquitto -c <path-to-config-file>/mosquitto.conf`

Output:

```
1603706412: mosquitto version 1.6.9 starting
1603706412: Using default config.
```

```
1603706412: Opening ipv4 listen socket on port 1883.
1603706412: Opening ipv6 listen socket on port 1883.
```

- Find out your computers local ip address and configure your mqtt broker in the ***cable_switch.yaml*** file

IP Address

```
ifconfig | grep inet // Linux, MacOS X
ipconfig // Windows
```

Add following information to your ***cable_switch.yaml*** file

```
mqtt:
  broker: YOUR_LOCAL_IP_ADDRESS
```

- Flash the firmware onto your NodeMCU with the command
  - `$ esphome cable_switch.yaml run`

Output:

```
[09:08:38][C][mqtt.binary_sensor:018]: MQTT Binary Sensor 'My first Binary Sensor':
[09:08:38][C][mqtt.binary_sensor:019]:   State Topic: 'cable_switch/binary_sensor/my_first_bir
```

- Make sure you installed ***node-red*** successfully, for reference check the prerequisites section
- Open a new terminal and start ***node-red*** with the command `$ node-red`

Output:

```
26 Oct 11:02:29 - [info] Server now running at http://127.0.0.1:1880/
26 Oct 11:02:29 - [info] Starting flows
26 Oct 11:02:29 - [info] Started flows
```

- Open a browser and go to http://127.0.0.1:1880
- Drag and drop a ***mqtt in*** node and a ***debug*** node onto the main frame
- Connect the two gray dots between the nodes
- Double-click the ***mqtt in*** node and click the ***pencil symbol***
- Enter a name like `mosquitto_local` for your local mqtt broker mosquitto, enter `localhost` in the server field and click ***Add***
- Copy `cable_switch/binary_sensor/my_first_binary_sensor/state` into the topic field and click ***Done***
- Click ***Deploy*** in the top right corner and click the ***bug symbol*** to see the debug output

- When connecting and disconnecting the ***D1*** and ***G*** pins on your NodeMCU you should see messages in the debug window

LED

ESPHome firmware

- Copy the information from your ***cable_switch.yaml*** file to a new file named ***led.yaml***
- Change the name specified in the yaml file to `esp_led`

```
esphome:
  name: cable_switch  #change to esp_led (On-site workshop: add a unique identifier)
  platform: ESP8266
  board: nodemcuv2
```

TASK: LED1

- Go to https://esphome.io/components/light/monochromatic.html
- Add a light and an output component to your yaml file
- Use jumper wires to connect the specified pin and a ***3V*** pin to your LED
- Flash the firmware onto your NodeMCU with
  - `$ esphome led.yaml run`

TASK: LED2

- Look for output starting with

```
[13:37:44][C][mqtt.light:054]: ...
```

- Find out which mqtt topic will turn the LED on and off

Configure node-red

- If not running anymore, start node-red and mosquitto again
- Go to http://127.0.0.1:1880 in your browser
- Add a ***mqtt in*** node to find out what information will turn
- Add a ***mqtt out*** node and use `mosquitto_local` as mqtt broker
- Configure the topic you found in the terminal output above
- Add two ***inject*** nodes and send the json objects that will turn the LED on/off

TASK: LED3

- Find out what json objects to send to `esp_led/light/led_light/command`

- Hint: The LED sends state messages via MQTT

TASK: LED4

- Use your Cable Switch from the last step to turn your LED on and off

TASK: (optional) LED5

- Dim your LED with your smartphone
- See section ***Additional*** for smartphone app suggestions

Wireless Socket

Setup Transceiver

- Wire up the CC1101 antenna as shown in the picture CC1101_wiring.png
- Go to `<tng-automation>/esp-smarthome/radio_transceiver.yaml` and change following information

```
wifi:
  ssid: "YOUR_WIFI_SSID"
  password: "YOUR_WIFI_PASSWORD"

mqtt:
  broker: YOUR_LOCAL_IP_ADDRESS
```

- Start mqtt broker mosquitto with `$ mosquitto -c <path-to-config-file>/mosquitto.conf`
- Flash the firmware onto your NodeMCU with
  - `$ esphome radio_transceiver.yaml run`
- Received RF timings are sent to the ***radio_transceiver/radio/433toMQTT*** topic
- MQTT messages with timings to ***radio_transceiver/radio/MQTTto433*** are sent via RF

Control wireless socket

- Go to `<tng-automation>/esp-smarthome` and inspect the ***wireless_socket_on*** and ***wireless_socket_off*** files
- Start node-red with `$ node-red` and open http://127.0.0.1:1880 in your browser

TASK: SOCKET1

- Turn your wireless socket on/off by sending timings from the files ***wireless_socket_on/off*** via mqtt

De/Encode Signals

- Instead of sending the recorded timings from the ***wireless_socket_on/off*** files we try to decode the timings and send a binary code

- Open terminal in `<tng-automation>/node-red/on-off-keying` and run:
  - `$ npm install`
  - `$ npm run build`
- On startup, node-red should print out a line like `7 Nov 20:48:30 - [info] User directory : home/<user>/.node-red`
- This is where node-red will keep user specific data like the flows you created, and where we can also install plugins
- Go to the node-red user directory (e.g. home//.node-red) and run
  - `$ npm install <tng-automation>/node-red/on-off-keying`
- Restart node-red and reload the node-red web interface
- There should now be four new nodes: ***ook_decode***, ***ook_encode***, ***ook_split***, ***ook_concat***
- Add an ***ook_decode*** node and double click it to see the configuration options

TASK: SOCKET2

- Inspect the `<tng-automation>/esp-smarthome/wireless_socket_on` file and find the correct patterns for zero, one, and start to configure the ***ook_decode*** node
- Use the ***file in*** node as input for the ***ook_decode*** node and a ***debug*** node as output
- When triggering the inject node you should see the 24 bit binary code necessary to turn on the wireless socket

END TASK

- Use an ***inject*** node to send the binary code as a string to an ***ook_encode*** node
- Configure the ***ook_encode*** node with the correct patterns for zero, one, and start
- Connect the output of the ***ook_encode*** with the ***mqtt out*** node
- Repeat this process with the ***wireless_socket_off*** file
- Now you should be able to turn your socket on and off using the correct binary code

TASK: SOCKET3

- Use your MQTT smartphone app to turn your socket on and off

TASK: (optional) SOCKET4

- Use your Cable Switch to send the 24 bit array to turn your socket on/off

TASK: (optional) SOCKET5

- Indicate the current status of your wireless socket with your LED

Weather Station

Decode Timings

- Output the timings sent to `radio_transceiver/radio/433toMQTT` in the node-red debug window
- Your NodeMCU should still be sending 433 MHz signals to the specified topic, otherwise flash the radio_transceiver.yaml file onto your device again
- Remove the backplate of your weather station and press the ***TX*** button to send 433 MHz signals
- You should see 433 MHz timings in the debug window

TASK: WEATHER1

- Try to find the correct patterns for zero, one, and start to configure an ***ook_decode*** node
- You should now see a binary array of length 40 in your debug window when pressing the ***TX*** button

Decode Binary

- Add a ***function*** node and use the 40 bit binary code as input

TASK: WEATHER2

Weather Station 40 Bits:

- Look at the blog post https://github.com/merbanan/rtl_433/blob/master/src/devices/infactory.c
- Find out how temperature and humidity are encoded in the 40 bit array

Weather Station 42 Bits:

- Look at the blog post https://forum.fhem.de/index.php/topic,58110.0.html
- Find out how temperature and humidity are encoded in the 42 bit array

TASK: WEATHER3

- Extract temperature and humidity from the binary code as described in the blog post from TASK: WEATHER2
- You can use following function to calculate the decimal number from a binary array

```
function binaryToNumber(input) {
    return input.reduceRight(
        ((previousValue, currentValue, currentIndex, array) =>
            Math.pow(2, (array.length - currentIndex - 1))
                * currentValue + previousValue), 0);
}
```

- Send temperature and humidity to your smartphone app

TASK: WEATHER4

- Send temperature and humidity to smartphone

TASK: (optional) WEATHER5

- Turn on a fan (wireless socket) depending on room temperature

HomeAssistant (optional)

- Install HomeAssistant and run it in a virtual machine or on a raspberry pi
- Connect a smart home speaker (Amazon Alexa, Google Nest, ...) to your wireless socket and weather station

Issues

No MQTT Message received

- If node-red doesn't receive the mqtt messages from your nodemcu, check if mosquitto is running in loopback mode
- Fix1: Disable mosquitto autostart, restart node-red, mosquitto and nodemcu
- Fix2: Use your local ip address instead of "localhost"

Start Node-Red

- If you cant start node-red on windows check if your Execution Policy is set correctly
- Fix: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned

Deployment Node-Red

- On Windows it can happen that re-deployment in node-red does not update the specified mqtt topic
- Fix: Restart mosquitto broker

Firewall Settings for MQTT

- Make sure your firewall settings allow the mosquitto broker to send and receive mqtt messages
- Fix: Einstellungen -> ***Firewall- & Netzwerkschutz*** -> ***Zugriff von App durch Firewall zulassen*** -> Select Mqtt

Additional

MQTT on smartphone

- Download an MQTT App for your smartphone
  - MQTT Dash(IoT, Smart Home) (for Android)s

    - IoT OnOff (for iOS)
    - MQTTTool (for iOS)
- Control your wireless socket with your smartphone
- View temperature and humidity information on your smartphone