EESTech Challenge 2019 - Tasks 3-4

Tasks and study material

## Learning - MQTT

See <https://sites.google.com/view/eestech-2019-zurich/study-material#h.p_QYNAB0wwl3as>

### Secure MQTT connection

Use this library: MQTT by Joel Gaehwiler (<https://github.com/256dpi/arduino-mqtt>)

Look at the example [AdafriutHuzzahESP8266Secure](https://github.com/256dpi/arduino-mqtt/blob/master/examples/AdafruitHuzzahESP8266Secure/AdafruitHuzzahESP8266Secure.ino). This uses "WiFiClientSecure net" as the transport layer (TLS+TCP) and "MQTTClient client" is using that.

It's calling "net.setInsecure();" to skip certificate validation. You can start with that and add validation later.

Note: On Macs, for the mosquitto\_\* commandline clients you want to use --cafile instead of --capath. The certificates are here:

[EESTech Challenge files Drive folder](https://drive.google.com/drive/folders/1Kjtf9Ja7W_PvOKqX2YID7vxm_NW6oWjQ)

The "DST Root CA X3.crt" file is the root certificate for the server.

The "certs.ar" file contains all root certificates and can be used with WiFiClientSecure to load a CertStore.

## Task 3.1 - MqttConnect

**Goal: Connect to MQTT and publish a message.**

**Score: 20**

Connect to the given MQTT broker and on successful connection, send the following message:

Topic: /teams/teamX/devices/<chipId>/events/connection

Payload: "1"

Retained: true

All topics a device needs to interact with will have the /teams/teamX/devices/<chipId>/ prefix.

<chipId> is the HEX chipId that is reported under /status?format=json.

Subtopics within this are:

* events/#: The device is publishing messages about events happening on the device.
* state: The device is publishing messages to this when its state changes.
* commands/#: The device is subscribing to these and executing commands sent by something else.
* config: The device is subscribing to this topic and receives configuration.

You don't have to use all these topics yet, but keep them in mind. It's worth creating a constant for the topic prefix and append different suffixes to it.

Evaluation: We'll evaluate this task automatically, stay tuned.

## Task 3.2 - MqttLastWill

**Goal: Set the last will for the MQTT connection.**

**Score: 10**

On MQTT connections it's possible to set a "last will", which is a message that is dispatched when the client is disconnected. This is good for keeping track of whether a client is connected or not.

Topic: /teams/teamX/devices/<chipId>/events/connection

Payload: "0"

Retained: true

When the device sends a message on connection and also sets a will to be sent on disconnection, then by listening to these we can tell if the client is connected.

## Task 3.3 - MqttLedControl

**Goal: Control the LED using MQTT.**

**Score: 20**

Similarly how the LED could be controlled through HTTP, now make it possible to turn the LED on/off using MQTT.

Topic: /teams/teamX/devices/<chipId>/commands/led

Payload: "1" → turn on, "0" → turn off

Don't forget to update the led status that is served by HTTP (text and JSON).

In addition to that, publish an event message for the change:

Topic: /teams/teamX/devices/<chipId>/events/led\_state

Payload: "1" → LED is on, "0" → LED is off

Publish the event even if the LED state didn’t change.

**Evaluation: This works now!**

Subscribe to the topic wildcard /evaluation/teams/teamX/#

Send a message (just one, e.g. from your laptop):

Topic: /evaluation/teams/teamX/tasks/MqttLedControl/test\_request

Payload: your chipId

mosquitto\_pub … -t /evaluation/teams/<teamX>/tasks/MqttLedControl/test\_request -m <chipId>

You'll see the test progress and result in the output.

## Learning - handling a button, GPIO, polling

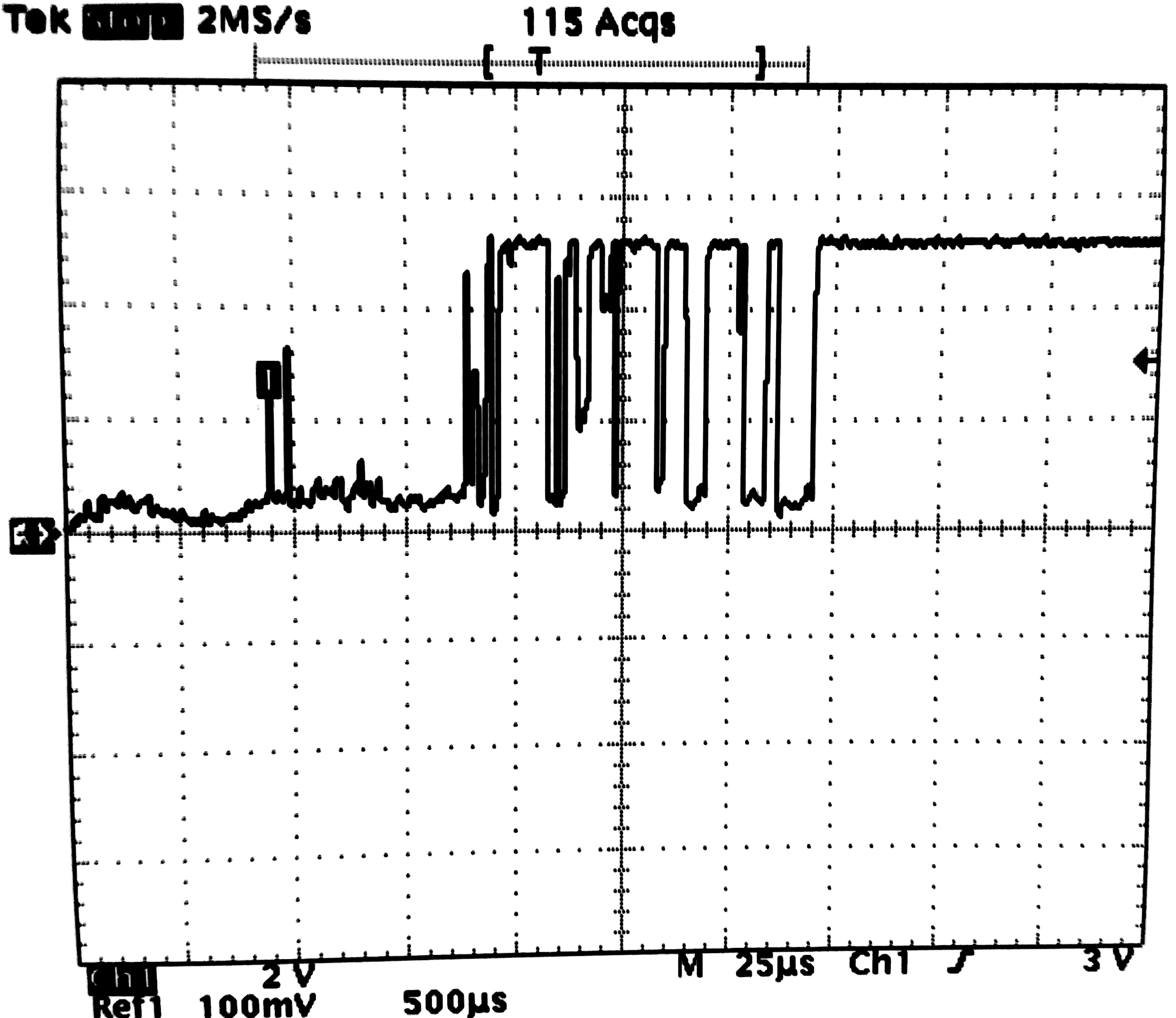
**GPIO**

Quoting [Wikipedia](https://en.wikipedia.org/wiki/General-purpose_input/output):

*A general-purpose input/output (GPIO) is an uncommitted digital signal pin on an integrated circuit or electronic circuit board whose behavior—including whether it acts as input or output—is controllable by the user at run time.*

**Contact bounce**

When a mechanical button is pressed or released, the electrical contact can be made and broken a couple of times before settling on the changed state. This is called bouncing, so the signal needs to be "debounced", see <https://en.wikipedia.org/wiki/Switch#Contact_bounce>.



The easiest solution is to sample the button state e.g. only every 50ms, then a bounce will not be detected as repeated button presses and releases.

Debouncing can be done in [hardware](http://www.labbookpages.co.uk/electronics/debounce.html) or in [software](https://www.arduino.cc/en/tutorial/debounce).

**millis()**

Both for software debouncing and for measuring time between events, the [millis()](https://www.arduino.cc/reference/en/language/functions/time/millis/) function is useful.

## Task 3.4 - MqttButton

**Goal: Handle a button, send MQTT events when pressed/released**

**Score: 20**

Connect a button to the ESP and report when it's pressed or released over MQTT.

When the button is pressed, it should be reported as:

Topic: /teams/teamX/devices/<chipId>/events/button/pressed

Payload: ""

When the button is released, it should be reported as:

Topic: /teams/teamX/devices/<chipId>/events/button/released

Payload: <length of button press in milliseconds, OK to have only 50ms resolution>

E.g. if the button was pressed for 310 ms, the following messages are sent:

/teams/teamX/devices/<chipId>/events/button/pressed ""

/teams/teamX/devices/<chipId>/events/button/released "300"

Hint: GPIO0 / D3 has a pull-up resistor already and usually has a "Flash" button on many ESP8266 boards. Connecting a button between this pin and the ground is the easiest. It will still need to be debounced.

## Hardware - Photoresistor GL5528

See <https://sites.google.com/view/eestech-2019-zurich/study-material#h.p_f3kMuNDQoFaU>

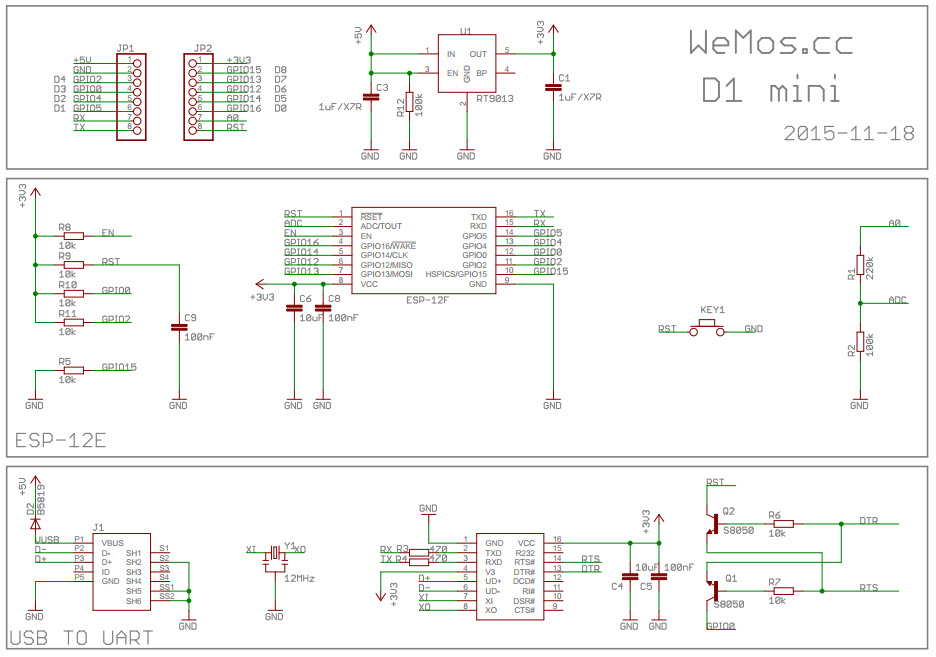
## Task 4.1 - LightSensor

**Goal: Report measurements from the light sensor periodically.**

**Score: 20**

Use the photoresistor and a 3.3kOhm resistor as a voltage divider and connect them to the analog input of the ESP board. Use analogRead(A0) to get a value between 0 and 1024 corresponding to the analog voltage on the input.

Note: the ESP analog input (ADC) is prepared for voltages between 0 and 1V, but on the Wemos D1 mini board there's a voltage divider made of a 220k and 100k resistor, so it's OK to put a voltage between 0V and 3.3V on the A0 pin of the board.



Report the analog input value periodically every 10 seconds to the following topic:

Topic: /teams/teamX/devices/<chipId>/events/light

Payload: a number as string between "0" - "1024"

## Hardware - DHT11

See <https://sites.google.com/view/eestech-2019-zurich/study-material#h.p_g3zPK-n_nMNi>

## Task 4.2 - TemperatureSensor

**Goal: Report measurements from the DHT11 sensor periodically**

**Score: 30**

Use the DHT11 sensor to get temperature and humidity values and report then as events every 10 seconds.

Topic: /teams/teamX/devices/<chipId>/events/dht

Payload format: {"temperature":25.3,"humidity":45.5}