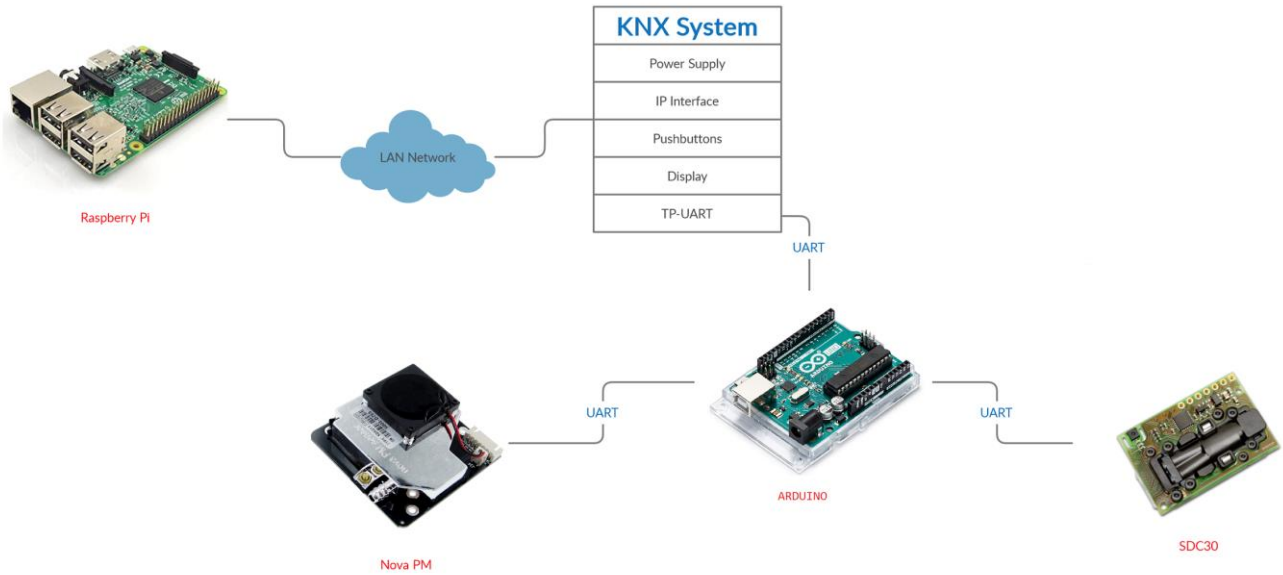
Smart Home Project

Team IP1
Robin Reiman
Florian Graf
Wodran Van de Sande
Stijn Van Riel
Koen Vorsters

Introduction

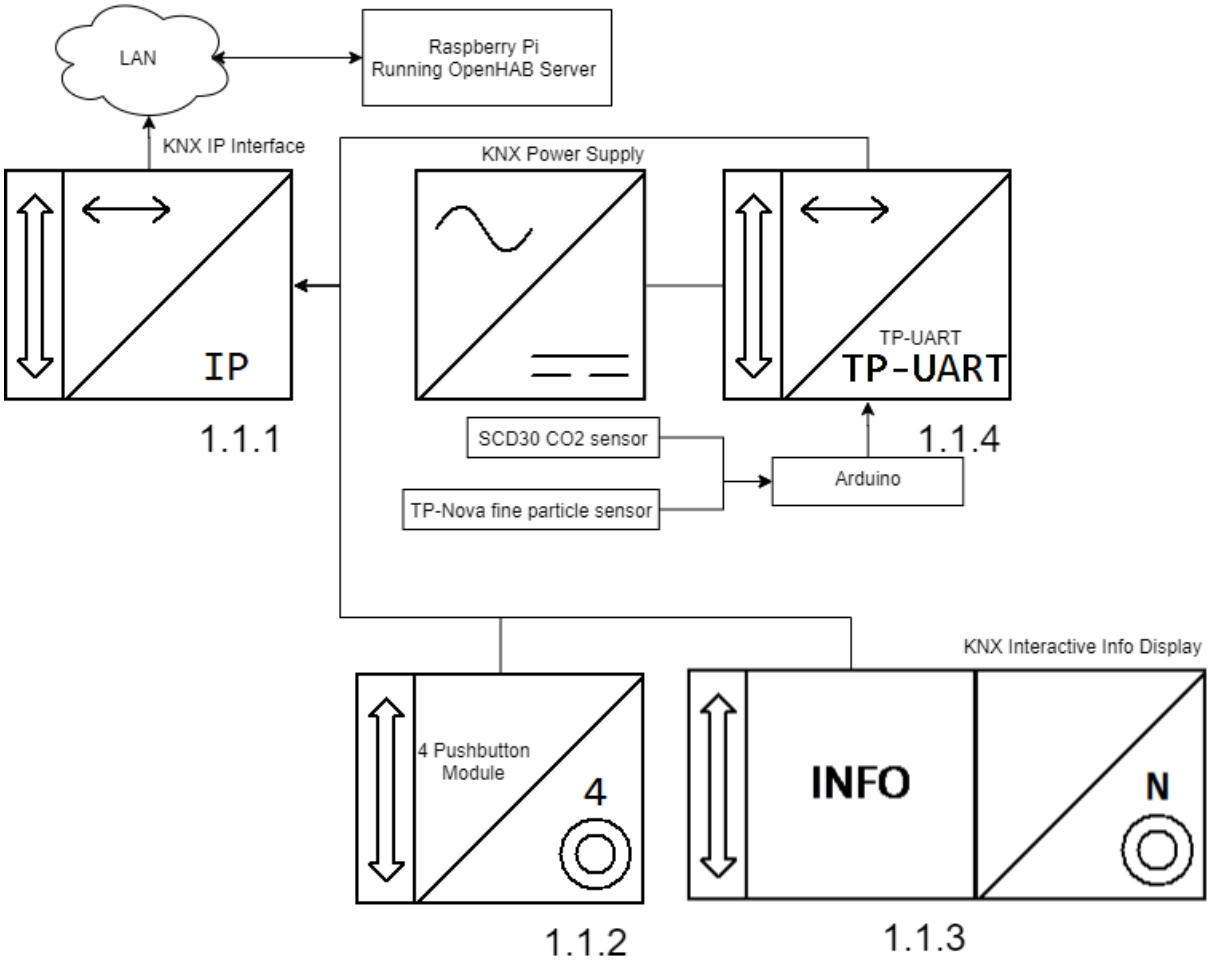
Project description



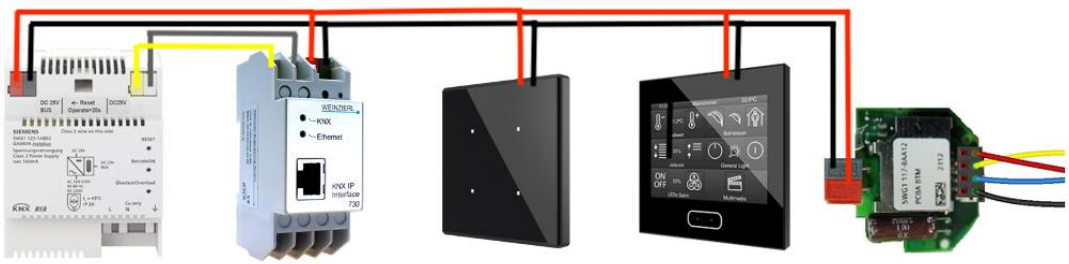
Our end goal is to create a self-built KNX sensor, as such a sensor costs quite a lot of money.
To measure the data, we use Arduino-compatible sensors, such as the Nova PM Sensor and the SDC30. These send their data to an Arduino, which in turn sends its data to the KNX system using a TP-UART module to communicate.
Our KNX System also communicates with a Raspberry Pi running OpenHABian, allowing us to integrate other smart systems such as Smart Lighting, Google Home Assistant or Apple HomeKit. The data that is read by the KNX system then becomes available on the entire system.
Another advantage of the OpenHAB system is the ability to set up rules using the “If … Then, Else … ” and “Other” principles. for example: when there is too much CO2 in the air we can start the ventilation automatically.
As an extra we also created a Grafana screen, showing the historical data of our sensor system. In addition to that we also made a Prototype PCB-Schematic.

Hardware

Block Diagram

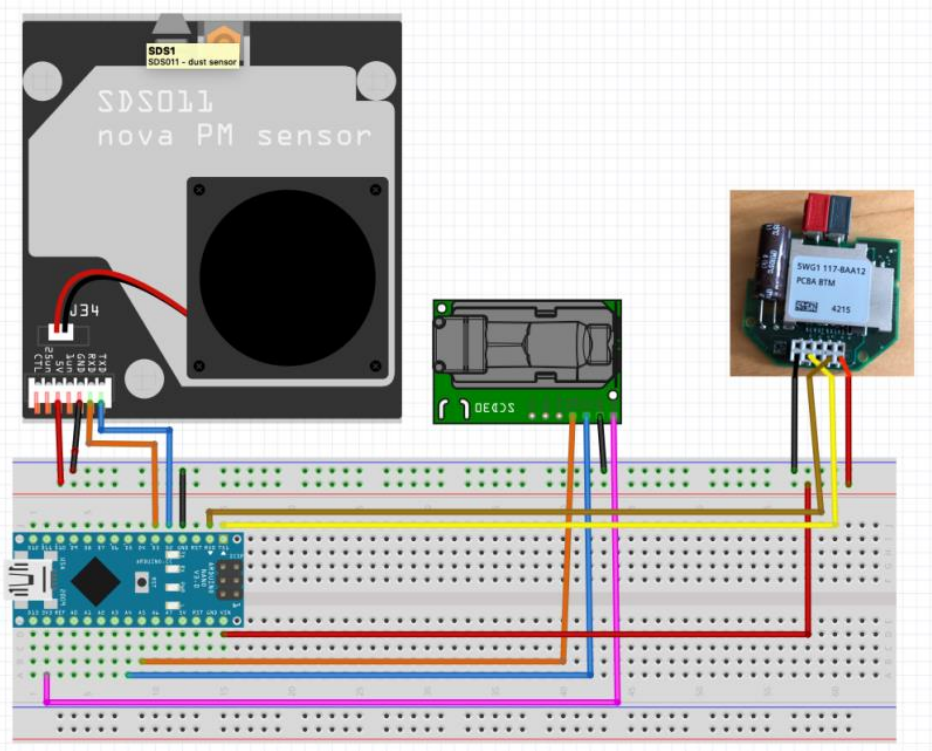


KNX System



KNX uses its power as data lines, so it doesn't need any special connections to be able to transmit data through its system.

Arduino Sensor System



All sensors are connected to the Arduino, which in turn is connected to the TP-UART.

Overview of Interfaces

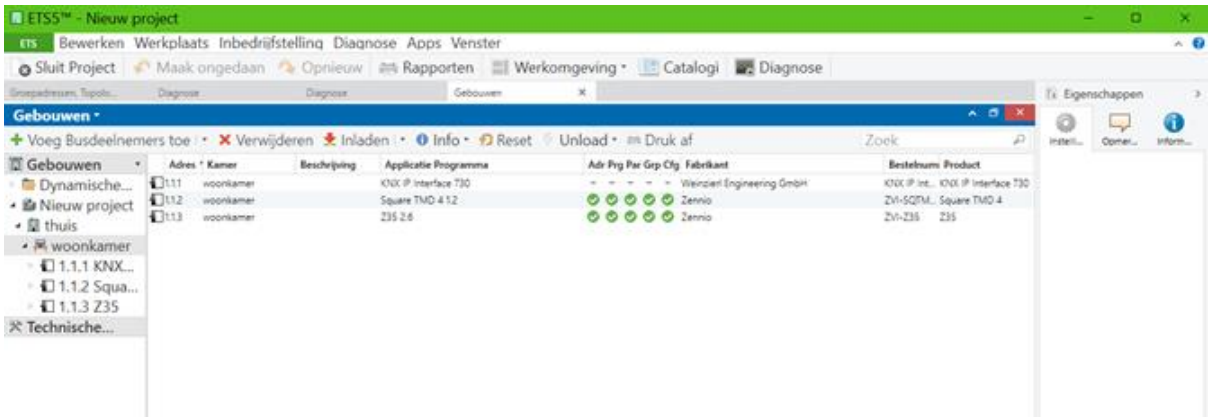
Name	**Usage**	**Electrical (Voltage level, IO Standard)**	**Mechanical (Plug type and pinout)**	**Protocol & Specification**
KNX	Communication between KNX devices	29V	2 Wires	PL110 Powerline @ 110kHz
UART	TP-UART & Nova PM Sensor	Not defined (typically 3.3 V or 5 V)	RX TX	UART
I²C	SDC30	Not defined (typically 3.3 V or 5 V)	SCL SDA	I²C 100kHz

Bill of Materials

Component Name	Model	Average Price	Usecase
KNX Power Supply	Siemens 5wg1125-1ab02	~€ 200.00	This power supply will convert our AC Current from the wall to a voltage that the KNX Components can use.
KNX IP interface	Weinzierl KNX IP interface 730	~€ 250.00	To program the KNX Interface and read out the data, we use the IP Interface.
KNX Pushbutton	Zennio Square TMD 4	~€ 100.00	This standard KNX Component has 4 buttons that we can use to control different KNX Components.
KNX Visualization	Zennio Z35	~€ 250.00	This KNX Component has a built in touch screen. It is able to dynamically show control buttons and sensor data.
Microcontroller	Arduino Uno	~€ 10.00	This microcontroller will read our sensor data and act as a bridge between the sensors and the TP-UART
Particulate Sensor	Nova PM Sensor SDS 11	~€ 15.00	A high-precision particulate sensor, capable of sensing PM2.5 and PM10 particles. This sensor will be connected to our ESP32 using the serial pins.
CO2 Sensor	SCD-30	~€ 60.00	This high accuracy CO2 sensor can detect CO2 in the air starting from 400 parts per million. It is also connected to the ESP32's Serial Pins.
TP-UART	Siemens 5wg1 117-8aa12	~€ 40.00	This component allows us to connect our Microcontroller to the KNX System.
Raspberry Pi	Raspberry Pi 3 Model B+	~€ 30.00	We will use the Raspberry Pi to display our KNX Data on an OpenHAB system

Software

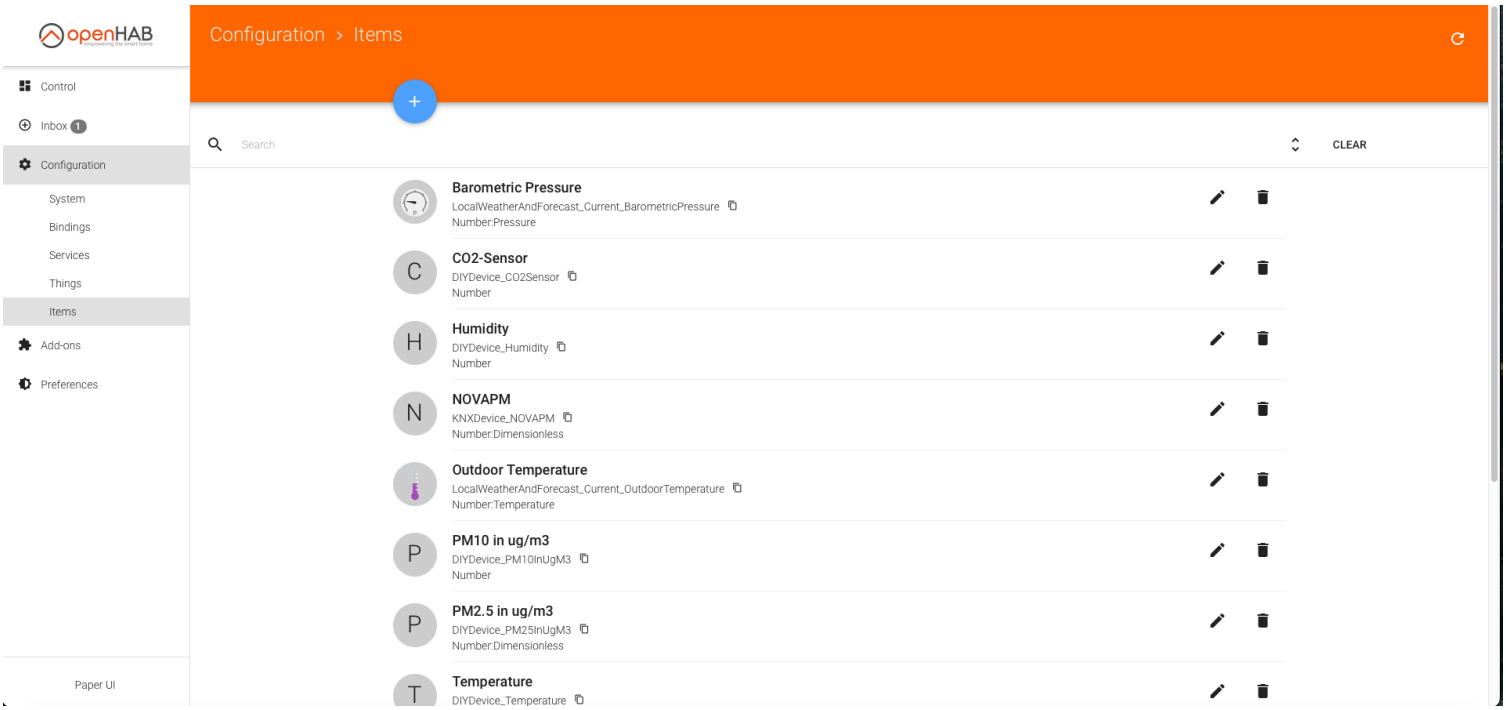
ETS5

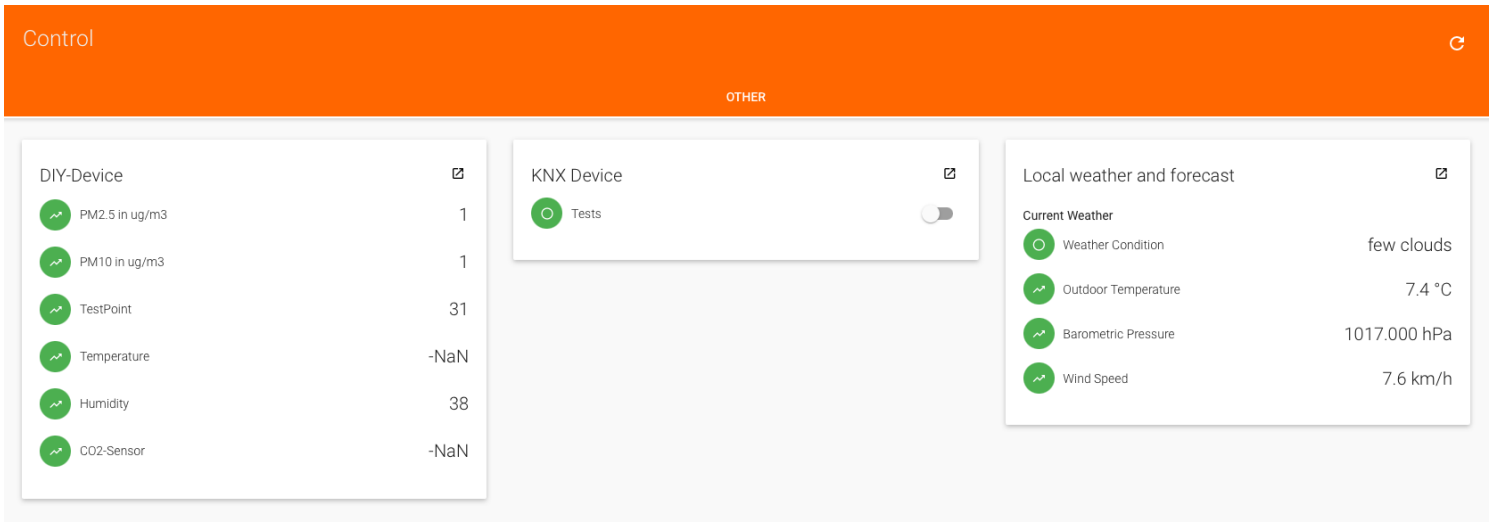


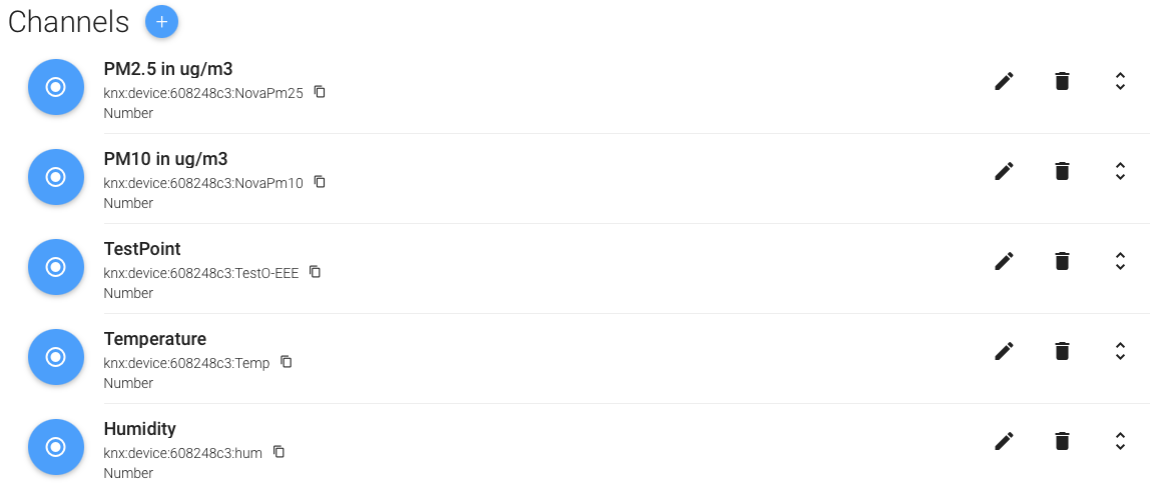
The ETS software allows us to program the KNX System. We use the free variant, which limits us to only 5 devices, but that is enough for our (limited) system.

Openhabian

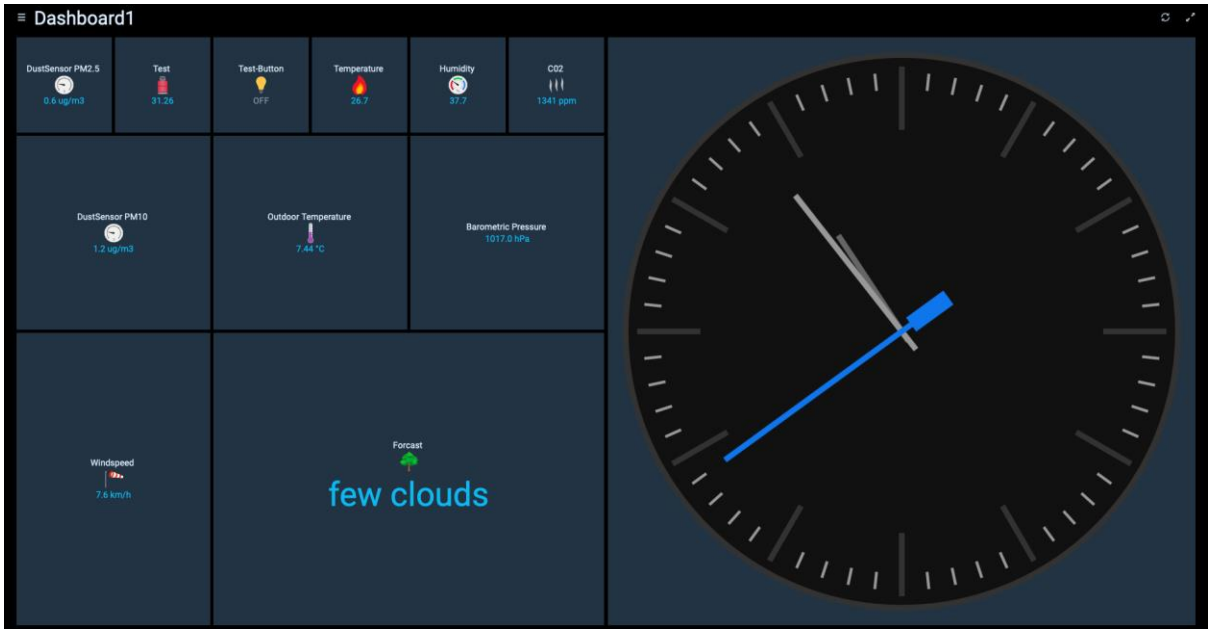
Running the OpenHABian OS on our Raspberry Pi enables us to set up the OpenHAB system very easily. With the OpenHAB Bindings there is almost no setup required to connect the KNX system or other smart home appliances.





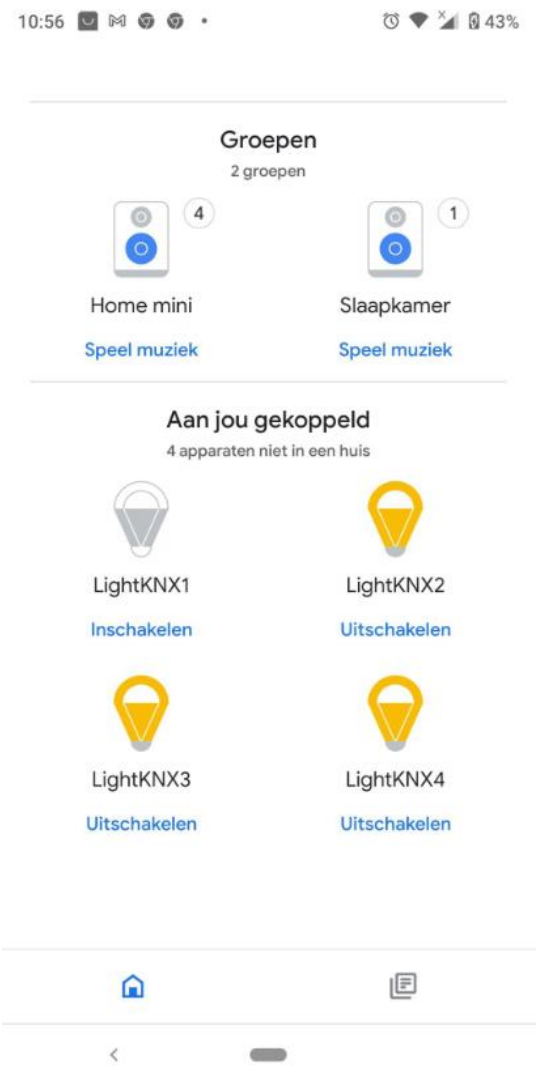


With a HABPanel we can display the sensor data and button status on an attractive screen. If we wanted to create a physical control panel in the home, we could use a tablet and display this screen.



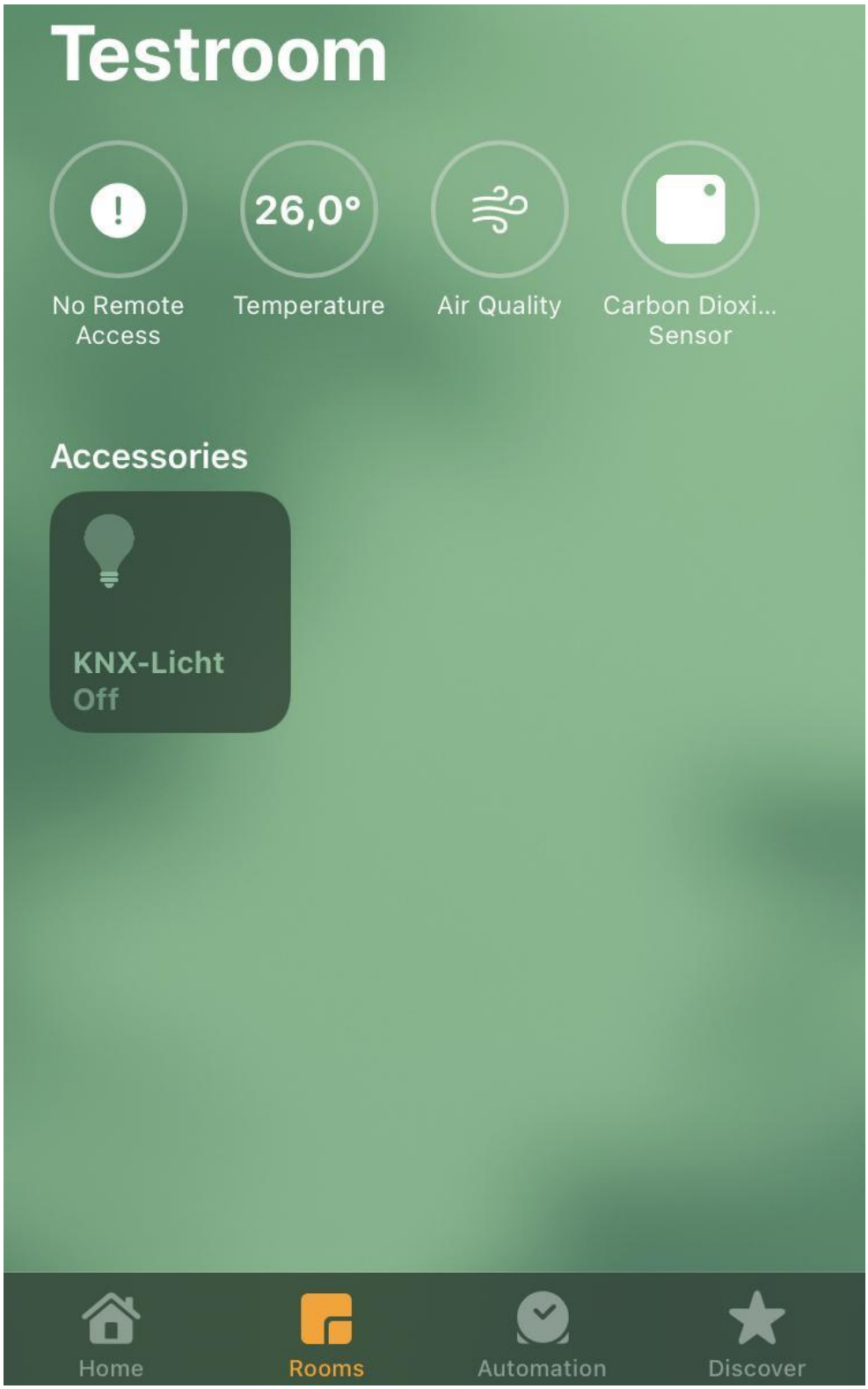
Google Assistant

using an OpenHAB binding, we can easily add Google Assistant commands that interact with the KNX System, such as asking the assistant to read out our sensor data.



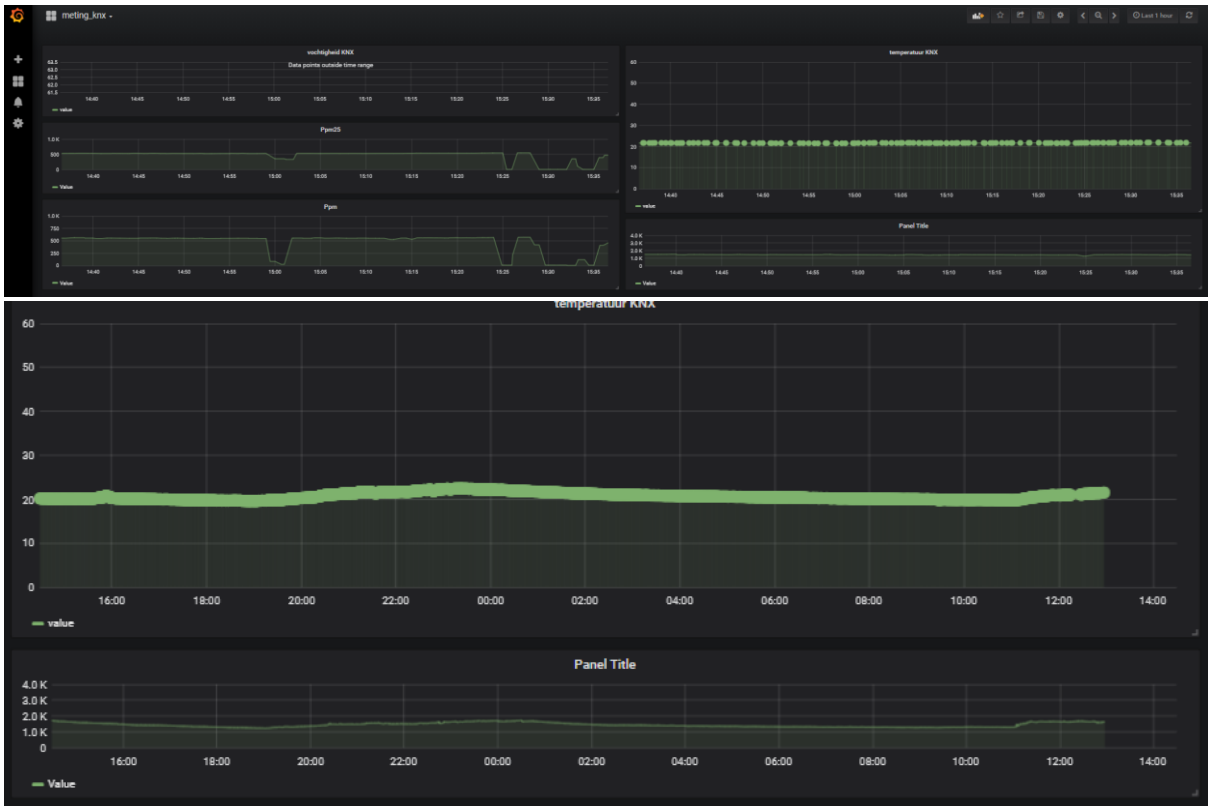
Apple HomeKit

The connection to Apple HomeKit allows us to control the system within the Apple Ecosystem.. In addition to that, its possible to control it with Siri. Simply install under “Misc” the HomeKit Service and check the default device number. This Device Number should be added with an IOS-Device under add Accessory and then the connection works.



Grafana

When our data is received by the Raspberry Pi, it is written away to an InfluxDB Database. Using Grafana, we can visualise the historical data so that we can chart the measurements.



Manual

Overview of provided components

Amount	Component Name	Model
1	KNX Power Supply	Siemens 5wg1125-1ab02
1	KNX IP Interface	Weinzierl KNX IP interface 730
1	KNX Touch Buttons	Zennio Square TMD 4
1	KNX Interactive Display	Zennio Z35
1	Arduino Uno	ARDUINO UNO REV3 A000066
1	Dust Sensor	Nova PM Sensor
1	CO^2 Sensor	SDC-30
1	TP-UART	Siemens 5wg1 117-8aa12
1	Set of assorted cables	/

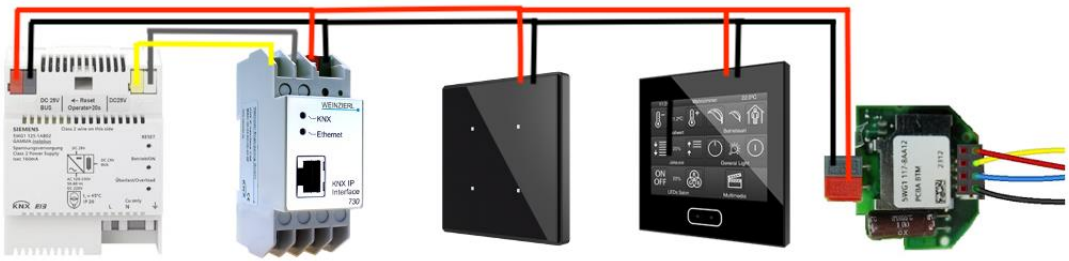
Installation Guide

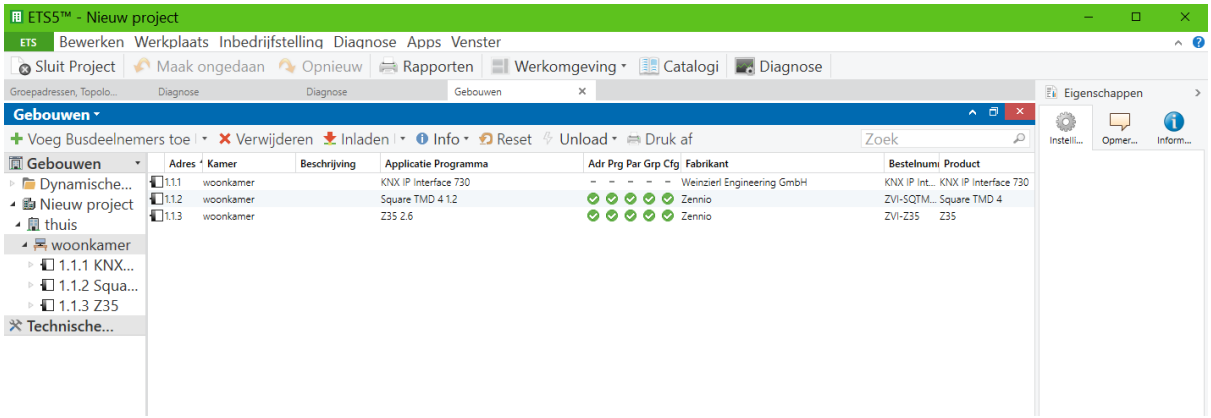
In this chapter we will discuss the necessary installation. This way you can also set up the system yourself. Once the system is set up it is quite easy to add other smart things and check them.

To get started,

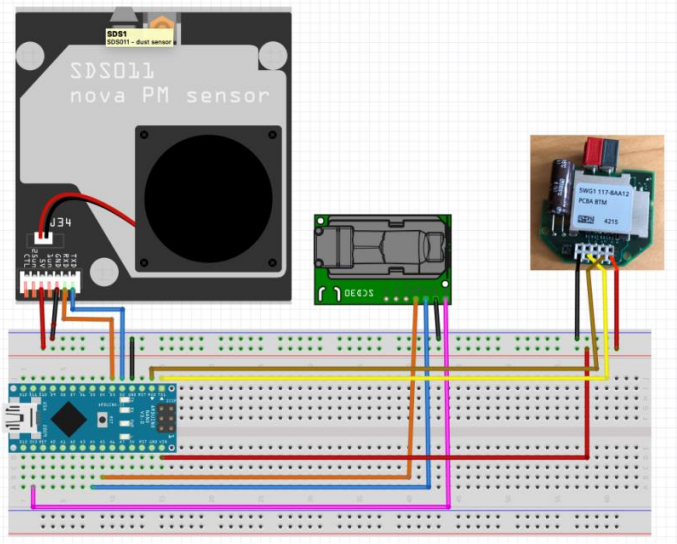
Hardware and Software installation

Install the components of the KNX System. Wire the components as seen in the picture below. After that we connect the sensors to the Arduino and made sure we could read out the sensors. Next you install the ETS5 software on your pc. You connect the KNX module IP 730 to the PC.





In the ETS5 software there is a bus monitor which makes it possible to view the transmitted data. These push buttons have to be addressed in the ETS5 software. This is quite simple and can be set by the graphical interface. Then connect the Tp Uart to the Arduino and connect the sensors to the microcontroller. (see picture)



Now the sensor data should also be visible in the bus monitor.

If this is successful, you can start integrating the KNX system with Openhab.

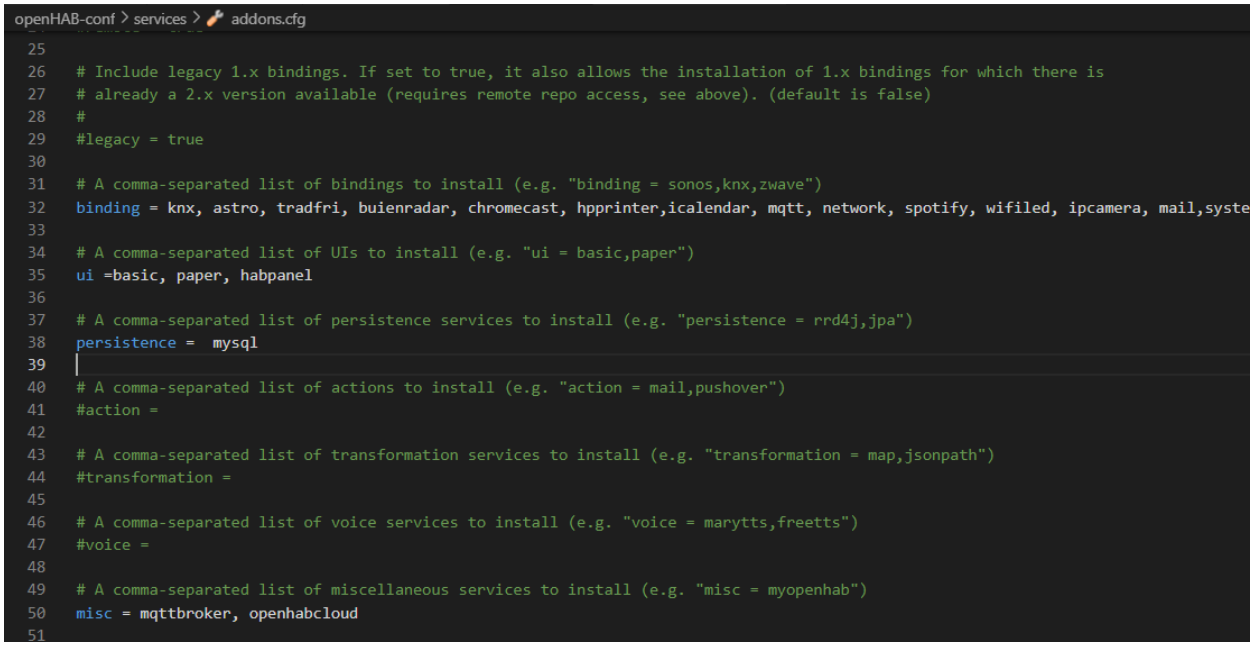
OpenHAB

Start by installing OpenHABian on the Raspberry Pi.
Download the files from https://www.openhab.org/download/ and flash them on a SD card.

When this is finished insert the SD card into the RPi and let the RPi install the software. This should take around 30 minutes. After that, reboot. You can now view the interface in a web browser on the Raspberry Pi's IP Adress. In this case it should be:http://192.168.0.158:8080.
A simple tip is to attach a screen and keyboard to the RPI. At startup, you can see on the screen which IP address your RPI has.

Now there are a number of ways to configure the Openhab software.
You can work through the PaperUI and configure bindings, things and items through this graphical interface.
It is quite simple but also more limited in its possibilities.
It is also possible to write this configuration in the files themselves. You can use Putty to connect via ssh to modify the files. Visual Studio Code has native support for writing code through SSH. This software can be installed and makes it much easier to modify things in the files. After the installation of Visual Studio Code you install the Openhab extension in this software. You open the folder of the RPI (192.168.0.158) and can program the files.

The first file you set up is the addons.cfg. Here you set which bindings need to be installed at startup. Once you set up this file, only this file will be viewed at startup. In the image below you can view our file.



We install the KNX binding and a number of other bindings in order to set up the Smarthome products. To be able to use the objects (things) you need the necessary bindings. Furthermore, we will specify which interfaces we want to make available. Like building the PaperUI and habpanel for the display. We also set the persistence to store the Data that we will generate in our system in a Database. After you have set this you can reboot the system and the necessary bindings will be installed.

Now we create a file in the folder “things”. In this file we will create all objects.

In the picture below you can see the example of the KNX system. In the first place you see the Bridge this "thing" is the KNX ip730. We tell the code which IP address it uses. This can be found in the ETS5 software and put the type on “tunnel”.
For the correct settings or extra options you can always check the Openhab website to see what you need to adjust.

```
Bridge knx:ip:bridge [
    type="TUNNEL",
    ipAddress="192.168.0.157",
    autoReconnectPeriod=60 //optional, do not set <30 sec.
] {
    Thing device knx_device "knx_device_pushbuttons" @ "knx_system" [
        //readInterval=3600 //optional, only used if reading values are present
    ] {
        //Items drukknoppen
        Type switch         : demoSwitch         "LightKNX1"                    [ ga="0/0/2" ]
        Type switch         : demoSwitch1        "LightKNX2"                    [ ga="0/0/3" ]
        Type switch         : demoSwitch2        "LightKNX3"                    [ ga="0/0/4" ]
        Type switch         : demoSwitch3        "LightKNX4"                    [ ga="0/0/1" ]
        //knx Sensoren
        Type number         : knxtemp            "knx temperatuur"               [ ga="0/0/7" ]
        Type number         : knxsensor          "knx "                         [ ga="0/0/10" ]
        Type number         : knxsensor2         "knx2"                         [ ga="0/0/9" ]
        Type number         : knxsensor3         "knx3"                         [ ga="0/0/6" ]
        Type number         : knxhumidity         "knxhumidity"                 [ ga="0/0/11" ]
```

We now make an object for each push button and to measure the sensors.

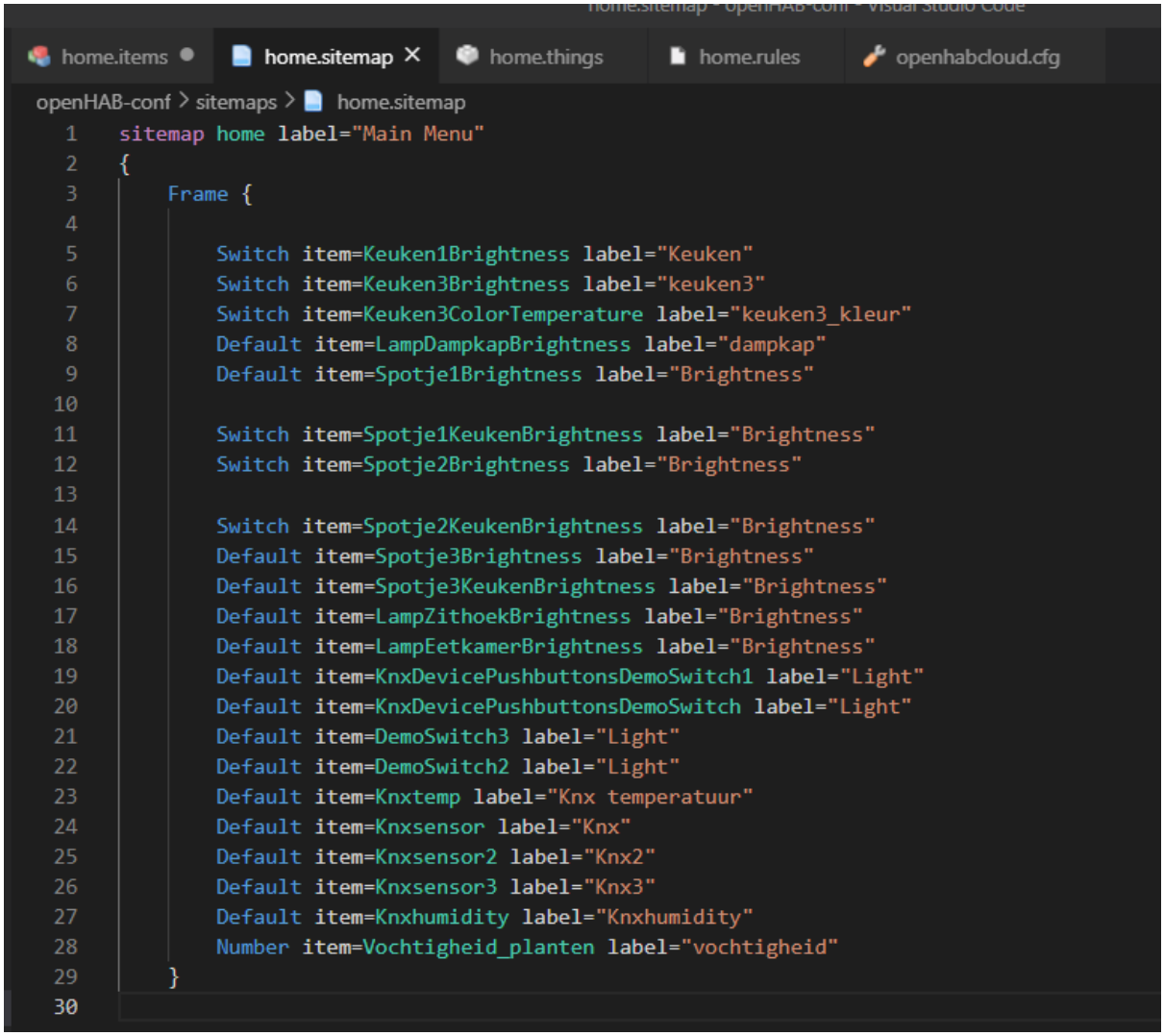
First, we indicate the Type then the name of the Object and give it a label. Lastly, you display the action that happens.

Next we will create an Item file in the directory items and set all the Items here. As you can see in the picture. Each item consists of a Type: a switch, a number, Dimmer,...

After the type you place the name of the item and then the Label. Then you specify which channel is used. Between the label and channel we have added extra information to a number of items. This enables us to use these items in Google Assistant at a later stage.

```
//items file

//items knx
Switch      KnxDevicePushbuttonsDemoSwitch     "LightKNX1"   [ "Switchable" ]                                  {channel="knx:device:bridge:kn
Switch      KnxDevicePushbuttonsDemoSwitch1    "LightKNX2"   [ "Switchable" ]                                  {channel="knx:device:bridge:kn
Switch      DemoSwitch2                        "LightKNX3"   [ "Switchable" ]                                  {channel="knx:device:bridge:kn
Switch      DemoSwitch3                        "LightKNX4"   [ "Switchable" ]                                  {channel="knx:device:bridge:kn
Group       sensorsknx                             "thermostat"                  { ga="Thermostat"}
Number      Knxtemp                             "Knxtemperatuur"               (sensorsknx)                  {channel="knx:device:bridge:knx_devi
Number      Knxsensor                           "Knx"                                                        {channel="knx:device:bridge:kn
Number      Knxsensor2                          "Knx2"                                                       {channel="knx:device:bridge:kn
Number      Knxsensor3                          "Knx3"                                                       {channel="knx:device:bridge:kn
Number      Knxhumidity                         "Knxhumidity"                                          {channel="knx:device:bridge:knx_device:knx
```

As a last file you create a file called “home.sitemap” in the “sitemap” folder. This file is used to make the Things visible in the Displays.



Depending on which database you prefer to work with, you can install it on your Raspberry Pi. For our project we worked with Mysql as well as influxDB. To set the data of the Things we use a file in persistence. for example: “mysql.persist”. Here you can determine which items are stored and when. In this example we store all data of the Items in a mysql database every minute. You can also set the file mysql.cfg correctly in services. Here you place the username and passwords and connection needed.

home.items ● | home.sitemap | mysql.persist ✕ | home.thi

openHAB-conf > persistence > mysql.persist

```
1   Strategies {
2   everyMinute : "0 * * * * ?"
3   everyHour : "0 0 * * * ?"
4   everyDay : "0 0 0 * * ?"
5   every2Minutes : "0 */2 * ? * *"
6   }
7
8   Items {
9   *: strategy = everyChange
10  *: strategy = everyMinute
11  }
```

You can also set the file mysql.cfg correctly in services. Here you place the username and passwords and connection needed.

Grafana

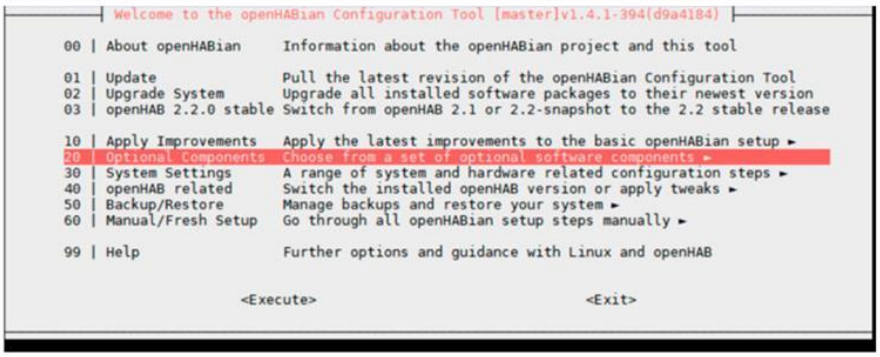
Now we connect the database to Grafana.

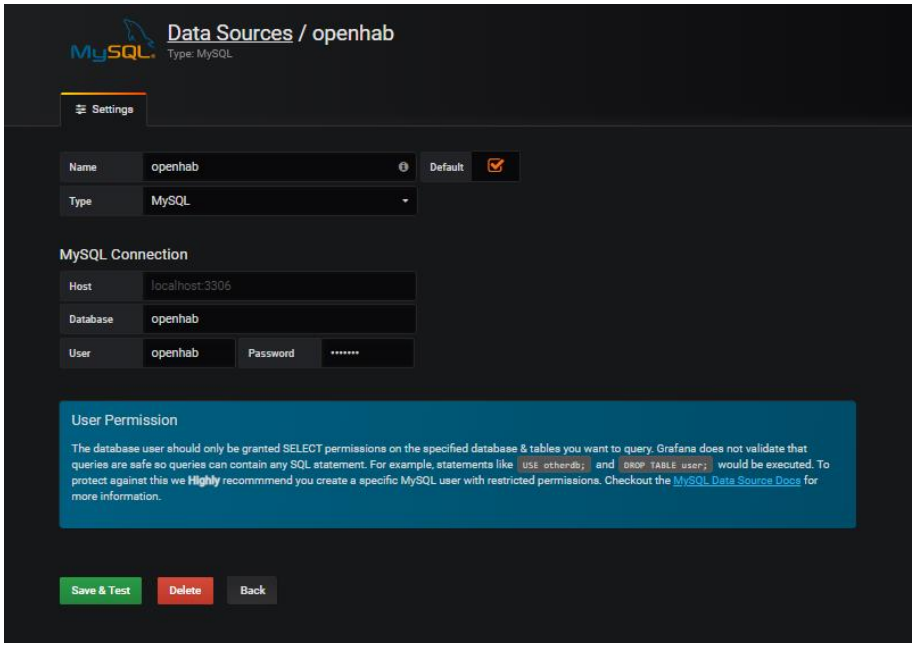
First you install Grafana. You can do this with the command “sudo openhabian-config”. Then you choose optional components and install Grafana. During the installation it will ask for a username and password. You will need these later to make the connection.

```
1 sudo openhabian-config
```

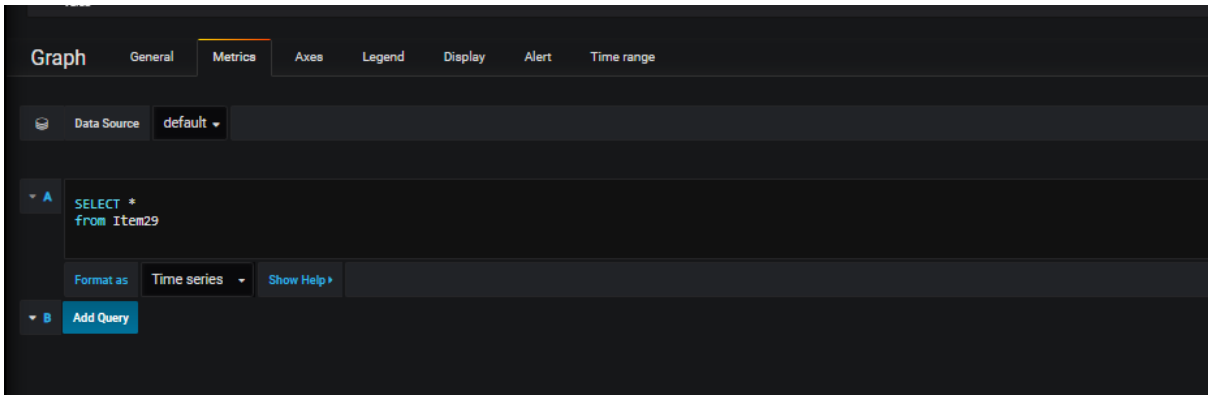
This will open the OpenHabian configuration menu.

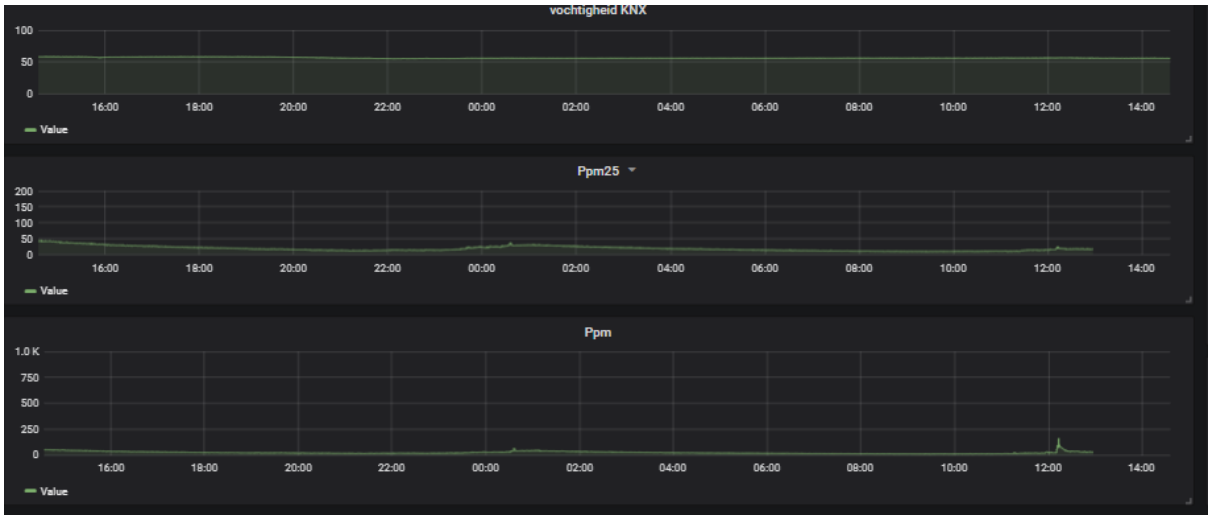
1. Select Optional Components



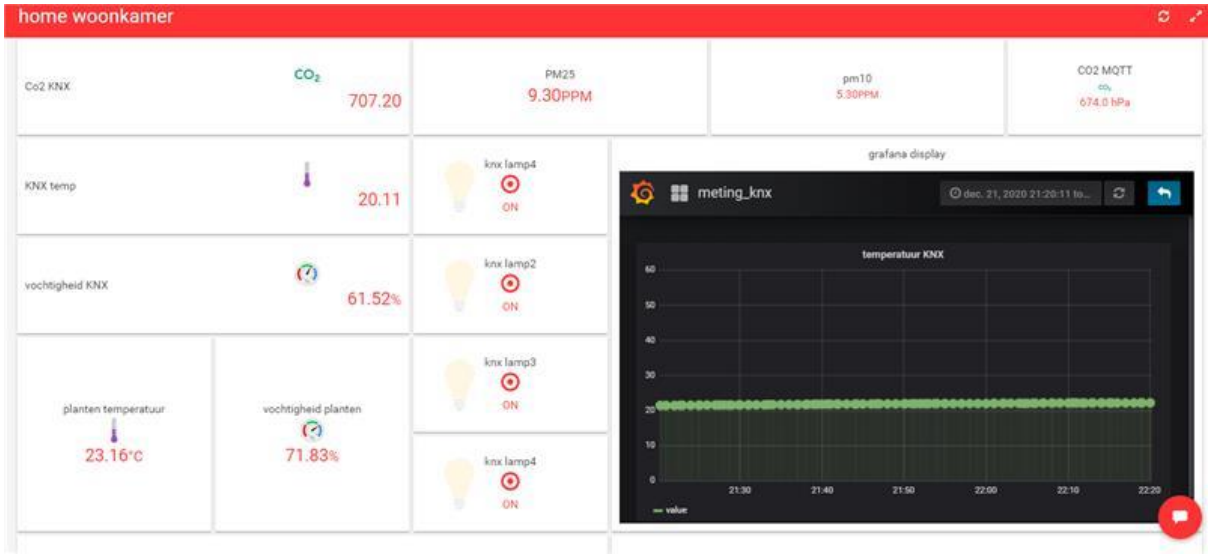


Now you can access the url 192.168.0.158:3000 via the web browser to open Grafana. Here you create a database connection and enter the username and password. Then you can test the connection. If it is successful you can create graphs. You have to change the time in the database to time with a lowercase letter. This way Grafana can place the data points correctly in the graph based on its timestamp.





From these graphs we create an Iframe which we can then insert into Habpanel and view these graphs through our display in any browser, on any device. To go to Habpanel and create these displays, go to 192.168.0.158:8080 via the web browser and select Habpanel. Here you can then create a display.

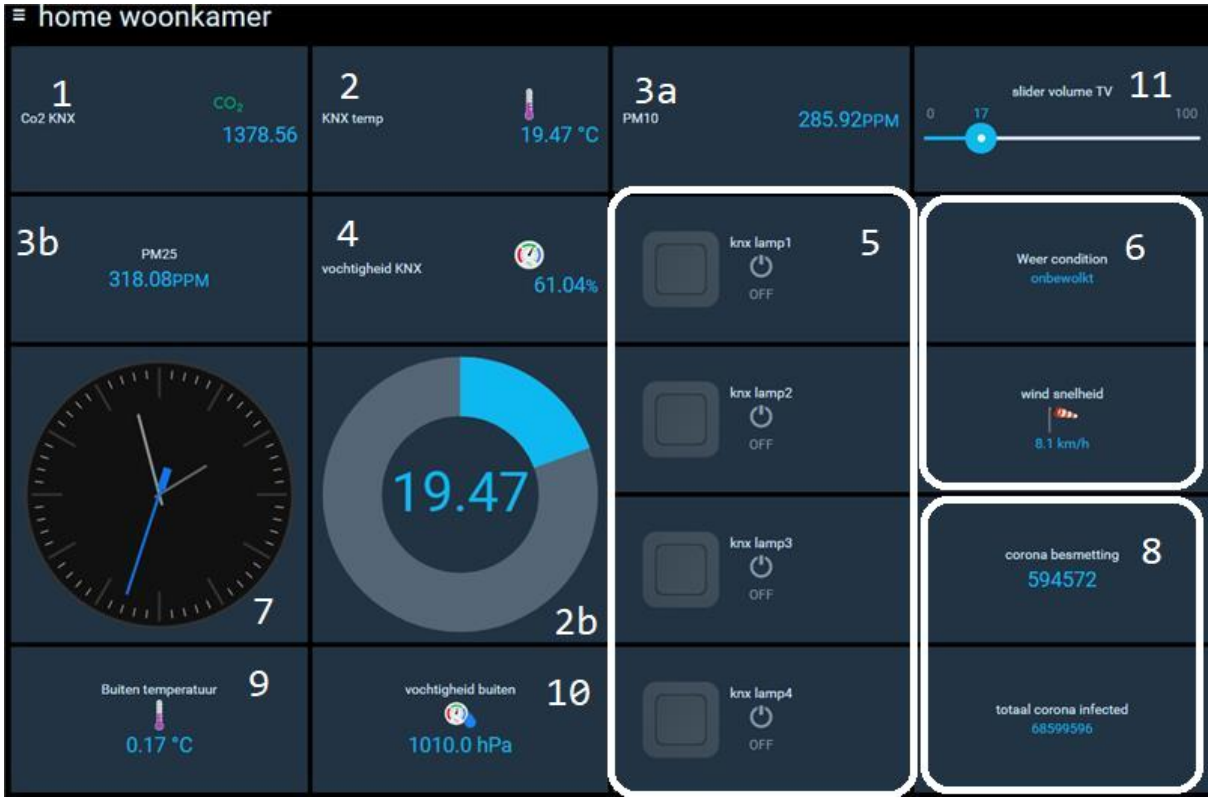


To communicate with the Google Assistant with the items we first need to make the Openhab available online. for this we use the Openhabcloud connector. This will ask for a username and password together with a unique ID. Then take your smartphone and go to the Google Home application. Here you click on the plus sign to add a new device and type openhab. click on the openhab icon. This will take you to the openhab website with the question to give permission. Press allow. Now the items with the extra code like switch and temperature will be available in the Google Home system. Now it is possible to control these items via voice commands

User's Manual

To access the OpenHAB paperUI, go to 192.168.168.21:8080
To access the Grafana sheets, go to 192.168.168.21:3000

OpenHAB Panel



On the Home screen, you can see:

1. The CO^2 sensor
2. The current air temperature (in degrees celsius) and in a graph (2b)
3. The PM10 particles (a) and the PM2.5 particles (b) (in parts per million)
4. The relative air moisture
5. The status of the pushbuttons
6. The weather condition
7. The current time
8. The amount of COVID-19 Infections
9. The Outside Temperature
10. The air pressure
11. A TV Volume slider

Pushbuttons

When you push the first button, the Interactive display is turned on or off.
When you push the second button, the lights are turned on or off.

Interactive Display

On the first screen you are presented with three buttons:

1. To configure the temperature settings for your thermostat
2. To view measurements made by the sensors
3. To configure the KNX buttons

On the info screen of the interactive display, you can observe the PM10 and PM2.5 particles, The relative Humidity, the air temperature and the CO^2 Concentration.
To go back to the home menu, click on the center button below the display.

On this screen, you can configure your thermostat temperature by pushing the + or - buttons on the screen.
To turn off the heating, press the “OFF” button

Troubleshooting

During the project, we encountered several problems:

Not enough power for sensors

Because both sensors require 5v to function, the microcontroller is unable to supply enough power to both sensors, causing it to malfunction during high loads.

We can fix this in a number of ways:

1. We use a secondary power supply to power the sensors separately.
2. We use a different microcontroller, capable of supplying a higher voltage
3. We use different sensors, that require less power

Sensor data is not received by KNX System

Because KNX is, in comparison to the TP-UART, a quite slow system, it is necessary to delay the data before sending it. Otherwise some of the data would not be read by the KNX system.

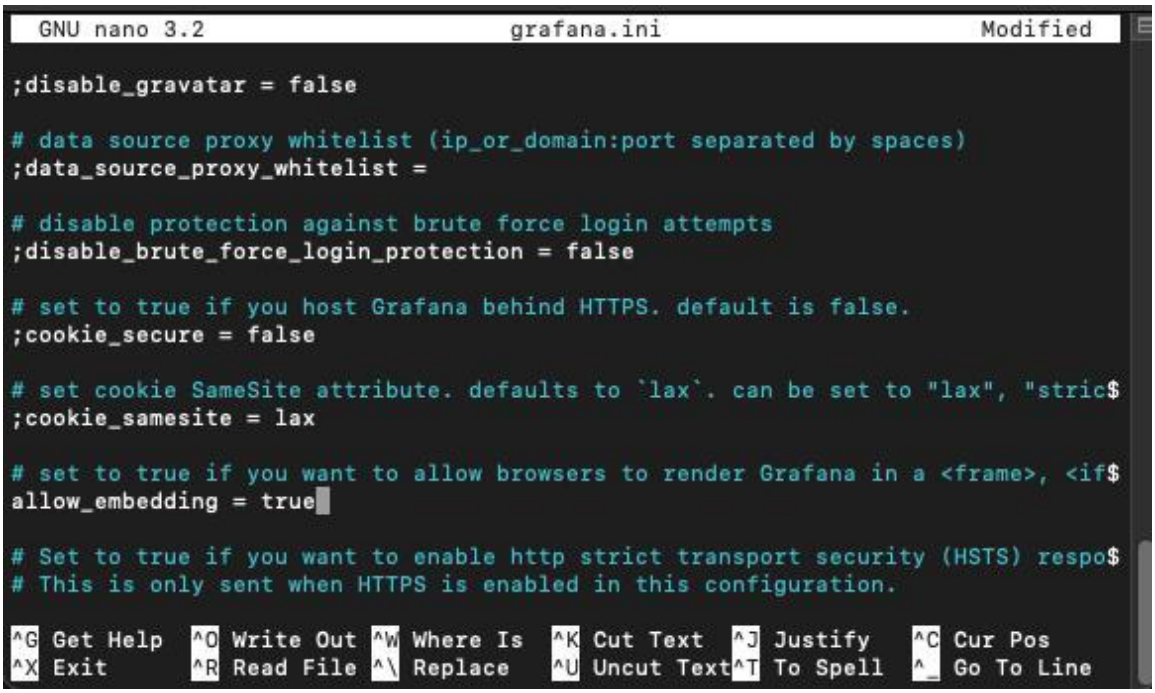
Covid

Because of the limitation imposed by the Covid-19 Pandemic, only 2 of the 5 students have the needed hardware to create the setup. This prevents us from doing meaningful collaboration on the hardware side.

The other 3 students try to balance it by working more on the documentation and general troubleshooting. But because they haven't worked with the hardware, the overall knowledge is limited to being theoretical.

Grafana

Our HABPanel's Grafana iFrame wasn't working. Most commonly this is caused by the the "embedded" Safety-Feature of your Webbrowser. In some browers you can switch the Safety-Feature off. If you're not able to switch it of you have to configure in the grafana.ini the "allow_embedding" to true like it's show in the picture picture you have to comment it in and set it to "true"


GNU nano 3.2 grafana.ini Modified
;disable_gravatar = false
data source proxy whitelist (ip_or_domain:port separated by spaces)
;data_source_proxy_whitelist =
disable protection against brute force login attempts
;disable_brute_force_login_protection = false
set to true if you host Grafana behind HTTPS. default is false.
;cookie_secure = false
set cookie SameSite attribute. defaults to `lax`. can be set to "lax", "stric$
;cookie_samesite = lax
# set to true if you want to allow browsers to render Grafana in a <frame>, <if$
allow_embedding = true
# Set to true if you want to enable http strict transport security (HSTS) respo$
# This is only sent when HTTPS is enabled in this configuration.
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Testing

To make sure the project works (and to find bugs), we created test cases and listed the requirements they need to pass them.

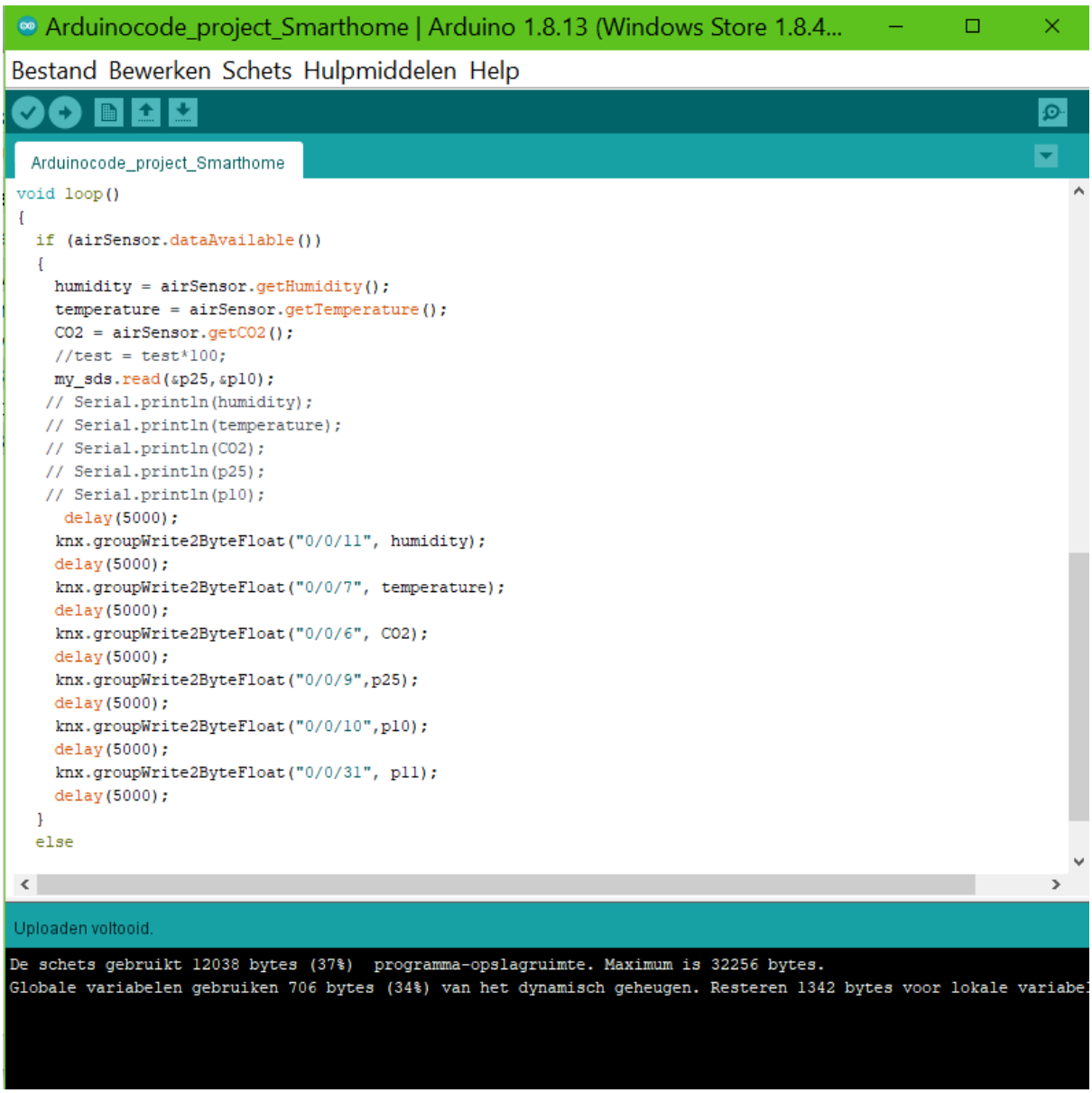
Test ID	Criteria	Verification method	Verification criteria	expected	measured	Test result
1.0	Proper connection of the Nova PM sensor and the microcontroller.	1. Run a UART connection testprogramm on the microcontroller. 2. Read measurement data on the microcontroller.	1. The test has to detect the sensor. 2. Some PM2.5 and PM5 particle values get send by the microcontroller.	/	/	Pass
1.1	Nova PM sensor measuring the correct values for PM2.5 and PM5 particles.	Researching standard values for the area the sensor is tested in and comparing them to the measured values.	Maximum difference of 10% between the measurement and the standard value.	PM2.5: 11-44 PM10: 4-13	PM2.5: 10-15 PM10: 5-10	Pass
2.0	Proper connection between all devices of the KNX-System	Trying to implement every device in the ETS-software.	No errors in the ETS-Software.	/	/	Pass
2.1	The KNX-System recognises if a button of the Zennio Square TMD 4 gets triggered.	Manually triggering all four buttons of the Zennio Square TMD 4.	The ETS-Software switches the status of the software button.	/	/	Pass
2.2	The Zennio Z35 shows the test data.	Configuring a test screen in the ETS-Software and displaying it on the Zennio Z35.	The Zennio Z35 correctly displays the test screen.	/	/	Pass

3.0	Proper connection of the SDC30 CO2 sensor and the microcontroller.	1. Run a I²C connection test program on the microcontroller 2. Read measurement data on the microcontroller	1. The test has to detect the sensor. 2. Some CO2, temperature, humidity and air pressure values are sent by the microcontroller.	/	/	Pass
3.1	SCD30 sensor measuring the correct values for CO2 concentration, temperature, humidity and air pressure.	**CO2:** Researching standard CO2 concentration values for the area the sensor is tested in and comparing them to the measured values. **Temperature:** Getting a reliable temperature value at time of testing with a thermometer and comparing it to the measured value. **Humidity:** Getting a reliable humidity value at time of testing with a hygrometer and comparing it to the measured value.	**Co2:** Maximum difference of 10% between the measurement and the standard value. **Temperature:** Maximum difference of 10% between the measurement and the reliable value. **Humidity:** Maximum difference of 10% between the measurement and the reliable value.	**CO2:** 440-2000 **Temp.:** 23°C **Hum.:** 40-60 %	**CO2:** 460-1600 **Temp.:** 23.1°C **Hum.:** 40-45 %	Pass
4.0	The TP UART can send the correct data to the KNX System	Display the Sensor data on the KNX system and in the Arduino's serial monitor	The Arduino and the KNX system show the same numbers	/	/	Pass
5.0	The OpenHAB system is discoverable in the LAN	Attempt to connect to the OpenHAB webpage	The user can connect to the webpage and is shown the intro screen.	/	/	Pass
5.1	The KNX system can connect to OpenHAB	The OpenHAB KNX Binding successfully connects	The KNX Binding shows "Online" in the OpenHAB Panel	/	/	Pass
5.2	The KNX system's data can be sent to OpenHAB	Create a display panel to show the sensor data	The sensor data on the OpenHAB Panel is the same as the KNX debug screen			Pass

Code & Configuration Files

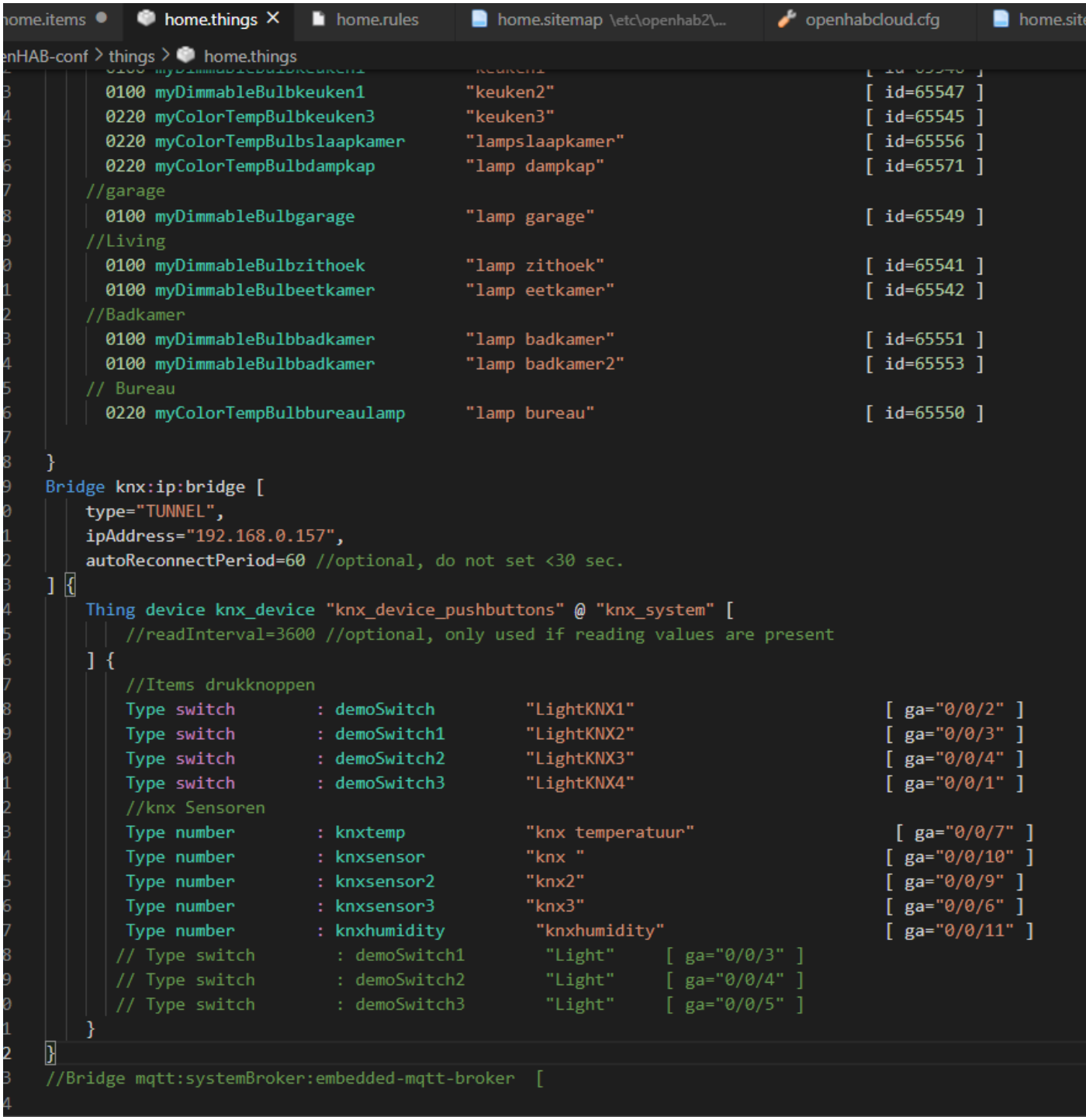
Microcontroller

This code will control the sensors, as well as send data to the KNX system using the TP-UART



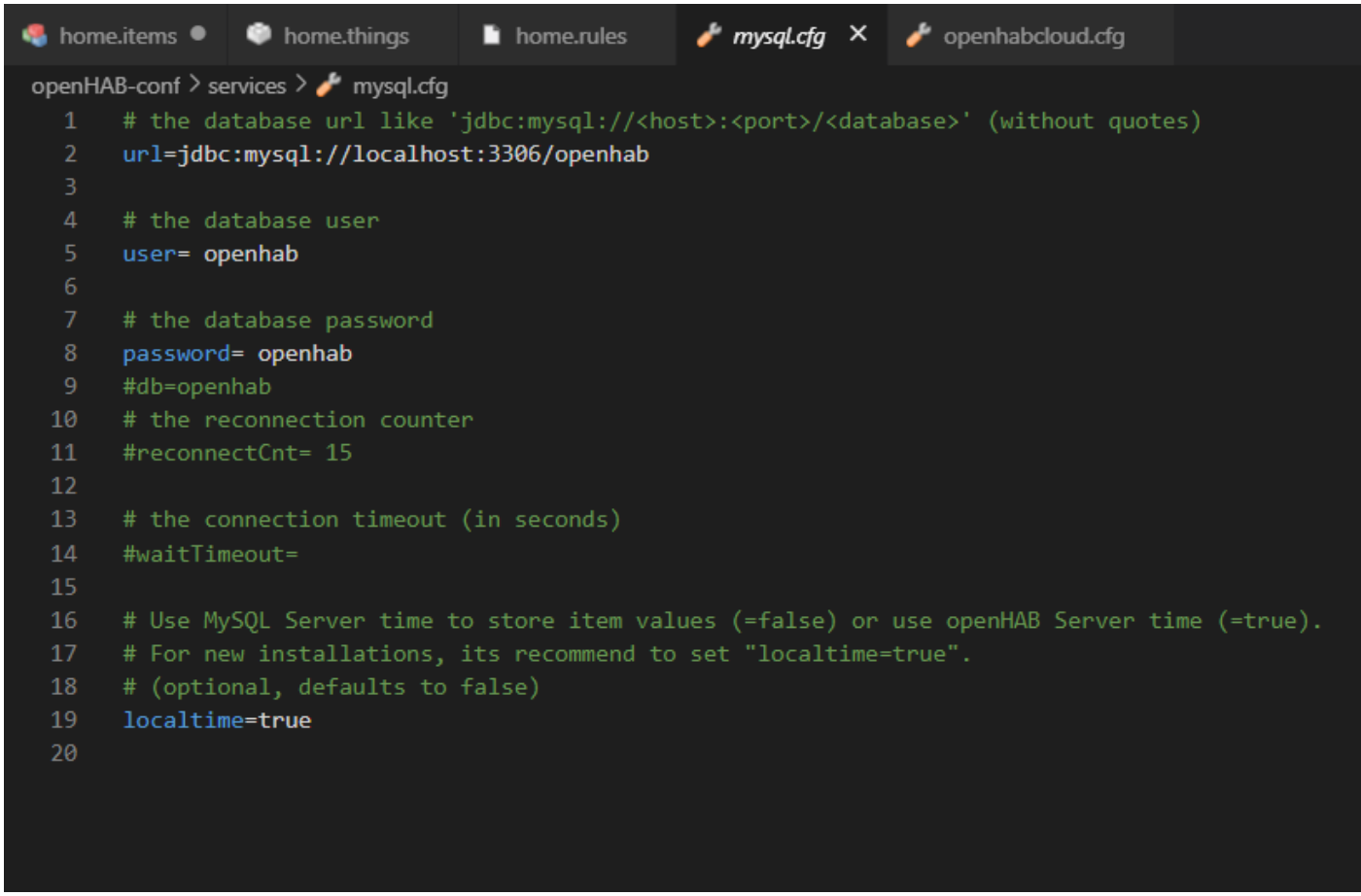
OpenHAB Configuration

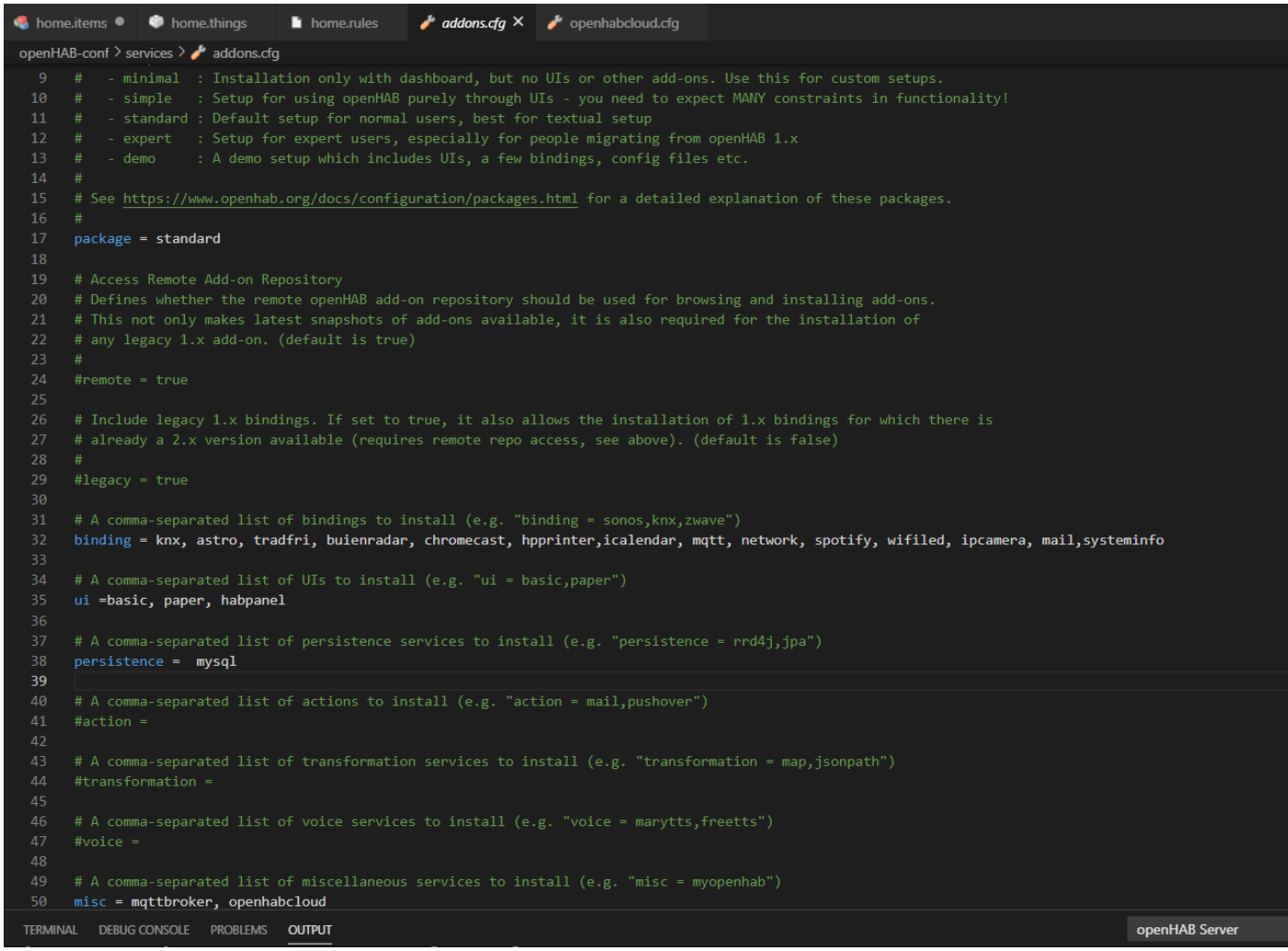
```
//items file

//items knx
Switch      KnxDevicePushbuttonsDemoSwitch     "LightKNX1"   [ "Switchable" ]                                                {channel="knx:device:bridge:k
Switch      KnxDevicePushbuttonsDemoSwitch1    "LightKNX2"   [ "Switchable" ]                                                {channel="knx:device:bridge:k
Switch      DemoSwitch2                        "LightKNX3"   [ "Switchable" ]                                                {channel="knx:device:bridge:k
Switch      DemoSwitch3                        "LightKNX4"   [ "Switchable" ]                                                {channel="knx:device:bridge:k
Group       sensorsknx                             "thermostat"                  { ga="Thermostat"}
Number      Knxtemp                             "Knxtemperatuur"               (sensorsknx)                          {channel="knx:device:bridge:knx_dev
Number      Knxsensor                           "Knx"                                                                       {channel="knx:device:bridge:k
Number      Knxsensor2                          "Knx2"                                                                      {channel="knx:device:bridge:k
Number      Knxsensor3                          "Knx3"                                                                      {channel="knx:device:bridge:k
Number      Knxhumidity                         "Knxhumidity"                                                 {channel="knx:device:bridge:knx_device:kn


//items ikea
Dimmer   Keuken1Brightness             "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbkeuken1:brightness"}
Dimmer   Keuken3Brightness             "Brightness"                      {channel="tradfri:0220:mygateway:myColorTempBulbkeuken3:brightness"}
Dimmer   Keuken3ColorTemperature       "Color temperature"    <colorwheel>   {channel="tradfri:0220:mygateway:myColorTempBulbkeuken3:color_temperature"}
Dimmer   LampBadkamerBrightness        "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbbadkamer:brightness"}
Dimmer   LampBureauBrightness          "Brightness"                      {channel="tradfri:0220:mygateway:myColorTempBulbbureaulamp:brightness"}
Dimmer   LampBureauColorTemperature    "Color temperature"    <colorwheel>   {channel="tradfri:0220:mygateway:myColorTempBulbbureaulamp:color_temperature"
Dimmer   LampDampkapBrightness         "Brightness"                      {channel="tradfri:0220:mygateway:myColorTempBulbdampkap:brightness"}
Color    LampDampkapColorTemperature   "Color temperature"    <colorwheel>   {channel="tradfri:0220:mygateway:myColorTempBulbdampkap:color_temperature"}
Dimmer   LampEetkamerBrightness        "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbeetkamer:brightness"}
Dimmer   LampGarageBrightness          "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbgarage:brightness"}
Dimmer   LampZithoekBrightness         "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbzithoek:brightness"}
Dimmer   LampslaapkamerBrightness      "Brightness"                      {channel="tradfri:0220:mygateway:myColorTempBulbslaapkamer:brightness"}
Dimmer   LampslaapkamerColorTemperature "Color temperature"   <colorwheel>   {channel="tradfri:0220:mygateway:myColorTempBulbslaapkamer:color_temperature"
Dimmer   Spotje1Brightness             "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbspot1:brightness"}
Dimmer   Spotje2Brightness             "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbspot2:brightness"}
Dimmer   Spotje3Brightness             "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbspot3:brightness"}
Dimmer   Spotje3KeukenBrightness       "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbkeukenspot3:brightness"}
Dimmer   Spotje1KeukenBrightness       "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbkeukenspot1:brightness"}
Dimmer   Spotje2KeukenBrightness       "Brightness"                      {channel="tradfri:0100:mygateway:myDimmableBulbkeukenspot2:brightness"}



//chromecast google mini
```


home.items
home.things
home.rules
home.sitemap \etc\openhab2\...
openhabcloud.cfg
home.site
enHAB-conf > things > home.things
0100 myDimmableBulbkeuken1 "keuken2" [id=65547]
0220 myColorTempBulbkeuken3 "keuken3" [id=65545]
0220 myColorTempBulbslaapkamer "lampslaapkamer" [id=65556]
0220 myColorTempBulbdampkap "lamp dampkap" [id=65571]
//garage
0100 myDimmableBulbgarage "lamp garage" [id=65549]
//Living
0100 myDimmableBulbzithoek "lamp zithoek" [id=65541]
0100 myDimmableBulbeetkamer "lamp eetkamer" [id=65542]
//Badkamer
0100 myDimmableBulbbadkamer "lamp badkamer" [id=65551]
0100 myDimmableBulbbadkamer "lamp badkamer2" [id=65553]
// Bureau
0220 myColorTempBulbbureaulamp "lamp bureau" [id=65550]
}
Bridge knx:ip:bridge [
type="TUNNEL",
ipAddress="192.168.0.157",
autoReconnectPeriod=60 //optional, do not set <30 sec.
] {
Thing device knx_device "knx_device_pushbuttons" @ "knx_system" [
//readInterval=3600 //optional, only used if reading values are present
] {
//Items drukknoppen
Type switch : demoSwitch "LightKNX1" [ga="0/0/2"]
Type switch : demoSwitch1 "LightKNX2" [ga="0/0/3"]
Type switch : demoSwitch2 "LightKNX3" [ga="0/0/4"]
Type switch : demoSwitch3 "LightKNX4" [ga="0/0/1"]
//knx Sensoren
Type number : knxtemp "knx temperatuur" [ga="0/0/7"]
Type number : knxsensor "knx " [ga="0/0/10"]
Type number : knxsensor2 "knx2" [ga="0/0/9"]
Type number : knxsensor3 "knx3" [ga="0/0/6"]
Type number : knxhumidity "knxhumidity" [ga="0/0/11"]
// Type switch : demoSwitch1 "Light" [ga="0/0/3"]
// Type switch : demoSwitch2 "Light" [ga="0/0/4"]
// Type switch : demoSwitch3 "Light" [ga="0/0/5"]
}
}
//Bridge mqtt:systemBroker:embedded-mqtt-broker [

openHAB-conf > sitemaps > home.sitemap

```
sitemap home label="Main Menu"
{
    Frame {

        Switch item=Keuken1Brightness label="Keuken"
        Switch item=Keuken3Brightness label="keuken3"
        Switch item=Keuken3ColorTemperature label="keuken3_kleur"
        Default item=LampDampkapBrightness label="dampkap"
        Default item=Spotje1Brightness label="Brightness"

        Switch item=Spotje1KeukenBrightness label="Brightness"
        Switch item=Spotje2Brightness label="Brightness"

        Switch item=Spotje2KeukenBrightness label="Brightness"
        Default item=Spotje3Brightness label="Brightness"
        Default item=Spotje3KeukenBrightness label="Brightness"
        Default item=LampZithoekBrightness label="Brightness"
        Default item=LampEetkamerBrightness label="Brightness"
        Default item=KnxDevicePushbuttonsDemoSwitch1 label="Light"
        Default item=KnxDevicePushbuttonsDemoSwitch label="Light"
        Default item=DemoSwitch3 label="Light"
        Default item=DemoSwitch2 label="Light"
        Default item=Knxtemp label="Knx temperatuur"
        Default item=Knxsensor label="Knx"
        Default item=Knxsensor2 label="Knx2"
        Default item=Knxsensor3 label="Knx3"
        Default item=Knxhumidity label="Knxhumidity"
        Number item=Vochtigheid_planten label="vochtigheid"
    }

```


home.items
home.things
home.rules
mysql.cfg
openhabcloud.cfg
openHAB-conf > services > mysql.cfg
# the database url like 'jdbc:mysql://<host>:<port>/<database>' (without quotes)
url=jdbc:mysql://localhost:3306/openhab
# the database user
user= openhab
# the database password
password= openhab
#db=openhab
the reconnection counter
#reconnectCnt= 15
the connection timeout (in seconds)
#waitTimeout=
# Use MySQL Server time to store item values (=false) or use openHAB Server time (=true).
# For new installations, its recommend to set "localtime=true".
# (optional, defaults to false)
localtime=true


home.items
home.things
home.rules
addons.cfg
openhabcloud.cfg
openHAB-conf > services > addons.cfg
- minimal : Installation only with dashboard, but no UIs or other add-ons. Use this for custom setups.
- simple : Setup for using openHAB purely through UIs - you need to expect MANY constraints in functionality!
- standard : Default setup for normal users, best for textual setup
- expert : Setup for expert users, especially for people migrating from openHAB 1.x
- demo : A demo setup which includes UIs, a few bindings, config files etc.
#
See https://www.openhab.org/docs/configuration/packages.html for a detailed explanation of these packages.
#
package = standard
# Access Remote Add-on Repository
# Defines whether the remote openHAB add-on repository should be used for browsing and installing add-ons.
# This not only makes latest snapshots of add-ons available, it is also required for the installation of
# any legacy 1.x add-on. (default is true)
#
#remote = true
Include legacy 1.x bindings. If set to true, it also allows the installation of 1.x bindings for which there is
already a 2.x version available (requires remote repo access, see above). (default is false)
#
#legacy = true
# A comma-separated list of bindings to install (e.g. "binding = sonos,knx,zwave")
binding = knx, astro, tradfri, buienradar, chromecast, hpprinter,icalendar, mqtt, network, spotify, wifiled, ipcamera, mail,systeminfo
# A comma-separated list of UIs to install (e.g. "ui = basic,paper")
ui =basic, paper, habpanel
# A comma-separated list of persistence services to install (e.g. "persistence = rrd4j,jpa")
persistence = mysql
# A comma-separated list of actions to install (e.g. "action = mail,pushover")
#action =
A comma-separated list of transformation services to install (e.g. "transformation = map,jsonpath")
#transformation =
A comma-separated list of voice services to install (e.g. "voice = marytts,freetts")
#voice =
# A comma-separated list of miscellaneous services to install (e.g. "misc = myopenhab")
misc = mqttbroker, openhabcloud
TERMINAL
DEBUG CONSOLE
PROBLEMS
OUTPUT
openHAB Server

Conclusion

During this project we created a KNX module that enables us to read out both Co2 values and the PM sensor within the KNX system. We linked this KNX system to the Openhab platform. This allowed us to use the components of the KNX system in the Openhab system. We then linked this to our own Smart home system Google assistant and Apple Home pod. This makes it possible to control the components with our voice via the Google Home Mini, smartphone, pc,...
We store the data of the entire Openhab system in a database and use this data to display the data in graphs.
Finally, we made these graphs available in the displays. This way, we can monitor and control the smarthome system via any web browser via any device. As a team we found this a very nice challenge. We worked hard on this project and were able to realize a lot. In our Demo video, which you can watch via this link https://youtu.be/Zj_WIVsVeXs, you can see for yourself what we have realized.

Sources

KNX:
https://www.knx.org/nl/
https://github.com/thorsten-gehrig/arduino-tpuart-knx-user-forum

Openhabian:
Documentation about openhab
https://www.openhab.org/docs/
Community for Openhab
https://community.openhab.org/

Setting up Things Channels and Bindings

https://www.youtube.com/watch?v=6HbGVWjL6II&list=PLstoNuAW8N-IkB48_3cXWBpKdAVHUt_wd&index=36&t=0s

https://www.youtube.com/watch?v=R-SrZvKHXdA&t=141s

Database

Installing mysql and php myadmin
https://pimylifeup.com/raspberry-pi-phpmyadmin/
https://pimylifeup.com/raspberry-pi-mysql/

Grafana

https://grafana.com/
https://www.smarthomeblog.net/openhab-persistence-grafana-dashboard/

Google Assistant

Apple HomeKit
https://www.apple.com/ios/home/

https://www.openhab.org/docs/ecosystem/google-assistant/
Complete Guide Connect Google Home (Ok Google) To OpenHAB 2
https://www.youtube.com/watch?v=vPNSFhcq3Ps&t=370s
Setup OpenHAB Cloud Connector and myopenhab.org
https://www.youtube.com/watch?v=joz5f4ejJVc&list=RDCMUC1WPn_mBd7eDmz7lMSXR5bA&index=18

Ios - Home
https://www.apple.com/ios/home/