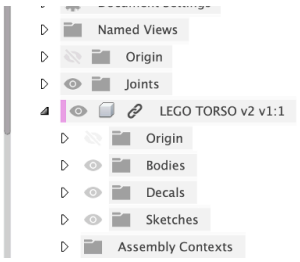


Koen Verbeeck

Embedded System Applications - Eindopdracht

Omschrijving opdracht

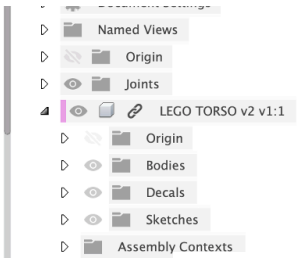
We hebben gezien hoe we een MQTT systeem opzetten met je ESP32 en Raspberry Pi en de uitgelezen data kan worden weergegeven in Grafana.
Voor de eindopdracht ga je je bestaande project uitbreiden met drukknoppen, sensoren en actuatoren om zo een mini automatisatie te bekomen.
Je bent volledig vrij in het kiezen van je hardware en aansturing maar ik verwacht een minimum aan complexiteit

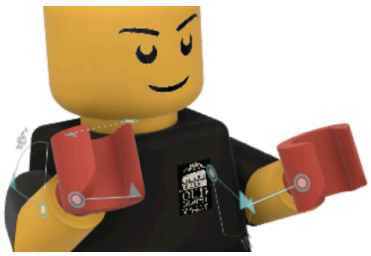
Het project bevat minimum:
- ESP32 die zowel published **EN** subscribed op verschillende topics (bidirectioneel)
- Raspberry Pi die zowel published **EN** subscribed op verschillende topics (bidirectioneel)
- Visuele weergave van de data op grafana.

Vb: Je ESP32 en Raspberry Pi hebben elk een drukknop en led aangesloten. Als je op de drukknop van de ESP drukt wordt een MQTT publish bericht gestuurd waarop de raspberry pi is ge-subscribed en laat deze een led branden. Je kan op Grafana vervolgens kijken of de led aan of uit is.
Indien je op de drukknop op de Raspberry Pi drukt zal er een MQTT publish gestuurd worden waar de ESP32 op ge-aboneerd is en zo een led laat branden op de ESP32.
Je kan temperatuur uitlezen op de ESP32 en weergeven op een LCD verbonden met de Raspberry pi

Dit alles documenteer je en upload je samen met je code op canvas. Je maakt een filmpje waarin je je project voorstelt. (! niet enkel de werking tonen maar ook gesproken uitleg bij je project !)

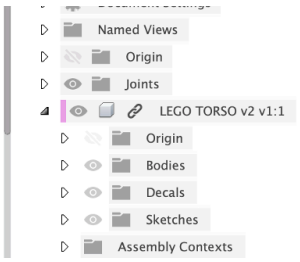
De documentatie bevat:
- Intro in MQTT en Grafana
- Je project beschrijving (welk probleem ga je oplossen?, waarom, ...)
- Hoe ben je te werk gegaan (Van installeren MQTT tot schrijven van code, testen, ...)
- Beschrijf de werking van je ESP32 code (welke stappen doorloopt het programma, ...)
- Beschrijf de werking van je Raspberry Pi code (welke stappen doorloopt het programma, ...)
- Hoe controleer je de werking van MQTT (worden pakketten doorgestuurd, ontvangen, ...)





Koen Verbeeck

Index



Koen Verbeeck

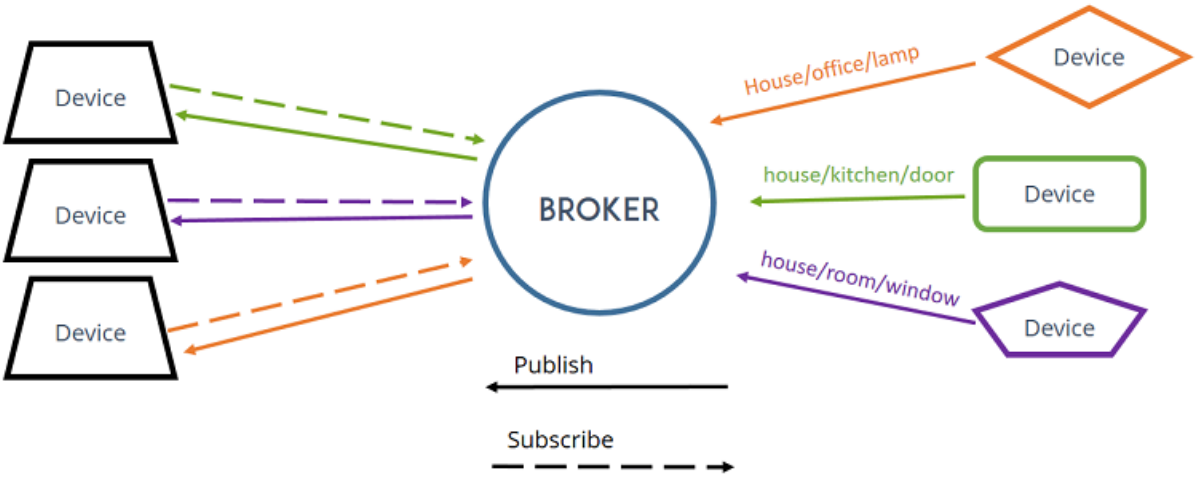
Intro

MQTT-Protocol

MQTT is een open protocol dat bestaat om op een gestandaardiseerde manier boodschappen uit te wisselen via ip-netwerken: MQTT (Message Queuing Telemetry Transport).

MQTT is ontwikkeld door IBM als een protocol dat efficiënt gebruikmaakt van de beschikbare netwerkbandbreedte en allerlei soorten data kan doorsturen. In MQTT staat de broker (de server) centraal, die de communicatie tussen zenders en ontvangers in goede banen leidt. Die zenders en ontvangers (de clients) worden in het MQTT-protocol publishers (uitgevers) en subscribers (abonnees) genoemd.

Een broker kan meerdere clients hebben en elke client kan ook zowel zenden als ontvangen. Eigenlijk werkt de broker als tussenpartij zodat zenders en ontvangers niet van elkaars bestaan hoeven af te weten om boodschappen uit te wisselen.



Publisher

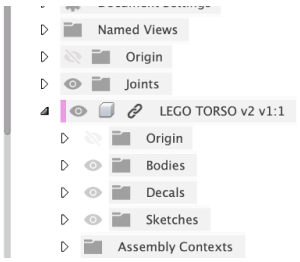
- Stuurt informatie naar de Broker
- Bv. Weerstation stuurt temperatuur door

Subscriber

- Krijgt informatie binnen van Broker
- Bv. Dashboard dat de temperatuur toont
- Heeft geen info over de publisher nodig

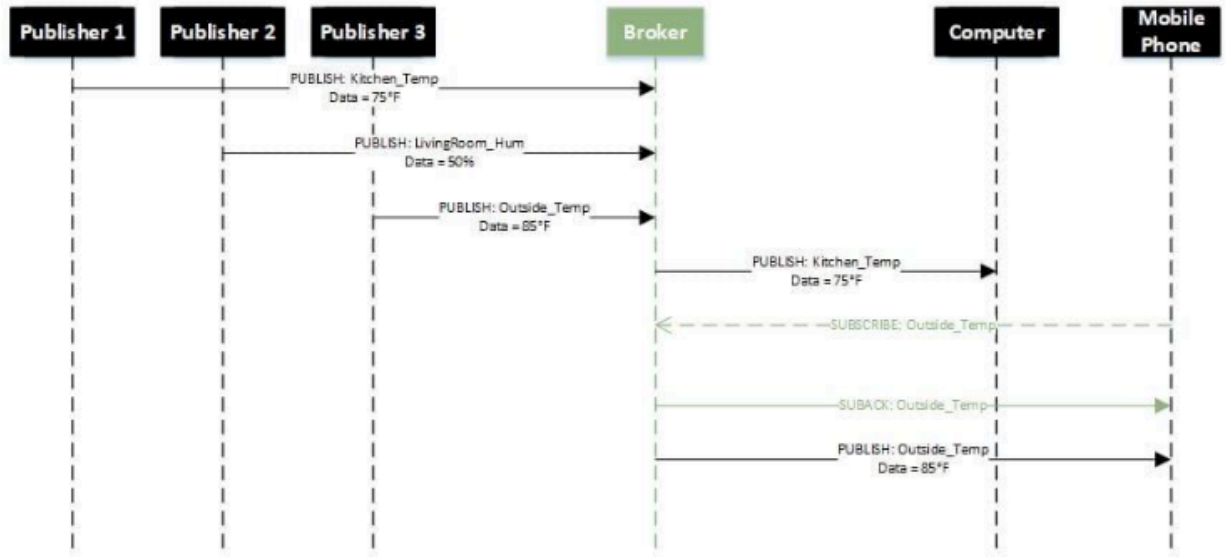
Broker – Server

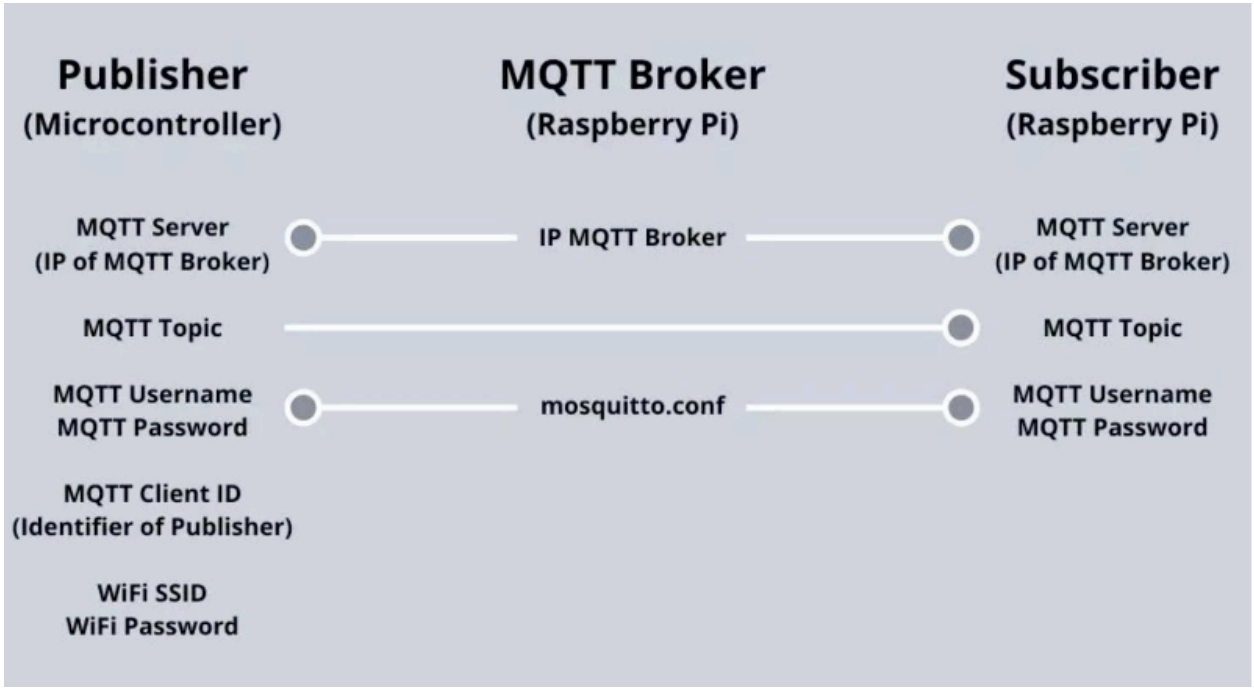
- Slaat het laatste bericht gepubliceerd in een bepaalde topic op

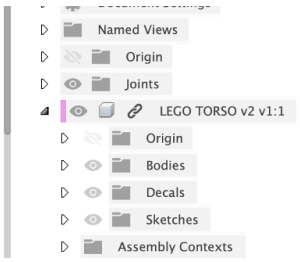


Koen Verbeeck

Elke MQTT-boodschap heeft een onderwerp (topic) en een inhoud (payload). Een client die in een onderwerp geïnteresseerd is, abonneert zich daarop bij de broker. Een client die een boodschap wil sturen, publiceert zijn inhoud door een boodschap met een specifiek onderwerp naar de broker te sturen. Zodra de broker een boodschap ontvangt, stuurt hij die door naar alle clients die op dat onderwerp zijn geabonneerd.







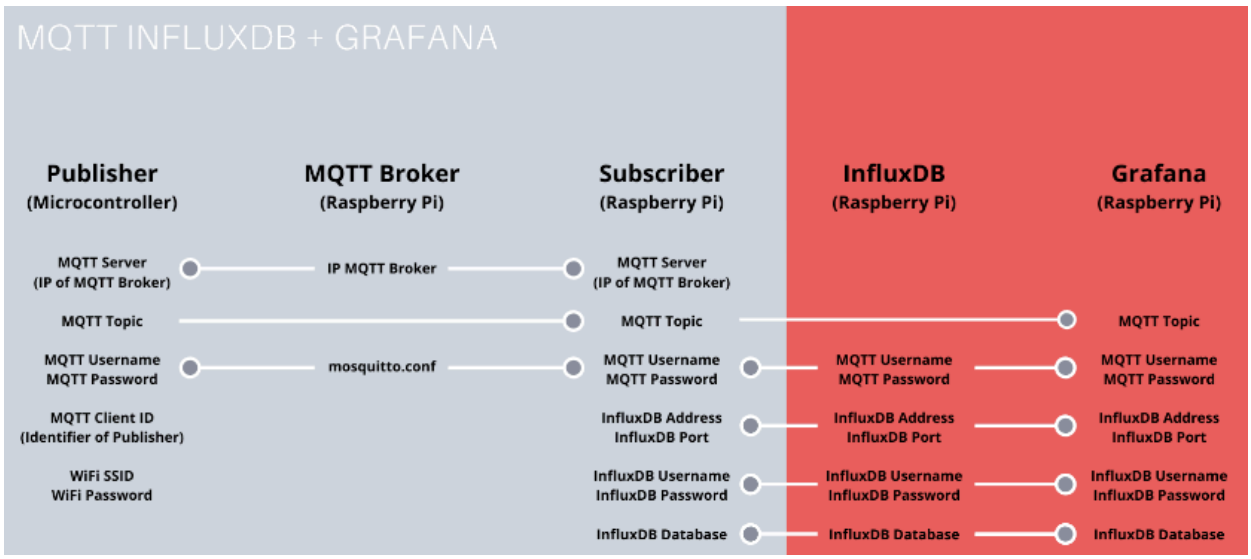
Koen Verbeeck

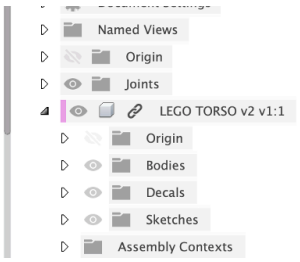
Grafana

Grafana is een open-sourceplatform voor gegevensbewaking, -analyse en -visualisatie dat wordt geleverd met een webserver waarmee het overal toegankelijk is.

In de webinterface kunnen gebruikers Grafana-dashboards met panelen maken om statistieken in de loop van de tijd weer te geven.

InfluxDB is een open source database die specifiek is gemaakt voor het opslaan van tijd gebonden data. Grafana is ideaal voor het visualiseren van al die data.



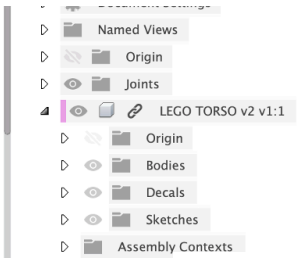


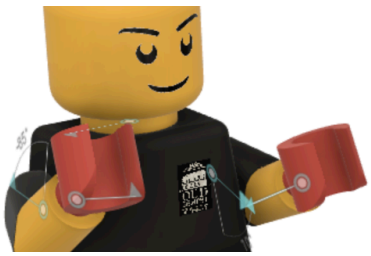
Koen Verbeeck

Beschrijving Project

Het doel van dit project is een serre (green-house) te automatiseren. Initieel was het de bedoeling om een simpel project te maken waardoor mijn vrouw op de hoogte gebracht zou worden wanneer de plantjes dorst zouden hebben. Dit project groeide al snel uit naar een volledige automatisatie waardoor ik overgeschakeld ben op het volledig automatiseren van een serre. Het project meet en voert acties uit op onderstaande zaken:

- Controle van temperatuur en vochtigheid
- Controle van de bodem vochtigheid
- Controle van regenval
- Controle van Lichtintensiteit
- Lokale weergave van data (Huidige Data) door middel van OLED, servo's en LEDs
- Grafana (historische en huidige data en alarmen)
- Het automatisch bewateren van de planten
- Het automatisch openen van het serre dak bij te hoge temperaturen
- Het automatisch verwarmen van de serre
- Automatische verlichting
-
- Initieel was het de bedoeling om de sensoren te verdelen over 2 ESP32s, door een onoplosbaar issue waarbij de 2 ESP32s niet gelijktijdig kunnen subscriben op eenzelfde topic heb ik dit plan moeten laten varen en dus alles op 1 ESP32 gezet.





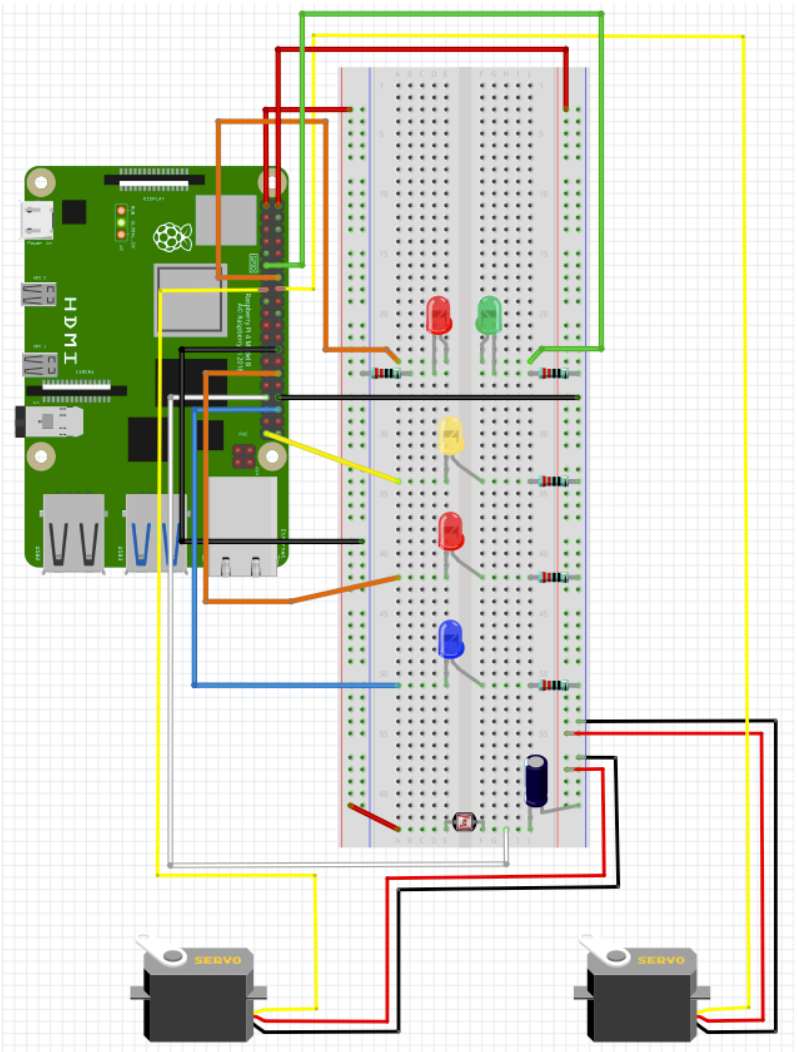
Koen Verbeeck

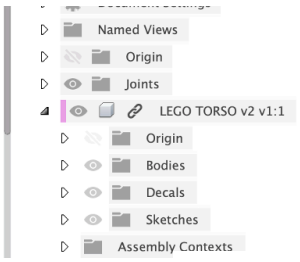
Schematics

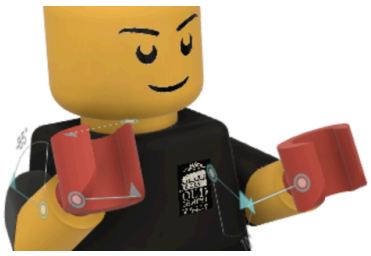
Raspberry Pi 4B

Deze zal fungeren als MQTT-Broker, MQTT-Subscriber en MQTT-Publisher, hier zal tevens een Python Script op draaien als bridge tussen Broker en de Influx Database.

- 1 x Raspberry Pi 4B
- 5 x LEDs
- 5 x 220 Ohm weerstanden
- 1 x LDR
- 1 x 10uF Capacitor
- 2 x Servo Motors
- 1 x Breadboard
- Assortiment Dupont male - female draden





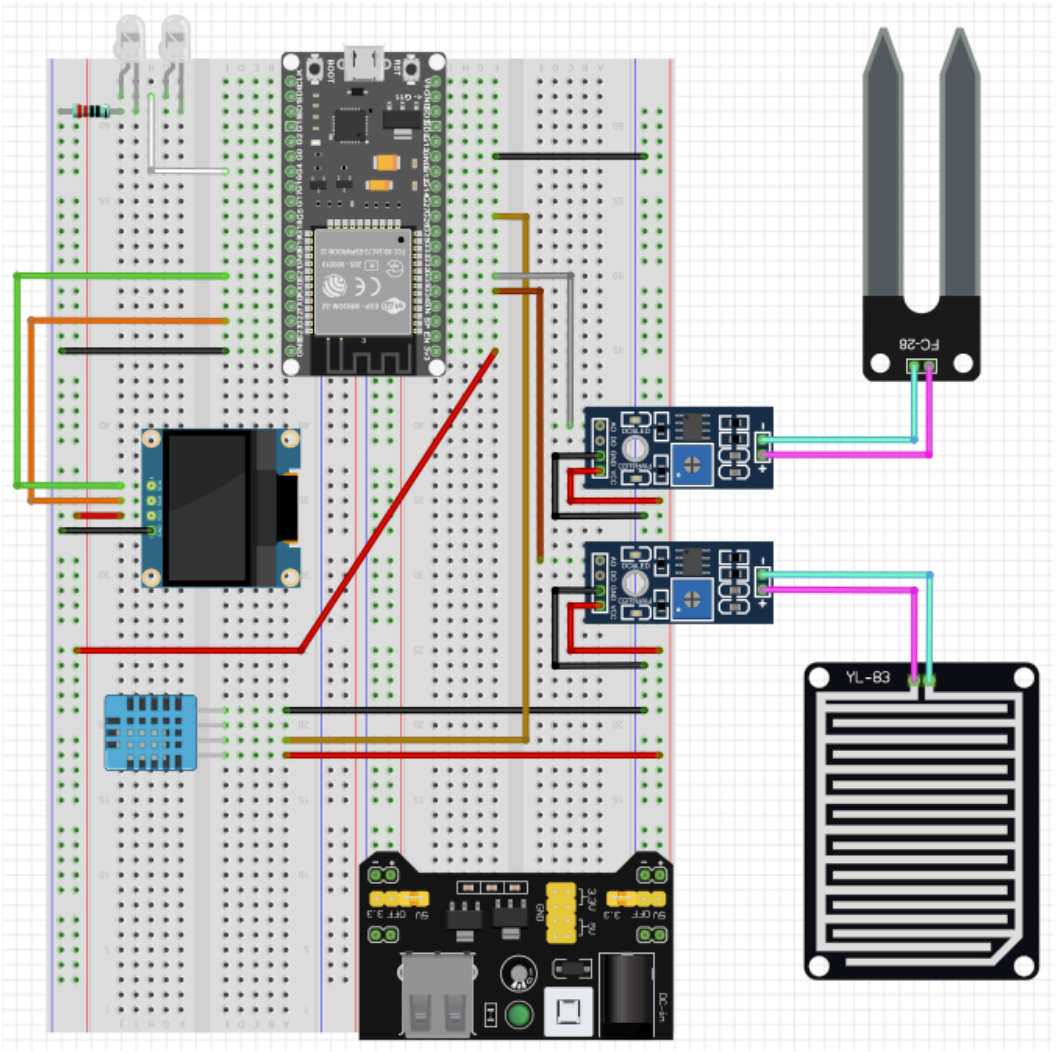


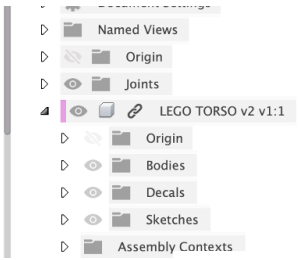
Koen Verbeeck

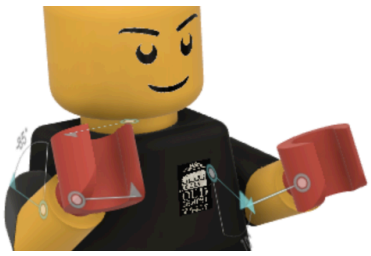
ESP32

Deze zal fungeren als MQTT-Subscriber en MQTT-Publisher.

- 1 x ESP-WROOM-32
- 1 x OLED 0,96” 128x64 Display
- 1 x DHT11
- 1 x 150 Ohm weerstand
- 2 x LEDs
- 1 x YL-83 Control Board + Rain Sensor Detection Board
- 1 x YL-83 Control Board + Hygrometer
- 2 x Breadboard
- 1 x YwRobot Power Supply 5V
- Assortiment Dupont male - male draden
- Assortiment Dupont male - female draden



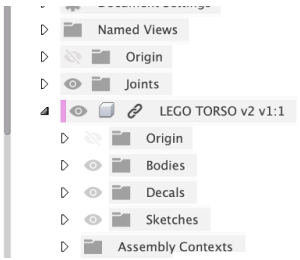


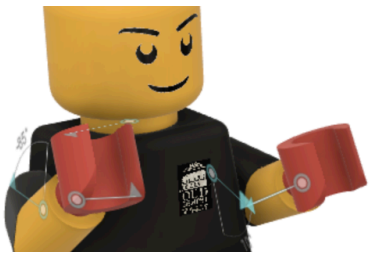


Koen Verbeeck

Stappen Project

1. Het opzetten van de Broker (Raspberry Pi) - Raspbian
2. Het instaleren van de laatste versie van Python en nodige libraries op de Broker
3. Het installeren van Influx Database en Grafana
4. Sytemctl: Automatisch opstarten van Influx en Grafana bij herstart Raspberry
5. Influx: aanmaken nieuwe database met de nodige rechten
6. Grafana: configureren + aanmaken van de nodige Dashboards en panels en linken met de Influx Database
7. Het configureren van een Bridge.py script dat er voor zal zorgen dat alles wat de Broker Published in de Influx Database terechtkomt en dus ook in Grafana
8. Het automatisch runnen van het Bridge script bij een reboot van de Raspberry Pi via een bashscript en Crontab
9. Raspberry Pi: Het installeren van de hardware volgens schema
10. Het schrijven en testen van de het Pyhon subscriber script
11. Het schrijven en testen van de het Pyhon publisher script
12. ESP32: Het installeren van de hardware volgens schema
13. Het schrijven en testen van de het C++ publisher/subscriber script
14. Testen en aanpassen waar nodig, komt de data juist binnen in Influx/Grafana en worden de juiste acties uitgevoegd op de data, verbind de ESP32 met Wi-Fi en de MQTT Broker, verbind de MQTT Broker met Wi-Fi, werken alles scripts naar behoren?





Koen Verbeeck

Werking ESP32 Code

Zie de comments voor uitgebreide uitleg.

Importeren van de libraries

Rain_Soil_DHT11_ESP32_MQTT_to_Influx_with_OLED_v2

```
// Include the necessary libraries
#include <DHT.h> // DHT11 sensor
#include <WiFi.h> // Wi-Fi Client
#include <ESP32Ping.h> // Ping Client for testing
#include <PubSubClient.h> // MQTT Client
#include <Adafruit_GFX.h> // Supporting library for OLED screen
#include <Adafruit_SSD1306.h> // OLED screen
#include <Wire.h> // I2C bus
```

Definiëren van de sensoren, actuatoren, LEDs en schermen

```
/*
   Lighthing
*/
#define whiteLight 4

/*
   DHT11 Sensor (Temperature and Humidity)
*/
#define DHTPIN 26  // for ESP32
#define DHTTYPE DHT11  // DHT11 or DHT22

DHT dht(DHTPIN, DHTTYPE);

/*
   Soil Moisture Sensor YL-69 or HL-69
*/
#define soilPin 35
// you can adjust the threshold value, 0 - 4095 (12 bits), lower value is wetter.
int soilThresholdValue = 3000;

/*
   Rain Sensor FC-37 or YL-83
*/
#define rainPin 34
// you can adjust the threshold value, 0 - 4095 (12 bits), lower value is wetter.
int rainThresholdValue = 2000;

/* OLED 128x64 Monochrone Screen
*/
#define SCREEN_WIDTH 128 // Width of the screen
#define SCREEN_HEIGHT 64 // Height of the screen

// Initialize the OLED screen
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

Bitmap voor het start-up scherm van de OLED

```
// BITMAP Koen_LEGO start-up screen
static const uint8_t image_data_KoenLego2[1024] = {
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xf0, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xf7, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```

Initialiseren en definieeren van de MQTT Client

```
/*
   MQTT client - ESP32
*/
const char* ssid = "BLYNK";
const char* password = "Bucky196";

const char* lighting_topic = "home/RPi4B/Lighting"; // MQTT Topic

const char* mqttServer = "192.168.10.124"; // MQTT Broker // KoenRPI4B.local
const int mqttPort = 1883; // MQTT Port
const char* mqttUser = "koen"; // Username Mosquitto
const char* mqttPassword = "k03n123"; // Password Mosquitto
const char* clientID = "ESP32_SOIL"; // MQTT client ID

WiFiClient espClient;
PubSubClient client(espClient);
```

IP address dat je wil pingen (Broker IP voor testen)

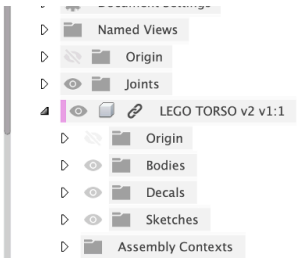
```
const IPAddress remote_ip(192, 168, 10, 124); // IP address of MQTT Broker for ping >> KoenRPI4B.local
```

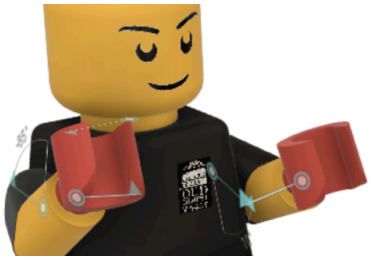
Functie voor het connecteren met je Wi-Fi Access Point

```
void setup_wifi() {
  // Delay for 10ms
  delay(10);
  // Print an empty line
  Serial.println();
  // Print to which SSID ESP32 is trying to connect
  Serial.print(F("Connecting to "));
  Serial.println(ssid);

  // Initialize Wi-Fi on ESP32 and connect to the configured WLAN
  WiFi.begin(ssid, password);

  // As long as ESP32 is not connected to the configured WLAN
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println(F("Trying to Connect"));
  }
  Serial.print(F("\nConnected to the "));
  Serial.print(ssid);
  Serial.println(F(" Wi-Fi network\n"));
}
```

Functie voor het testen van de signaalsterkte van het Wi-Fi AP

```
void RSSI_Info() {
  // Request Wi-Fi Signal Strength 3 times
  for (int i = 0; i < 3; i++) {
    Serial.print(F("RSSI: "));
    Serial.print(WiFi.RSSI());
    Serial.println(F("dBm"));
    delay(1000);
  }
  Serial.println();
}
```





Koen Verbeeck

Functie voor het pingen van de MQTT Broker

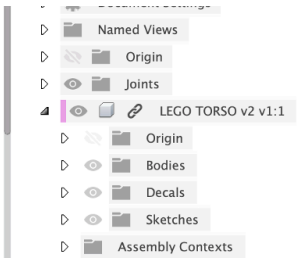
```
void ping_MQTT_Broker() {
  // Send 3 pings to the MQTT Broker
  Serial.print(F("Pinging MQTT Broker 3 times: "));
  Serial.println(remote_ip);
  for (int i = 0; i < 3; i++) {
    if (Ping.ping(remote_ip)) {
      Serial.println(F("Success !!"));
    }
    else {
      Serial.println(F("No reply"));
    }
  }
  Serial.println();
}
```

Functie voor het weergeven van nuttige DHCP info zoals je IP address

```
void DHCP_Info() {
  // Print the ESP32 IP info received from DHCP
  Serial.print(F("IP address: "));
  Serial.println(WiFi.localIP());
  Serial.print(F("Subnet Mask: "));
  Serial.println(WiFi.subnetMask());
  Serial.print(F("Default Gateway: "));
  Serial.println(WiFi.gatewayIP());
  Serial.println();
}
```

Callback functie voor het verwerken van inkomende MQTT berichten. Hier zullen ook de LEDs aangestuurd worden

```
// Callback function when an MQTT message is received
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);

  //Serial.print("Message:");
  //for (int i = 0; i < length; i++) {
  //  Serial.print((char)payload[i]);
  //}

  Serial.print((char)payload[0]);

  if ((char)payload[0] == '1') {
    digitalWrite(whiteLight, HIGH);
  }
  else {
    digitalWrite(whiteLight, LOW);
  }

  Serial.println();
  Serial.println("-----------------------");
}
```

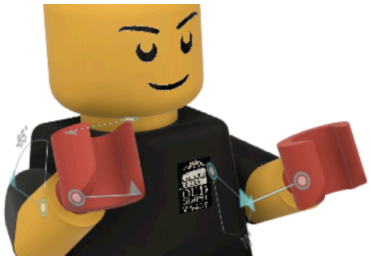


Koen Verbeeck

Initialiseren van de sensoren, actuatoren, LEDs en schermen

```
220 void setup() {
221
222   Serial.begin(115200);
223
224   /*
225     Lighthing
226   */
227   pinMode(whiteLight, OUTPUT);
228
229   /* OLED 128x64 Monochrone Screen
230   */
231   // Initialize the OLED screen on I2C address 0x3C
232   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
233     Serial.println(F("SSD1306 allocation failed"));
234     for (;;);
235   }
236   delay(2000); // Pause for 2 seconds
237
238   // Clear the buffer.
239   display.clearDisplay();
240
241   // Draw Start-up screen bitmap on the screen
242   // const tImage KoenLego2 = { image_data_KoenLego2, 128, 64, 8 }; >> info from bitmap (LCD Image Converter)
243   display.drawBitmap(0, 0, image_data_KoenLego2, 128, 64, 1);
244   display.display();
```

```
246   /*
247     Soil Moisture Sensor YL-69 or HL-69
248   */
249   pinMode(soilPin, INPUT);
250   pinMode(red_soil, OUTPUT);
251   pinMode(green_soil, OUTPUT);
252   digitalWrite(green_soil, LOW);
253   digitalWrite(red_soil, LOW);
254
255   /*
256     Rain Sensor FC-37 or YL-83
257   */
258   pinMode(rainPin, INPUT);
259   pinMode(red_rain, OUTPUT);
260   pinMode(green_rain, OUTPUT);
261   digitalWrite(green_rain, LOW);
262   digitalWrite(red_rain, LOW);
263
264   /*
265     DHT11 Sensor (Temperature and Humidity)
266   */
267   dht.begin();
```



Initialiseren van de MQTT Client + uitvoeren voorgaande functies

```
/*
  Initialize MQTT client - ESP32
*/
setup_wifi();
RSSI_Info();
DHCP_Info();
// ping_MQTT_Broker();
client.setServer(mqttServer, mqttPort);

// set a callback function for when data is received from broker
client.setCallback(callback);

// connect to broker
while (!client.connected()) {
  Serial.println("Connecting to MQTT...");
  if (client.connect("ESP32Client", mqttUser, mqttPassword )) {
    Serial.println("connected");
  }
  else {
    /*
        int state ()
        Returns the current state of the client. If a connection attempt fails,
        this can be used to get more information about the failure.

        All of the values have corresponding constants defined in PubSubClient.h.

        Returns:
        -4 : MQTT_CONNECTION_TIMEOUT - the server didn't respond within the keepalive time
        -3 : MQTT_CONNECTION_LOST - the network connection was broken
        -2 : MQTT_CONNECT_FAILED - the network connection failed
        -1 : MQTT_DISCONNECTED - the client is disconnected cleanly
        0 : MQTT_CONNECTED - the client is connected
        1 : MQTT_CONNECT_BAD_PROTOCOL - the server doesn't support the requested version of MQTT
        2 : MQTT_CONNECT_BAD_CLIENT_ID - the server rejected the client identifier
        3 : MQTT_CONNECT_UNAVAILABLE - the server was unable to accept the connection
        4 : MQTT_CONNECT_BAD_CREDENTIALS - the username/password were rejected
        5 : MQTT_CONNECT_UNAUTHORIZED - the client was not authorized to connect
    */
    Serial.print("failed with state ");
    Serial.println(client.state());
    delay(2000);
  }
}
```

Subscriben op de nodige MQTT Topics

```
  // subscribe to topic
  client.subscribe(lighting_topic);
}
```

De client.loop() zal de connectie met de Broker in tact houden alsook luisteren naar inkomende MQTT berichten.

```
/*
  MQTT client - ESP32
*/
client.loop();

```

Uitlezen van de DHT11 sensor

```
/*
  DHT11 Sensor (Temperature and Humidity)
*/
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();

// check if returns are valid and print the sensor data
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("Failed to read from DHT");
}
else {
  Serial.print("Temperature: ");
  Serial.print(temperature - 2); // Temperature correction
  Serial.println("°C");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");
}
```

Uitgelezen data omvormen naar een char array om de data te kunnen versturen via MQTT in een Topic

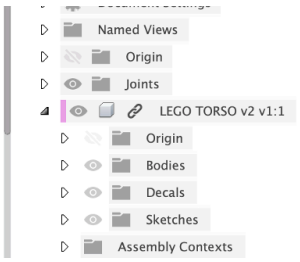
```
// Covert float (temperature) to char array
// Must be at least 10 char for the float payload
// home/esp32/dht11_temp b'22.400000'
char temp_array[10];
sprintf(temp_array, "%f", temperature - 2); // Temperature correction
// Covert float (humidity) to char array
// Must be at least 10 char for the float payload
// home/esp32/dht11_humi b'54.000000'
char humi_array[10];
sprintf(humi_array, "%f", humidity);

// Only publish data if the sensor is sending actual data (connected)
if (not(isnan(temperature) || isnan(humidity))) {
  client.publish("home/esp32/dht11_temp", temp_array);
  client.publish("home/esp32/dht11_humi", humi_array);
}
```

Uitlezen van de rain en moisture soil sensor

```
/*
  Soil Moiture Sensor YL-69 or HL-69
*/
// read the input on analog pin 35 (ADC 1 Channel 7):
int soilVal = analogRead(soilPin);
Serial.print("Soil Moisture Sensor value is: ");
Serial.print(soilVal);

if (soilVal > soilThresholdValue) {
  Serial.println(" - Time to water your plant");
}
else {
  Serial.println(" - Plant doesn't need watering");
}

/*
  Rain Sensor FC-37 or YL-83
*/
// read the input on analog pin 34 (ADC 1 Channel 6):
int rainVal = analogRead(rainPin);
Serial.print("Rain Sensor value is: ");
Serial.print(rainVal);
if (rainVal < rainThresholdValue) {
  Serial.println(" - It's raining");
}
else {
  Serial.println(" - It's dry");
}
```

Uitgelezen data omvormen naar een char array om de data te kunnen versturen via MQTT in een Topic

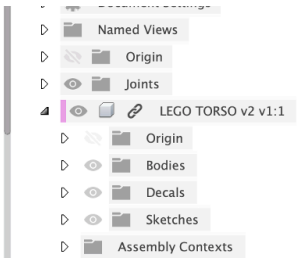
```
// Covert integer (soilVal) to char array
// Must be at least 10 char for the unsigned int (%u) payload
// home/esp32/dht11_temp b'22.400000'
char soil_array[10];
sprintf(soil_array, "%u", soilVal);
// Covert integer (rainVal) to char array
// Must be at least 10 char for the unsigned int (%u) payload
// home/esp32/dht11_humi b'54.000000'
char rain_array[10];
sprintf(rain_array, "%u", rainVal);

// Publish rain and soil sensors data
client.publish("home/esp32/soil", soil_array);
client.publish("home/esp32/rain", rain_array);
```



Koen Verbeeck

Data weergeven op het OLED scherm

```
399  /*
400   * Display the data on the OLED screen
401   */
402  // Clear the OLED buffer
403  display.clearDisplay();
404  // Set the text color to white,
405  // failing to set this wil give you a black screen
406  display.setTextColor(WHITE);
407  display.invertDisplay(false);
408  // display current temperature in °C
409  display.setFont();
410  display.setTextSize(1);
411  display.setCursor(0, 0);
412  display.print("Curr. Temp.: ");
413  display.print(temperature - 2); // Read temperature minus correction
414  display.print(" ");
415  display.cp437(true);
416  display.write(167);
417  display.print("C");
418  // display current humidity in %
419  display.setCursor(0, 10);
420  display.print("Curr. Hum:");
421  display.print(humidity);
422  display.print(" ");
423  display.print("%");
424  // display current status Soil Moisture Sensor
425  display.setCursor(0, 20);
426  display.print("Status Soil Sensor:");
427  display.setCursor(0, 30);
428  display.print(soilVal);
429  display.print(" ");
430  if (soilVal > soilThresholdValue) {
431    display.print("Soil is Dry");
432  }
433  else {
434    display.print("Soil is wet");
435  }
436  // display current status Rain Sensor
437  display.setCursor(0, 40);
438  display.print("Status Rain Sensor:");
439  display.setCursor(0, 50);
440  display.print(rainVal);
441  display.print(" ");
442  if (rainVal < rainThresholdValue) {
443    display.print("It's raining");
444  }
445  else {
446    display.print("It's dry");
447  }
450  // "display.display()" is NOT necessary after every single drawing command,
451  // you can batch up a bunch of drawing operations and then update the screen
452  // all at once by calling "display.display()".
453  display.display();
454
455  // client.disconnect();  // disconnect from the MQTT broker
456  // Serial.println("Disconnecting from MQTT...");
457  delay(1000 * 1); // Wait for 2 seconds
458 }
```



Koen Verbeeck

Werking RPi4b Code

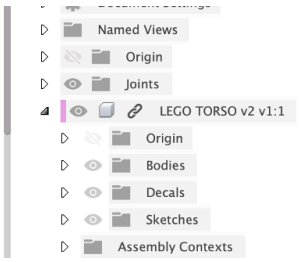
Bridge Influx

Deze code zal luisteren naar berichten die de Broker Published, het enige wat we hier dienen aan te passen is onderstaande data die we geconfigureerd hebben in de Broker en Influx

```
INFLUXDB_ADDRESS = '127.0.0.1'
INFLUXDB_USER = 'koen'
INFLUXDB_PASSWORD = [redacted]
INFLUXDB_DATABASE = 'garden'

MQTT_ADDRESS = '127.0.0.1'
MQTT_USER = 'koen'
MQTT_PASSWORD = [redacted]
MQTT_TOPIC = 'home/+/+'
MQTT_REGEX = 'home/([^/]+)/([^/]+)'
MQTT_CLIENT_ID = 'MQTTInfluxDBBridge'
```

Publisher

```
# Library to control MQTT
import paho.mqtt.client as mqtt
# The Capacitor will block the flow of current to ground when
# fully charged !!!
from gpiozero import LightSensor

from time import sleep

# GPIO.add_event_detect(PIN, GPIO.BOTH, handle) is generating error:
# RuntimeError: Failed to add edge detection
#
# Solution: This issue is because an unknown conflict using Pin7 (GPIO4)
#which appears to have a secondary function of GCLK. Moving it to GPIO13,
# which has no secondary functions allows it to work,
sensor = LightSensor(13)

# Define Variables
# broker is located on this machine (loopback address)
MQTT_BROKER = "127.0.0.1"
MQTT_PORT = 1883
MQTT_KEEPALIVE_INTERVAL = 60
MQTT_TOPIC = "helloTopic"
MQTT_MSG = "hello MQTT"

# Define on_publish event function
def on_publish(client, userdata, mid):
    print("Message Published...")
```

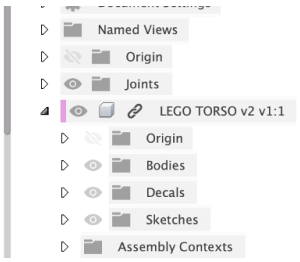


Koen Verbeeck

```python
while True:
    # Create instance of client
    client = mqtt.Client()
    # Define username and password stored in Broker password file
    client.username_pw_set("koen", "k03n123")
    # Register publish callback function
    client.on_publish = on_publish
    # Connect to (broker, port, keepalive-time)
    client.connect(MQTT_BROKER, MQTT_PORT, MQTT_KEEPALIVE_INTERVAL)

    sensorVal = sensor.value
    print(sensorVal)

    if float(sensorVal < 0.7):
        print("It's Dark")
        client.publish("home/RPi4B/Lighting", 1)  # publish
    elif float(sensorVal >= 0.7):
        print("It's Light")
        client.publish("home/RPi4B/Lighting", 0)  # publish

    sleep(1) # Depends on keep-alive time?
    # Disconnect from MQTT_Broker
    client.disconnect()
```

Subscriber

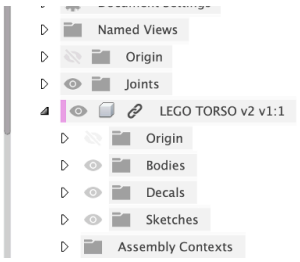
```python
# Library to control MQTT
import paho.mqtt.client as mqtt
# Library to control the LEDs
from gpiozero import LED
# Library to control Servo's
from gpiozero import AngularServo

#Temperature
red_temp = LED(27)  # red temp. alarm LED
green_temp = LED(18)  # green temp. status LED
tempThresholdValueHigh = 25.00  # °C
tempThresholdValueLow = 20.00  # °C

# Soil Moisture
# you can adjust the threshold value, 0 - 4095 (12 bits), lower value is wetter.
soilThresholdValue = 3000
servo_soil = AngularServo(22, min_angle=0, max_angle=180)  # Servo(22)

#Rain
# you can adjust the threshold value, 0 - 4095 (12 bits), lower value is wetter.
rainThresholdValue = 2000
servo_rain = AngularServo(23, min_angle=0, max_angle=180)  # Servo(23)

# Greenhouse Dynamic Roof
yellow_roof = LED(21)  # Dynamic Roof open is on, closed is off.

# Heating
red_heating = LED(5)  # Heating on is on, heating off is off.
```



Koen Verbeeck

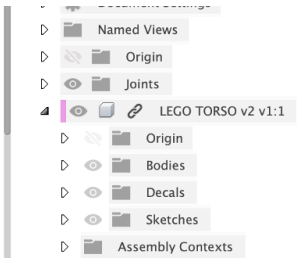
```python
# Sprinklers
blue_sprinklers = LED(26)  # Sprinklers on is on, sprinklers off, is off.

# The callback for when the client connects to the broker
def on_connect(client, userdata, flags, rc):
    # Print result of connection attempt
    # Connection Return Codes:
    # 0: Connection successful
    # 1: Connection refused - incorrect protocol version
    # 2: Connection refused - invalid client identifier
    # 3: Connection refused - server unavailable
    # 4: Connection refused - bad username or password
    # 5: Connection refused - not authorised
    # 6-255: Currently unused.
    if rc == 0:
        print("\nConnection is good \nReturned code =", rc)
    else:
        print("Bad connection Returned code= ", rc)
    # Subscribe to the topics by using a wild card,
    # receive any messages published in it.
    client.subscribe("home/esp32/#")


```

```python
# The callback for when the client connects to the broker
def on_connect(client, userdata, flags, rc):
    # Print result of connection attempt
    # Connection Return Codes:
    # 0: Connection successful
    # 1: Connection refused - incorrect protocol version
    # 2: Connection refused - invalid client identifier
    # 3: Connection refused - server unavailable
    # 4: Connection refused - bad username or password
    # 5: Connection refused - not authorised
    # 6-255: Currently unused.
    if rc == 0:
        print("\nConnection is good \nReturned code =", rc)
    else:
        print("Bad connection Returned code= ", rc)
    # Subscribe to the topics by using a wild card,
    # receive any messages published in it.
    client.subscribe("home/esp32/#")
```



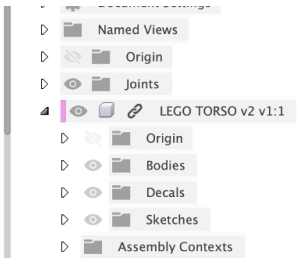
Koen Verbeeck

```python
def on_message(client, userdata, msg):
    # Without the command below, the message will be decoded in Bytes ('bXXX'),
    # so we need to convert it to UTF-8 before using it.
    msg.payload = msg.payload.decode("utf-8")
    # Print a received msg
    print("Message received >>> " + msg.topic + " >>> " + str(msg.payload))
    print("Topic is: " + msg.topic)
    print("Payload is: " + msg.payload)

    # Temperature Control
    if msg.topic == "home/esp32/dht11_temp":
        if float(msg.payload) > tempThresholdValueHigh:
            print(" - To hot, Opening Greenhouse Roof")
            green_temp.off()
            red_temp.on()
            yellow_roof.on()
            red_heating.off()
        elif float(msg.payload) < tempThresholdValueLow:
            print(" - To cold, Greenhouse Heating On")
            green_temp.off()
            red_temp.on()
            yellow_roof.off()
            red_heating.on()
        else:
            print(" - Temperature is OK")
            green_temp.on()
            red_temp.off()
            yellow_roof.off()
            red_heating.off()

    # Soil Control
    if msg.topic == "home/esp32/soil":
        if int(msg.payload) > soilThresholdValue:
            print(" - Watering your plant")
            servo_soil.angle = 0
            blue_sprinklers.on()
        else:
            print(" - Plants don't need watering")
            servo_soil.angle = 180
            blue_sprinklers.off()

    # Rain Control
    if msg.topic == "home/esp32/rain":
        if int(msg.payload) < rainThresholdValue:
            print(" - It's raining")
            servo_rain.angle = 180
        else:
            print(" - It's dry")
            servo_rain.angle = 0
```



Koen Verbeeck

```
104
105  # broker is located on this machine (loopback address)
106  broker = "127.0.0.1"
107  # Create instance of client
108  # !!!!! with client ID “digi_mqtt_test” !!!!
109  client = mqtt.Client()
110  # Define callback function for successful connection
111  client.on_connect = on_connect
112  # Define callback function for receipt of a message
113  client.on_message = on_message
114  # Define username and password stored in Broker password file
115  client.username_pw_set("koen", "k03n123")
116  # Connect to (broker, port, keepalive-time)
117  client.connect(broker, 1883, 60)
118  # Start networking daemon
119  client.loop_forever()
120
```



Koen Verbeeck

Filmpje Werking

https://www.youtube.com/watch?v=sMnkvHMmoqA