# SMART GARDEN

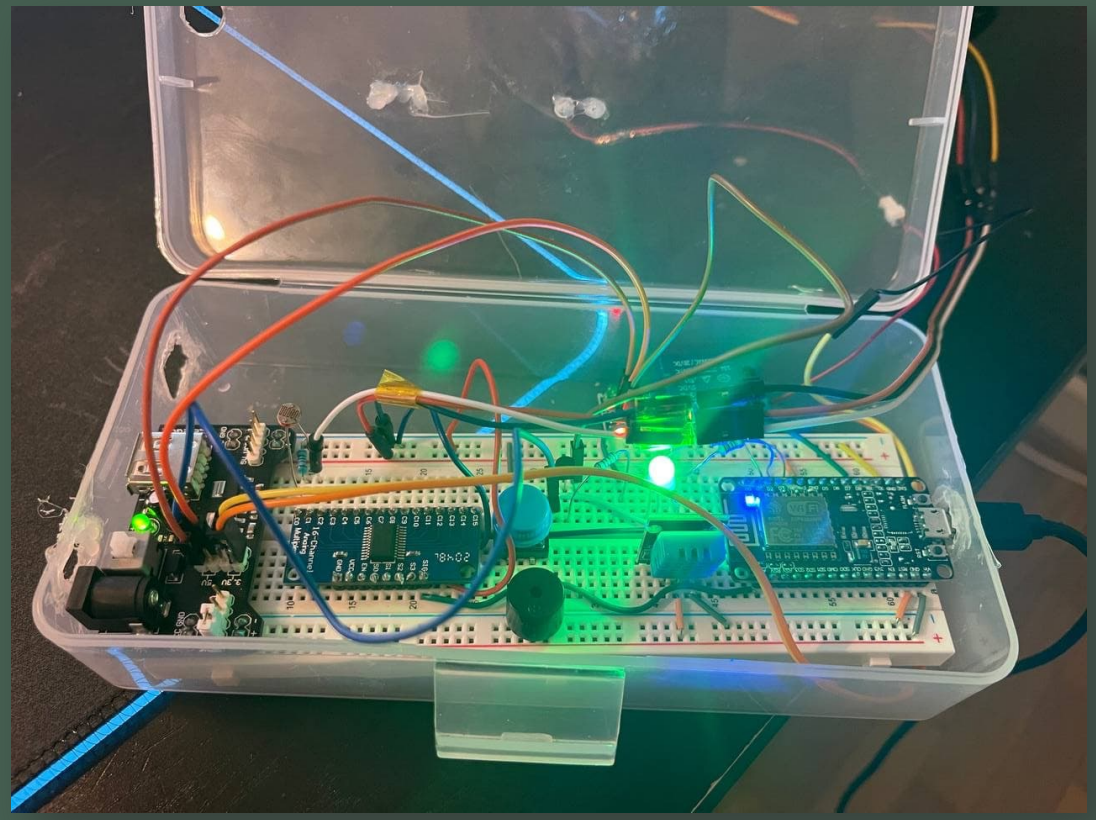
Esame di Laboratorio IoT
a.a. 2020/2021

Gianlorenzo Occhipinti
829524

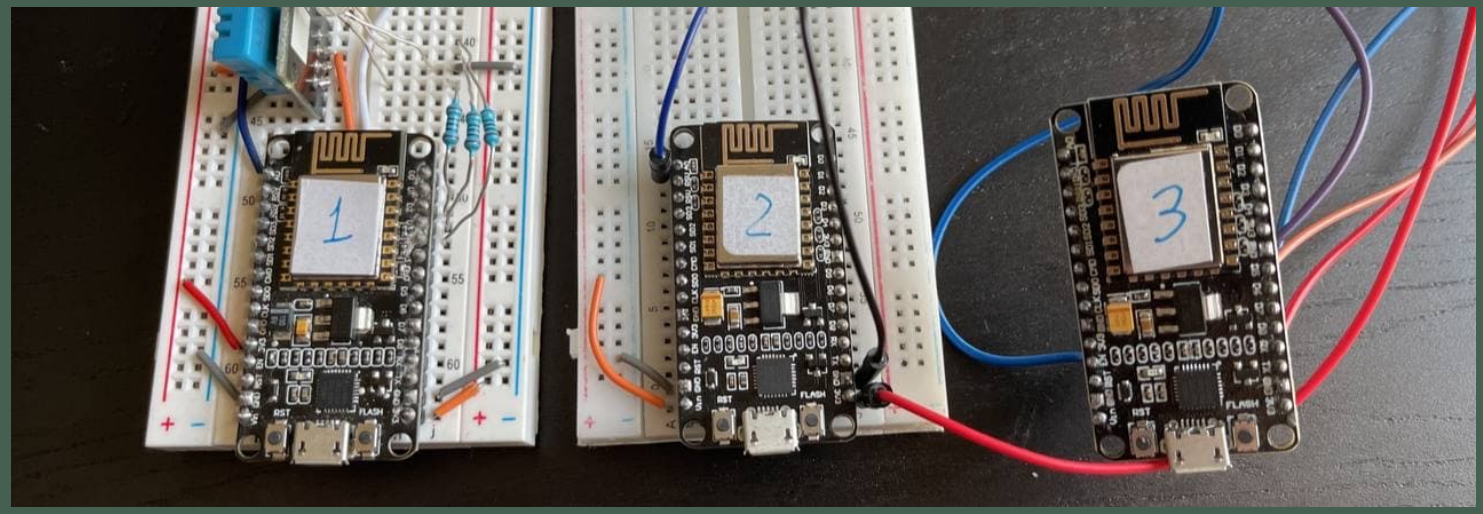
# Project description

- Smart Garden is an autonomous, fully configurable system that monitor the moisture level of the soil and water the plants in a smart way.
- The system has a feature called smart rain, if the next day will rain with a high probability, the plants won't be watered if their moisture soil level is not critical (under the half of the threshold).
- The system has several kinds of capabilities, most of them are optional, and this means that a user can build the perfect system only for what he needs.
- All the system works inside a net of microcontroller and in a decentralized way, so there isn't a hub which represent a single point of failure.

# After 1st assignment



# After 2nd assignment

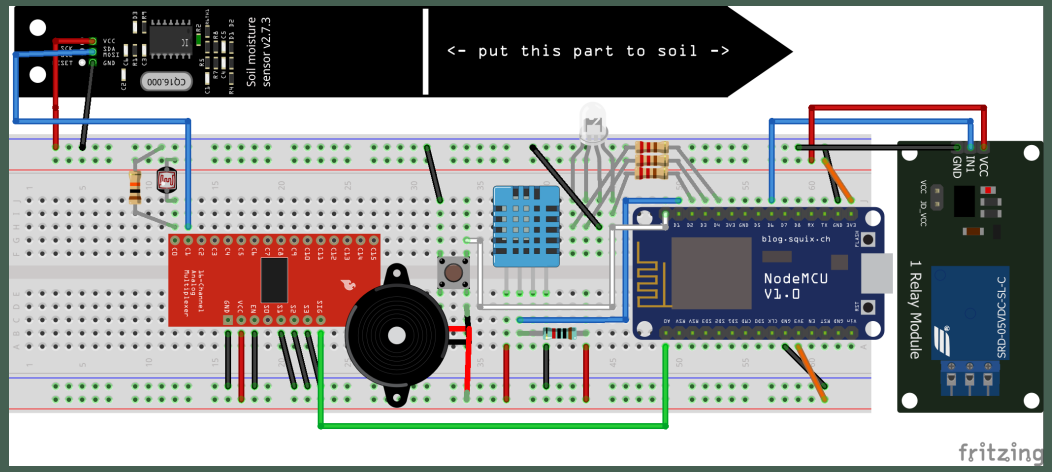



**SM1:**
- Led module
- DHT11 module
- BTN module
- light module

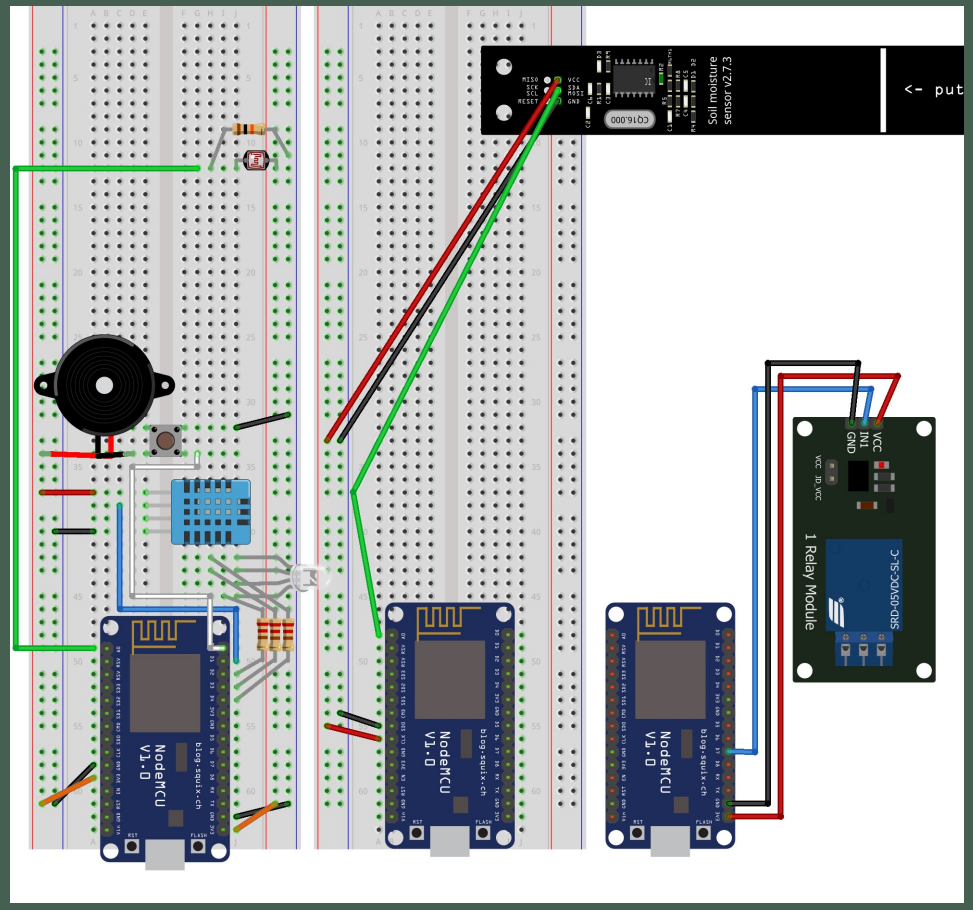
**SM2:**
- Moisture module

**SM3:**
- Water module

# Previous schema




1st assignment schema




2nd assignment schema

# SENSORS & MODULES

# Sensor types

## Read Only Sensor

- Sensor value logged every 5 seconds and pushed into the appropriate topic.
- Reads from read only sensor can be disabled.

- **Photoresistor**
- **Weather sensor (Weather API)**
- **Moisture sensor**
- **DHT11**
- **Water tank level**

## Input Sensor

- Sensor value logged every times an event occurs (e.g. button press).
- Reads from input sensor cannot be disabled.

- **State button**
- **Telegram bot**

# Modules

One node can contain multiple modules. List of available modules:

- **Button module:** handle the button and change the system status.
- **DHT11 module:** handle the DHT11 sensor.
- **Led module**: handle the system led that describes the system status.
- **Moisture module**: handle the moisture module.
- **Photoresistor module**: handle the photoresistor sensor.
- **Telegram module**: handle the interaction with the telegram bot.
- **Water engine module**: handle the water pump and virtualize water tank level sensor.
- **Weather module**: get the weather data from open weather.

**Undererlined** modules can enter into deep sleep mode if meet certain conditions.

# Node code example

**SM1:**
- Led module
- DHT11 module
- BTN module
- light module

**SM2:**
- Moisture module

**SM3:**
- Water module

src > systems > main1.cpp > loop()

```cpp
#include "sm_core.h"

SMCore smCore;

SM_DHT11 m_dht(D1);

SM_Photoresistor m_light(A0);
SM_StatusBtn m_btn(D0);
SM_StatusLed m_led(D2, D3, D4);


void setup() {
    smCore.coreSetup();
}

void loop() {
    smCore.coreLoop();
}
```

src > systems > main2.cpp > setup()

```cpp
#include "sm_core.h"

SMCore smCore;


SM_SoilMoisture m_moisture(A0);




void setup() {
    smCore.coreSetup();
}

void loop() {
    smCore.coreLoop();
}
```

src > systems > main3.cpp > ...

```cpp
#include "sm_core.h"

SMCore smCore;









SM_WaterEngine m_water(D7);

void setup() {
    smCore.coreSetup();
}

void loop() {
    smCore.coreLoop();
}
```

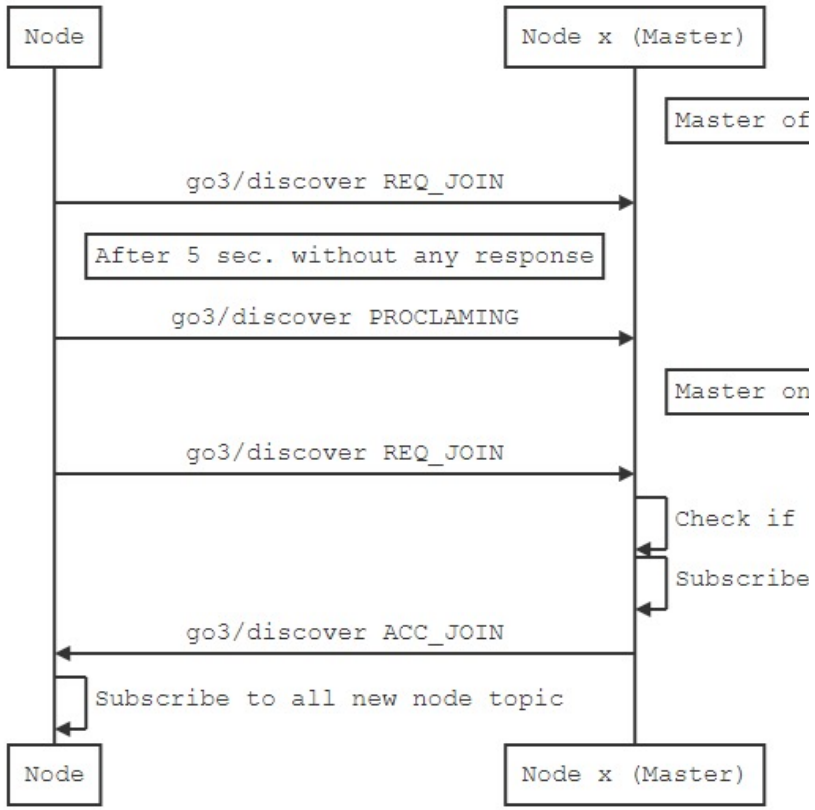
# SMARTGARDEN NETWORK PROTOCOL

# Network protocol

Roles of the nodes are not hard coded inside the code. Each node can be either a master node, with a procedure called **PROCLAIMING.**

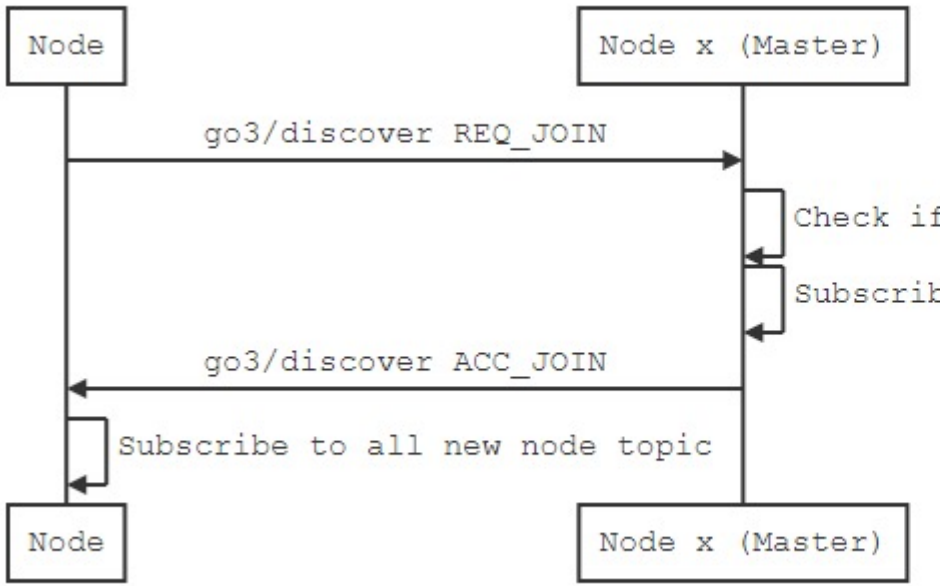
When a node joins into the smartgarden net, publish an MQTT packet into the discover topic (go3/discover) and if it doesn't receive any reply from a master node, proclaims himself as master node. After that, the node has both a slave client and a master client.

The slave client, publish an **REQ_JOIN** packed into the discover topic and when it's accepted from the master with an **ACC_JOIN** packet publish (if sensor's reading are enabled) their reading into the appropriate topic (given from master when accept the client)



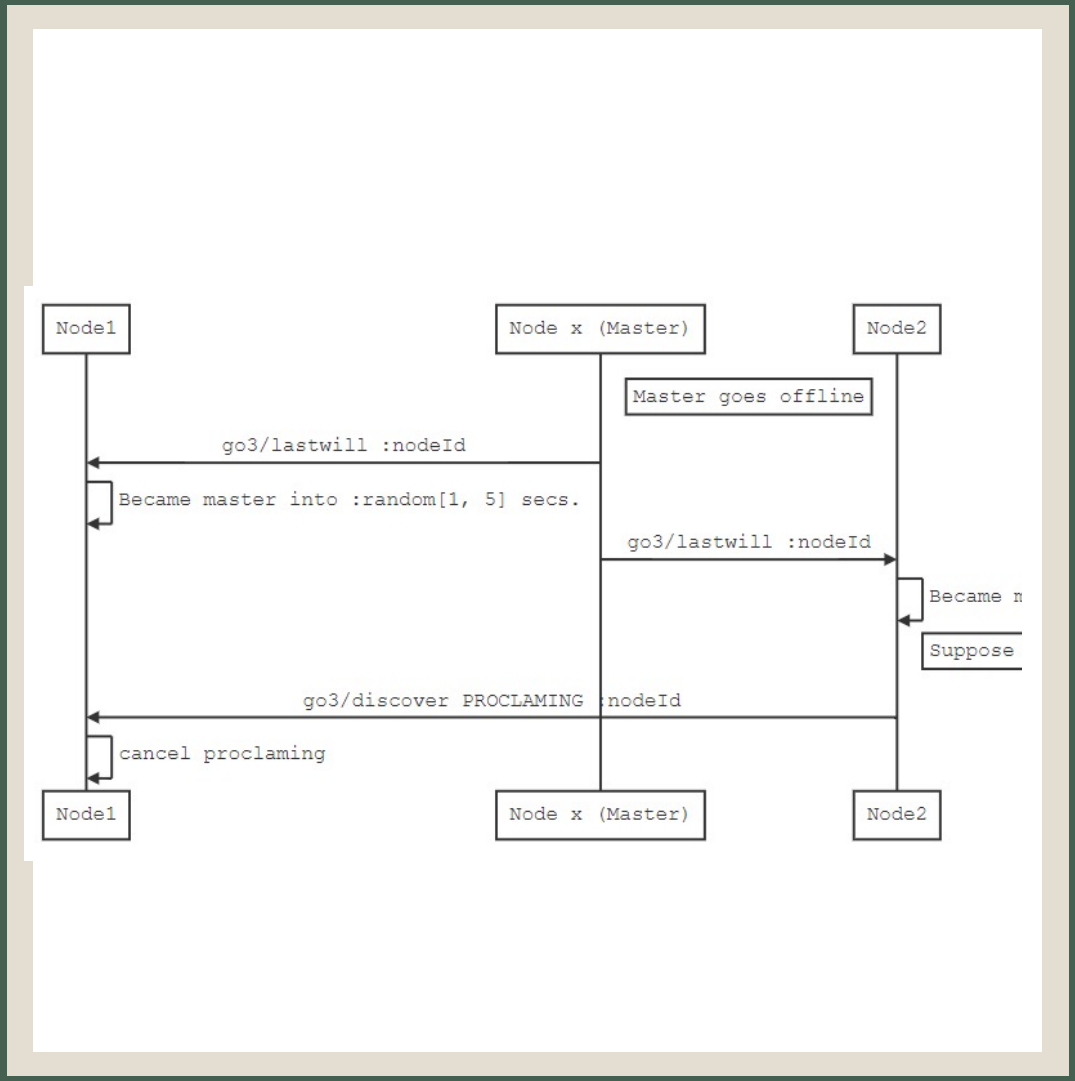

# Network protocol

- Roles of the nodes are not hard coded inside the code. Each node can be a master node, with a procedure called **PROCLAIMING.**
- When a node joins into the smartgarden net, publish an MQTT packet into the discover topic (go3/discover) and if it doesn't receive any reply from a master node (10s), proclaims himself as master node. After that the node can be considered both slave and master node.
- The proclaiming procedure can start if the node has a MASTER NODE flag defined.




# Network protocol

The slave client, publish an **REQ_JOIN** packed into the discover topic and when it's accepted from the master with an **ACC_JOIN** packet publish (if sensor's reading are enabled) their reading into the appropriate topic (given from master when accept the client)

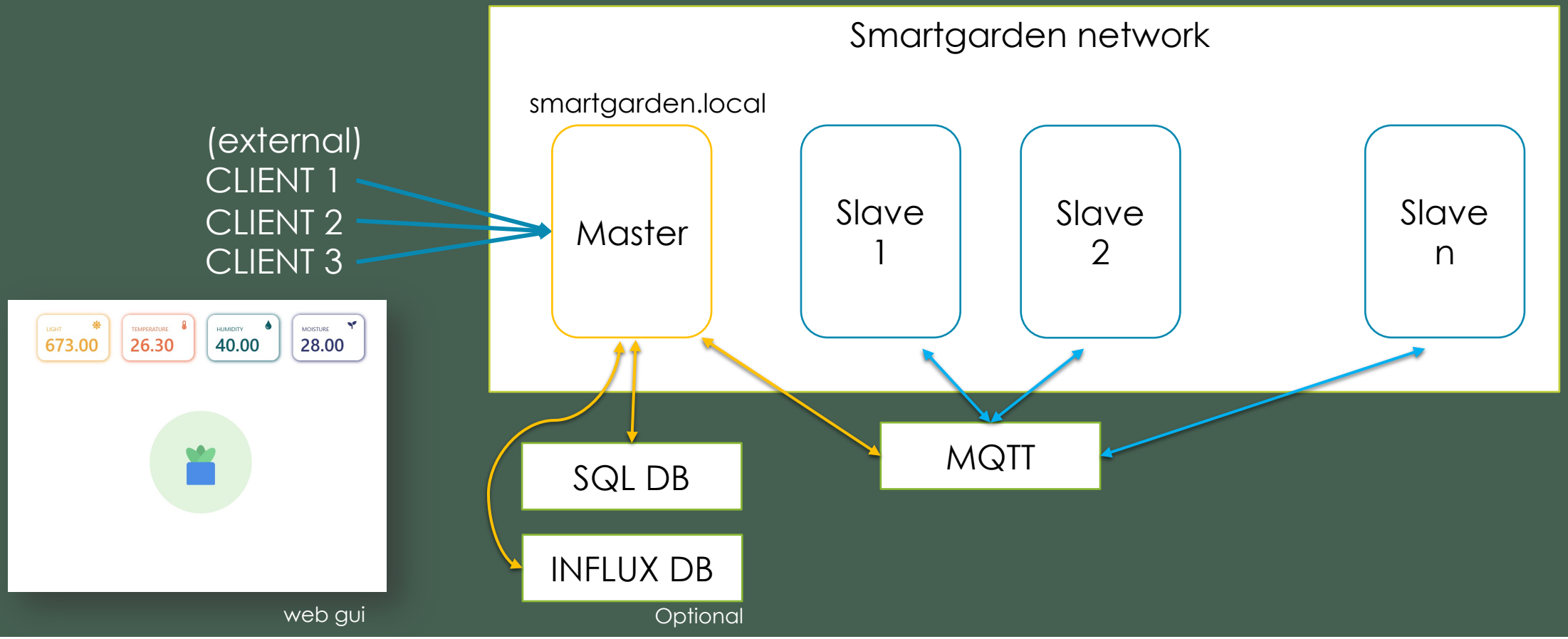


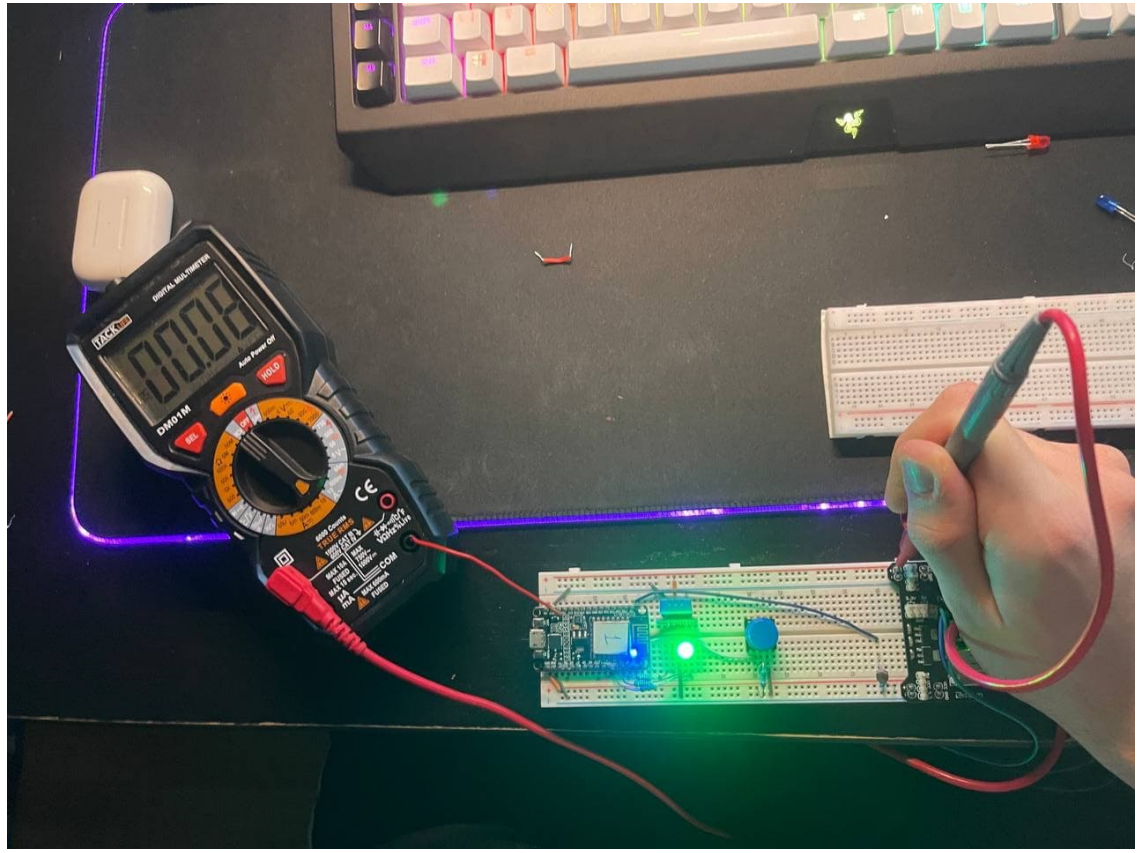

# Network protocol

If the master goes offline all the slaves receive a message into the last will topic, sets a random timer and who consume it before others, proclaims himself as master.

This election mechanisms starts only in the nodes that have the MASTER NODE flag defined.

# Network


Smartgarden network
smartgarden.local
(external)
CLIENT 1
CLIENT 2
CLIENT 3
Master
Slave 1
Slave 2
Slave n
LIGHT
673.00
TEMPERATURE
26.30
HUMIDITY
40.00
MOISTURE
28.00
web gui
SQL DB
INFLUX DB
Optional
MQTT



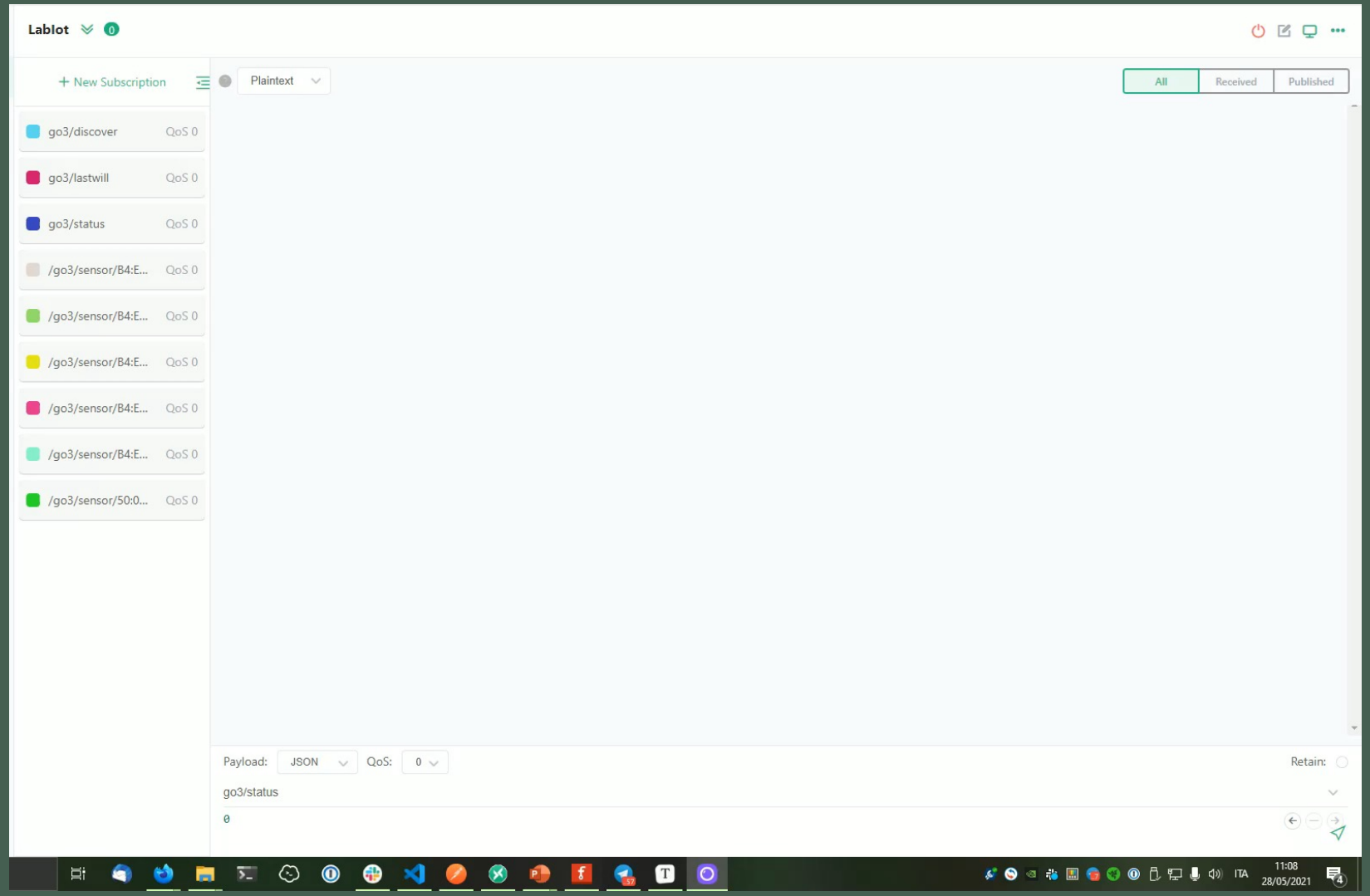

# NODE POWER CONSUMPTION

- All those measurements are taken with a digital multimeter.

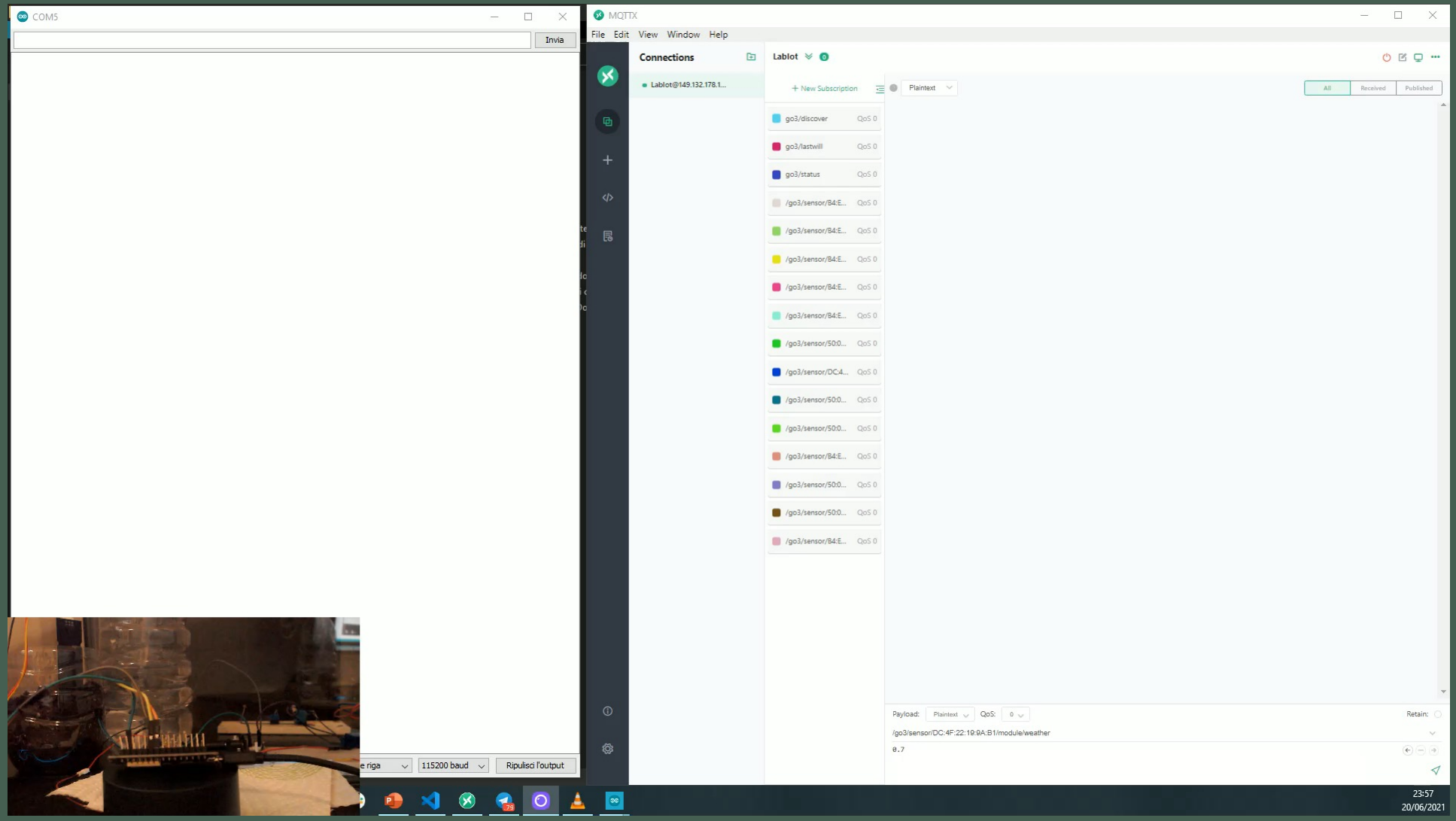
| Node type | Durability with a 18650 battery (3500 mAh) |
|---|---|
| Master node without modules | 44h |
| Slave node no deep sleep | 58h |
| Slave node with deep sleep (once every 8h) | 4177h = 172d |

# SYSTEM IN ACTION

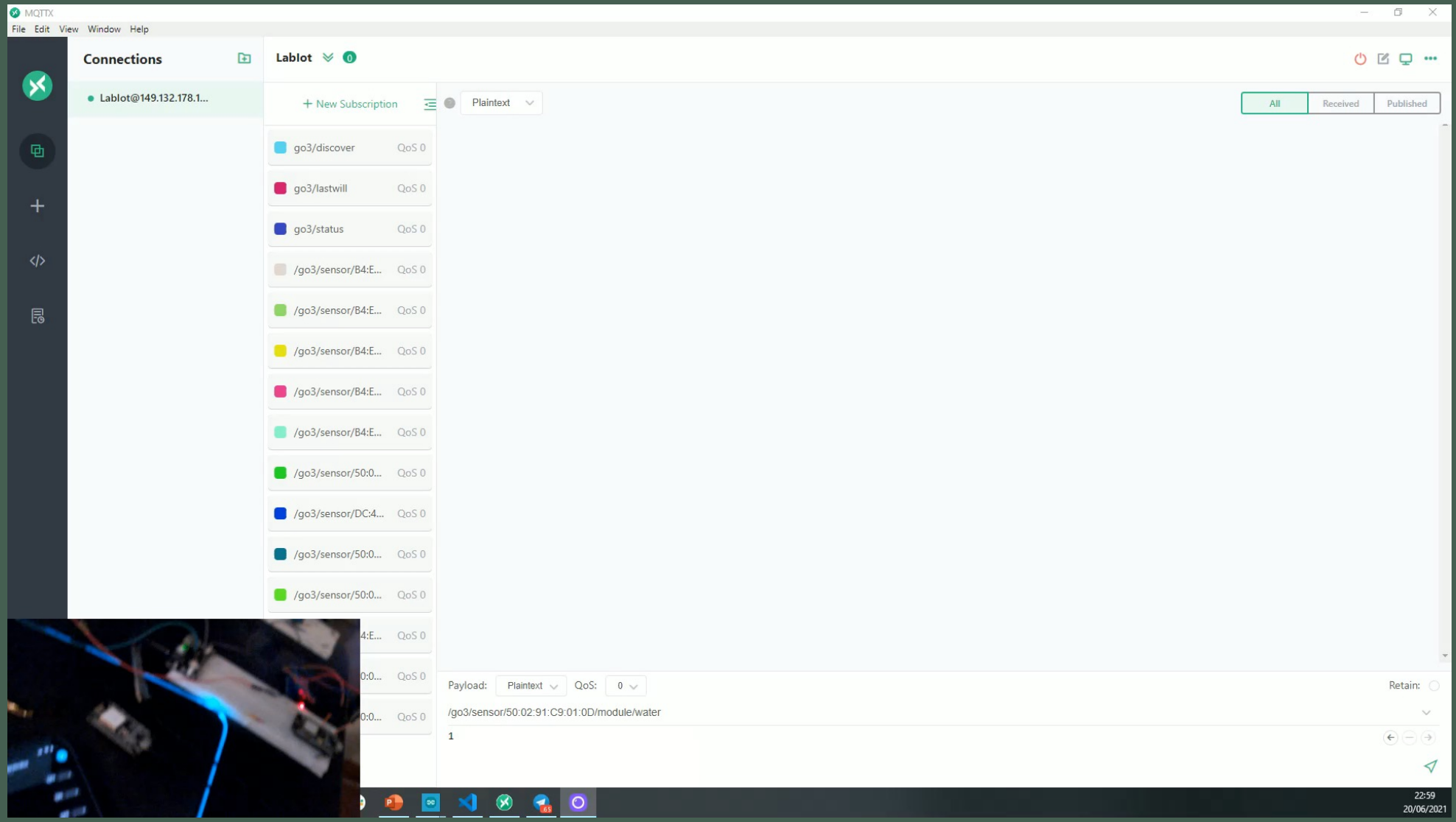
# NETWORK PROTOCOL



# SMART RAIN FEATURE



# TELEGRAM MODULE CONTROL



# A brief recap

- Smart Garden is a system compound by different nodes, each one with the needed modules. All the nodes communicate through a specific designed protocol constructed over MQTT.
- The current master node has a webserver reachable at “smartgarden.local” which provides:
  - /: homepage showing sensor values and system reading stats.
  - /stats: return a JSON with all sensors values (light, temp, hum, … )
  - /config: return a JSON with all master configs (sensorUpdateInterval, systemStastus, moistureTreshold, …)
    - All the config can also be set with a POST request, and also they will be immediately saved into the SQL database.
- The master node can optionally use an SQL DB for storing network topology, configs and alarms and it also can optionally use INFLUX DB for storing the sensor values.
- The system is almost always available at “smartgarden.local” because if the master node goes down, a random eligible slave node became the new master node.