Introductie

Voor dit onderdeel is heb ik een test opstelling gemaakt om een beter begrip te krijgen over hoe hardware in elkaar zit. Wat komt er allemaal bij kijken en hoe haal en schrijf je data naar hardware componenten.

Een goed beeld te krijgen wat er allemaal bij komt kijken heb ik ervoor gekozen om naast een temperatuursensor een oled scherm aan te sluiten.
Om ook inzicht te kijgen in hoe je deze data(temperatuur) nu kan versturen naar bijvoorbeeld een MQTT broker of dat je deze beschikbaar stelt via een API, heb ik besloten om dit ook beschikbaar te maken in de firmware.

Inhoudsopgave

Wat, waarom, hoe?

Om goed weer te kunnen geven wat ik allemaal gedaan heb, welke stappen ik heb doorlopen en welke keuzes ik gemaakt heb. Zal ik antwoord geven op de vragen wat, waarom en hoe.

Wat

Het doel is om aan te tonen hoe je een sensor uit kan lezen en deze data, beschikbaar kan stellen. Om mijzelf uit te dagen heb ik ervoor gekozen om nog een OLED scherm toe te voegen en de data beschikbaar te stellen via een API en te versturen naar een MQTT broker. Deze technieken kunnen van pas komen voor het groepsproject.

Zowel het OLEd scherm als de sensor communiceren via het I2C protocol. I²C - Wikipedia

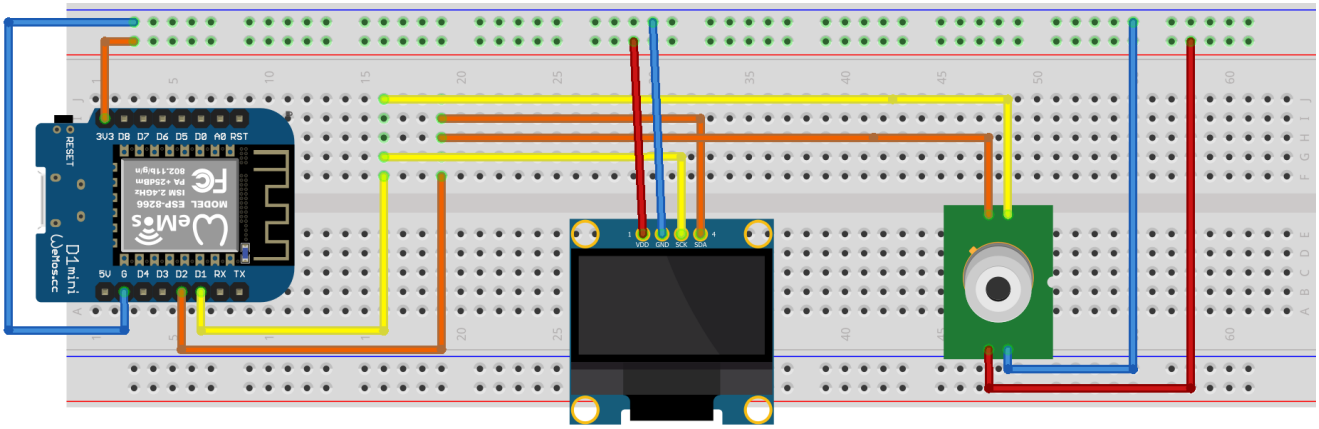
Kort samengevat, het programmeren van een microcontroller, zodat deze een sensor uitleest en de data als een object aanbiedt. Dit gebeurt via een API en naar een MQTT broker.

Waarom

Om een goed gevoel en een beter inzicht te krijgen wat er nu voor nodig is om een sensor uit te lezen en deze data beschikbaar te stellen in een lees- en handelbaar formaat. En deze data via bepaalde protocollen beschikbaar te stellen zodat deze als input gebruikt kan worden door andere processen/services.

Hoe

Voor mijn proof of concept heb ik de onderstaande situatie nagebouwd.



<<<<<<< HEAD

De eerste uitdaging was de adressen te achterhalen van het OLED display en de temperatuur module. Elk I²C module heeft een eigen adres om aan te spreken.
Voordeel van I²C modules is dat het mogelijk is om 255 modules op 1 draad aan te sluiten. Je kan het vergelijken met een aantal huizen in een straat.

Om het adres te achterhalen heb ik gebruik gemaakt van de volgende code:

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(115200);
  Serial.println("\nI2C Scanner");
}

void loop() {
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ ) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address<16) {
        Serial.print("0");
      }
      Serial.println(address,HEX);
      nDevices++;
    }
    else if (error==4) {
      Serial.print("Unknow error at address 0x");
      if (address<16) {
        Serial.print("0");
      }
      Serial.println(address,HEX);
    }
  }
```

```
  if (nDevices == 0) {
    Serial.println("No I2C devices found\n");
  }
  else {
    Serial.println("done\n");
  }
  delay(5000);
}
```

Deze code scant de I²C bus of er modules aanwezig zijn. Als deze correct aangesloten zijn dan ze hebben spanning dan zal de code het adres geprint worden.

De volgende adressen zijn naar voren gekomen:
MLX90614: 0×5A
OLED: 0×3D

Er is iemand al zo aardig geweest om voor de temperatuursensor een library hiervoor te maken. Om de waarden van de sensor uit te kunnen lezen kan men de al gemaakte methodes in `<Adafruit_MLX90614.h>` gebruiken.

Voor het uitlezen van de omgevings- en object temperatuur kunnen de volgende functies aangeroepen worden. De methode `.readAmbientTempC()` geeft een double terug met de waarde van de omgevingstemperatuur in graden Celcius.
De methode `readObjectTempC()` geeft een double terug met de waarde van het object voor de sensor in graden Celcius.

Voor het OLED scherm kan er gebruik gemaakt worden van de `<Adafruit_SSD1306.h>` library.

=======

"./img/07_hardware/Beer.cool.board_bb.png" is not created yet. Click to create.

De code is vervolgens uitgebreid dat deze netjes de data naar een MQTT broker zendt en dat deze een API beschikbaar maakt. /getTemperatures. Deze geeft vervolgens de waarde van de sensor terug.

```
/\*\*\*\*\*\*\*\*\*\*

 Present pictures on oled during boot and show ambient and object temperature

 API json formatting, hand made json

\*\*\*\*\*\*\*\*\*\*/

// Importing libraries

#include <Arduino.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>
```

```
#include <Adafruit\_MLX90614.h>

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>

#include <Adafruit\_I2CDevice.h>



// Declaring variables

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels



// Declaration for an OLED SSD1306 display connected to I2C (SDA, SCL pins)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);



// Declaration for an infrared temperature sensor MLX90614

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614(0x5A);



// MQTT server

const char\* mqtt\_server = "192.168.2.10";

const char\* topic = "infrared-scanner"; // this is the \[root topic\]



// WiFi settings

const char\* ssid = "IOT-Devices";

const char\* pswd = "8m4^ARuCVX5f!1fsH&vp7zT!";



// Variable used for timing of messages, 10.000 = 10 seconds
```

```
long timeBetweenMessages = 10000;



// Create objects

WiFiClient espClient;

ESP8266WebServer server(80);

PubSubClient client(espClient);



// Additional variables

long lastMsg = 0;

int value = 0;

int status = WL\_IDLE\_STATUS; // the starting Wifi radio's status



// Method for setting up Wi-Fi

void setup\_wifi() {

 delay(10);



 // We start by connecting to a WiFi network

 Serial.println();

 Serial.print("Connecting to ");

 Serial.println(ssid);

 WiFi.begin(ssid, pswd);

 while (WiFi.status() != WL\_CONNECTED) {

 delay(500);

 Serial.print(".");

 }

 Serial.println("");

 Serial.println("WiFi connected");
```

```
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}

//
void callback(char\* topic, byte\* payload, unsigned int length) {
 Serial.print("Message arrived \[");
 Serial.print(topic);
 Serial.print("\] ");
 for (int i = 0; i < length; i++) {
 Serial.print((char)payload\[i\]);
 }
 Serial.println();

 // Switch on the LED if an 1 was received as first character
 if ((char)payload\[0\] == '1') {
 digitalWrite(BUILTIN\_LED, LOW); // Turn the LED on (Note that LOW is the voltage level
 // but actually the LED is on; this is because
 // it is acive low on the ESP-01)
 } else {
 digitalWrite(BUILTIN\_LED, HIGH); // Turn the LED off by making the voltage HIGH
 }
}

// Method for getting MAC address and convert to string
String macToStr(const uint8\_t\* mac)
{
 String result;
```

```
 for (int i = 0; i < 6; ++i) {
 result += String(mac\[i\], 16);
 if (i < 5)
 result += ':';
 }
 return result;
}


// Method for composing hostname
String composeClientID() {
 uint8\_t mac\[6\];
 WiFi.macAddress(mac);
 String clientId;
 clientId += "esp-";
 clientId += macToStr(mac);
 return clientId;
}


// Method for reconnecting to MQTT
void reconnect() {
 // Loop until we're reconnected
 while (!client.connected()) {
 Serial.print("Attempting MQTT connection...");


 String clientId = composeClientID() ;
 clientId += "-";
 clientId += String(micros() & 0xff, 16); // to randomise. sort of
```

```
  // Attempt to connect

  if (client.connect(clientId.c\_str())) {

  Serial.println("connected");

  // Once connected, publish an announcement...

  client.publish(topic, ("connected " + composeClientID()).c\_str() , true );

  // ... and resubscribe

  // topic + clientID + in

  String subscription;

  subscription += topic;

  subscription += "/";

  subscription += composeClientID() ;

  subscription += "/in";

  client.subscribe(subscription.c\_str() );

  Serial.print("subscribed to : ");

  Serial.println(subscription);

  } else {

  Serial.print("failed, rc=");

  Serial.print(client.state());

  Serial.print(" wifi=");

  Serial.print(WiFi.status());

  Serial.println(" try again in 5 seconds");

  // Wait 5 seconds before retrying

  delay(5000);

  }

  }

}
```

```
// Methods for API calls

// Method for /helloWorld GET

void getHelloWord() {

 DynamicJsonDocument doc(512);

 doc\["name"\] = "Hello world";



 Serial.print(F("Stream..."));

 String buf;

 serializeJson(doc, buf);

 server.send(200, "application/json", buf);

 Serial.println(F("done."));

}



// Method for /getTemperatures GET

void getTemperatures() {

 DynamicJsonDocument doc(512);

 String ambientTemp;

 String objectTemp;

 ambientTemp.concat(mlx.readAmbientTempC());

 objectTemp.concat(mlx.readObjectTempC());

 doc\["ambientTemperature"\] = ambientTemp;

 doc\["objectTemperature"\] = objectTemp;

 Serial.print(F("Stream..."));

 String buf;

 serializeJson(doc, buf);

 server.send(200, "application/json", buf);

 Serial.println(F("done."));
```

```
}



// Method for serving settings

// Call url: http://192.168.2.79/settings?signalStrength=true&chipInfo=true&freeHeap=true

void getSettings() {

 // Allocate a temporary JsonDocument

 // StaticJsonDocument<512> doc;

 DynamicJsonDocument doc(512);

 doc\["ip"\] = WiFi.localIP().toString();

 doc\["gw"\] = WiFi.gatewayIP().toString();

 doc\["nm"\] = WiFi.subnetMask().toString();

 if (server.arg("signalStrength")== "true"){

 doc\["signalStrengh"\] = WiFi.RSSI();

 }

 if (server.arg("chipInfo")== "true"){

 doc\["chipId"\] = ESP.getChipId();

 doc\["flashChipId"\] = ESP.getFlashChipId();

 doc\["flashChipSize"\] = ESP.getFlashChipSize();

 doc\["flashChipRealSize"\] = ESP.getFlashChipRealSize();

 }

 if (server.arg("freeHeap")== "true"){

 doc\["freeHeap"\] = ESP.getFreeHeap();

 }

 Serial.print(F("Stream..."));

 String buf;

 serializeJson(doc, buf);

 server.send(200, F("application/json"), buf);
```

```
 Serial.println(F("done."));

}



void setRoom() {

 String postBody = server.arg("plain");

 Serial.println(postBody);

 DynamicJsonDocument doc(512);

 DeserializationError error = deserializeJson(doc, postBody);

 if (error) {

 // if the file didn't open, print an error:

 Serial.print(F("Error parsing JSON "));

 Serial.println(error.c\_str());

 String msg = error.c\_str();

 server.send(400, F("text/html"),

 "Error in parsin json body! <br>" + msg);

 } else {

 JsonObject postObj = doc.as<JsonObject>();

 Serial.print(F("HTTP Method: "));

 Serial.println(server.method());

 if (server.method() == HTTP\_POST) {

 if (postObj.containsKey("name") && postObj.containsKey("type")) {

 Serial.println(F("done."));

 // Here store data or doing operation



 // Create the response

 DynamicJsonDocument doc(512);

 doc\["status"\] = "OK";
```

```
  Serial.print(F("Stream..."));

  String buf;

  serializeJson(doc, buf);

  server.send(201, F("application/json"), buf);

  Serial.println(F("done."));

  }else {

  DynamicJsonDocument doc(512);

  doc\["status"\] = "KO";

  doc\["message"\] = F("No data found, or incorrect!");

  Serial.print(F("Stream..."));

  String buf;

  serializeJson(doc, buf);

  server.send(400, F("application/json"), buf);

  Serial.println(F("done."));

  }

  }

  }

}

// Set routing

void restServerRouting() {

  server.on("/", HTTP\_GET, \[\]() {

  server.send(200, F("text/html"),

  F("Welcome to the REST Web Server"));

  });

  // Handle get requests

  server.on(F("/helloWorld"), HTTP\_GET, getHelloWord);

  server.on(F("/settings"), HTTP\_GET, getSettings);

  server.on(F("/getTemperatures"), HTTP\_GET, getTemperatures);
```

```
 // Handle post requests
 server.on(F("/setRoom"), HTTP\_POST, setRoom);
}

// Manage not found URL
void handleNotFound() {
 String message = "File Not Found\\n\\n";
 message += "URI: ";
 message += server.uri();
 message += "\\nMethod: ";
 message += (server.method() == HTTP\_GET) ? "GET" : "POST";
 message += "\\nArguments: ";
 message += server.args();
 message += "\\n";
 for (uint8\_t i = 0; i < server.args(); i++) {
 message += " " + server.argName(i) + ": " + server.arg(i) + "\\n";
 }
 server.send(404, "text/plain", message);
}

void printStringOled(bool clearDisplay,int textSize, String color,int x, int y, bool
println,String stringToDisplay){
//  display.clearDisplay()        - all pixels are off
//  display.drawPixel(x,y, color) - plot a pixel in the x,y coordinates
//  display.setTextSize(n)        - set the font size, supports sizes from 1 to 8
//  display.setCursor(x,y)        - set the coordinates to start writing text
//  display.print(“message”)      - print the characters at location x,y
```

```
//  display.display()             - call this method for the changes to make effect

 if(clearDisplay){

 display.clearDisplay();

 }

 display.setTextSize(textSize);

 display.setTextColor(WHITE);

 display.setCursor(x, y);

 display.println(stringToDisplay);

 display.display();

}



String IpAddress2String(const IPAddress& ipAddress)

{

 return String(ipAddress\[0\]) + String(".") +

 String(ipAddress\[1\]) + String(".") +

 String(ipAddress\[2\]) + String(".") +

 String(ipAddress\[3\]);

}

// 'Beer.Cool first image', 128x64px

const unsigned char BeerCool2 \[\] PROGMEM = {

 0xff, 0xff, 0xf3, 0x8f, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xff, 0xdf, 0xf3, 0xff, 0xce, 0xec, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xff, 0xd9, 0xe9, 0xff, 0xbf, 0xff, 0xbf, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xff, 0xdf, 0x1b, 0xff, 0x7f, 0xff, 0xdf, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xff, 0xdf, 0xdb, 0xfe, 0xff, 0xff, 0xc8, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
 0xff, 0xff, 0xdf, 0xdb, 0xfe, 0xff, 0xff, 0xe0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xff, 0xdf, 0xd8, 0x00, 0x3f, 0xff, 0xe0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xf0, 0x00, 0x1f, 0xb0, 0x00, 0xff, 0xfe, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0x80, 0x00, 0x1f, 0xb0, 0x01, 0xff, 0xff, 0xe0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0x80, 0x00, 0x01, 0xc0, 0x01, 0xff, 0xff, 0xf8, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0x80, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xe8, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0x80, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xc8, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0x80, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xd0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xd0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x01, 0xd0, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x06, 0x64, 0xcc, 0x00, 0x78, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc6, 0x66, 0x60, 0xcc, 0x00, 0x78, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc7, 0x66, 0x60, 0xcc, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc7, 0x66, 0x60, 0xe8, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc7, 0x46, 0x60, 0xcc, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
 0xc7, 0x76, 0x60, 0xcc, 0x40, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc3, 0x76, 0x62, 0xee, 0xc0, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc3, 0x66, 0x66, 0x40, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc2, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xc0, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x01, 0x80, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x91, 0x21, 0x11, 0x80, 0x38, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe1, 0x12, 0x33, 0x19, 0x80, 0x38, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe3, 0x02, 0x3b, 0x19, 0x80, 0x38, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe3, 0x07, 0x1b, 0x1d, 0x80, 0x38, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe3, 0x07, 0x1b, 0x19, 0x80, 0x30, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe3, 0x07, 0x1b, 0x19, 0x88, 0x30, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe3, 0x03, 0x11, 0x99, 0x90, 0x30, 0x00, 0xf0, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe1, 0x91, 0x20, 0x80, 0x00, 0x30, 0x00, 0xf8, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0xfc, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf6, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf6, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf6, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf6, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf6, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0xf7, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0xe1, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x1e, 0x00, 0x00, 0xe0, 0x00, 0x1c, 0x4c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xe0, 0x00, 0x7b, 0x00, 0x00, 0xe0, 0x00, 0xff, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xf0, 0x03, 0xf3, 0x00, 0x00, 0xe0, 0x00, 0xfe, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xf0, 0x03, 0xf7, 0xbf, 0xf8, 0xe0, 0x00, 0xf0, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xf7, 0xcb, 0x9f, 0xf9, 0xe0, 0x00, 0x83, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xf6, 0x7b, 0xdf, 0xf9, 0xe0, 0x00, 0x1f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xed, 0xfd, 0xcf, 0xf1, 0xe0, 0x00, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xef, 0xfd, 0xcf, 0xf1, 0xe0, 0x00, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xef, 0xfe, 0xdf, 0xf9, 0xc0, 0x00, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xf7, 0xff, 0x9f, 0xfc, 0x00, 0x00, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xf3, 0xff, 0xbf, 0xff, 0x00, 0x01, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xfb, 0xff, 0x3f, 0xff, 0xfc, 0x0f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
```

```
 0xff, 0xf9, 0xfe, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xfd, 0xf1, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xfe, 0xc7, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,

 0xff, 0xfe, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};




void setup() {

 pinMode(BUILTIN\_LED, OUTPUT); // Initialize the BUILTIN\_LED pin as an output

 // Turn of blue led

 digitalWrite(BUILTIN\_LED, HIGH);



 Serial.begin(115200);



 // Start looking for the oled display

 if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64

 Serial.println(F("SSD1306 allocation failed"));

 for(;;);

 }

 delay(2000);

 // Initialize infrared temp sensor

 mlx.begin();



 // Print to display next step, connecting to Wi-Fi

 printStringOled(true,1,"WHITE",0,20,true,"WiFi connecting....");

 delay(1000);
```

```
WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, pswd);

Serial.println("");

// Wait for connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

printStringOled(false,1,"WHITE",0,20,false,".");

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

printStringOled(true,1,"WHITE",0,20,true,"Connected to: ");

printStringOled(false,1,"WHITE",0,30,true,ssid);

printStringOled(false,1,"WHITE",0,40,false,"Ip address: ");

printStringOled(false,1,"WHITE",0,40,true,"WiFi connecting....");

printStringOled(true,1,"WHITE",0,20,true,IpAddress2String(WiFi.localIP()));

delay(3000);

// Activate mDNS this is used to be able to connect to the server

// with local DNS hostmane esp8266.local

if (MDNS.begin("esp8266")) {

display.clearDisplay();

display.setCursor(0, 20);

display.println("MDNS responder started");

display.display();
```

```
  Serial.println("MDNS responder started");

  delay(1000);

  }

  // Set server routing

  restServerRouting();

  // Set not found response

  server.onNotFound(handleNotFound);



  // Start server

  server.begin();

  Serial.println("HTTP server started");

  display.clearDisplay();

  display.println("HTTP server started");

  display.display();

  delay(1000);

  // Connecting to MQTT server

  client.setServer(mqtt\_server, 1883);

  client.setCallback(callback);

  display.setTextSize(1);

  display.setTextColor(WHITE);

  display.setCursor(0, 20);

  // Display static text

  display.println("Showing some images");

  display.display();

  delay(1000);

  display.clearDisplay();

  display.drawBitmap(0,0,BeerCool2,128,64,WHITE);

  display.display();
```

```
 delay(3000);
}

double readAmbientTemperatureC(){
 return mlx.readAmbientTempC();
}

double readObjectTemperatureC(){
 return mlx.readObjectTempC();
}

void displaytext(String text, int textSize, int times, int x, int y, bool cp437, int symbol){
// Changing Font Size
//  display.clearDisplay();
 display.setTextColor(WHITE);
 display.setCursor(x,y);
 display.setTextSize(textSize);
 if(cp437){
 display.cp437(true);
 } else{
 display.cp437(false);
 }
 if(symbol > 0){
 display.write(167);
 }
 display.print(text);
}
```

```
// Counter to track loops
int i = 0;
void loop() {
 // Server ready
 server.handleClient();


 // confirm still connected to mqtt server
 if (!client.connected()) {
 reconnect();
 }
 client.loop();
 // Display ambient and object temperture on oled display
 String someText;
 display.clearDisplay();
 display.setTextSize(1);
 display.setCursor(0,0);
 display.print("Ambient temperature: ");
 display.setTextSize(2);
 display.setCursor(0,10);
 someText.concat(readAmbientTemperatureC());
 display.print(someText);
 display.print(" ");
 display.setTextSize(1);
 display.cp437(true);
 display.write(167);
 display.setTextSize(2);
 display.print("C");
```

```
display.setTextSize(1);

display.setCursor(0,35);

display.print("Object temperature: ");

display.setTextSize(2);

display.setCursor(0,45);

display.print(mlx.readObjectTempC());

display.print(" ");

display.setTextSize(1);

display.cp437(true);

display.write(167);

display.setTextSize(2);

display.print("C");

display.display();

delay(1000);

i++;

Serial.print("Ambient = ");

Serial.print(mlx.readAmbientTempC());

Serial.print("\*C\\tObject = ");

Serial.print(mlx.readObjectTempC());

Serial.println("\*C");



long now = millis();

if (now - lastMsg > timeBetweenMessages ) {

lastMsg = now;

++value;
```

```
// Composing MQTT message

String payload = "{\\"micros\\":";

payload += micros();

payload += ",\\"counter\\":";

payload += value;

payload += ",\\"client\\":";

payload += composeClientID();

payload += ",\\"abientTemp\\":";

payload += mlx.readAmbientTempC();

payload += ",\\"objectTemp\\":";

payload += mlx.readObjectTempC();

payload += "}";



// Composing published topic

String pubTopic;

pubTopic += topic ;

pubTopic += "/";

pubTopic += composeClientID();

pubTopic += "/out";



Serial.print("Publish topic: ");

Serial.println(pubTopic);

Serial.print("Publish message: ");

Serial.println(payload);



// Publish data to MQTT server

client.publish( (char\*) pubTopic.c\_str() , (char\*) payload.c\_str(), true );

}
```

```
}
```