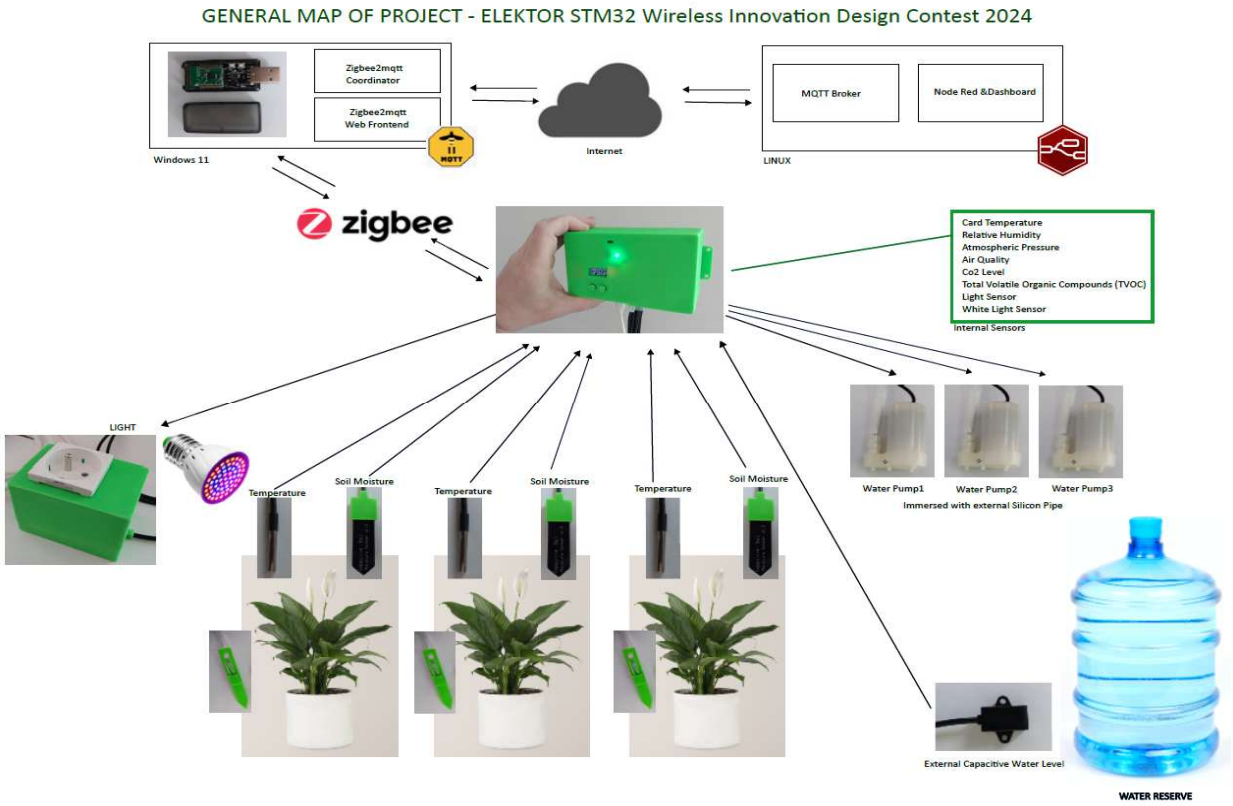
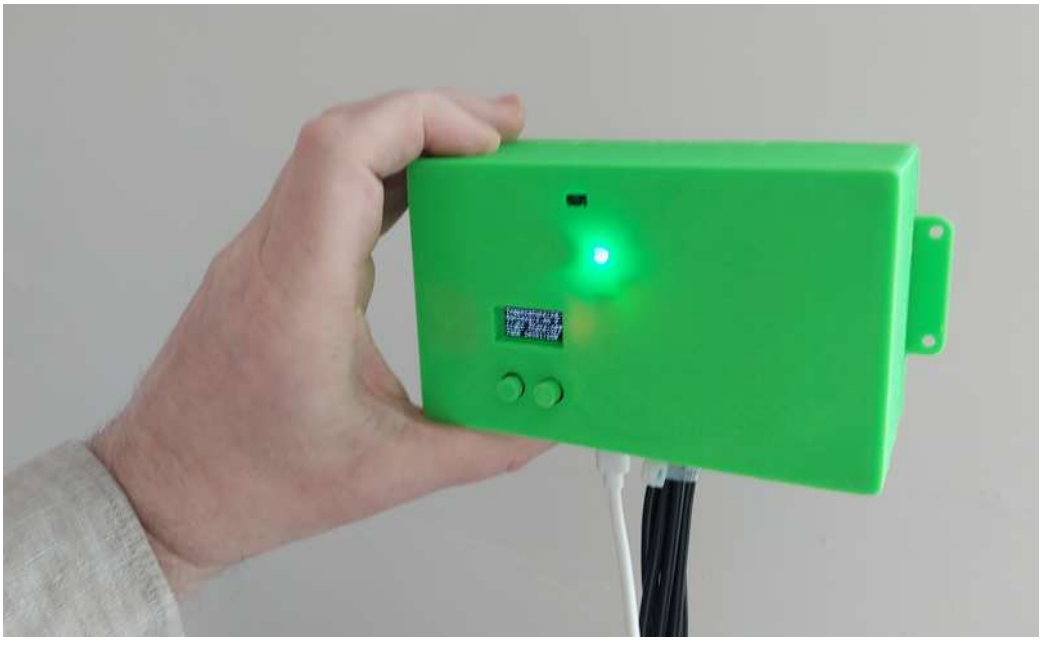
PROJECT FOR ELEKTOR STM32 Wireless Innovation Design Contest

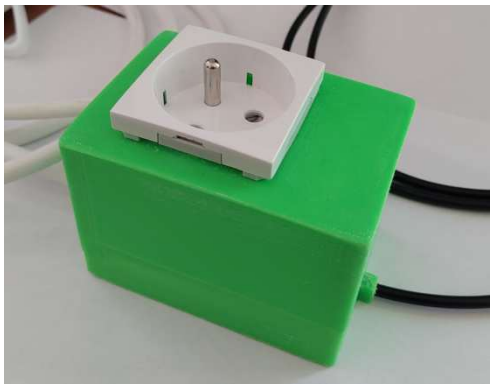
Environmental measurement center, lighting and water supply for indoor plants or greenhouse

Alain ROMASZEWSKI update 24/02/2024

The goal of this project is to create with stm32wb5mmg-dk card a device allowing environmental measurements as well as monitoring of soil humidity and soil temperature of plantations with the management of an automation allowing automated watering and additional illumination according to the measurements.







Project description

The objective of this project is to create an environmental measurement unit, lighting and water supply for indoor plants or in a greenhouse using the stm32wb5mm-dk kit with communication in the Zigbee protocol to a server Mqtt linux on the internet through an Mqtt2Zigbee gateway under Windows 11, the information and automations then being processed by the Node-RED application under Linux with web user interface.
The persistence of data is performed by Node-RED to a MariaDB database under Linux.

A box has been designed and printed on a 3D printer to accommodate the stm32wb5mm-dk card which will be fixed with 3 screws, the main diode is visible through the top case and a window is available to view the Oled display. The two buttons are mechanically transferred to the exterior. The two USB sockets are available and a way is provided for all the wiring. The sensors are fixed on the upper part, each with a receptacle and a window to the outside.

The device must carry out measurements of ambient temperature, ambient humidity, atmospheric pressure (BME280 component), Co2 level, air quality and volatile organic compound or TVOC level. (Ens160 component). An illumination sensor is present to measure the available light. Finally, a capacitive water level sensor is used and fixed to the low level of the water reserve.

Some measurements are taken at each plant using a capacitive sensor and a temperature probe (DS18B20 component). Capacitive sensors are measured with a 16-bit converter (ADS1115 component).

Most sensors can be used via the board's available I2C1 bus. One port is used to create a 1-wire bus for the 3 DS18B20s.

A switching board equipped with power MOSFETs is responsible for relaying the ports of the main board in order to manage 3 watering pumps and an external static relay for switching a special lamp for plants.

A switching power supply module provides complementary 3v3 power from the 5v source and powers the Shield. This card can also charge a battery and provide 3v3 if 5v is no more present.

The 3 pumps operating from 3v to 5v with brushless motors are immersed in the water reserve and a silicone hose supplies each plant.

In the event of lack of humidity in a plant, the planned automation takes care of supplying water with a given watering time until the desired humidity level is reached.

Phone SMS sending from node-RED is planned when the ambient or plant temperature is too high, when the air quality is too poor or when the water level is too low.

Datas are available and accessible via the Zigbee network, the card behaves as a server and as a router.

Datas are read and refreshed locally every 500ms.

The data can be viewed on the Oled display by selecting different screens using buttons 1 and 2.

To force the Zigbee pairing of the card at startup, button 1 must be pressed for 3 seconds.

The available and modifiable data for the lamp and the operating time of the pumps are available to be used by a Zigbee2mqtt coordinator, this coordinator is responsible for retrieving/modifying the information and retransmitting/monitoring topics (path to each data) on a mqtt server.

By using Node-RED specialized in retrieving and modifying Topics, it is possible with a graphical editor to display, storage in dataBase and make some automations.

This project uses different skills: digital electronics, printed circuit design, 3D printing creation, microcontroller programming, internet networks, Mqtt server, Node-RED server, Mariadb database, Linux system.

My objectives are as follows:

- **The first objective** which constitutes this project is the commissioning of the stm32wb5mm-dk kit with the use of the sensors on the card and the use of additional sensors connected to a daughter card (shield) made on both sides. The design of a specific box printed in ABS or ASA, on a 3D printer. The card works in Router/CLUSTER client-server mode. Interface to Zigbee2mqtt, Node-RED interface.

- **The second objective** is the creation of a printed card (4 layers) using the integrated component stm32wb55mmg with its shield (daughter card) and a power supply circuit with battery and solar panel, allowing the exploitation of the low power mode of the component .

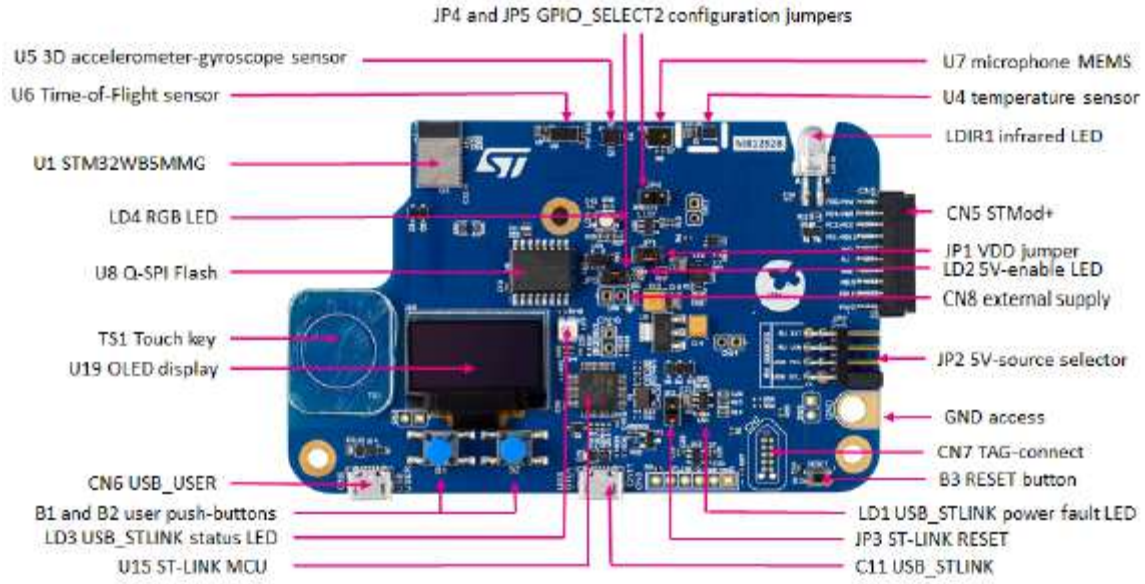
Resumption of certain sensors from the first part.

Add water quality TDS sensor. Add high level sensor and automatic supply of the water reserve.

- **The third objective** is to use and evaluate the Thread network in place of Zigbee.

Project Hardware

Motherboard: stm32wb5mm-dk.



This card consists of an integrated stm32wb55mmg component equipped with an M4 processor for processing and an M0+ coprocessor for BLUETOOTH, Zigbee and Thread communications.

We will deal here with Zigbee 3.0 mode. The card includes a OLED display of 128x64 pixels, 1 RGB diode, 2 push buttons, 1 sensitive key, 1 temperature sensor, 1 Gyroscopic and velocity sensor, 1 distance sensor, movements and a microphone.

The display is connected to the SPI interface and the sensors to the I2C3 bus. The card exposes different ports, an I2C1 bus, a Low Power UART and SPI bus among others.

Modules and sensors already present and used on the main card:

An OLED display for messages and displays.
A diode for signaling measurements taken.
First button to associate the server with the Zigbee coordinator when booting the system.
An STS22H temperature sensor integrated into the card (allowing you to know the temperature of the case and the card)
A gyroscope and velocity sensor (not necessarily useful here but exposed if necessary)
Two buttons used to scroll through the different displays. The same button can be used when booting the card to associate it with the Zigbee network.

Modules and sensors present on Shield card with Arduino type connectors:

A 16-bit (4-channel) ADS1115 type converter module. Allowing the measurement of battery voltage, the measurement of 3 soil humidity sensors.

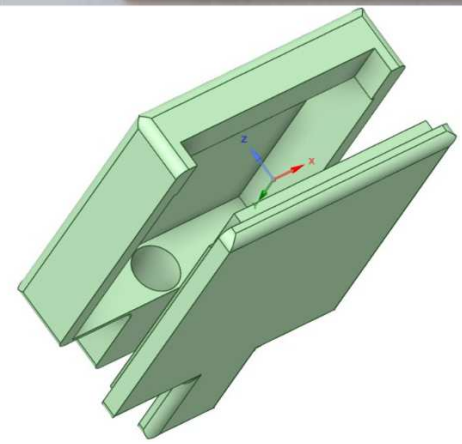
A BME280 type temperature, humidity and atmospheric pressure measurement module.

An ENS160 pollution and air quality measurement module, QA, Tvoc, Co2 measurement associated with an Aht25 sensor measuring temperature and hygrometry for the calibration of the ENS160 module.

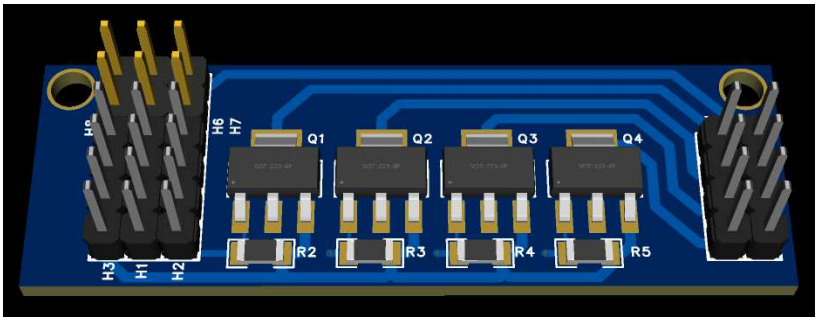
A VEML7700 type brightness measurement module.

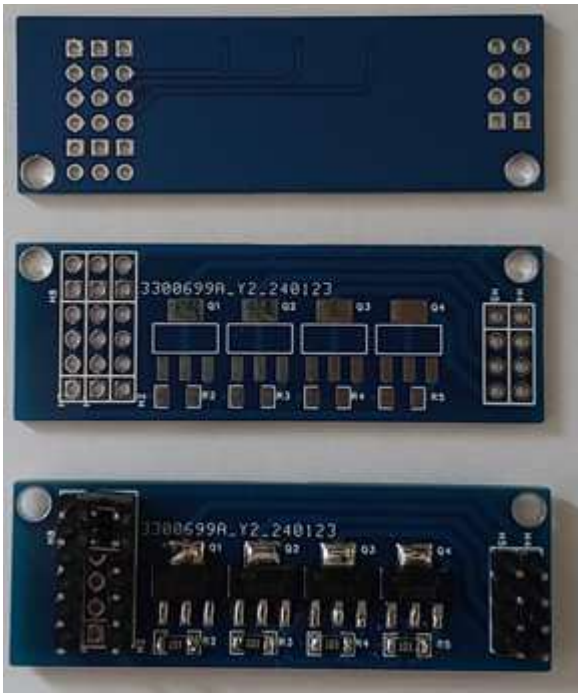
DS18B20 type remote temperature sensors encapsulated in stainless steel.

3 capacitive sensors type modules for soil hygrometric measurement modified to operate in 3v3 (removal of the voltage regulator) with printed box and silicon components isolation.



A power control module for activating 3 watering pumps. Either from 3v3/5v immersed brushless pumps and activating Relay for Light.





External box with Static Relay

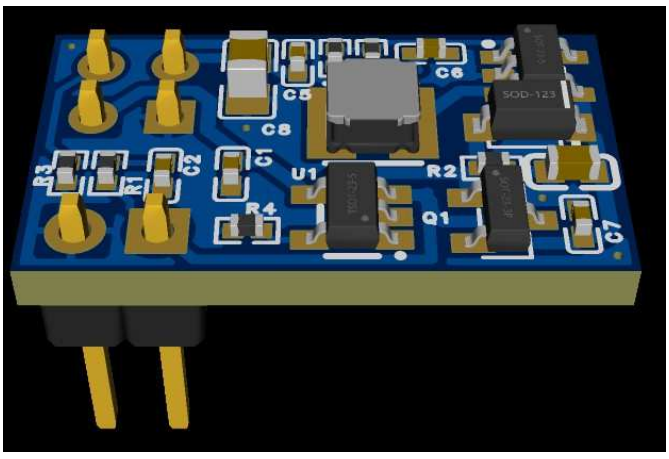
Schematic of connections in file joined.

Guide for pipe and pomp

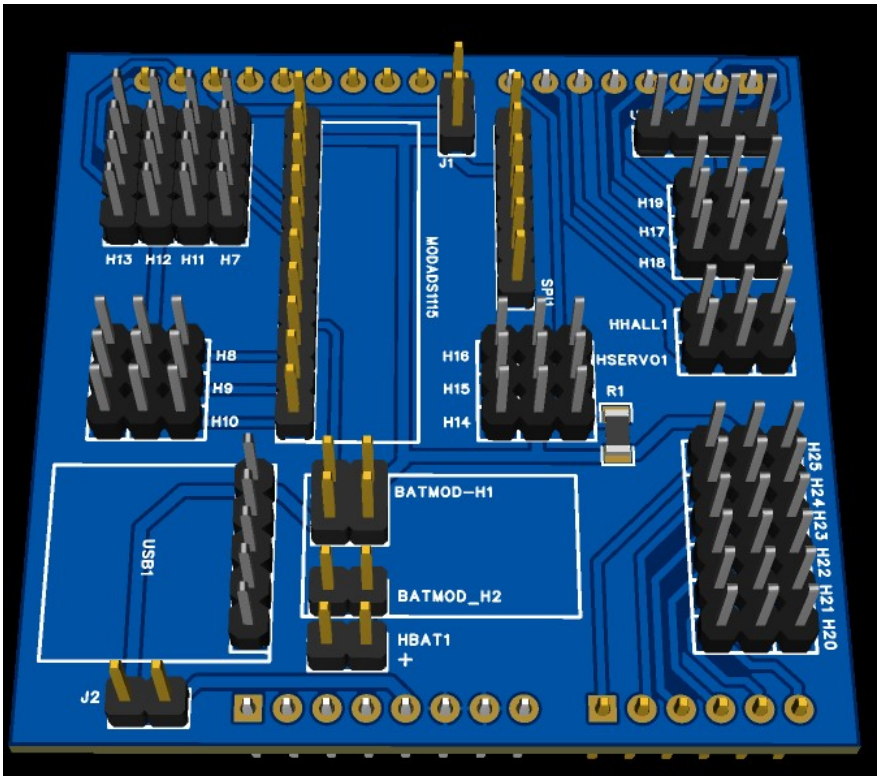
A capacitive module 3v3 detector for checking the low level of the water reserve.

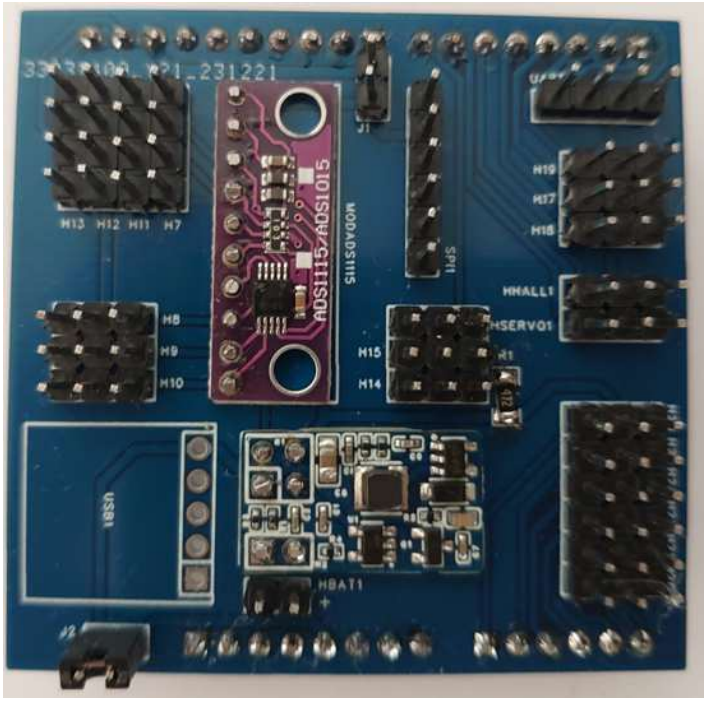
you need to attach the sensor using a double-sided adhesive to the lowest level of the tank.

A switching charging and power supply module to manage the charging of an 18650 battery used for pumps power and to provide up to 1A at 3.4 volts for the shield here, the input voltage is 5 volts

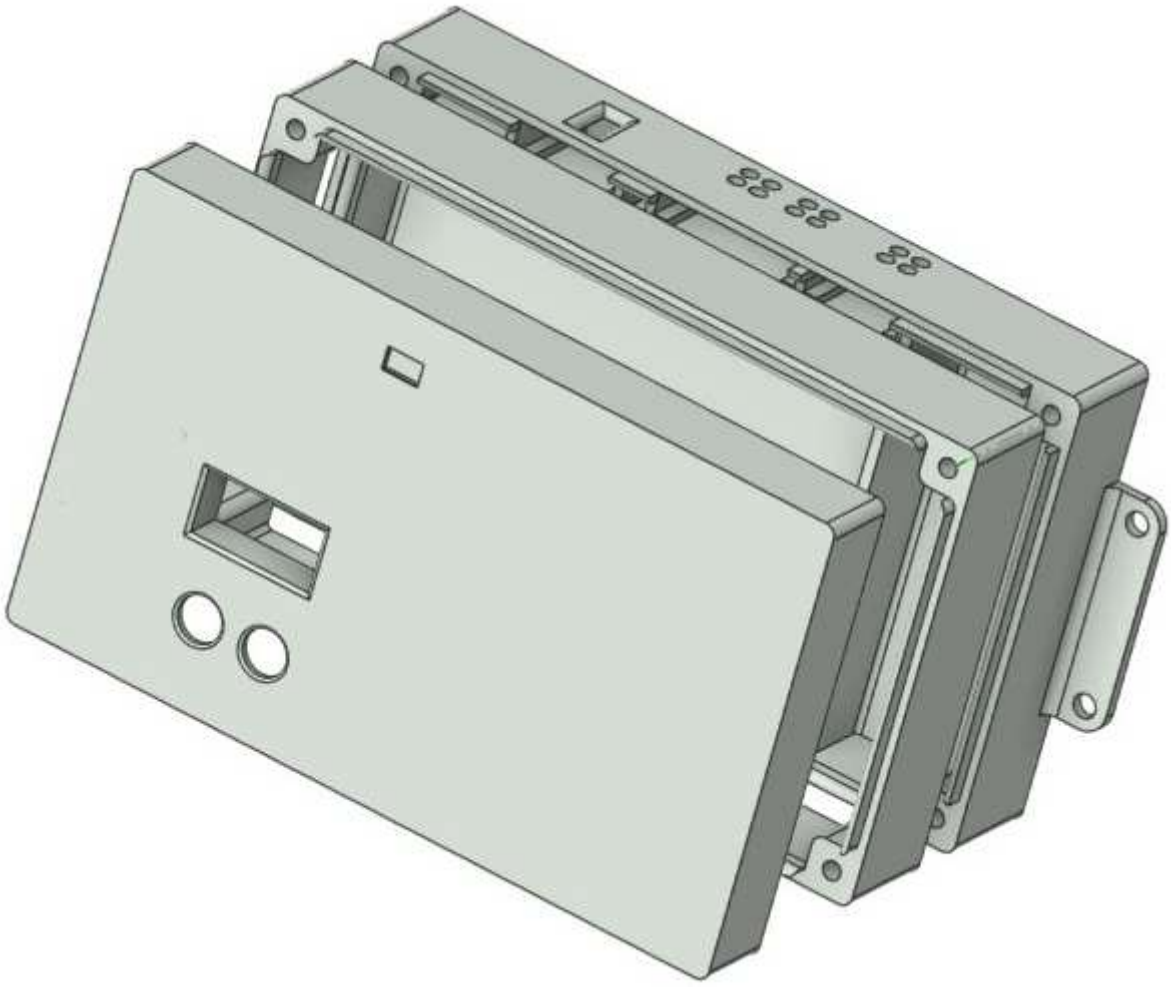


A shield card connected to the Arduino connectors on the stm32wb5mm-dk board.





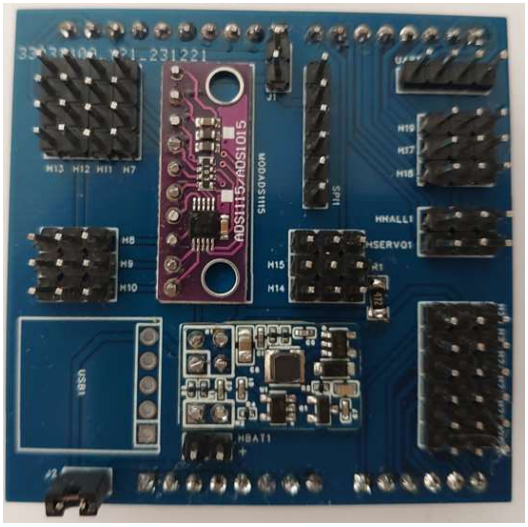
<u>The whole thing is placed in an ABS case printed with a 3D printer.</u>



Front of the printed case

stm32wb5mm-dk card mounted on the front part

Shield assembled



Power part fixed in top

Assembled case with shield and sensors

Bottom case integration of sensors

All I2C sensors modules are connected with default configuration : gnd,vcc,sda,scl pin to pin.

The measurement chain is as follows:

The stm32wb5mmg-dk module and its daughter card exhibit regular measurements as a Zigbee server/router.

The zigbee2mqtt gateway client installed on a Windows 11 machine with a compatible Zigbee USB key is responsible for powering an MQTT event server installed on a remote server via the Internet. The gateway informs about the state of the network and the state of the attributes of the modules that have been associated with this gateway.

The results requested from the sensors are organized in a tree of Topics (access paths) on the MQTT server. MQTT GET requests are used to query the clusters of measurement results present on the remote Zigbee server.

It is also possible to transmit MQTT requests to modify a state or a value commanding an action to the Zigbee Remote client in order to activate, for example, a pump or a valve for a period of time.

Finally, the Node-RED application allows you to manage MQTT events and launch attribute readings from Clusters and launch actions via Mqtt set requests.

In passing, the events are stored in a MARIADB database to ensure the persistence of information and statistical processing of events.

Quick summary of the architecture of a Zigbee Network

The Zigbee protocol works using a mesh architecture and follows a communication model based on the master-slave principle. Here is an explanation of how the Zigbee protocol works:

Mesh Topology : Zigbee uses a mesh network topology, which means that each node (device or device) can communicate directly with other nodes nearby. Each node can act as a transmitter, a receiver and optionally a router.

Zigbee Nodes : There are three main types of Zigbee nodes:

1. **Network coordinator (Zigbee Coordinator)** : It is the main node which initializes and organizes the formation of the network.
2. **Network router (Zigbee Router)** : This node participates in the routing of information in order to facilitate its delivery. It can participate in the Zigbee network as End Nodes.
3. **Zigbee End Devices** : These are simple devices that join the Zigbee network. They are usually battery powered and are designed to minimize their power consumption.

Network Formation : When setting up a Zigbee network, the coordinator first emits a signal inviting other devices to join the network. Interested end nodes communicate with the coordinator to join the network. Once associated, endpoints can transmit data to this coordinator.

Master-Slave Communication: Zigbee uses a master-slave model, where the coordinator acts as the master of the network, and the end nodes are the slaves. End nodes can serve data to the coordinator, they cannot communicate directly with each other in most cases. However, using mesh topology, data can be routed through other nodes in the network to reach its destination with router node.

Security : Zigbee incorporates security mechanisms such as data encryption and device authentication.

Energy management : Zigbee is designed to be energy efficient, especially for battery-powered endpoints. Devices can go into sleep mode to save power and can be woken up when communication is needed.

Quick summary about the Zigbee protocol:

Zigbee is based on the 802.15.4 physical protocol sharing the 2.4Ghertz WiFi frequency band. The band is divided into 16 channels (11 to 26), some channels are vacant and can be more particularly used in our case (15, 20, 25 for example). Of course all devices connected to the main coordinator must use the same channel as this one.

Each Router or Endpoint must first contact its coordinator in order to establish an exchange of keys and create a secure connection (or association). This session/association must be memorized between the Router/Endpoint and the coordinator.

This operation must be done once the network is installed, the coordinator authorizing this association only over a short period of time, each device having a unique 64-bit MAC address, the coordinator is also able to select and secure the association.

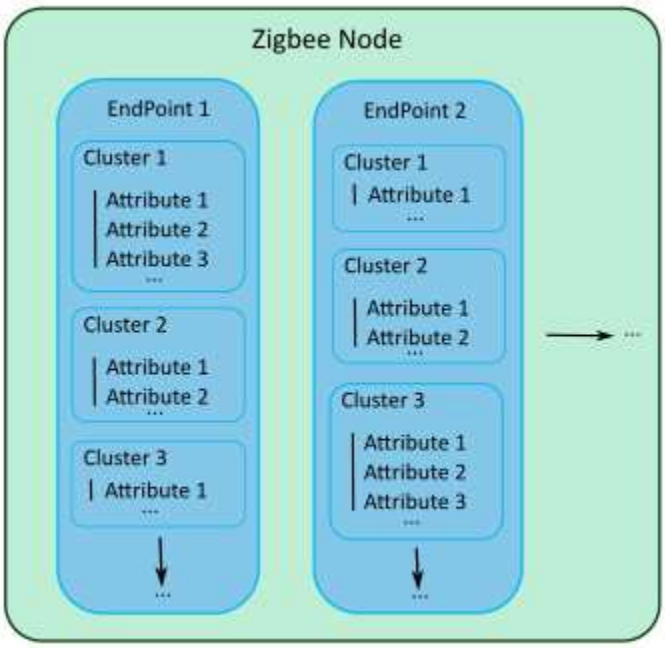
During the association, the coordinator retrieves a unique 64-bit address, allowing the Zigbee Node to be identified.

The Zigbee Node is divided into 1 or more EndPoints (between 1 and 240).

Each Endpoint has one or more Clusters representing a function standardized by the Zigbee or proprietary protocol with default attributes depending on the functions.

For each Cluster, it is therefore necessary to define its main function, its default and/or complementary attributes which can then be translated into a set of values or actions initiated by the coordinator.

Each attribute has a standardized 16-bit identifier, a value with a type (8, 16, 32, 64 bits, strings, enumeration, table, etc.). It can be read, written and/or reported on the network periodically or even persistent (its value is then memorized by the EndPoint). For Alarm or command mode the attribute can be transmitted to the coordinator as soon as the event occurs.



In our case of physical measurements to be taken, we will work as a query server on the attributes, the client coordinator being responsible for regularly querying the attributes in order to retrieve and store the information.

To carry out actions, the coordinator can modify attribute values on the EndPoint, the latter upon receipt will execute the requested tasks, indicating its state by modifying attributes.

At the development level, ST Microelectronics has released a pre-compiled library for Cluster management. This is also accompanied by a Microcode intended for the M0+ communication coprocessor, depending on the final need and the desired protocol.

It is possible to add attributes to an existing Cluster depending on the desired final application.

Microcontroller Software Development Tool

For development, we will use the STM32CubeIDE v1.14.0 tool in C language, based on an existing Zigbee example.

It was necessary to rewrite drivers in C language available in the Arduino universe.

The existing example uses a non-preemptive pseudo system provided as an example by the manufacturer allowing the Zigbee network layer to be managed and tasks to be carried out sequentially by priority.
https://www.st.com/en/development-tools/stm32cubeide.html

Zigbee2mqtt Gateway

This Open Source gateway works with the Node.js framework and is therefore programmed in Javascript. It can be installed under Linux or Windows environment.
It has a Web interface allowing you to manage the nodes and a configuration file allowing you to configure the gateway and recognize new nodes.

It can manage different Zigbee USB adapters on the market, here we will use a Zigbee 3.0 USB Dongle adapter based on Silicon Labs EFR32MG21 with pre-flashed EZSP V8.0.

It is responsible for transmitting the state of the gateway, the state of the nodes and the value of the node attributes to different Topics of an Mqtt server.

https://github.com/ROMASZEWSKI/zigbee2mqtt

https://github.com/ROMASZEWSKI/zigbee2mqtt-frontend

Managing MQTT data with Node-RED

This open source application works with the Node .js framework and is therefore programmed in Javascript. It can be installed under Linux or Windows environment.

It has a Web interface and allows you to process information flows from Mqtt servers, to work with databases (here MariaDB) and finally to generate events with scripts and graphic flows.

https://github.com/node-red/node-red

https://nodered.org/

3D design

3D design is carried out by the free DesignSpark Mechanical 6 software (free version of SpaceClaim).

https://fr.rs-online.com/web/generalDisplay.html?id=designspark/designspark-mechanical

https://www.ansys.com/products/3d-design/ansys-spaceclaim

3D FDM printing

Printing is done with PrusaSlicer 2.6 software on an upgraded ANET ET 4 PRO.

The creations use either ABS filament or ASA filament (UV resistant).

https://www.prusa3d.com/en/page/prusaslicer_424/

https://anet3d.com/pages/et4

Printed Circuit Board Design

The design uses EasyEDA PRO v2.1.42 software and the ultra-fast, low-cost production line that goes with it.

https://easyeda.com/

Design and description of the functions of the shield card with Arduino connectors

This card has a soldered ADS1115 module (Fig.2) and a complementary soldered 5v -> 3.3v 1A switching power supply module with the possibility of battery charging so as not to overuse the power supply of the main map. The battery is not really useful at the moment but useful in subsequent projects.

It includes all the 2.54mm Dupont Male connectors for the various sensors.

Description of the main application

Installing or upgrade First the Firmware for M0+ Co-processor

the STM32WB5x processor works with an M0 co-processor which uses firmware provided by ST. This firmware is available in the STM32CubeWB PACK available here:

the stm32WB5x processor works with an M0 co-processor which uses firmware provided by ST. This firmware is available in the STM32CubeWB PACK available here:

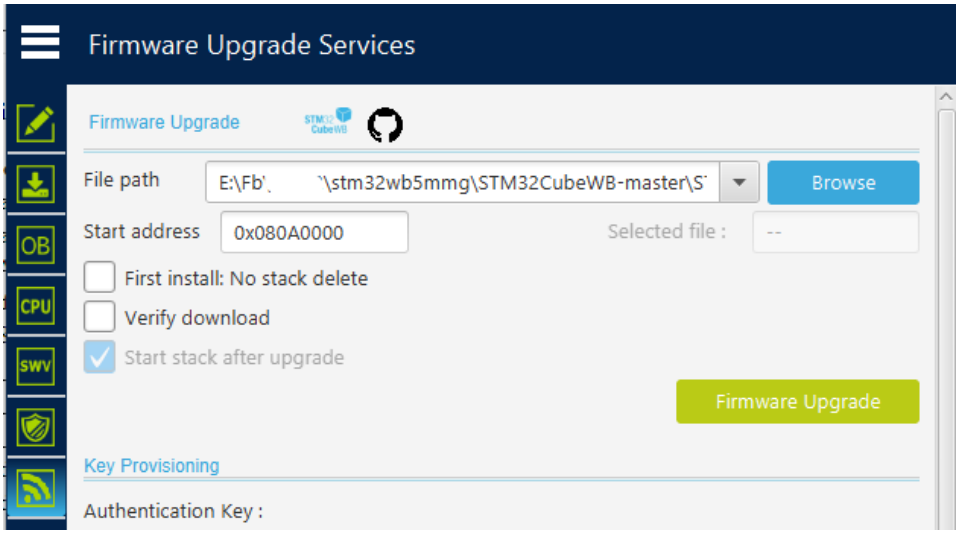
https://github.com/STMicroelectronics/STM32CubeWB/tree/master/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x

We will use the **stm32wb5x_Zigbee_FFD_fw.bin** file in its revision 1.18.

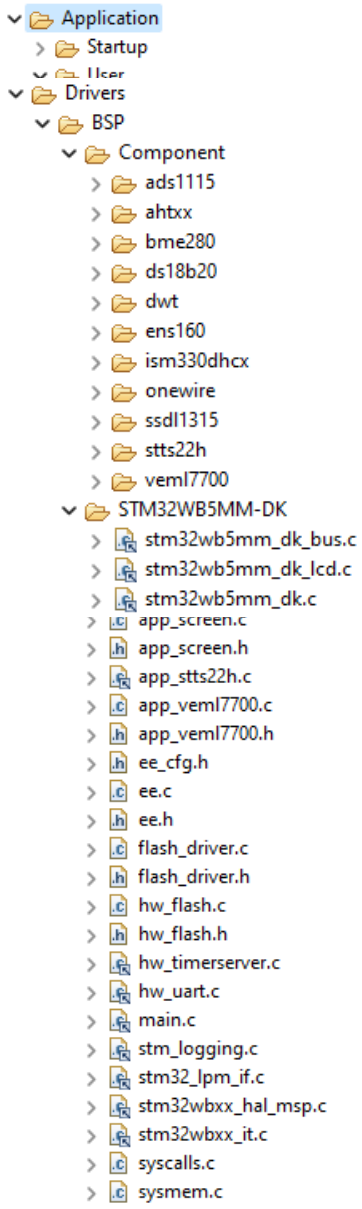
Programming of the coprocessor is carried out via the FUS interface accessible via the **STM32CubeProgrammer** software.

Firmware file : stm32wb5x_Zigbee_FFD_fw.bin

Start Address : 0x080A0000



Description of the Drivers



In this directory we find the .c and .h drivers for the different sensors used:

ads1115: 16-bit AD Converter Driver.
ahtxx: Temperature and humidity sensor driver for calibration of the ENS160
bme280: Temperature, relative humidity and atmospheric pressure sensor driver.
ds18b20: Driver for 1-wire temperature sensors.
dwt: microsecond delay generator driver for onewire.
ens160: CO2, QA and Tvoc sensor driver.
ism330dhcx: onewire gyroscope driver :
1-wire sensor driver
ssdl1315: Oled display driver
sts22h: temperature sensor driver of the
veml7700 card: light sensor driver.
The STM32WB5MM-DK directory which includes the button, RGB LED and Oled display drivers.

Application modules

app_ads1115: High level control of AD converter.

app_ahtxx: High level control of the temperature/humidity sensor of the ENS160
app_automat: Control of automations: Pumps, Light, level.

app_bme280: High level control of the bme280 temperature/humidity/pressure sensor.
app_clusters: definitions and activation of Zigbee clusters.

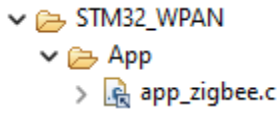
app_ds18b20: High level control of DS18B20 1-wire temperature sensors.
app_ens160: High level control of the pollution sensor: Co2, QA, TVoc.
app_entry: Groups the Zigbee protocol initialization subroutines.
app_ScheduleMeasure: Management of cyclical measurements and updating of Zigbee attributes and management of automations. Display of measurements by APP_DBG. **MAIN PROGRAM**
app_screen: Management of screens displaying the measurement results with selection by button 1 and 2 to scroll.
app_sts22h: High level control of the board temperature sensor.
app_veml7700: High level control of the light sensor.

ee, flash_driver,hw_flash: Set of subroutines used to emulate an eeprom on a Flash memory area of the processor. Used to save Zigbee connection information.
hw_timerserver: Management of timers for delays and the sequencer.

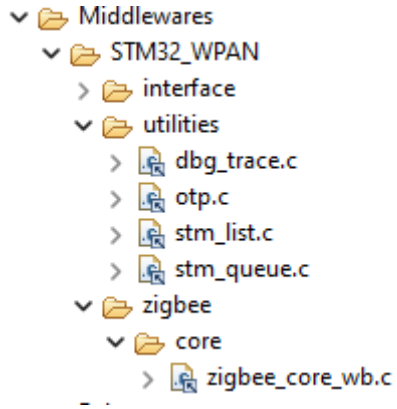
hw_uart: Serial interface management for APP_DBG displays.

main: Management of basic processor initialization.

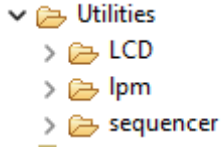
stm_logging: management of debug displays to the serial link.



app_zigbee: this module configures the Zigbee layer, different calls are made to the modules described previously.



The application is based on the STM32_WPAN Middlewares library allowing the use of Zigbee communication.



Finally, the Utilities directory containing high-level management of the Oled display as well as sequencer for the management and synchronization of non-preemptive tasks.

```
18:39:32 **** Incremental Build of configuration Release for project Zigbee_TempMeas_Server_Coord ****
make -j12 all
arm-none-eabi-size  Zigbee_TempMeas_Server_Coord.elf
   text    data     bss     dec     hex filename
 119100    1925  124826  245851   3c05b Zigbee_TempMeas_Server_Coord.elf
Finished building: default.size.stdout


18:39:32 Build Finished. 0 errors, 0 warnings. (took 397ms)
```

How the application works

The application configures on channel 15 Zigbee, an EndPoint 1 server, and a Temperature Measurement type cluster, proprietary attributes are then added to this standardized cluster. This choice was made after trying to create proprietary Cluster but after some trouble with the pre-compiled zcl library, on the other hand the application is stable and simple.

Cyclically, all sensors are interrogated and the results placed in each corresponding attribute of the Cluster, the delay is 20 seconds by default, but can be modified with a cluster attribute (ATTREFRESHMEAS).

The board's temperature sensor is used via the I2C3 interface.
The ADS1115, ENS160, BME280, VEML7700 and AHT25 sensors are used via the I2C1 interface. The soil humidity sensors have been modified to work at 3.3v and are connected to 3 inputs of the ADS1115 A/D converter. The first converter measures the Battery charging voltage.

The 3 DS18B20 1-wire bus type temperature sensors are connected to 1 PORT bit.
The Capacitive water tank level sensor is connected to 1 PORT bit.

The AHT25 temperature/humidity sensor supplied with the ENS160 must be interrogated less often because it tends to heat up, which is a shame for the measurement!

The activation of the pumps is managed by an activation time set in ms by their **PlantWaterAlimentationX** attribute (where X value is 1,2 or 3).

If this attribute is different from 0, the pump is started and the attribute is decremented by time to 0 and the pump is stopped. Each pump is on 1 PORT bit. The ON/OFF pump status is reported on a Cluster attribute **PlantWaterAlimentationStateX**.

If the Light attribute is 1 the light is on. 1 port bit. With the attribute **PlantLight**.

List of attributes

The **app_clusters.h file** lists the attributes used, each attribute has a 16-bit identifier.

```
#define ATTCARDBATT          0x0101
#define ATTREFRESHMEAS       0x0102

#define ATTENVLUM            0x0300
#define ATTENVPRESS          0x0301
#define ATTENVHUM            0x0302
#define ATTENVTEMP           0x0303
#define ATTENVQA             0x0304
#define ATTENVCO2            0x0305
#define ATTENVTVOC           0x0306
#define ATTENVR1             0x0307
#define ATTENVR2             0x0308
#define ATTENVR3             0x0309
#define ATTENVR4             0x030A
#define ATTWHITELUM          0x030B
#define ATTTANKLEVEL         0x030C


#define ATTPLANTTEMP1        0x0310
#define ATTPLANTIDDS1        0x0311
#define ATTPLANTSHUM1        0x0312
#define ATTPLANTWALIM1       0x0313
#define ATTPLANTWALIMSTATE1  0x0314

#define ATTPLANTTEMP2        0x0320
#define ATTPLANTIDDS2        0x0321
#define ATTPLANTSHUM2        0x0322
#define ATTPLANTWALIM2       0x0323
#define ATTPLANTWALIMSTATE2  0x0324

#define ATTPLANTTEMP3        0x0330
#define ATTPLANTIDDS3        0x0331
#define ATTPLANTSHUM3        0x0332
#define ATTPLANTWALIM3       0x0333
#define ATTPLANTWALIMSTATE3  0x0334
```

The **app_clusters.c file** declares the list of attributes to add to the Cluster.

```
static const struct ZbZclAttrT attr_list_Main[] = {
    {
        ATTCARDBATT,ZCL_DATATYPE_UNSIGNED_16BIT,ZCL_ATTR_FLAG_WRITABLE,0,
        NULL,
        {0, 5000},{0, 0xffff}
    },
    {
        ATTENVTEMP,ZCL_DATATYPE_SIGNED_16BIT,ZCL_ATTR_FLAG_WRITABLE,0,
        NULL,
        {-4000, 14500},{0, 0xffff}
    },
    {
        ATTENVLUM,ZCL_DATATYPE_UNSIGNED_16BIT,ZCL_ATTR_FLAG_WRITABLE,0,
        NULL,
        {0, 65535},{0, 0xffff}
    },
    {
        ATTWHITELUM,ZCL_DATATYPE_UNSIGNED_16BIT,ZCL_ATTR_FLAG_WRITABLE,0,
        NULL,
        {0, 65535},{0, 0xffff}
    },
```

Each attribute must be declared with its type, its minimum and maximum value, as well as its cyclical reportability or modification to the network.

The temperature in °C * 100 is a signed 16-bit integer.
The atmospheric pressure in hPa *10 is an unsigned 16-bit integer, not corrected with the altitude value.
The relative humidity and soil humidity in % *10 is an unsigned 16-bit integer.
The battery voltage in mV is an unsigned 16-bit integer. The Co2 in ppm is an unsigned 16-bit integer.
The Tvoc in pbm is an unsigned 16-bit integer.
The values in ohms of resistors R1, R2, R3 and R4 for gas detection are unsigned 32-bit integers.
The air quality is an unsigned 8-bit integer.
The motor and light on/off control is an unsigned 8-bit integer.
The water supply control is an unsigned 16-bit integer representing the time in ms.
The on/off pump state On/Off is an unsigned 8-bit integer.
The air quality sensor is an unsigned 8-bit integer.
The on/off tank level is an unsigned 8-bit integer.
The identifier DS18B20 is an unsigned 64-bit integer.

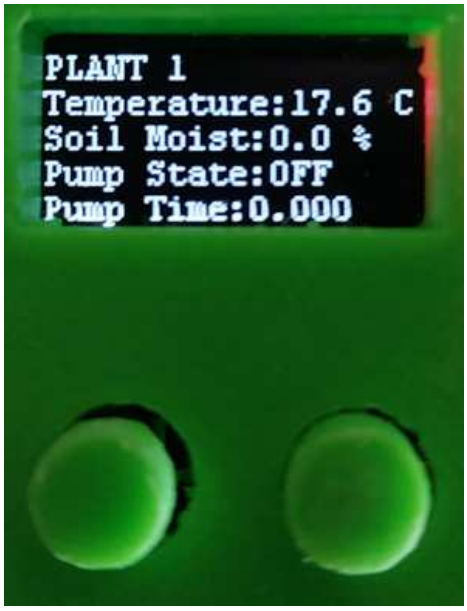
In order not to consume too much current on the card ports, a control module comprising 4 N-channel power MOSFETs makes it possible to control the pump motors and a relay for the lamp.

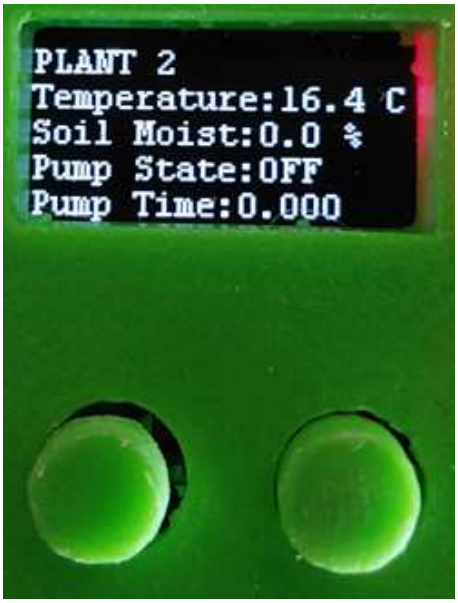
The **app_automat.c file** manages the automation and actions of the pumps, the lamp and the level sensor.

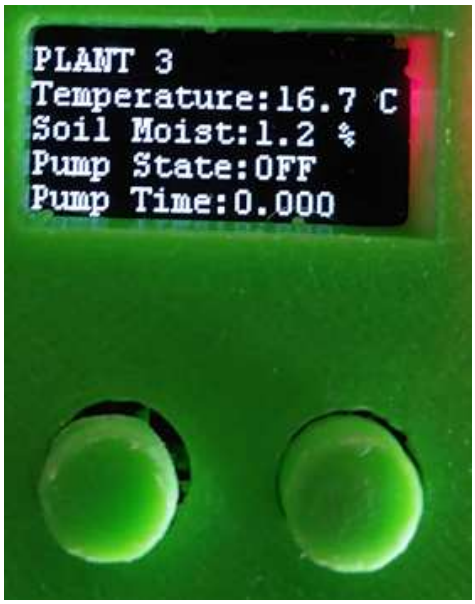
The **app_SheduleMeasure.c file** cyclically retrieves the value of the sensors by placing them in the Cluster attributes, a display is reported on the serial link.

The **app_screen.c file** manages the display of information screens, the different screens can be selected using the two buttons on the card for cyclical selection.

Capture of different screens








DS18B20 ID List
28da1956b5013c1b
286e1256b5013cb0
28e90156b5013ced

Installation and configuration of the Zigbee2Mqtt application

The Zigbee coordinator will be the Zigbee2Mqtt application installed on Windows 11 with a Zigbee 3.0 USB adapter based on Silicon Labs EFR32MG21 with EZSP v8.

This application written in JavaScript with the Node.js framework will offer management of the coordinator via a Web interface and allow you to configure and transmit the state of each attribute of the Cluster to a Topic Set or Get of the remote Mqtt server. So by subscribing to an Mqtt Topic it is possible either to retrieve the value of the attribute or to modify it.

It will be necessary to modify the definition of Clusters in order to be able to correctly visualize the attributes.

measuredValue
minMeasuredValue
maxMeasuredValue
tolerance
minPercentChange
minAbsoluteChange
BatteryVoltage
DelayRefreshMeas
Luminosity
Pressure
Humidity
TemperatureEnv
AirQuality
Co2
Tvoc
R1
R2
R3
R4
WhiteLuminosity

TankLevel
PlantTemperature1
PlantMACDS1
PlantSoilMoisture1
PlantWaterAlimentation1
PlantWaterAlimentationState1
PlantTemperature2
PlantMACDS2
PlantSoilMoisture2
PlantWaterAlimentation2
PlantWaterAlimentationState2
PlantTemperature3
PlantMACDS3
PlantSoilMoisture3
PlantWaterAlimentation3
PlantWaterAlimentationState3
PlantLight

You can discover this part in other documents and files join in zigbeetomqtt zip.

Installing an MQTT server

It is the MOSQUITTO application which will be used to manage an Mqtt server and will be installed on a Debian 12 VPS from OVH with ispconfig auto installation.

Installation and configuration of the Node-RED application

This application written in Javascript and Node.js Framework allows you to connect to the Mqtt server and manage the storage of attributes in a database, as well as to carry out automation and alert management actions. It will be installed on a Debian 12 VPS.

This application has a web interface to graphically manage and schedule events. Another dashbord graphical interface allows you to view information and take action via widgets.

Node-RED have the capacity to request mqtt server and act with Mariadb database.

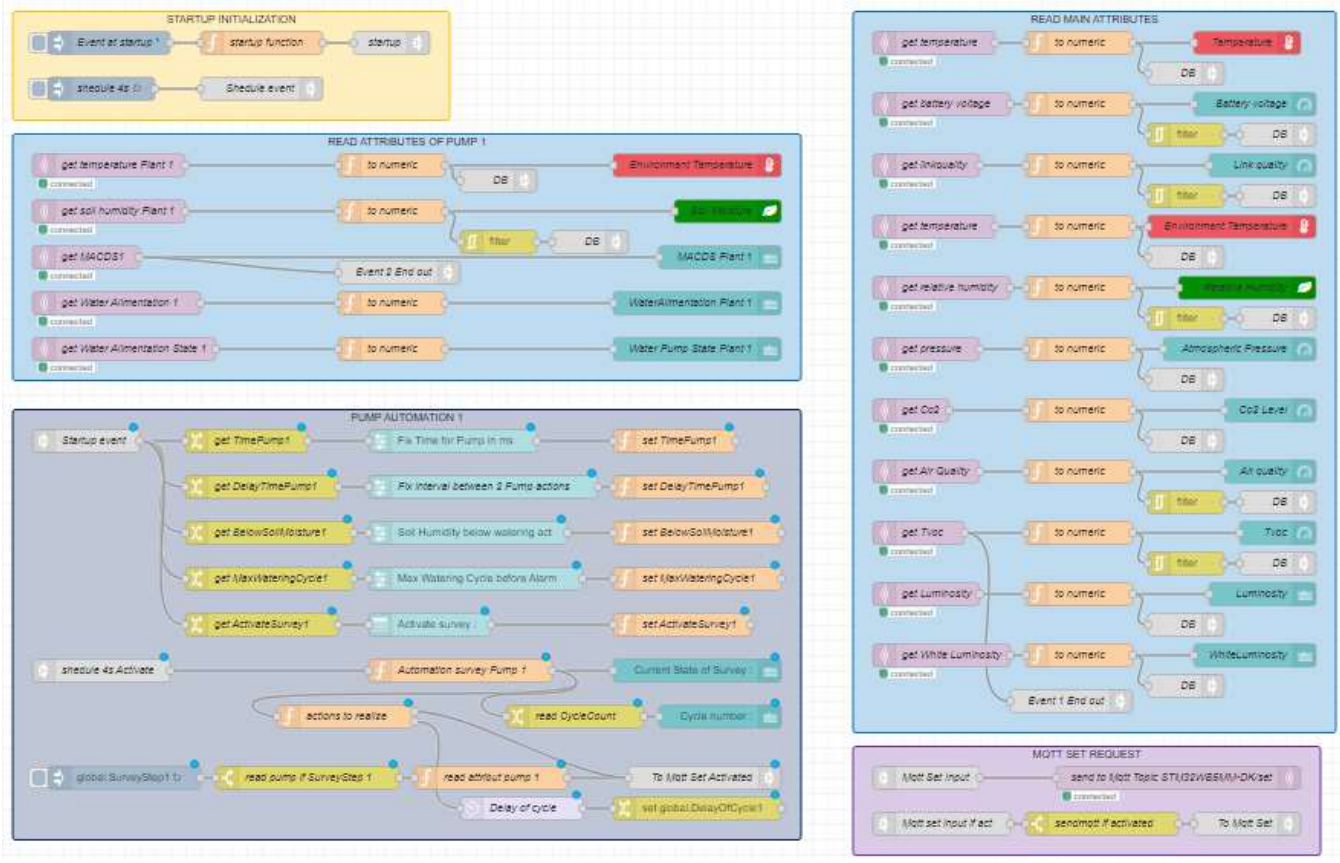
The flows of the project manage a light automation :

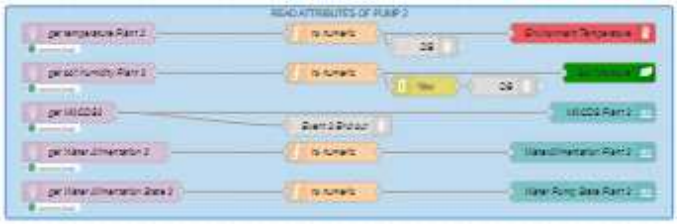
- Manual Light ON/OFF
- ON/OFF if brightness is too low.
- Sheduler with multiple times by week.
- Sheduler + brightness is too low.
- Solar Day/night + On if brightness is too low.

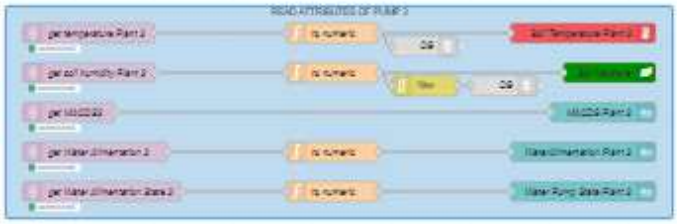
The flows of the project manage Pump automation :

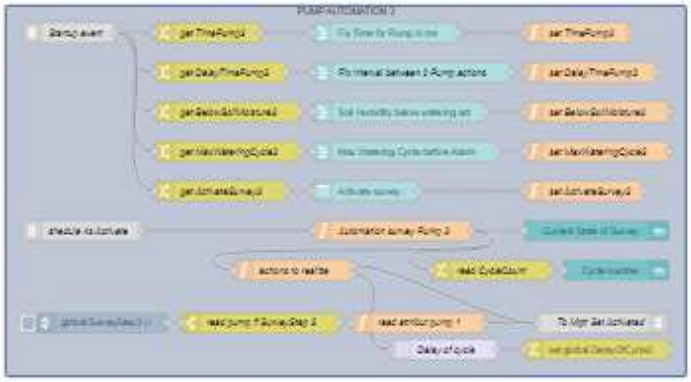
- If the level of soil moisture is too low, it is possible to activate a cycle of watering with some limits and if the water tank level is hight.(pumping delay and wait delay with a number of maximum cycle).

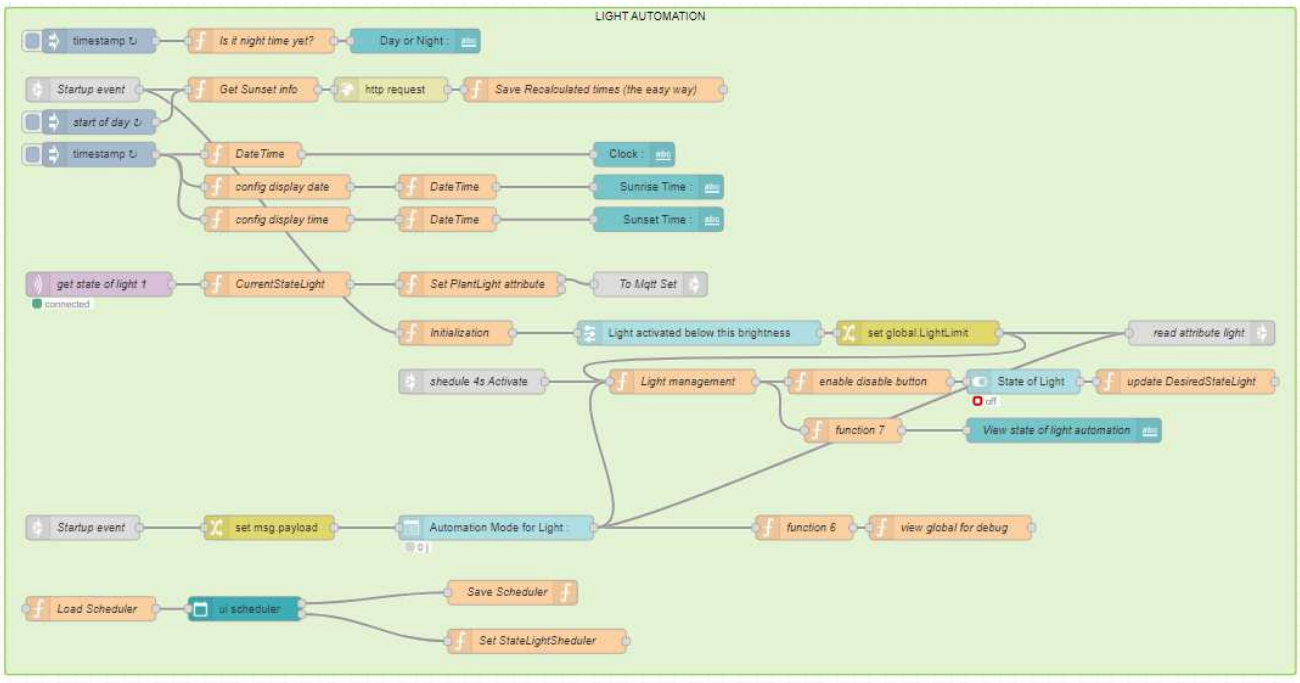
Screens of flows used :

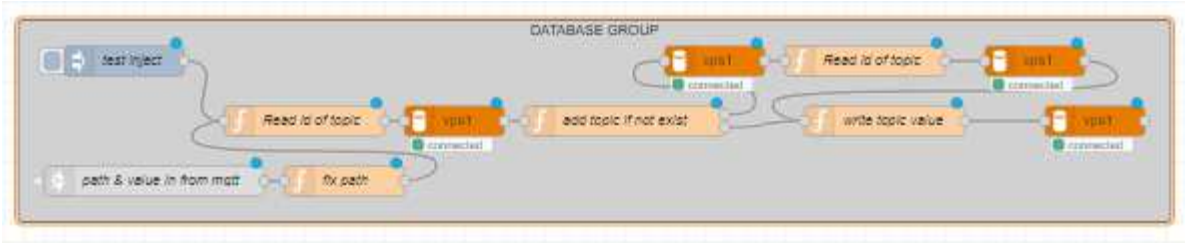






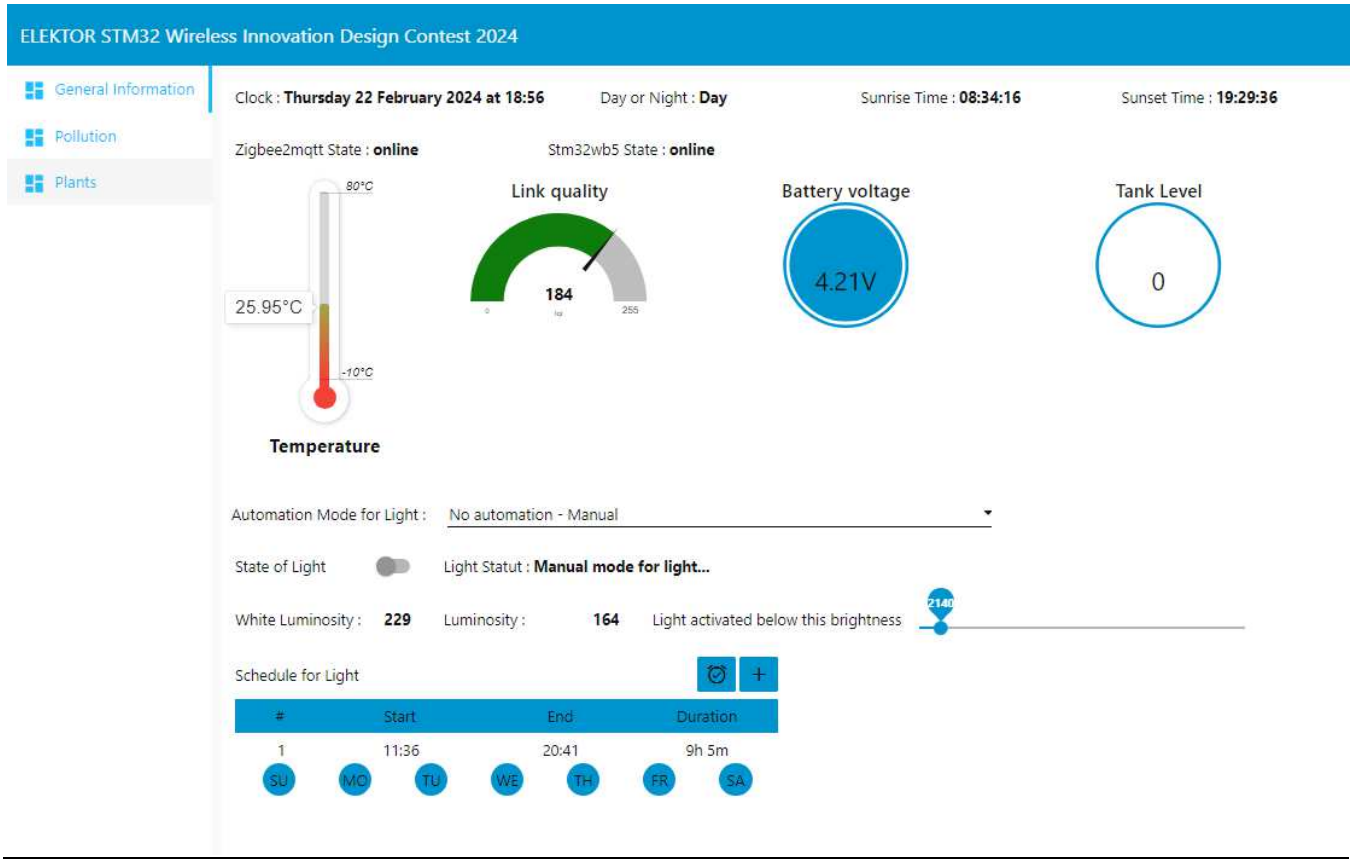


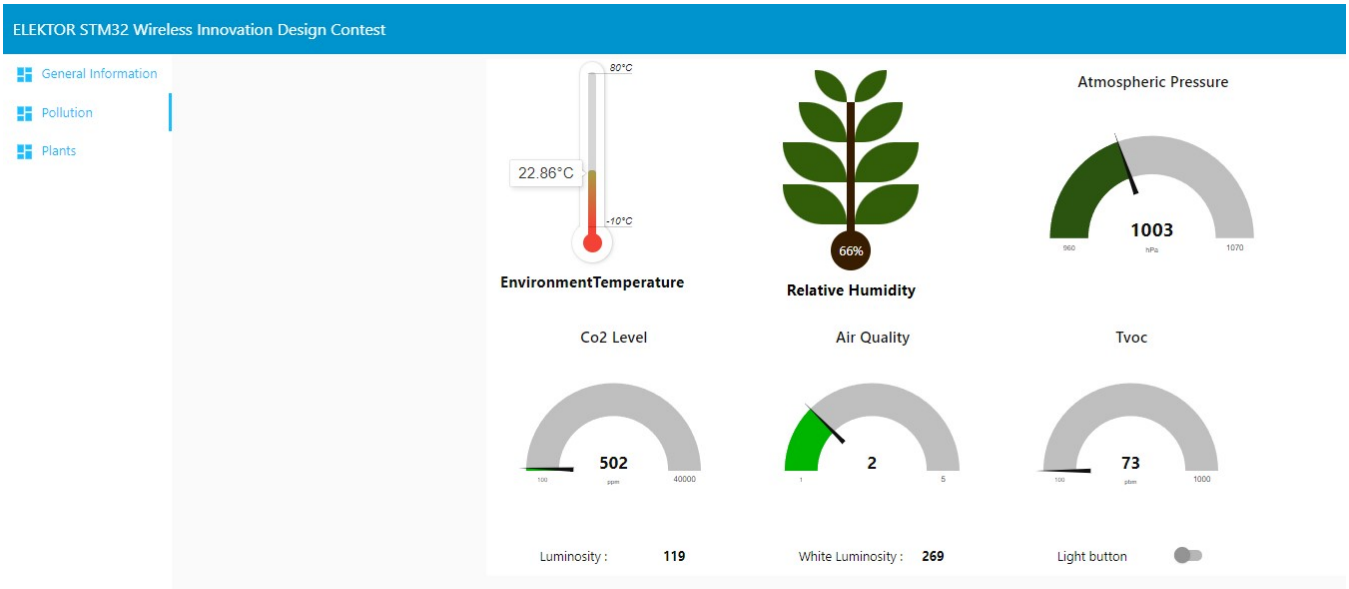

LIGHT AUTOMATION
timestamp t.i
Is it night time yet?
Day or Night :
Startup event
Get Sunset info
http request
Save Recalculated times (the easy way)
start of day t.i
timestamp t.i
DateTime
Clock :
config display date
DateTime
Sunrise Time :
config display time
DateTime
Sunset Time :
get state of light 1
connected
CurrentStateLight
Set PlantLight attribute
To Mqtt Set
Initialization
Light activated below this brightness
set global.LightLimit
read attribute light
shedule 4s Activate
Light management
enable disable button
State of Light
off
update DesiredStateLight
function 7
View state of light automation
Startup event
set msg.payload
Automation Mode for Light :
function 6
view global for debug
Load Scheduler
ui scheduler
Save Scheduler
Set StateLightSheduler

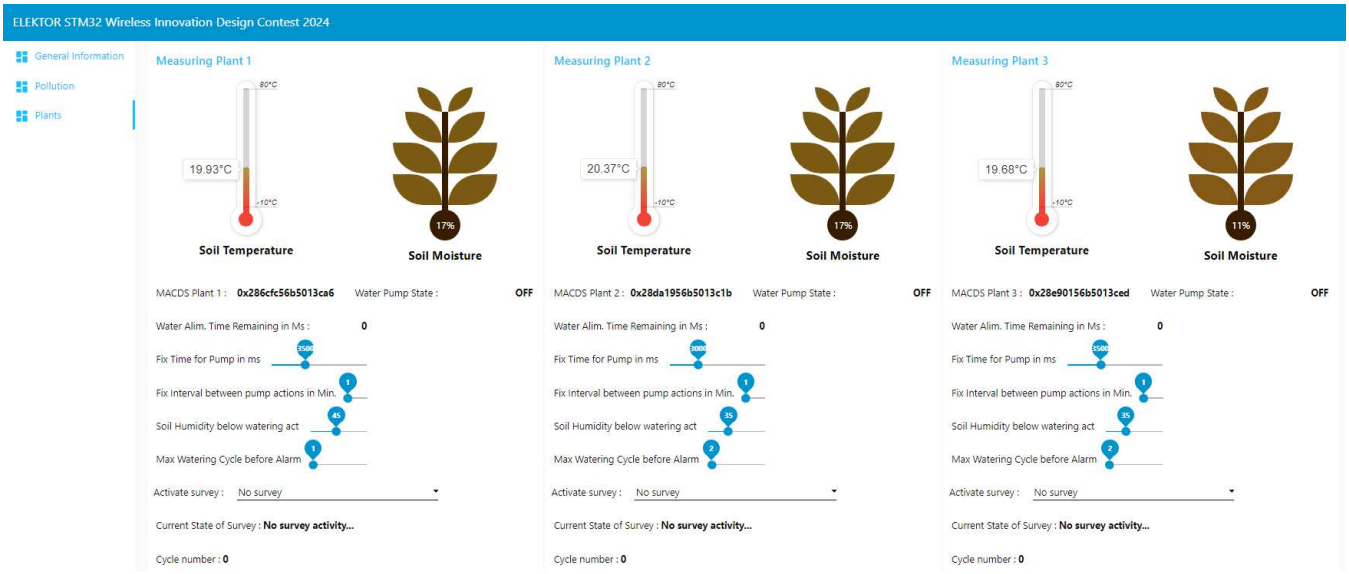


All flows and detailed explanations are joined with this project in nodered directory.

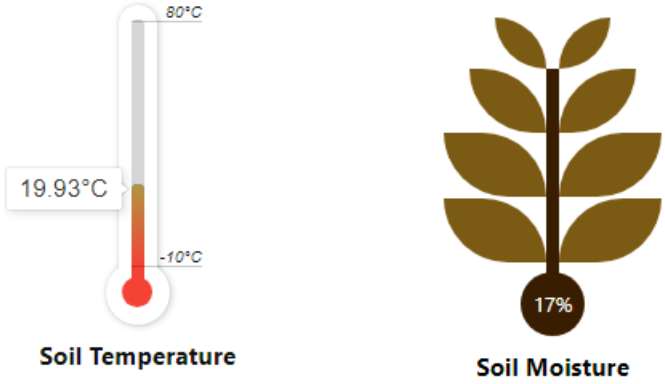
NodeRed Dashboard sample screens







Measuring Plant 1



MACDS Plant 1 : **0x286cfc56b5013ca6** Water Pump State : **OFF**

Water Alim. Time Remaining in Ms : **0**

Fix Time for Pump in ms 3500

Fix Interval between pump actions in Min. 1

Soil Humidity below watering act 45

Max Watering Cycle before Alarm 1

Activate survey : No survey

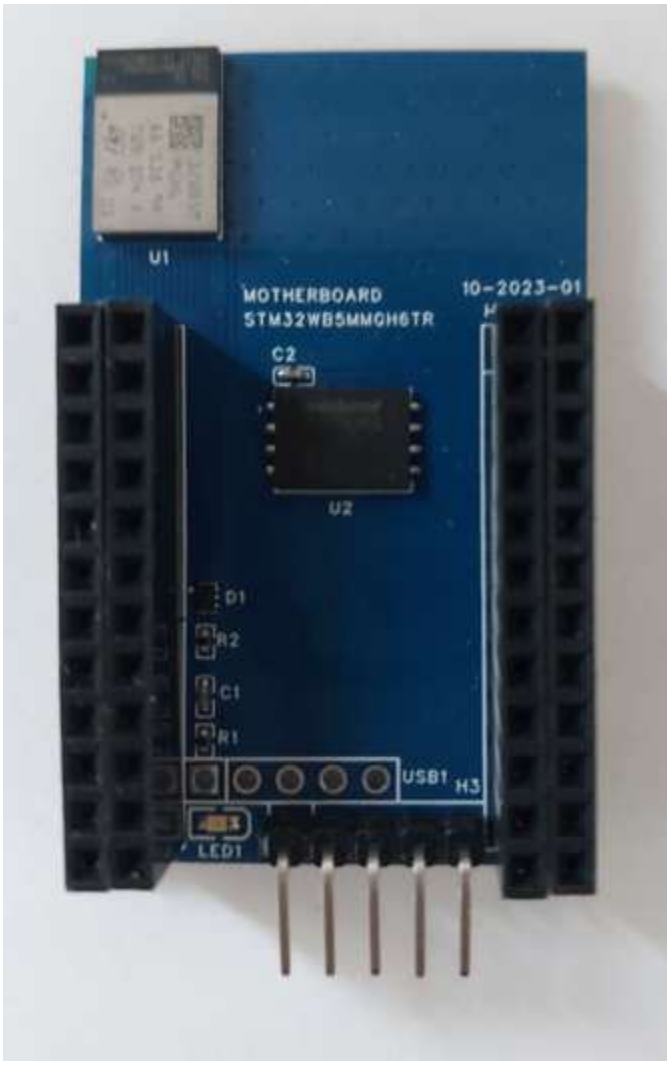
Current State of Survey : **No survey activity...**

Cycle number : **0**

RESSOURCES LINKS OF THIS PROJECT

https://project.sophe.com/stm32ContestAllProject.zip

Somes other details of my second goal in low power mode

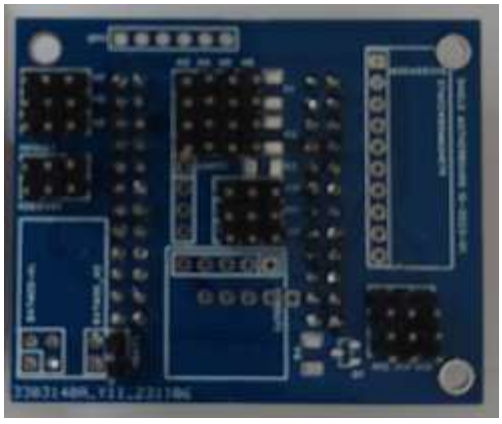


Creation of light motherboard (4 layers) with stm32wb5mmgh6tr and flash 32M for recording measure.

Very difficult to produce…

PCB attached to this project.

Programming with stm32programmer

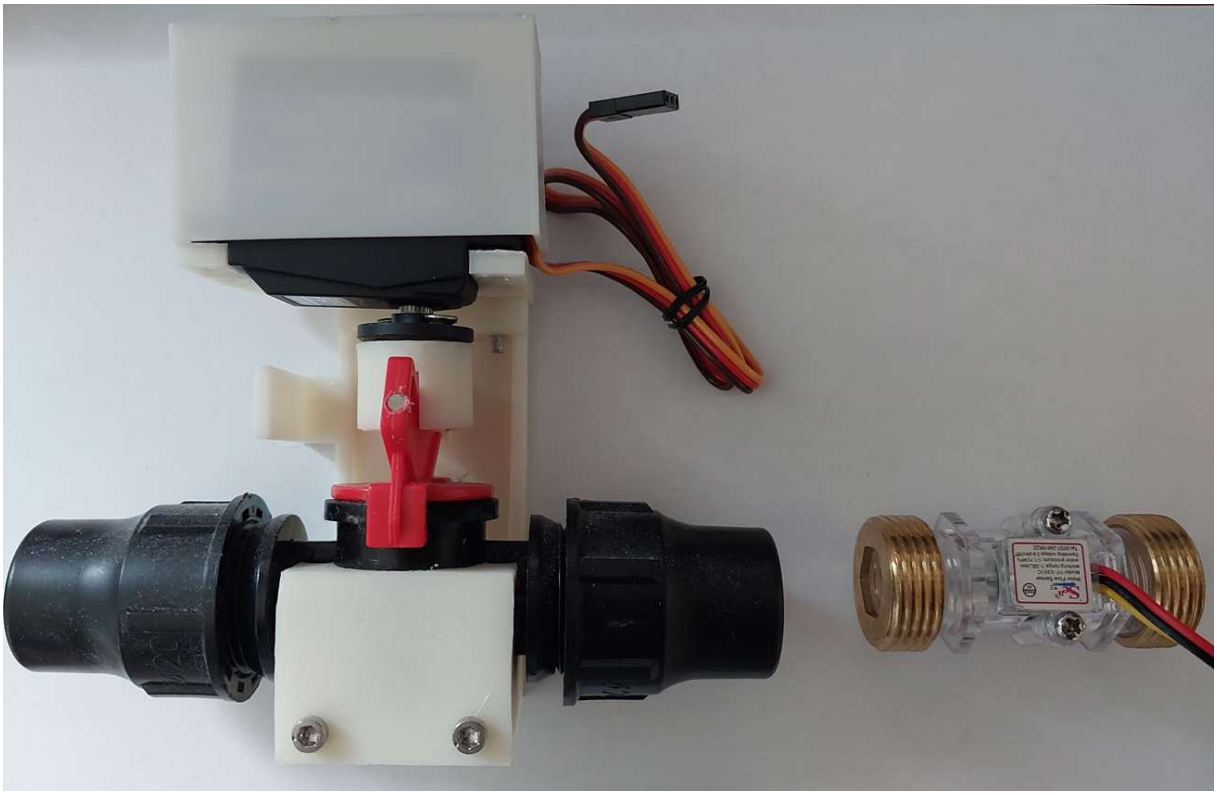


Shield for low power usage (2 layers)

PCB attached to this project

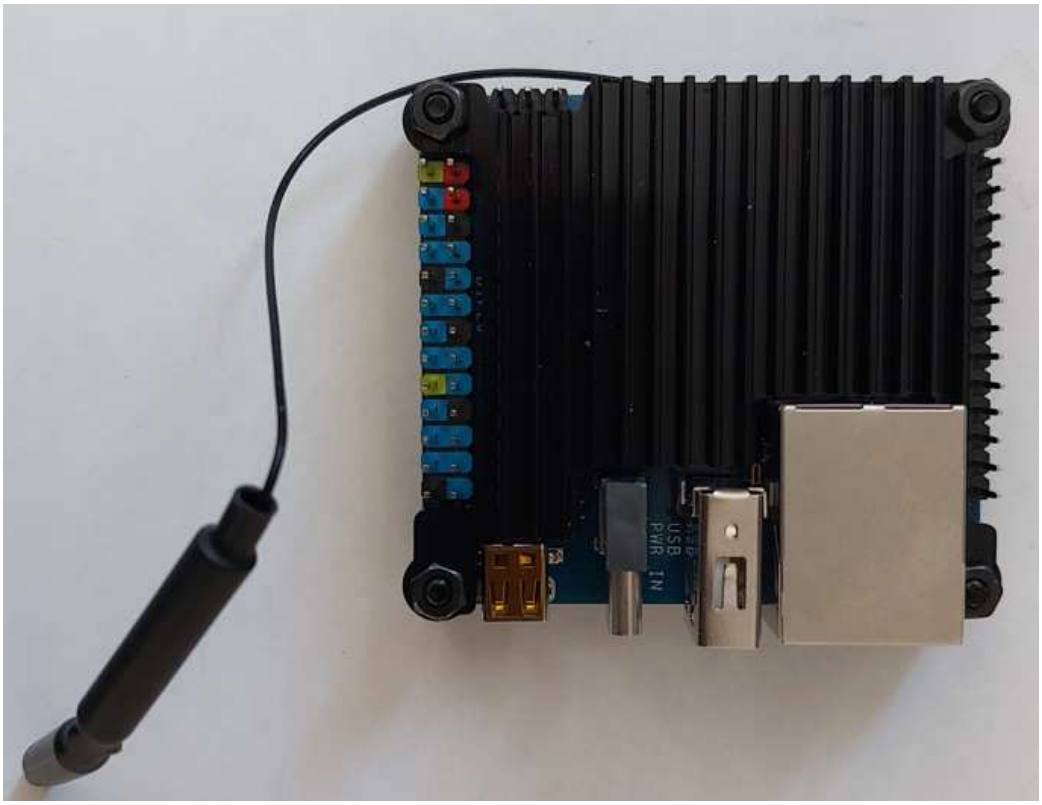
Solar Panel 5V

An irrigation valve drived by a servomotor for supply water by gravity with flow meter counting by Hall effect.



TDS Sensor for testing quality water connected on ADS1115.

Integrate Zigbee2mqtt Coordinator and Mosquitto in OrangePi Card Zero 3 with Linux or OpenWrt.




HDMI
PWR IN
Rayson
RS1032LC4
D28DB-53BT
2343FHLKG
H618
UART
ANT
SIP13