Bronbeschrijving

NFC Module

https://wiki.dfrobot.com/

Deze bron heb ik gebruikt voor het configureren en lezen van de NFC-module. Eerst moeten we de juiste library downloaden. Vervolgens kunnen we aan de slag.

Het was voor mij de eerste keer dat ik met deze sensor gewerkt heb en heb daarom eerst de voorbeeldcode uitgevoerd. Hierna werd al meteen duidelijk hoe deze sensor en zijn library werkte en kon ik dus op een eenvoudige manier de voorbeeldcode aanpassen naar mijn benodigdheden.

De eerste blok gebruik ik om te bepalen welke blok van de NFC-tag we gaan lezen en de grootte van hoeveel we van deze blok willen lezen.

```
// NFC Config Variables
#define BLOCK_SIZE 16
#define PN532_IRQ 2
#define POLLING 0

// The block to be read
#define READ_BLOCK_NO 2
```

Vervolgens gebruiken we de volgende code voor het lezen van de NFC-module en door te sturen naar de MQTT-broker.

```
if (nfc.scan())
{
  if (nfc.readData(dataRead, READ_BLOCK_NO) != 1)
  {
    Serial.println(" read failure!");
  }
  else
  {
    Serial.print("NFC string: ");
    Serial.println((char *)dataRead);
    // Send data to MQTT Broker
    mqttClient.publish(MQTT_NFC, (char *)dataRead);
  }
  nfcDelay = millis();
}
```

SCD4X

https://learn.adafruit.com/adafruit-scd-40-and-scd-41/arduino

Hier eigenlijk hetzelfde als bij de NFC-module. Het enige verschil bij de SCD4X sensor was dat ik een beetje harder moest zoeken naar juiste documentatie, aangezien er meerdere versies van deze sensor bestaan. Nadat ik de juiste documentatie had gevonden, was het weer zeer simpel om deze om te vormen naar wat ik nodig had.

We moeten voordat we iets doen opnieuw de juiste library downloaden. Hierna kunnen we beginnen doorgebruik te maken van onderstaande code voor het aanmaken van een SCD4X sensor object.

```
// SCD4X Sensor object and variables
SensirionI2CScd4x scd4x;
```

Hierna kunnen we volgende code gebruiken om het object klaar te maken om metingen te maken.

```
Serial.print("Inittializing SCD4X\n");
scd4x.begin(Wire);

// stop potentially previously started measurement
error = scd4x.stopPeriodicMeasurement();
if (error)
{
  Serial.print("Error trying to execute stopPeriodicMeasurement(): ");
  errorToString(error, errorMessage, 256);
  Serial.println(errorMessage);
}

uint16_t serial0;
uint16_t serial1;
uint16_t serial2;
error = scd4x.getSerialNumber(serial0, serial1, serial2);
if (error)
{
  Serial.print("Error trying to execute getSerialNumber(): ");
  errorToString(error, errorMessage, 256);
  Serial.println(errorMessage);
  exit(1);
}
else
{
  printSerialNumber(serial0, serial1, serial2);
  Serial.println("SCD4X Sensor Ready");
}

error = scd4x.startPeriodicMeasurement();
if (error)
{
  Serial.print("Error trying to execute startPeriodicMeasurement(): ");
  errorToString(error, errorMessage, 256);
  Serial.println(errorMessage);
}
```

Om dan ten slotte een functie te gebruiken om deze metingen op te vragen

```
uint16_t co2;
float temperature;
float humidity;

error = scd4x.readMeasurement(co2, temperature, humidity);
if (error)
{
  Serial.print("Error trying to execute readMeasurement(): ");
  errorToString(error, errorMessage, 256);
  Serial.println(errorMessage);
}
else if (co2 == 0)
{
  Serial.println("Invalid sample detected, skipping.");
}
else
{
  Serial.print("Co2:");
  Serial.print(co2);
  Serial.print(" ppm");
  Serial.print("\t");
  Serial.print("Temperature:");
  Serial.print(temperature);
  Serial.print(" °C");
  Serial.print("\t");
  Serial.print("Humidity:");
  Serial.print(humidity);
  Serial.println(" %");
```

Servo

https://github.com/madhephaestus/ESP32Servo

Het gebruiken van een servo op het esp32 platform is een beetje anders dan op het arduino platform. Hierbij gebruiken we namelijk een andere library. Vervolgens blijft de rest wel gewoon hetzelfde.

We maken een Servo object aan, vervolgens ‘attachen’ we deze op de juiste pin op de mcu om dan ten slotte met de write funtie de servo te laten bewegen.

```
// Servo setup
Servo door;
door.attach(D10);
door.write(180);
```

PubSubClient (MQTT)

https://github.com/knolleary/pubsubclient

Voor het MQTT-gedeelte wou ik eerst de library gebruiken waarmee we in de les al kennis hebben gemaakt (async-mqtt-client). Het probleem hiermee was dat deze niet goed samenwerkte met de NFC library. Daarom heb ik gekozen voor deze library. De documentatie en de voorbeelden maakte het makkelijk om te leren hoe ik deze library moest implementeren.

Om te beginnen moet je altijd een PubSubClient object aanmaken en het WiFiClient object meegeven.

```
WiFiClient espClient;
PubSubClient mqttClient(espClient);
```

Vervolgens in de setup functie zetten we onze broker en port instellingen juist en stellen we in welke functie moet worden opgeroepen als callback. Deze callback functie zal worden opgeroepen en uitgevoerd, elke keer als er een update wordt gestuurd naar een van de gesubscribeerde topics.

```
mqttClient.setServer(RPI_ADDRESS, 1883);
mqttClient.setCallback(callback);
```

Nadien gebruik ik volgende functie om te connecteren op eerder geconfigureerde MQTT-broker, ook subscriben we hier op de juiste topics

```
void reconnect()
{
  // Loop until we're reconnected
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (mqttClient.connect("ESP-IntelliHome"))
    {
      Serial.println("connected");
      //mqttClient.subscribe("test/led");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

Ten slotte gebruik ik de eerder gekozen callback functie om te bepalen wat er moet gebeuren als er een update naar een eerder gesubscribeerde topic gestuurd wordt.