Low-code
TETRA-project

Link naar de projectpagina

Case zelf gebruiken

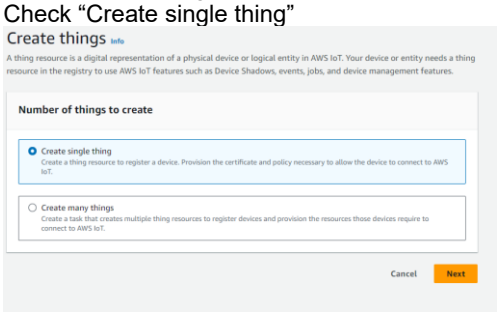
Getting started

In deze case wordt en ESP32 gebruikt voor het inlezen van analoge signalen en het sturen van digitale uitgangen. De ESP32 wordt via Wifi verbonden aan een netwerk. De gelezen waarden en het sturen van de digitale uitgangen gebeurt via AWS IoT. De communicatie verloopt over versleutelde MQTT aan de hand van certificaten

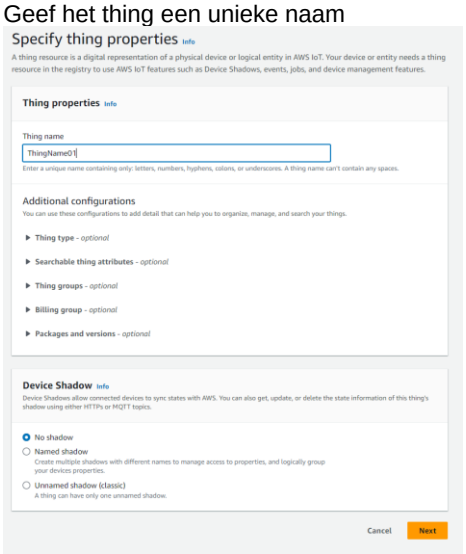
AWS
IoT Core connective

Om de ESP32 te laten communiceren met AWS IoT dient eerst een “thing” en certificaat aangemaakt te worden in AWS

- Open “IoT Core” service
- Open “Things”
- Open “Create things”
- Check “Create single thing”



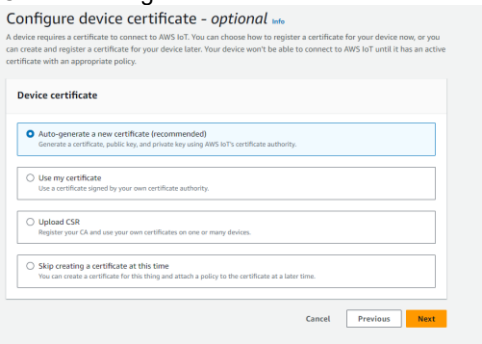
- Geef het thing een unieke naam



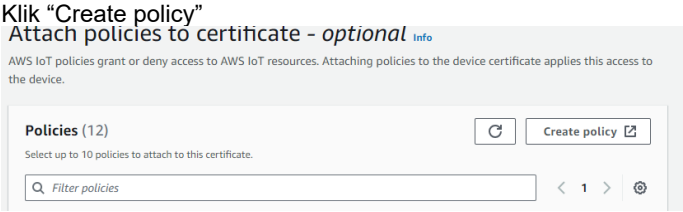
Thomas More
Campus De Nayer
Jan De Nayerlaan 5, 2860 Sint-Katelijne-Waver, België
+ 32 15 31 69 44 | info.denayer@thomasmore.be | www.thomasmore.be

LID VAN ASSOCIATIE KU LEUVEN

- Check “Auto-generate a new certificate”



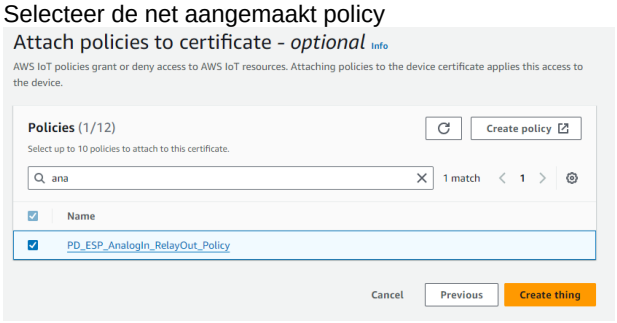
- Klik “Create policy”



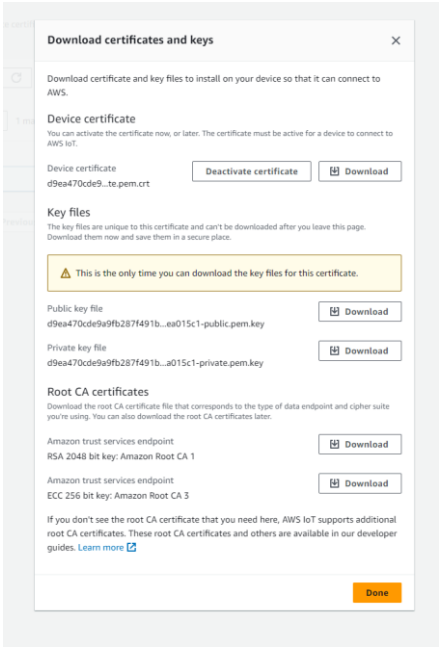
- Geef de policy rechten op de juiste iot actions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:eu-central-1:x:client/thing"
    },
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:eu-central-1:x:topicfilter/subscribe_topic"
    },
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "iot:Receive",
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:eu-central-1:x:topic/publish_topic
    }
  ]
}
```

- Selecteer de net aangemaakt policy



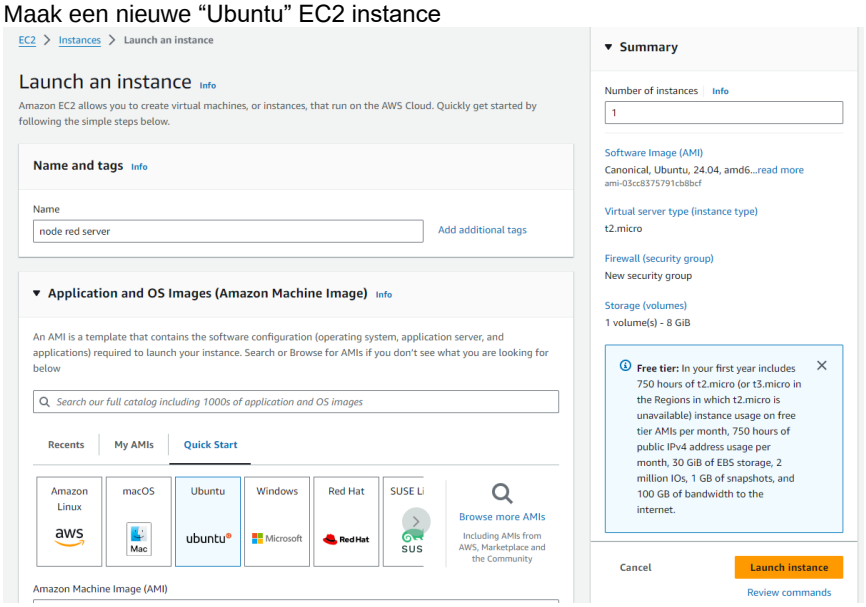
- Download alle certificaten



ECT instance

Node red zal draaien op een EC2 instance op AWS

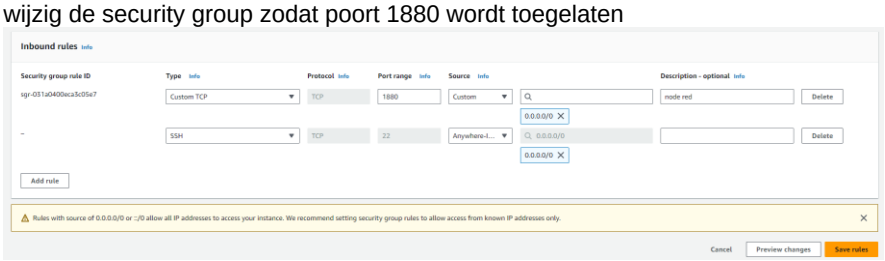
- Maak een nieuwe “Ubuntu” EC2 instance



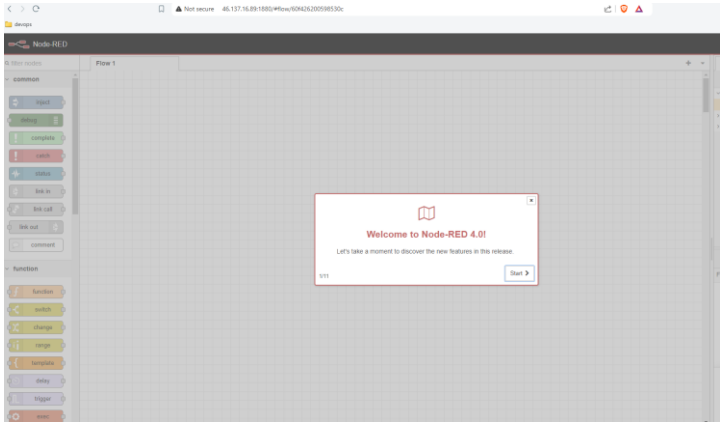
Thomas More
Campus De Nayer

Jan De Nayerlaan 5, 2860 Sint-Katelijne-Waver, België
+ 32 15 31 69 44 | info.denayer@thomasmore.be | www.thomasmore.be

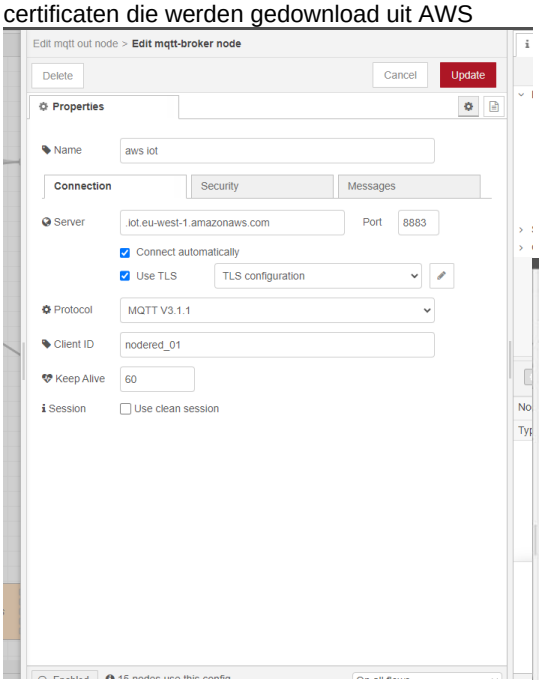
- Login op de EC2 instance en installeer node-red
    - `sudo npm install -g --unsafe-perm node-red`
- wijzig de security group zodat poort 1880 wordt toegelaten

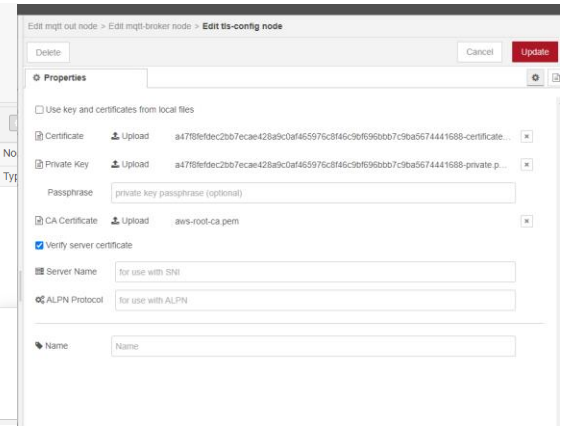


- start node red en open deze in een browser
    - sudo node-red



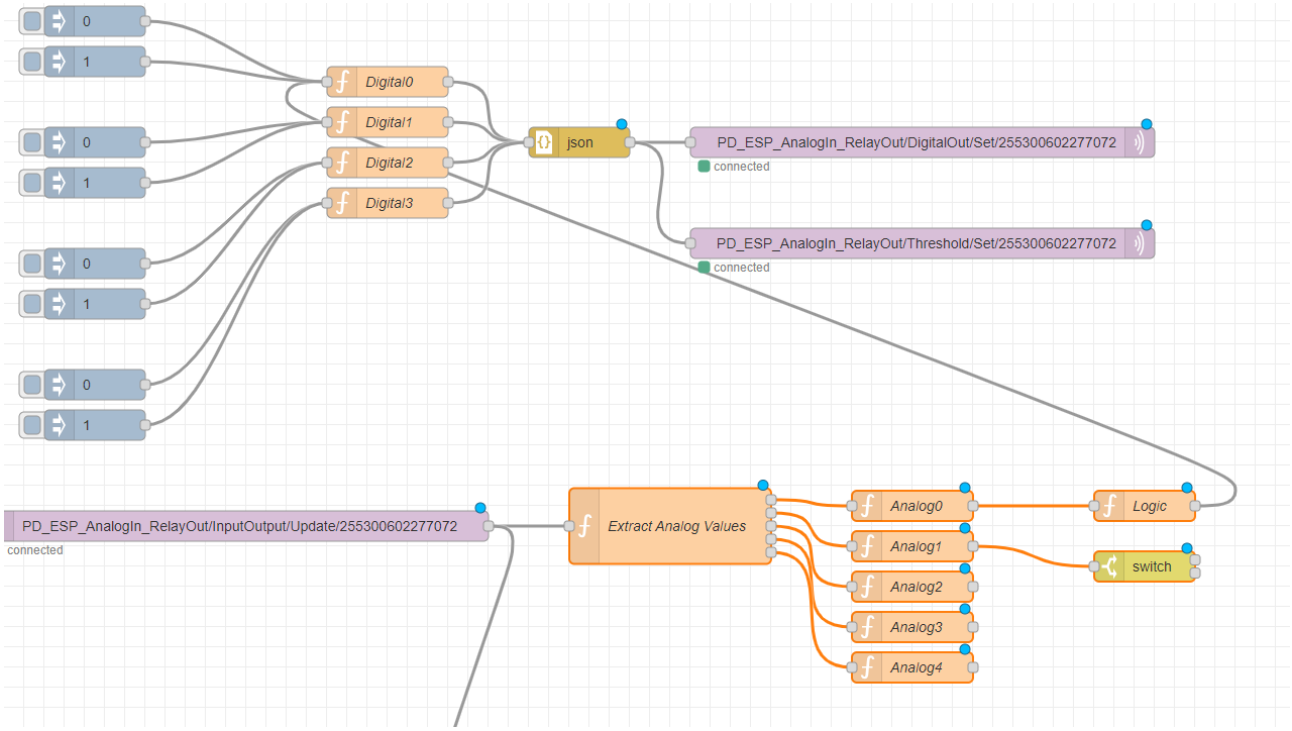
- importeer de node-red frows
- De connectie met AWS dient nog geconfigureerd te worden in node red. Wijzig de server naam en importeer de certificaten die werden gedownload uit AWS





Thomas More
Campus De Nayer

Jan De Nayerlaan 5, 2860 Sint-Katelijne-Waver, België
+ 32 15 31 69 44 | info.denayer@thomasmore.be | www.thomasmore.be



ESP32

Gebruik esp-idf om de firmware te uploaden naar een esp32.

- https://idf.espressif.com/

Aanpassingen in de code

- Plaats de AWS certificaten en de “certs” folder
- In bestadn aws.c
- Wijzig eventueel de namen van de key en certificaat bestanden in lijnen 43-48
- Vul de link naar aws HOST in op lijn 32

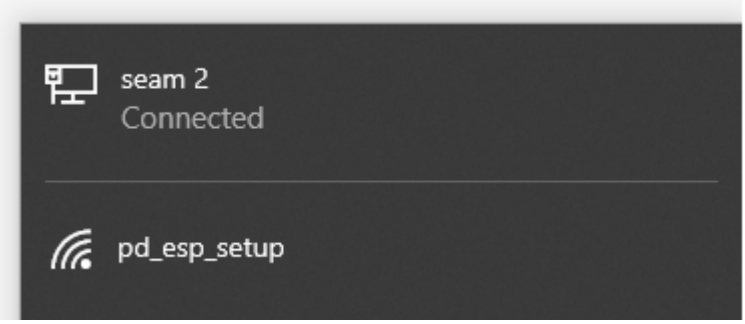
```
31
32  #define AWS_IOT_HOST ""
33  #define AWS_IOT_PORT 8883
34
35  extern Device_settings_t* DeviceSettings;
36
37
38
39  extern QueueHandle_t DataOutQueue;
40
41  const TickType_t QueueReadBlockTime = pdMS_TO_TICKS( 100 );
42
43  extern const uint8_t aws_root_ca_pem_start[] asm("_binary_aws_root_ca_pem_start");
44  extern const uint8_t aws_root_ca_pem_end[] asm("_binary_aws_root_ca_pem_end");
45  extern const uint8_t certificate_pem_crt_start[] asm("_binary_certificate_pem_crt_start");
46  extern const uint8_t certificate_pem_crt_end[] asm("_binary_certificate_pem_crt_end");
47  extern const uint8_t private_pem_key_start[] asm("_binary_private_pem_key_start");
48  extern const uint8_t private_pem_key_end[] asm("_binary_private_pem_key_end");
```

Flash naar esp

Het flashen van de firmware kan met volgend commando in de esp-idf console “idf.py build flash monitor –p COM4”.

```
C:\Users\u0118137\Documents\esp\PD_ESP_AnalogIn_RelayOut>idf.py build flash monitor -p COM8
Executing action: all (aliases: build)
Running ninja in directory c:\users\u0118137\documents\esp\pd_esp_analogin_relayout\build
Executing "ninja all"...
[1/8] Performing build step for 'bootloader'
ninja: no work to do.
[6/6] Generating binary image from built executable
esptool.py v3.1-dev
Merged 1 ELF section
Generated C:/Users/u0118137/Documents/esp/PD_ESP_AnalogIn_RelayOut/build/SMM_IOT_AWS.bin
Executing action: flash
Running ninja in directory c:\users\u0118137\documents\esp\pd_esp_analogin_relayout\build
Executing "ninja flash"...
[1/4] Performing build step for 'bootloader'
ninja: no work to do.
[1/2] cmd.exe /C "cd /D C:\Users\u0118137\esp-idf\componen...18137/esp-idf/components/esptool_py/run_serial_tool.cm
esptool.py esp32 -p COM8 -b 460800 --before=default_reset --after=hard_reset write_flash --flash_mode dio --flash_f
0m --flash_size 4MB 0x8000 partition_table/partition-table.bin 0xd000 ota_data_initial.bin 0x1000 bootloader/bootlo
bin 0x10000 SMM_IOT_AWS.bin
esptool.py v3.1-dev
Serial port COM8
Connecting....
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
```

De eerste maal dat de ESP32 wordt opgestart gedraagd de esp32 zich als access point met SSID “pd_esp_setup”



Surf naar 192.168.4.1 om het ssid en paswoord van het wifi access point waarmee je wil verbinden in te vullen



Na het verzenden herstart de esp32 en verbindt deze met het geconfigureerde access point. Daarna verbindt deze met AWS en start deze met het verzenden van de data over MQTT.

MQTT topics:

Aanpassen digitale output: PD_ESP_AnalogIn_RelayOut/DigitalOut/Set/
Lezen analoge input: PD_ESP_AnalogIn_RelayOut/InputOutput/Update/

Hardware

Voor het testen van deze case werd gebruik gemaakt van een ESP32-WROOM-32D met volgende aansluitingen:
ANALOG0 ADC1_CHANNEL_0

ANALOG1 ADC1_CHANNEL_3
ANALOG2 ADC1_CHANNEL_6
ANALOG3 ADC1_CHANNEL_7
ANALOG4 ADC1_CHANNEL_4

DIGITALOUT0 GPIO_NUM_18
DIGITALOUT1 GPIO_NUM_5
DIGITALOUT2 GPIO_NUM_17
DIGITALOUT3 GPIO_NUM_16
DIGITALOUT4 GPIO_NUM_19

Deze de pinnen waarop de analoge of digitale IO wordt aangesloten kan worden gewijzigd in het bestand “inputoutput.h”

main > C inputoutput.h > ANALOG0

```c
#pragma once

#ifndef INPUTOUTPUT_H_INCLUDE
#define INPUTOUTPUT_H_INCLUDE

#define ANALOG0 ADC1_CHANNEL_0
#define ANALOG1 ADC1_CHANNEL_3
#define ANALOG2 ADC1_CHANNEL_6
#define ANALOG3 ADC1_CHANNEL_7
#define ANALOG4 ADC1_CHANNEL_4

#define DIGITALOUT0 GPIO_NUM_18
#define DIGITALOUT1 GPIO_NUM_5
#define DIGITALOUT2 GPIO_NUM_17
#define DIGITALOUT3 GPIO_NUM_16
#define DIGITALOUT4 GPIO_NUM_19

#define maxvoltage 3000
#define minvoltage 142

#define ANALOG_READ_INTERVAL 5000

void init_inputoutput();
void set_io_output(int output, int value);

#endif
```

Contact

Glenn Adriaensens - glenn.adriaensens@thomasmore.be
Joris Dieltiens - joris.dieltiens@thomasmore.be

Projectlink: https://thomasmore.be/nl/lowcode