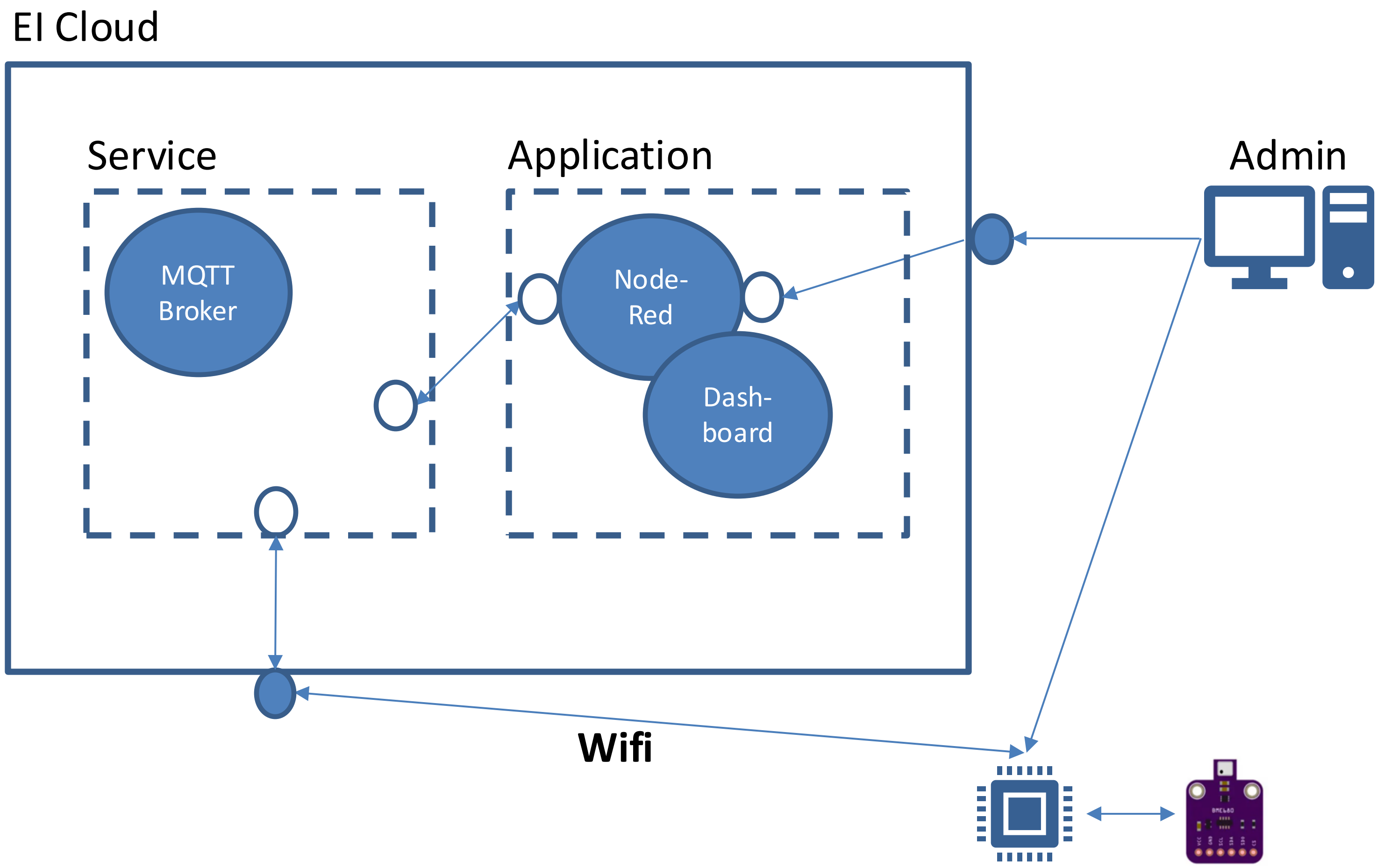


IoT Seminar Lab 3a:

Programming a µC

Connect to Cloud

Visualize sensor data

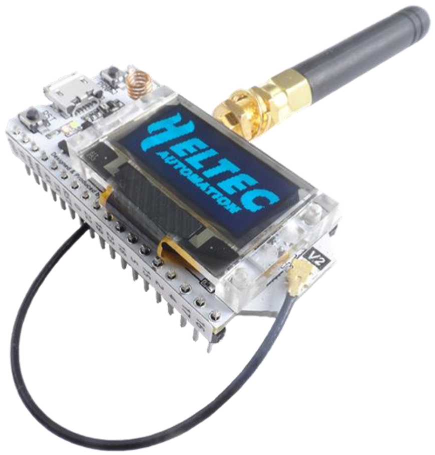
→ With WIFI

Hardware:
- µC: Heltec WiFi LoRa 32 (V2)
- Sensor: Bosch BME680

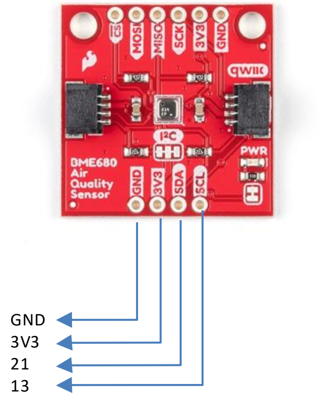
Preparations (µcontroller and sensorboard)

- ! Handle device carefully - Avoid electrostatic charge !
- Attach antenna – ! Do not operate controller without antenna !
- Connect Bosch BME680 sensor board to controller via I2C (PIN no):

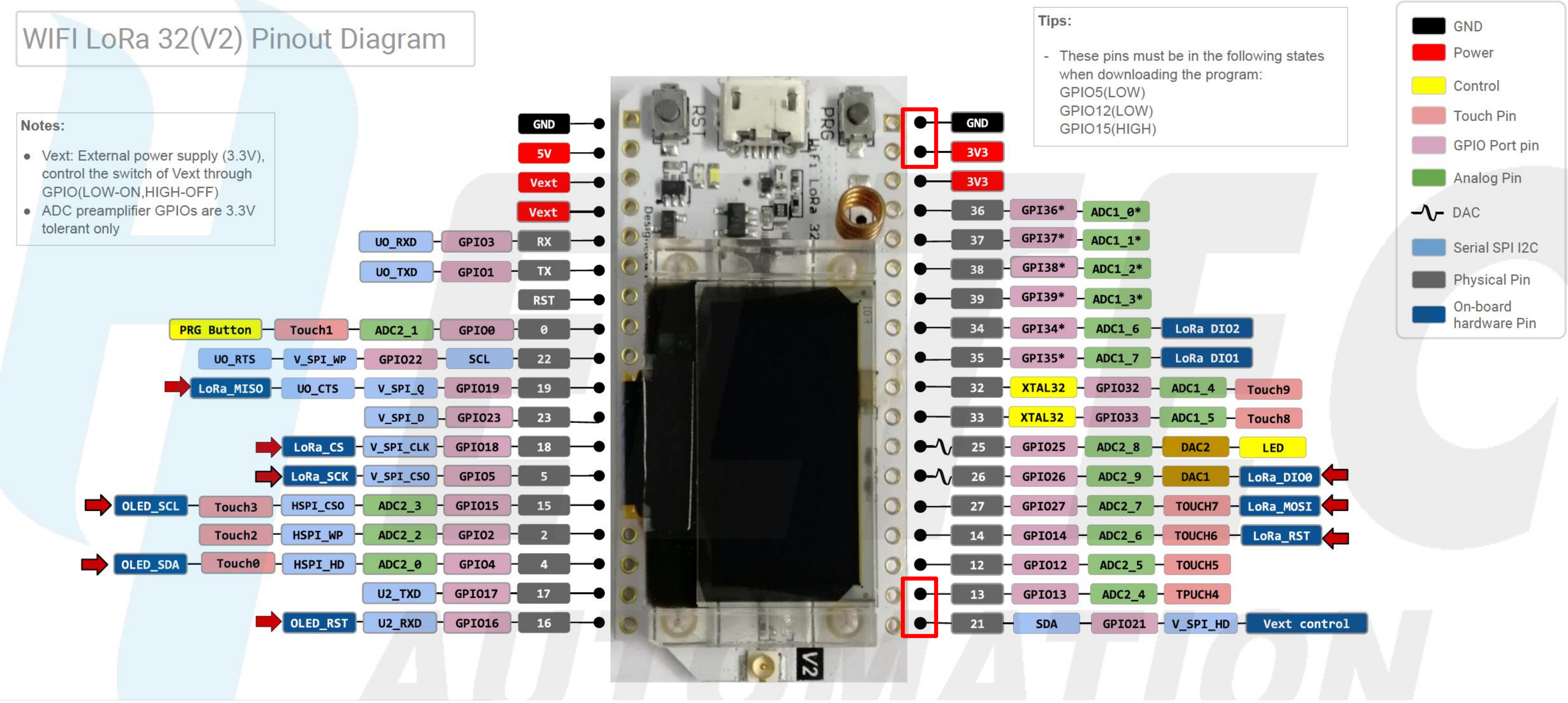
µc	BME680
GND	GND
3V3	3Vo
13 (Clock)	SCK / SCL
21 (Data)	SDI / SDA



- Connect now to PC with micro-USB
  - Driver installation typically not needed
- Find Datasheets in attached folder
  - Heltec: https://heltec.org/project/wifi-lora-32/



Heltec ESP32 WIFI/LoRa (V2) PIN-Layout



WIFI LoRa 32(V2) Pinout Diagram

Notes:

- Vext: External power supply (3.3V), control the switch of Vext through GPIO(LOW-ON,HIGH-OFF)
- ADC preamplifier GPIOs are 3.3V tolerant only

Tips:

- These pins must be in the following states when downloading the program:
  GPIO5(LOW)
  GPIO12(LOW)
  GPIO15(HIGH)

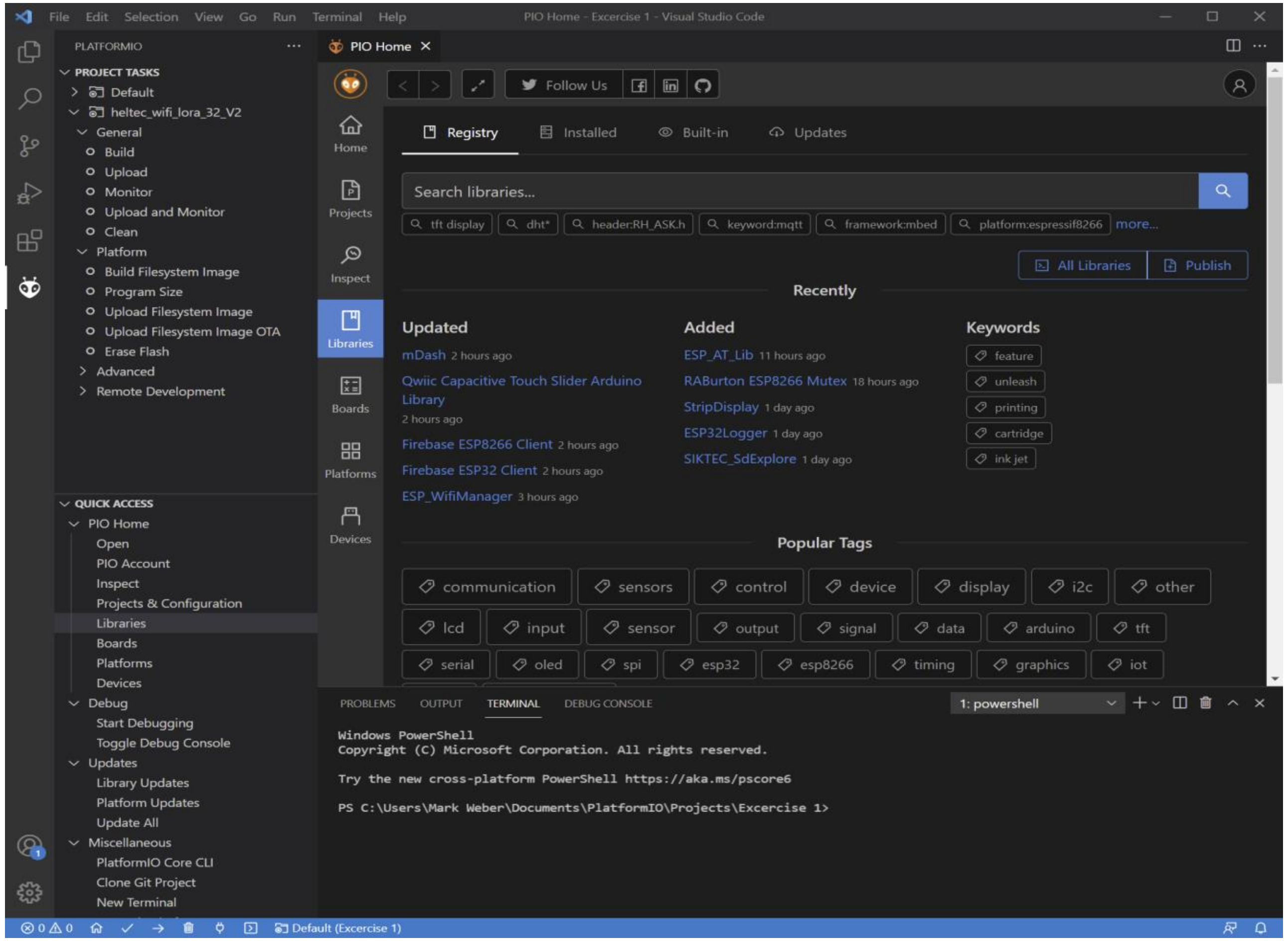
					GND
					5V
					Vext
					Vext
			UO_RXD	GPIO3	RX
			UO_TXD	GPIO1	TX
					RST
	PRG Button	Touch1	ADC2_1	GPIO0	0
	UO_RTS	V_SPI_WP	GPIO22	SCL	22
	LoRa_MISO	UO_CTS	V_SPI_Q	GPIO19	19
			V_SPI_D	GPIO23	23
		LoRa_CS	V_SPI_CLK	GPIO18	18
		LoRa_SCK	V_SPI_CSO	GPIO5	5
OLED_SCL	Touch3	HSPI_CSO	ADC2_3	GPIO15	15
	Touch2	HSPI_WP	ADC2_2	GPIO2	2
OLED_SDA	Touch0	HSPI_HD	ADC2_0	GPIO4	4
			U2_TXD	GPIO17	17
		OLED_RST	U2_RXD	GPIO16	16

GND					
3V3					
3V3					
36	GPI36*	ADC1_0*			
37	GPI37*	ADC1_1*			
38	GPI38*	ADC1_2*			
39	GPI39*	ADC1_3*			
34	GPI34*	ADC1_6	LoRa DIO2		
35	GPI35*	ADC1_7	LoRa DIO1		
32	XTAL32	GPIO32	ADC1_4	Touch9	
33	XTAL32	GPIO33	ADC1_5	Touch8	
25	GPIO25	ADC2_8	DAC2	LED	
26	GPIO26	ADC2_9	DAC1	LoRa_DIO0	
27	GPIO27	ADC2_7	TOUCH7	LoRa_MOSI	
14	GPIO14	ADC2_6	TOUCH6	LoRa_RST	
12	GPIO12	ADC2_5	TOUCH5		
13	GPIO13	ADC2_4	TPUCH4		
21	SDA	GPIO21	V_SPI_HD	Vext control	

■	GND
🟥	Power
🟨	Control
▨	Touch Pin
▨	GPIO Port pin
🟩	Analog Pin
∿	DAC
🟦	Serial SPI I2C
▨	Physical Pin
🟦	On-board hardware Pin

Preparations (IDE)

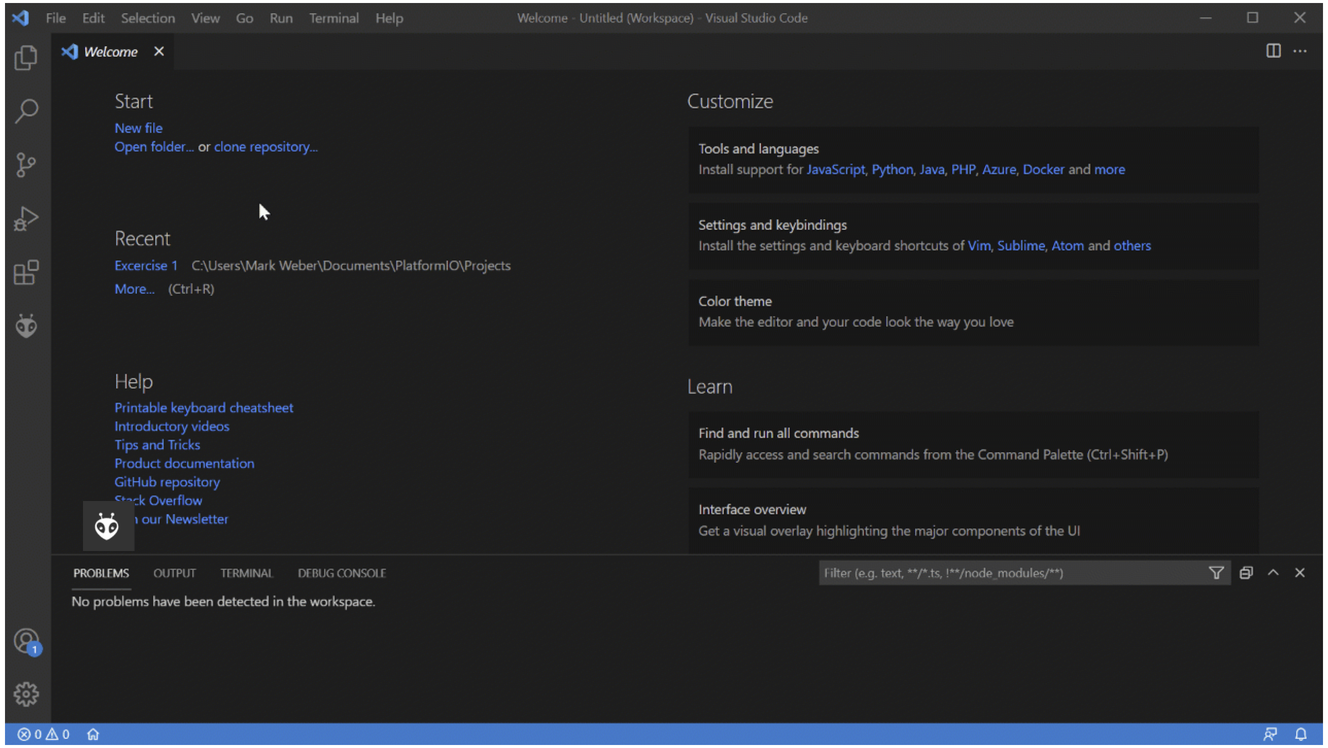
- typically pre-installed in lab, PlatformIO has to be installed in lab:
  - Install Visual Studio Code (https://code.visualstudio.com/)
  - Install PlatformIO (https://platformio.org/)



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Mark Weber\Documents\PlatformIO\Projects\Excercise 1>
```

Exercise 1 – “Hello World“

- Create a new Project (PIO Home → Open)
- Wait until Installer is finished!
- New Project:
  - IoTLab3a → select Heltec board→ Framework Arduino: **Heltec WiFi LoRa 32 (V2) (Heltech Automation)**
  - Add library to your project: → PIO Home → libraries → select Heltec ESP32 Dev-Boards **v1.1.2** → Add to your project → maybe restart DIE
  - Go to Exercise_1 → src → main.cpp
  - Add library in code too: #include <heltec.h>
  - Make sure to set baudrate to 115200 in platformio.ini: `monitor_speed=115200`
  - Build and Upload Code


File
Edit
Selection
View
Go
Run
Terminal
Help
Welcome - Untitled (Workspace) - Visual Studio Code
Welcome
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Filter (e.g. text, **/*.ts, !**/node_modules/**)
⊗ 0 ⚠ 0

Start
New file
Open folder... or clone repository...

Recent
Excercise 1 C:\Users\Mark Weber\Documents\PlatformIO\Projects
More... (Ctrl+R)

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
our Newsletter

Customize
Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love

Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl+Shift+P)
Interface overview
Get a visual overlay highlighting the major components of the UI

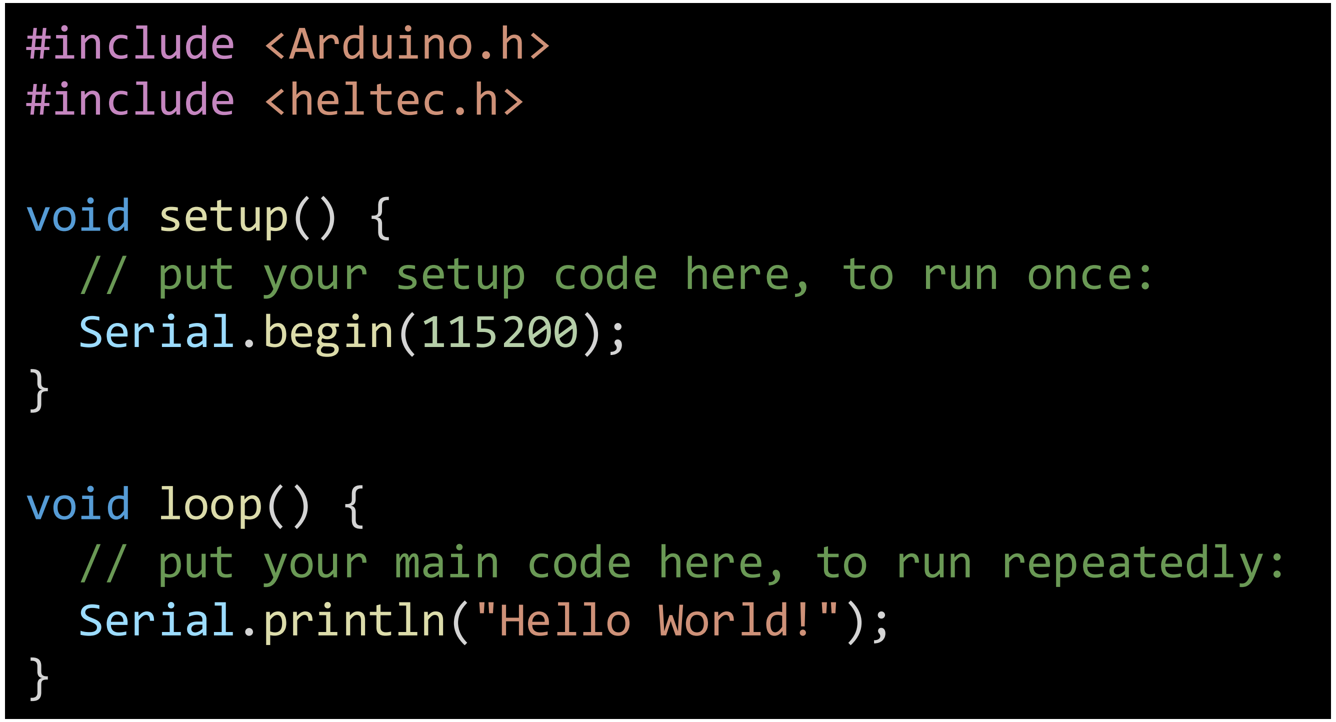
No problems have been detected in the workspace.

Exercise 1 – “Hello World“

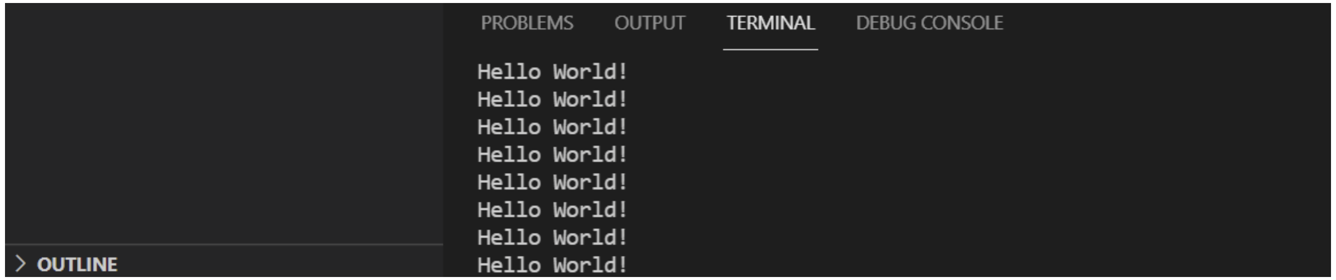
- Includes of **libraries**:
  ```
  #include <Arduino.h>
  #include <heltec.h>
  ```

- **setup()-function** for initial settings:
  ```
  Serial.begin(115200);
  ```

- **loop()-function** for main code which runs repeatedly:
  ```
  Serial.println("Hello World!");
  ```



```
#include <Arduino.h>
#include <heltec.h>

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World!");
}
```



```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

- Output in Serial Monitor (if BAUD-Rate was configured correctly in platformio.ini)
- HINT: code needs to be added in next steps

Exercise 2 – “Connect to local wifi“ (1/3)

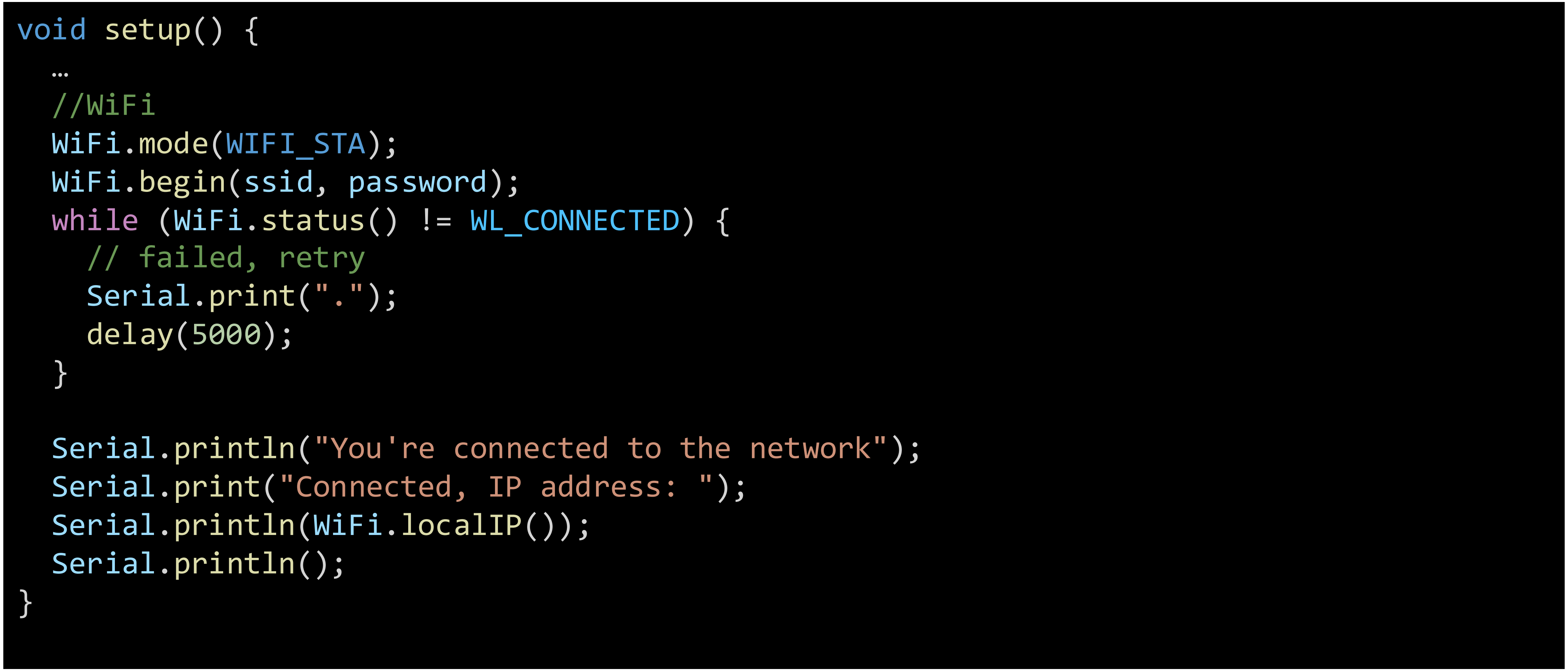
- Comment out `println("Hello World!");`
- Includes of additional **libraries** and create global **variables/objects**:

```
#include <WiFi.h>

//WiFi
WiFiClient wifiClient;
const char* ssid = "<SSID_for_wifi>"; // put in your WIFI SSID, provided in lab
const char* password = "<password_for_wifi>"; // put in WIFI password

void setup() {
  …
}

void loop() {
  …
}
```

WIFI Name:	THMnet
WIFI Password:	

Exercise 2 – “Connect to local wifi“ (2/3)

- in **setup()-function** add following code:



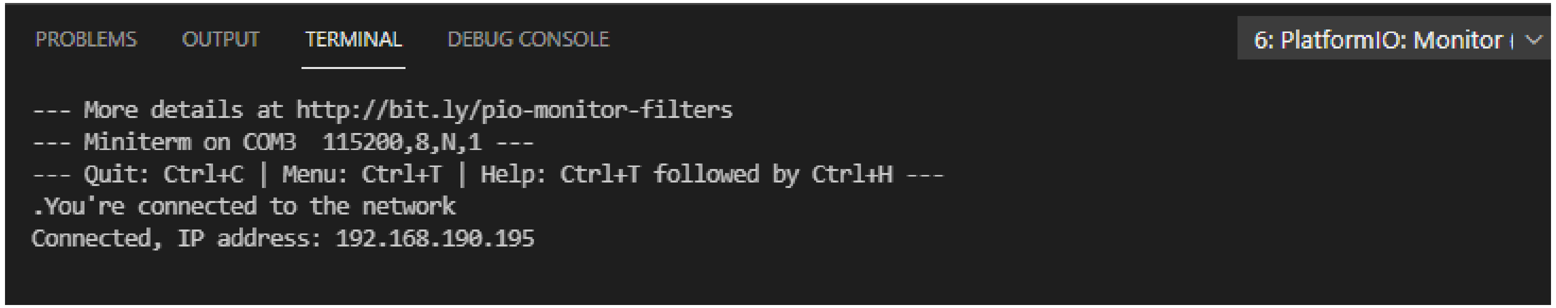
```
void setup() {
  …
  //WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }

  Serial.println("You're connected to the network");
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
}
```

Exercise 2 – “Connect to local wifi“ (3/3)

- in **loop()-function** add:

```
void loop() {
  while (WiFi.status() != WL_CONNECTED || WiFi.localIP() == IPAddress(0,0,0,0)) {
    WiFi.reconnect();
    delay(5000);
  }




}
```

- **Output** in Serial Monitor should show connected and local IP Address



```
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on COM3  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
.You're connected to the network
Connected, IP address: 192.168.190.195
```

Exercise 3 – “Connect MQTT Broker“ (1/4)

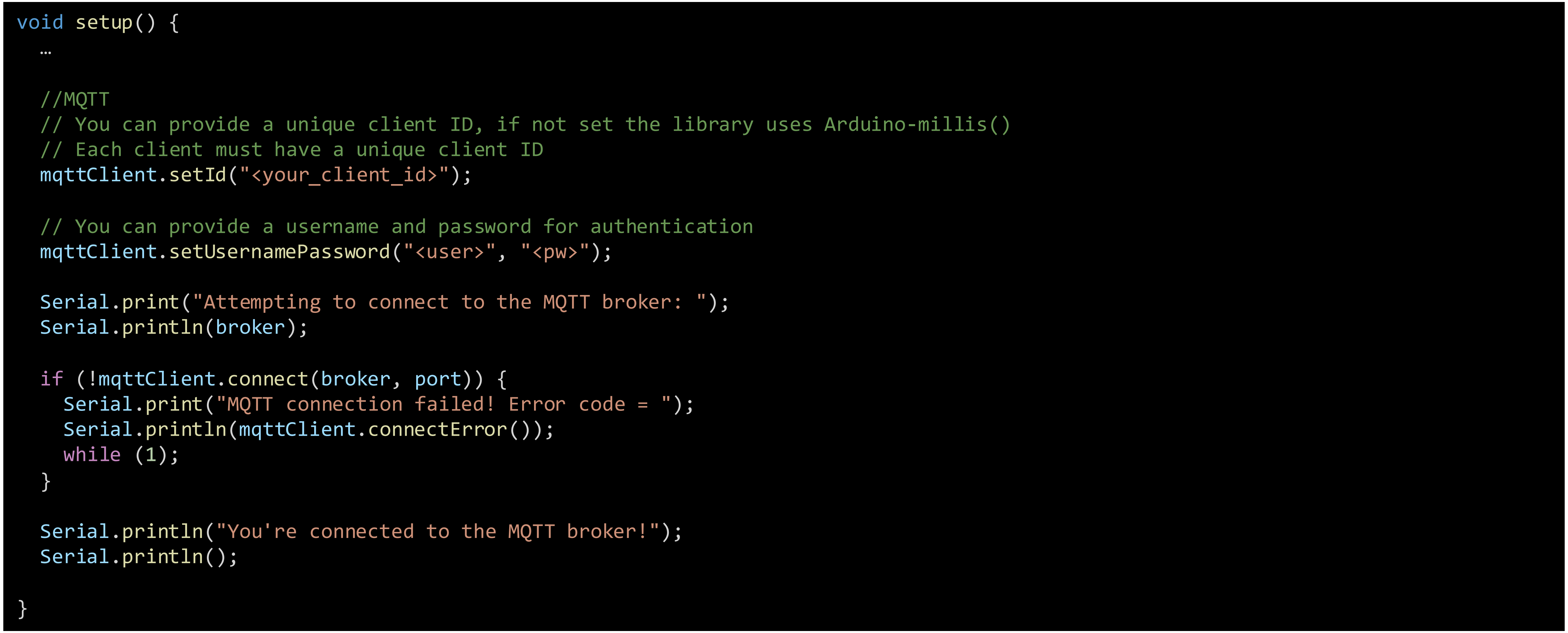
- **Install „ArduinoMQTTClient“ Library V0.1.8** in your project and create global **variables/objects**:

```
#include <ArduinoMqttClient.h>

//MQTT
MqttClient mqttClient(wifiClient);
const char broker[] = "<FQDN of Broker>"; // MQTT Broker FQDN, will be provided in lab
int        port     = 1993; // MQTT Port
const char topic[]  = "<topic>"; // MQTT Topic

void setup() {
 …
}

void loop() {
 …
}
```

MQTT FQDN:	MQTT.EI.THM.DE
MQTT Port:	1993
MQTT Topic:	THM/IoTLab/yourname/BME680
MQTT User:	iotlab
MQTT Password:	iotlab

Exercise 3 – “Connect MQTT Broker“ (2/4)

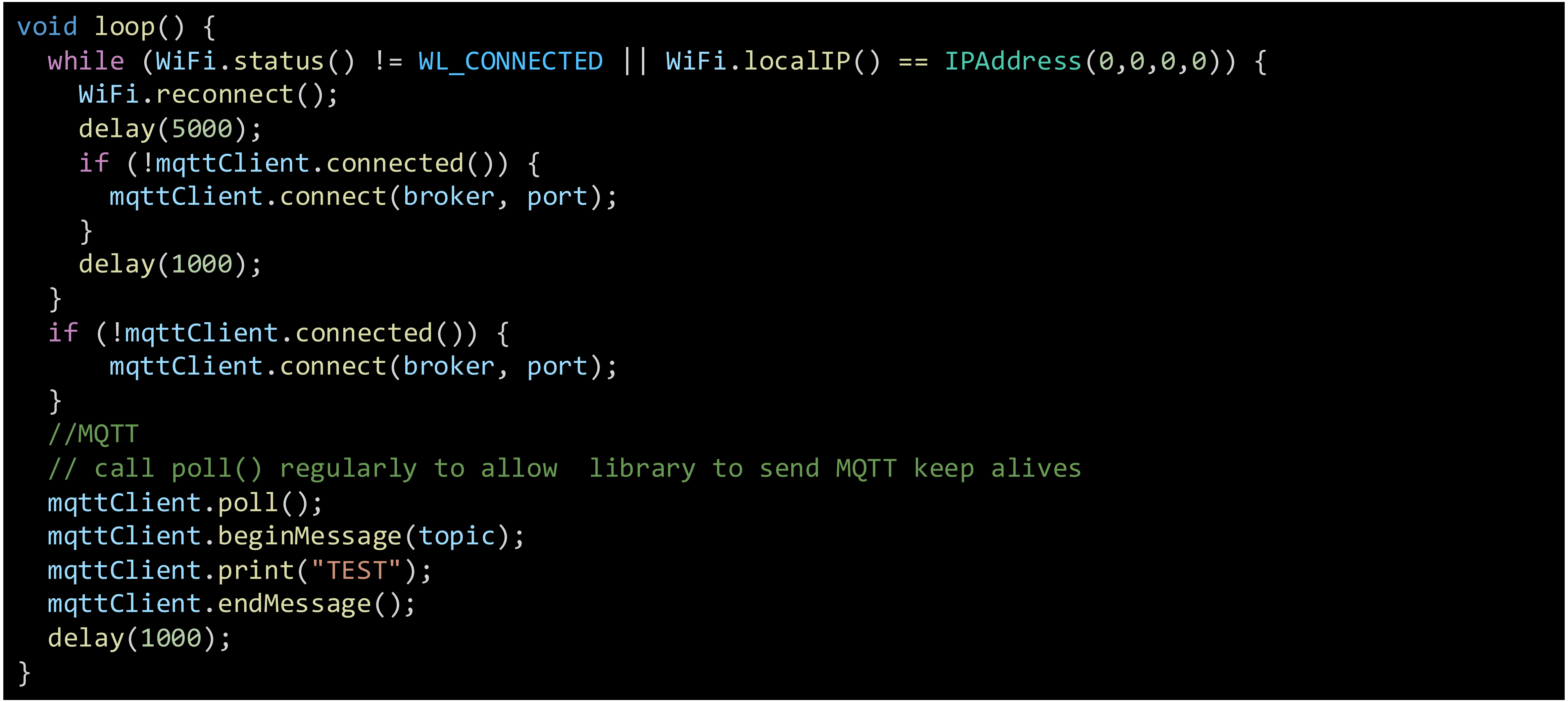
- in **setup()-function** add following code:



```
void setup() {
  …

  //MQTT
  // You can provide a unique client ID, if not set the library uses Arduino-millis()
  // Each client must have a unique client ID
  mqttClient.setId("<your_client_id>");

  // You can provide a username and password for authentication
  mqttClient.setUsernamePassword("<user>", "<pw>");

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());
    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();

}
```

Exercise 3 – “Connect MQTT Broker“ (3/4)

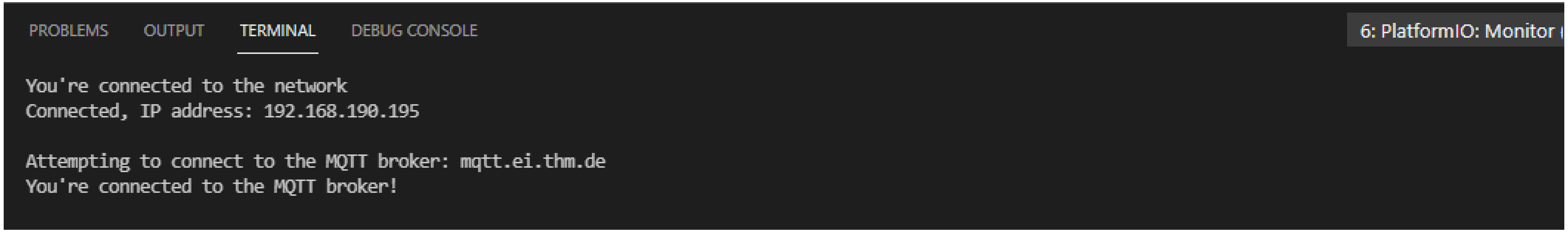
- in **loop()-function** add:



```
void loop() {
  while (WiFi.status() != WL_CONNECTED || WiFi.localIP() == IPAddress(0,0,0,0)) {
    WiFi.reconnect();
    delay(5000);
    if (!mqttClient.connected()) {
      mqttClient.connect(broker, port);
    }
    delay(1000);
  }
  if (!mqttClient.connected()) {
      mqttClient.connect(broker, port);
  }
  //MQTT
  // call poll() regularly to allow  library to send MQTT keep alives
  mqttClient.poll();
  mqttClient.beginMessage(topic);
  mqttClient.print("TEST");
  mqttClient.endMessage();
  delay(1000);
}
```

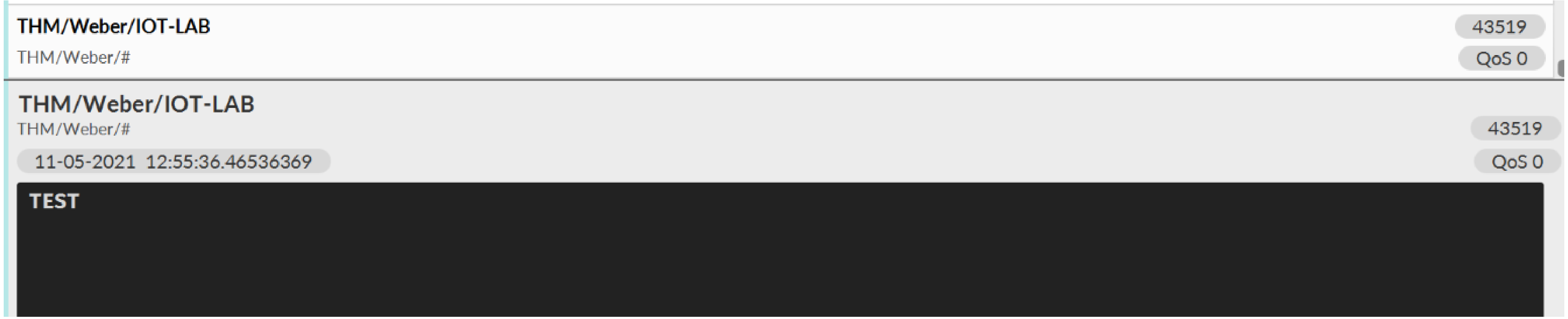
Exercise 3 – “Connect MQTT Broker“ (4/4)

- **Output** in Serial Monitor should show connected and local IP Address



```
You're connected to the network
Connected, IP address: 192.168.190.195

Attempting to connect to the MQTT broker: mqtt.ei.thm.de
You're connected to the MQTT broker!
```

- Check sending MQTT message to broker with **MQTT Explorer** e.g.:



THM/Weber/IOT-LAB
THM/Weber/#
43519
QoS 0

THM/Weber/IOT-LAB
THM/Weber/#
43519
11-05-2021 12:55:36.46536369
QoS 0

TEST

Exercise 4 – “Read Sensor Data“ (1/5)

- Ensure that BME680 sensor board is connected
- **Install „BSEC Software Library“ V1.8.1492 in your project** and create global **variables/objects**:

```
#include <bsec.h>
#include <Wire.h>

// BME680
#define SDA2 21 // Data_BME680
#define SCL2 13 // Clock_BME680
TwoWire I2Ctwo = TwoWire(1); // for BME680

// Helper functions declarations
void checkIaqSensorStatus(void);
void errLeds(void);

// Create an object of the class Bsec
Bsec iaqSensor;

String output; // output string: error msg or sensor data
```

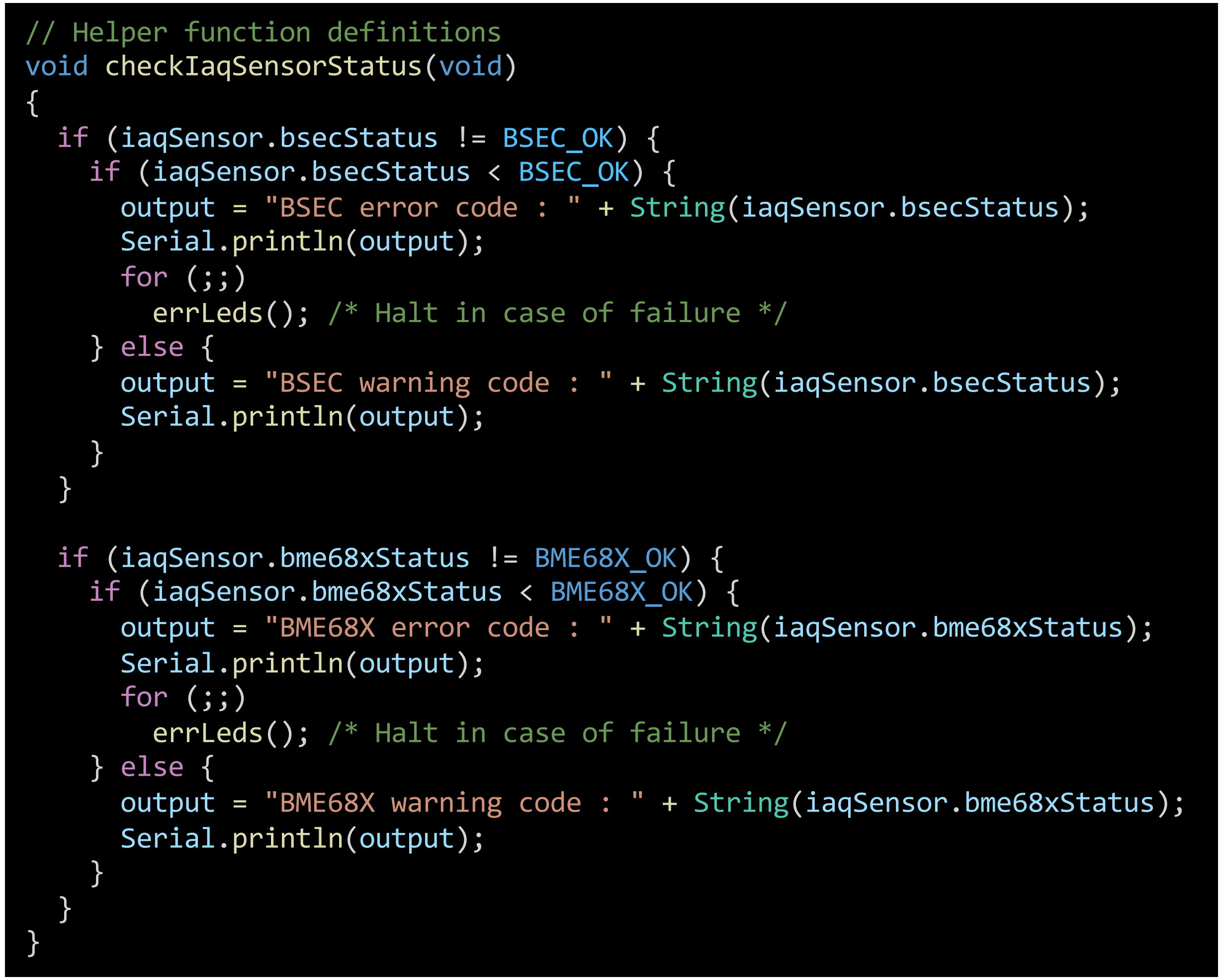
Program the ESP32

- We need to specify some versions of libraries we use
  - Navigate to your platformio.ini file in the document-tree
  - Change the version numbers to those (maybe you need to install these versions):

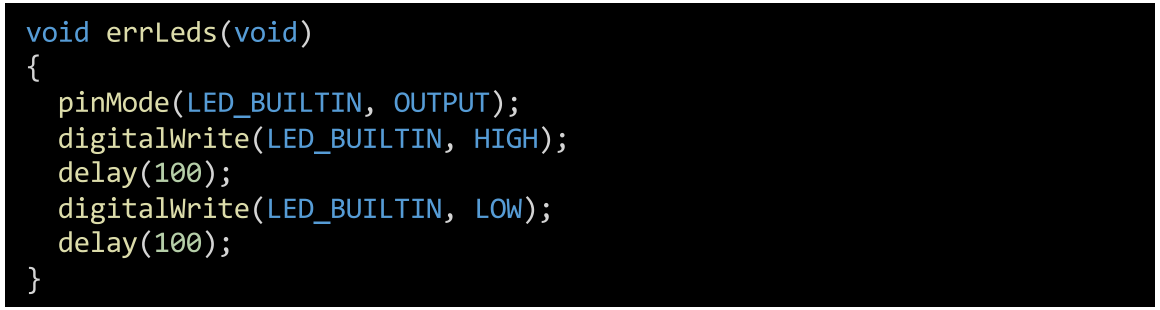
```
[env:heltec_wifi_lora_32_V2]
platform = espressif32@4.0.0
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.2
    arduino-libraries/ArduinoMqttClient@^0.1.8
    boschsensortec/BSEC Software Library@^1.8.1492
monitor_speed = 115200
```

Exercise 4 – “Read Sensor Data“ (2/5)

- Add **helper-functions** below **loop-function**:



```
// Helper function definitions
void checkIaqSensorStatus(void)
{
  if (iaqSensor.bsecStatus != BSEC_OK) {
    if (iaqSensor.bsecStatus < BSEC_OK) {
      output = "BSEC error code : " + String(iaqSensor.bsecStatus);
      Serial.println(output);
      for (;;)
        errLeds(); /* Halt in case of failure */
    } else {
      output = "BSEC warning code : " + String(iaqSensor.bsecStatus);
      Serial.println(output);
    }
  }

  if (iaqSensor.bme68xStatus != BME68X_OK) {
    if (iaqSensor.bme68xStatus < BME68X_OK) {
      output = "BME68X error code : " + String(iaqSensor.bme68xStatus);
      Serial.println(output);
      for (;;)
        errLeds(); /* Halt in case of failure */
    } else {
      output = "BME68X warning code : " + String(iaqSensor.bme68xStatus);
      Serial.println(output);
    }
  }
}
```



```
void errLeds(void)
{
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
}
```

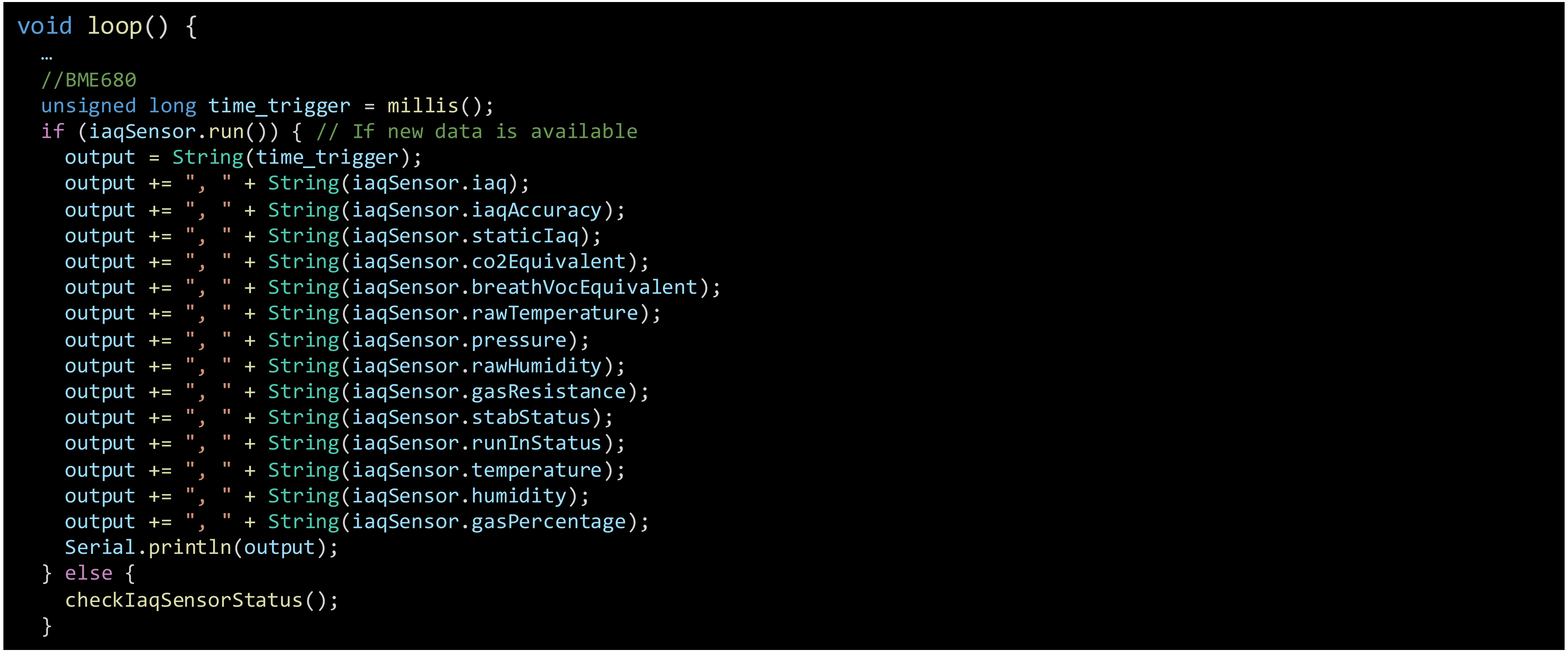
Exercise 4 – “Read Sensor Data“ (3/5)

- in **setup()-function** add following code… **Note: Use I2Ctwo.begin(SDA2, SCL2) below**

```
void setup() {
  …
  // BME680
  I2Ctwo.begin(SDA2,SCL2); // BME680 I2C 2nd Bus
  iaqSensor.begin(BME68X_I2C_ADDR_HIGH, I2Ctwo); // BME680 I2C 2nd Bus
  output = "\nBSEC library version " + String(iaqSensor.version.major) + "." + String(iaqSensor.version.minor) + "." +
String(iaqSensor.version.major_bugfix) + "." + String(iaqSensor.version.minor_bugfix);
  Serial.println(output);
  checkIaqSensorStatus();
  bsec_virtual_sensor_t sensorList[13] = {
    BSEC_OUTPUT_IAQ,
    BSEC_OUTPUT_STATIC_IAQ,
    BSEC_OUTPUT_CO2_EQUIVALENT,
    BSEC_OUTPUT_BREATH_VOC_EQUIVALENT,
    BSEC_OUTPUT_RAW_TEMPERATURE,
    BSEC_OUTPUT_RAW_PRESSURE,
    BSEC_OUTPUT_RAW_HUMIDITY,
    BSEC_OUTPUT_RAW_GAS,
    BSEC_OUTPUT_STABILIZATION_STATUS,
    BSEC_OUTPUT_RUN_IN_STATUS,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_TEMPERATURE,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_HUMIDITY,
    BSEC_OUTPUT_GAS_PERCENTAGE
  };
  iaqSensor.updateSubscription(sensorList, 13, BSEC_SAMPLE_RATE_LP);
  checkIaqSensorStatus();
  // Print the header
  output = "Timestamp [ms], IAQ, IAQ accuracy, Static IAQ, CO2 equivalent, breath VOC equivalent, raw temp[° C], pressure [hPa], raw relative humidity [%],
gas [Ohm], Stab Status, run in status, comp temp[° C], comp humidity [%], gas percentage";
  Serial.println(output);
}
```

Exercise 4 – “Read Sensor Data“ (4/5)

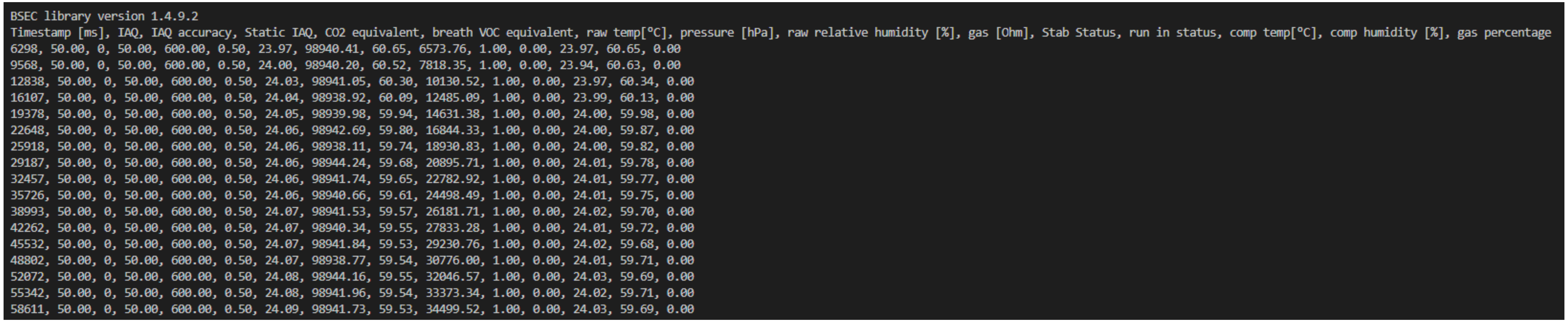
- in **loop()-function** add (behind `delay(1000);`):



```
void loop() {
  …
  //BME680
  unsigned long time_trigger = millis();
  if (iaqSensor.run()) { // If new data is available
    output = String(time_trigger);
    output += ", " + String(iaqSensor.iaq);
    output += ", " + String(iaqSensor.iaqAccuracy);
    output += ", " + String(iaqSensor.staticIaq);
    output += ", " + String(iaqSensor.co2Equivalent);
    output += ", " + String(iaqSensor.breathVocEquivalent);
    output += ", " + String(iaqSensor.rawTemperature);
    output += ", " + String(iaqSensor.pressure);
    output += ", " + String(iaqSensor.rawHumidity);
    output += ", " + String(iaqSensor.gasResistance);
    output += ", " + String(iaqSensor.stabStatus);
    output += ", " + String(iaqSensor.runInStatus);
    output += ", " + String(iaqSensor.temperature);
    output += ", " + String(iaqSensor.humidity);
    output += ", " + String(iaqSensor.gasPercentage);
    Serial.println(output);
  } else {
    checkIaqSensorStatus();
  }
```

Exercise 4 – “Read Sensor Data“ (5/5)

- **Output** in Serial Monitor should show BME680 Sensor Data



```
BSEC library version 1.4.9.2
Timestamp [ms], IAQ, IAQ accuracy, Static IAQ, CO2 equivalent, breath VOC equivalent, raw temp[°C], pressure [hPa], raw relative humidity [%], gas [Ohm], Stab Status, run in status, comp temp[°C], comp humidity [%], gas percentage
6298, 50.00, 0, 50.00, 600.00, 0.50, 23.97, 98940.41, 60.65, 6573.76, 1.00, 0.00, 23.97, 60.65, 0.00
9568, 50.00, 0, 50.00, 600.00, 0.50, 24.00, 98940.20, 60.52, 7818.35, 1.00, 0.00, 23.94, 60.63, 0.00
12838, 50.00, 0, 50.00, 600.00, 0.50, 24.03, 98941.05, 60.30, 10130.52, 1.00, 0.00, 23.97, 60.34, 0.00
16107, 50.00, 0, 50.00, 600.00, 0.50, 24.04, 98938.92, 60.09, 12485.09, 1.00, 0.00, 23.99, 60.13, 0.00
19378, 50.00, 0, 50.00, 600.00, 0.50, 24.05, 98939.98, 59.94, 14631.38, 1.00, 0.00, 24.00, 59.98, 0.00
22648, 50.00, 0, 50.00, 600.00, 0.50, 24.06, 98942.69, 59.80, 16844.33, 1.00, 0.00, 24.00, 59.87, 0.00
25918, 50.00, 0, 50.00, 600.00, 0.50, 24.06, 98938.11, 59.74, 18930.83, 1.00, 0.00, 24.00, 59.82, 0.00
29187, 50.00, 0, 50.00, 600.00, 0.50, 24.06, 98944.24, 59.68, 20895.71, 1.00, 0.00, 24.01, 59.78, 0.00
32457, 50.00, 0, 50.00, 600.00, 0.50, 24.06, 98941.74, 59.65, 22782.92, 1.00, 0.00, 24.01, 59.77, 0.00
35726, 50.00, 0, 50.00, 600.00, 0.50, 24.06, 98940.66, 59.61, 24498.49, 1.00, 0.00, 24.01, 59.75, 0.00
38993, 50.00, 0, 50.00, 600.00, 0.50, 24.07, 98941.53, 59.57, 26181.71, 1.00, 0.00, 24.02, 59.70, 0.00
42262, 50.00, 0, 50.00, 600.00, 0.50, 24.07, 98940.34, 59.55, 27833.28, 1.00, 0.00, 24.01, 59.72, 0.00
45532, 50.00, 0, 50.00, 600.00, 0.50, 24.07, 98941.84, 59.53, 29230.76, 1.00, 0.00, 24.02, 59.68, 0.00
48802, 50.00, 0, 50.00, 600.00, 0.50, 24.07, 98938.77, 59.54, 30776.00, 1.00, 0.00, 24.01, 59.71, 0.00
52072, 50.00, 0, 50.00, 600.00, 0.50, 24.08, 98944.16, 59.55, 32046.57, 1.00, 0.00, 24.03, 59.69, 0.00
55342, 50.00, 0, 50.00, 600.00, 0.50, 24.08, 98941.96, 59.54, 33373.34, 1.00, 0.00, 24.02, 59.71, 0.00
58611, 50.00, 0, 50.00, 600.00, 0.50, 24.09, 98941.73, 59.53, 34499.52, 1.00, 0.00, 24.03, 59.69, 0.00
```

- Try to „manipulate“ sensor data like temperature / humidity

Exercise 5 – “Show Data on OLED“ (1/4)

- Create global **variables/objects**:

```
//  OLED + LoRa
#define BAND    868E6  //for Lab3b / you can set band here directly,e.g. 868E6,915E6
#define SDA1 SDA_OLED
#define SCL1 SCL_OLED
TwoWire I2Cone = TwoWire(0); // OLED
```

Exercise 5 – “Show Data on OLED“ (2/4)

- in **setup()-function** add following code:

```
void setup() {
  …
  //OLED
  Heltec.begin(true /*DisplayEnable Enable*/, false /*LoRa Enable*/, true /*Serial Enable*/, false /*LoRa use PABOOST*/, BAND /*LoRa RF working band*/);
  Heltec.display -> clear();
  Heltec.display -> drawString(0, 0, "TEST");
  Heltec.display -> display();
}
```

Exercise 5 – “Show Data on OLED“ (3/4)

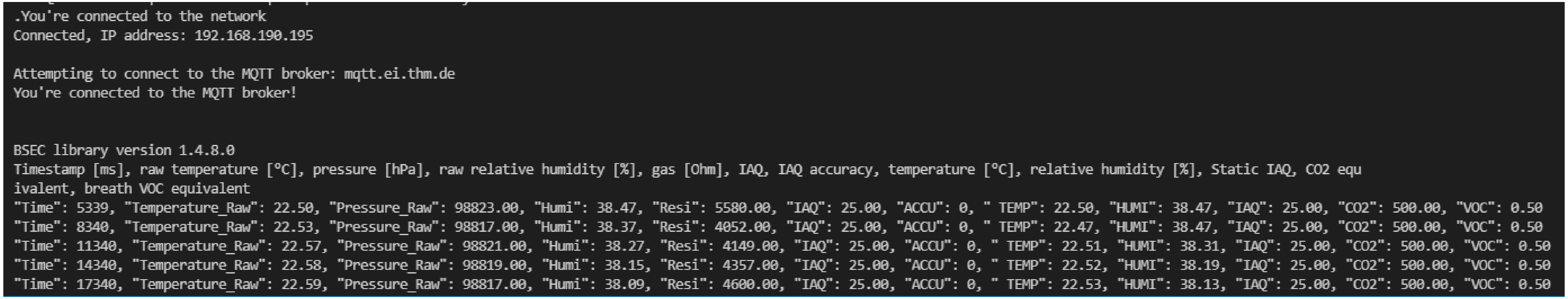
- in **loop()-function** add (behind `Serial.println(output);`):

```
void loop() {
     …
    //OLED
    Heltec.display -> clear();
    Heltec.display -> setFont(ArialMT_Plain_10);
    Heltec.display -> drawString(0, 0, "Temp: " + String(iaqSensor.temperature)+ " ° C");
    Heltec.display -> drawString(0, 10, "Humid: " + String(iaqSensor.humidity) + " %");
    Heltec.display -> drawString(0, 20, "IAQ: " + String(iaqSensor.iaq) + "IAQ_A: " + String(iaqSensor.iaqAccuracy));
    Heltec.display -> drawString(0, 30, "VOC: " + String(iaqSensor.breathVocEquivalent));
    Heltec.display -> drawString(0, 40, "Press: " + String(iaqSensor.pressure));
    Heltec.display -> drawString(0, 50, "CO2: " + String(iaqSensor.co2Equivalent));
    Heltec.display -> display();
    delay(100);
    Heltec.display -> clear();
    …
}
```

- **Output** in Serial Monitor should show connected and local IP Address

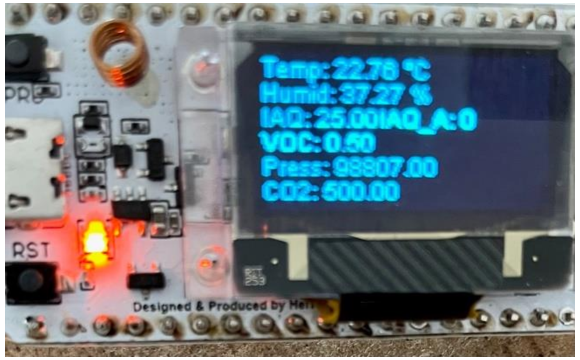
Exercise 5 – “Show Data on OLED“ (4/4)

- **Output** in Serial Monitor should show BME680 Sensor Data (same as before)



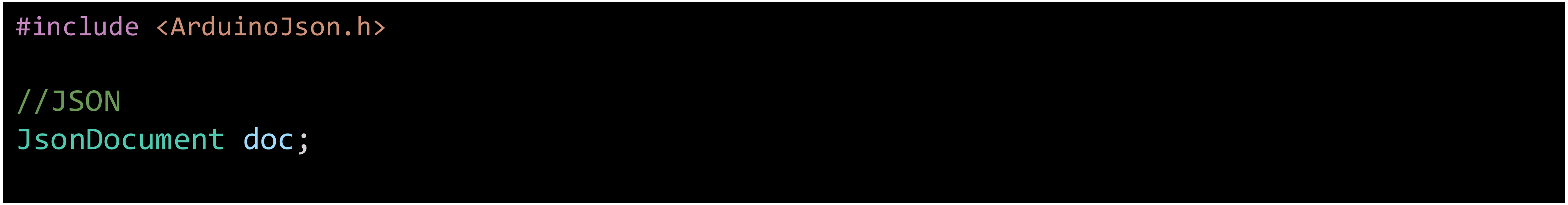
```
.You're connected to the network
Connected, IP address: 192.168.190.195

Attempting to connect to the MQTT broker: mqtt.ei.thm.de
You're connected to the MQTT broker!


BSEC library version 1.4.8.0
Timestamp [ms], raw temperature [°C], pressure [hPa], raw relative humidity [%], gas [Ohm], IAQ, IAQ accuracy, temperature [°C], relative humidity [%], Static IAQ, CO2 equ
ivalent, breath VOC equivalent
"Time": 5339, "Temperature_Raw": 22.50, "Pressure_Raw": 98823.00, "Humi": 38.47, "Resi": 5580.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.50, "HUMI": 38.47, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 8340, "Temperature_Raw": 22.53, "Pressure_Raw": 98817.00, "Humi": 38.37, "Resi": 4052.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.47, "HUMI": 38.47, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 11340, "Temperature_Raw": 22.57, "Pressure_Raw": 98821.00, "Humi": 38.27, "Resi": 4149.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.51, "HUMI": 38.31, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 14340, "Temperature_Raw": 22.58, "Pressure_Raw": 98819.00, "Humi": 38.15, "Resi": 4357.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.52, "HUMI": 38.19, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 17340, "Temperature_Raw": 22.59, "Pressure_Raw": 98817.00, "Humi": 38.09, "Resi": 4600.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.53, "HUMI": 38.13, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
```

- **OLED Display** should show sensor data additionally:



Exercise 6 – “Send JSON via MQTT “ (1/4)

- **Install „ArduinoJSON Library“ v7.1.0 in your project** and create global **variables/objects**:



```
#include <ArduinoJson.h>

//JSON
JsonDocument doc;
```

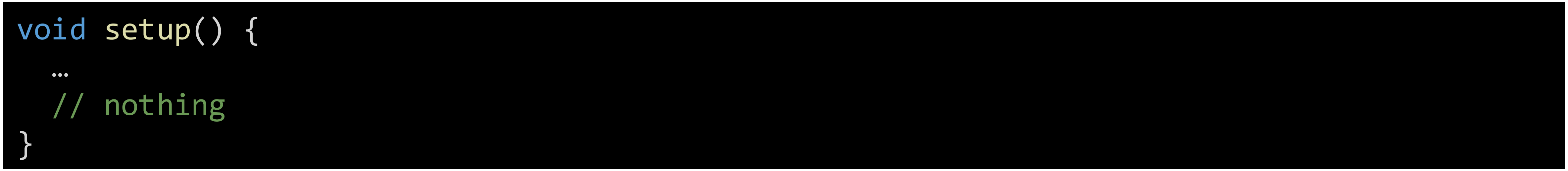
Program the ESP32

- We need to specify some versions of libraries we use
  - Navigate to your platformio.ini file in the document-tree
  - Change the version numbers to those (maybe you need to install these versions):

```
[env:heltec_wifi_lora_32_V2]
platform = espressif32@4.0.0
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.2
    arduino-libraries/ArduinoMqttClient@^0.1.8
    boschsensortec/BSEC Software Library@^1.8.1492
    bblanchon/ArduinoJson@^7.1.0
monitor_speed = 115200
```

Exercise 6 – “Send JSON via MQTT “ (2/4)

- in **setup()-function** add following code:



```
void setup() {
  …
  // nothing
}
```

Exercise 6 – “Send JSON via MQTT “ (3/4)

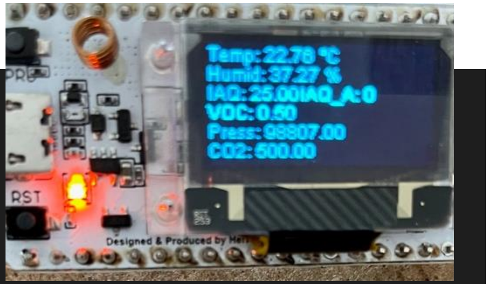
- in **loop()-function** add (behind `Heltec.display -> clear();`):
- Don‘t forget to comment lines of the TEST message from previous steps

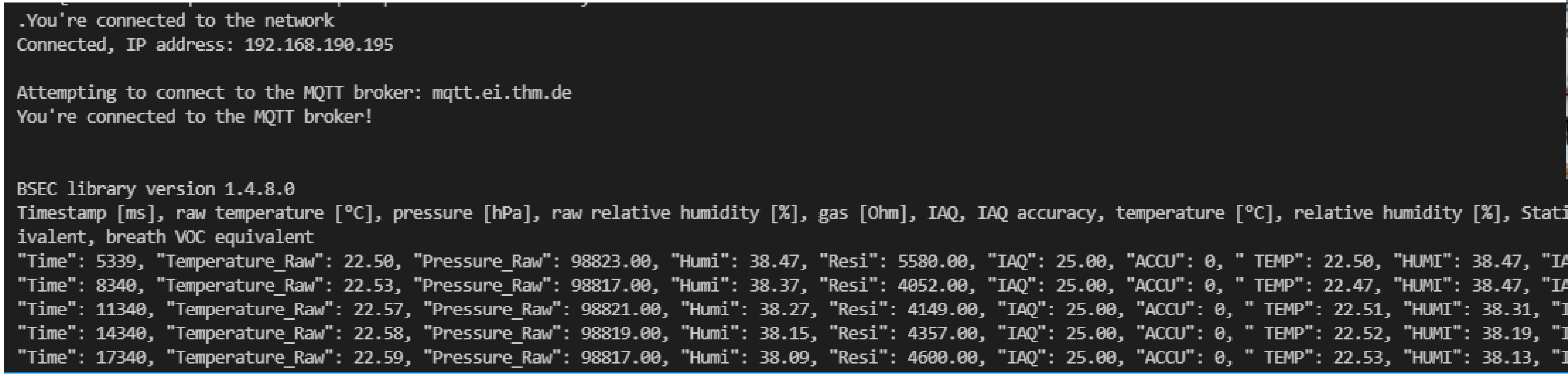
```
void loop() {
…
    //JSON via MQTT
    doc["SensorID"] =  "BME680_Weber";
    doc["Temp_R"] =  String(iaqSensor.rawTemperature);
    doc["Pres_R"] = String(iaqSensor.pressure);
    doc["Humid_R"] = String(iaqSensor.rawHumidity);
    doc["GasResi_R"] = String(iaqSensor.gasResistance);
    doc["IAQ"] = String(iaqSensor.iaq);
    doc["IAQ_A"] = String(iaqSensor.iaqAccuracy);
    doc["Temp"] = String(iaqSensor.temperature);
    doc["Humid"] = String(iaqSensor.humidity);
    doc["IAQ_S"] = String(iaqSensor.staticIaq);
    doc["CO2"] = String(iaqSensor.co2Equivalent);
    doc["VOC"] = String(iaqSensor.breathVocEquivalent);

    mqttClient.beginMessage(topic);
    serializeJson(doc, mqttClient);
    mqttClient.endMessage();
    doc.clear();

    //mqttClient.beginMessage(topic);
    //mqttClient.print("TEST");
    //mqttClient.endMessage();
}
```

- **Output** in Serial Monitor should show connected and local IP Address

Exercise 6 – “Send JSON via MQTT “ (4/4)

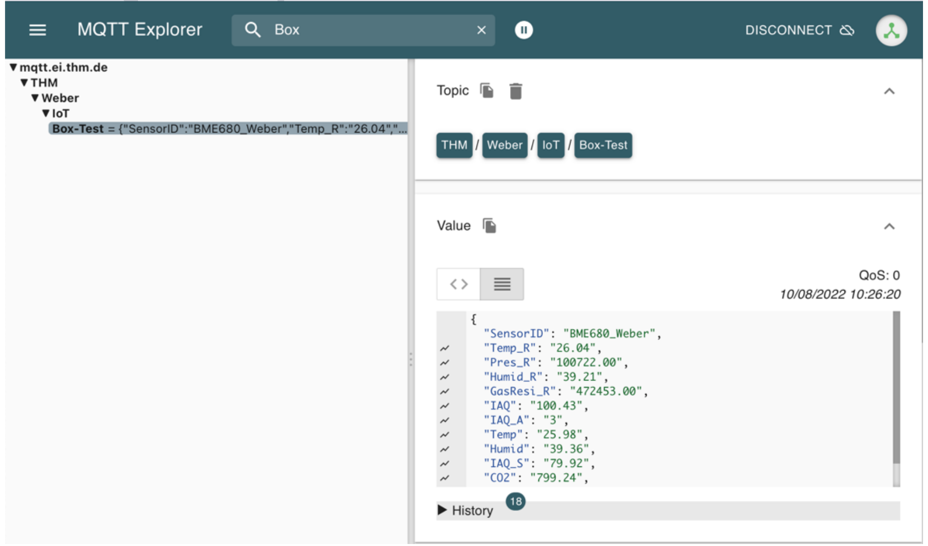
- **Output** in Serial Monitor and **OLED Display** should show BME680 Sensor Data





```
.You're connected to the network
Connected, IP address: 192.168.190.195

Attempting to connect to the MQTT broker: mqtt.ei.thm.de
You're connected to the MQTT broker!


BSEC library version 1.4.8.0
Timestamp [ms], raw temperature [°C], pressure [hPa], raw relative humidity [%], gas [Ohm], IAQ, IAQ accuracy, temperature [°C], relative humidity [%], Static IAQ, CO2 equ
ivalent, breath VOC equivalent
"Time": 5339, "Temperature_Raw": 22.50, "Pressure_Raw": 98823.00, "Humi": 38.47, "Resi": 5580.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.50, "HUMI": 38.47, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 8340, "Temperature_Raw": 22.53, "Pressure_Raw": 98817.00, "Humi": 38.37, "Resi": 4052.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.47, "HUMI": 38.47, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 11340, "Temperature_Raw": 22.57, "Pressure_Raw": 98821.00, "Humi": 38.27, "Resi": 4149.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.51, "HUMI": 38.31, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 14340, "Temperature_Raw": 22.58, "Pressure_Raw": 98819.00, "Humi": 38.15, "Resi": 4357.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.52, "HUMI": 38.19, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
"Time": 17340, "Temperature_Raw": 22.59, "Pressure_Raw": 98817.00, "Humi": 38.09, "Resi": 4600.00, "IAQ": 25.00, "ACCU": 0, " TEMP": 22.53, "HUMI": 38.13, "IAQ": 25.00, "CO2": 500.00, "VOC": 0.50
```

- **MQTT Explorer** should show sensor data as JSON String:



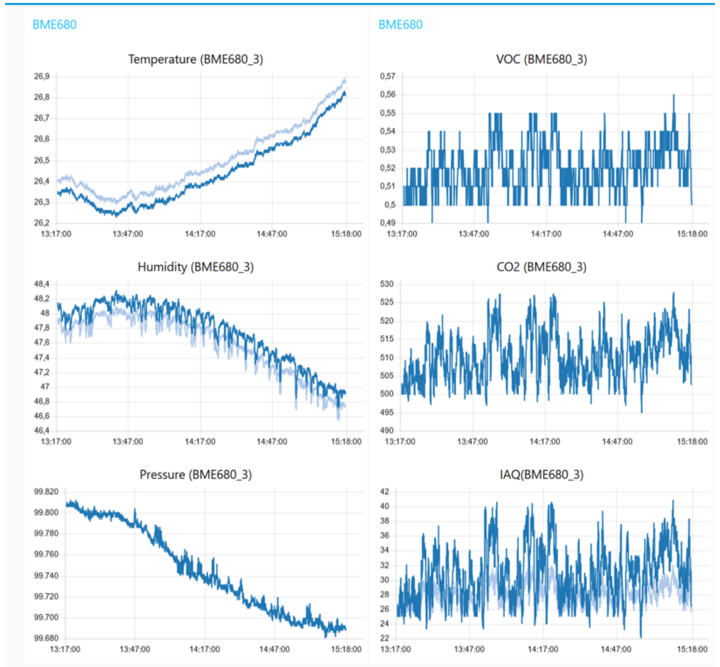
▼mqtt.ei.thm.de
 ▼THM
  ▼Weber
   ▼IoT
    Box-Test = {"SensorID":"BME680_Weber","Temp_R":"26.04","...

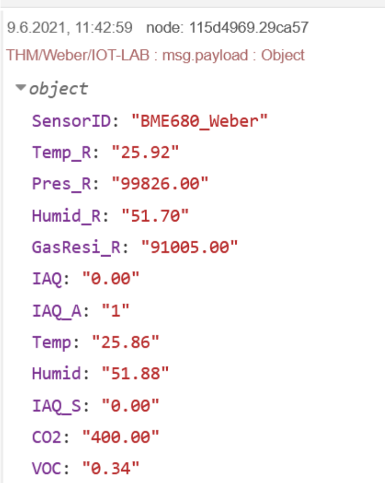
QoS: 0
10/08/2022 10:26:20

```
{
  "SensorID": "BME680_Weber",
  "Temp_R": "26.04",
  "Pres_R": "100722.00",
  "Humid_R": "39.21",
  "GasResi_R": "472453.00",
  "IAQ": "100.43",
  "IAQ_A": "3",
  "Temp": "25.98",
  "Humid": "39.36",
  "IAQ_S": "79.92",
  "CO2": "799.24",
```

Exercise 7 – “local Cloud“ (1/2)

- Login into your local docker cloud(from Lab 1)
- Use youe **NodeRed**
- **Create** new flow, **subscribe** to your topic and **show** sensor data in debug window
- Convert JSON to JS-Object with JSON Node (if needed)
- Visualize Data on Dashboard in Node-Red:

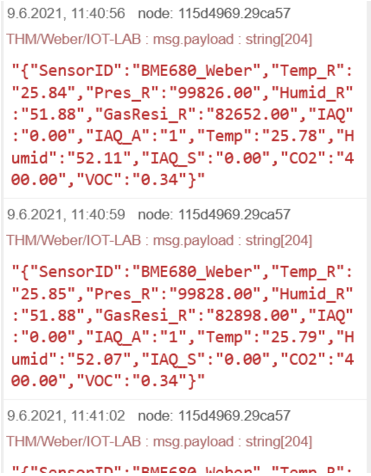




9.6.2021, 11:42:59 node: 115d4969.29ca57

THM/Weber/IOT-LAB : msg.payload : Object

```
▾object
  SensorID: "BME680_Weber"
  Temp_R: "25.92"
  Pres_R: "99826.00"
  Humid_R: "51.70"
  GasResi_R: "91005.00"
  IAQ: "0.00"
  IAQ_A: "1"
  Temp: "25.86"
  Humid: "51.88"
  IAQ_S: "0.00"
  CO2: "400.00"
  VOC: "0.34"
```



9.6.2021, 11:40:56 node: 115d4969.29ca57

THM/Weber/IOT-LAB : msg.payload : string[204]

```
"{"SensorID":"BME680_Weber","Temp_R":
"25.84","Pres_R":"99826.00","Humid_R"
:"51.88","GasResi_R":"82652.00","IAQ"
:"0.00","IAQ_A":"1","Temp":"25.78","H
umid":"52.11","IAQ_S":"0.00","CO2":"4
00.00","VOC":"0.34"}"
```

9.6.2021, 11:40:59 node: 115d4969.29ca57

THM/Weber/IOT-LAB : msg.payload : string[204]

```
"{"SensorID":"BME680_Weber","Temp_R":
"25.85","Pres_R":"99828.00","Humid_R"
:"51.88","GasResi_R":"82898.00","IAQ"
:"0.00","IAQ_A":"1","Temp":"25.79","H
umid":"52.07","IAQ_S":"0.00","CO2":"4
00.00","VOC":"0.34"}"
```

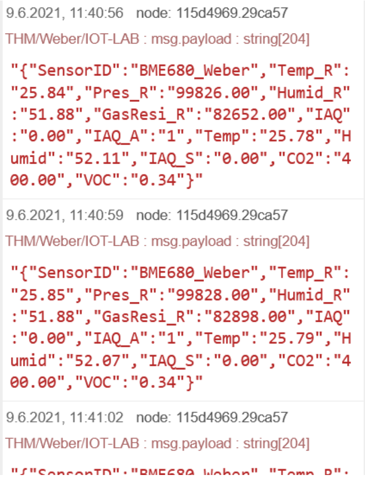
9.6.2021, 11:41:02 node: 115d4969.29ca57

THM/Weber/IOT-LAB : msg.payload : string[204]

```
"{"SensorID":"BME680_Weber","Temp_R":
```

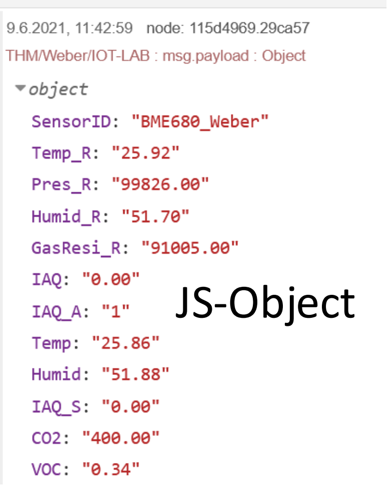
Exercise 7 – “EI Cloud“ (2/2)

- Login into EI Cloud (https://cloud.ei.thm.de)
- Use your „Node-Red“ you created as Swarm in Lab 2 on your Endpoint
- **Create** new flow, **subscribe** to your topic and **show** sensor data in debug window
- Convert JSON to JS-Object with JSON Node if needed
- Store and Visualize Data on
  - (optional) Dashboard in Node-Red
  - InfluxDB & Grafana



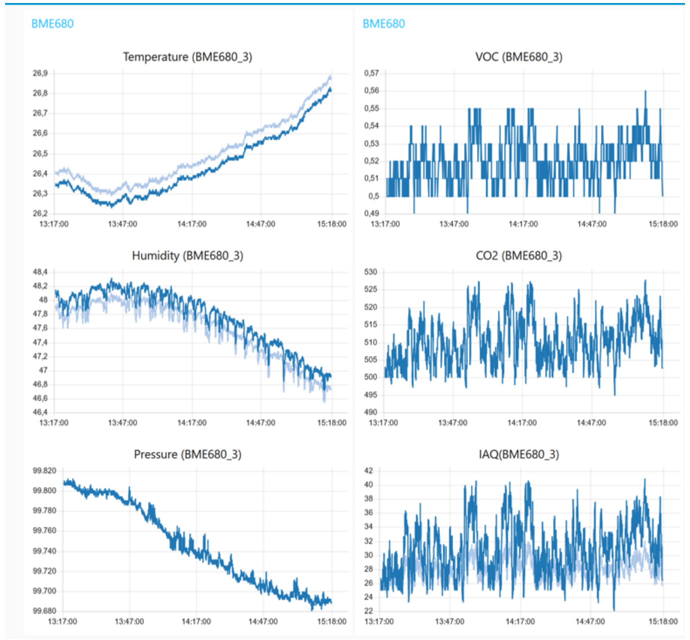
```
9.6.2021, 11:40:56   node: 115d4969.29ca57
THM/Weber/IOT-LAB : msg.payload : string[204]
"{"SensorID":"BME680_Weber","Temp_R":
"25.84","Pres_R":"99826.00","Humid_R"
:"51.88","GasResi_R":"82652.00","IAQ"
:"0.00","IAQ_A":"1","Temp":"25.78","H
umid":"52.11","IAQ_S":"0.00","CO2":"4
00.00","VOC":"0.34"}"
9.6.2021, 11:40:59   node: 115d4969.29ca57
THM/Weber/IOT-LAB : msg.payload : string[204]
"{"SensorID":"BME680_Weber","Temp_R":
"25.85","Pres_R":"99828.00","Humid_R"
:"51.88","GasResi_R":"82898.00","IAQ"
:"0.00","IAQ_A":"1","Temp":"25.79","H
umid":"52.07","IAQ_S":"0.00","CO2":"4
00.00","VOC":"0.34"}"
9.6.2021, 11:41:02   node: 115d4969.29ca57
THM/Weber/IOT-LAB : msg.payload : string[204]
```

JSON



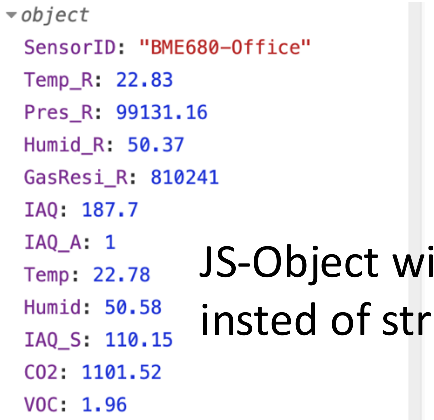
```
9.6.2021, 11:42:59   node: 115d4969.29ca57
THM/Weber/IOT-LAB : msg.payload : Object
▾object
  SensorID: "BME680_Weber"
  Temp_R: "25.92"
  Pres_R: "99826.00"
  Humid_R: "51.70"
  GasResi_R: "91005.00"
  IAQ: "0.00"
  IAQ_A: "1"
  Temp: "25.86"
  Humid: "51.88"
  IAQ_S: "0.00"
  CO2: "400.00"
  VOC: "0.34"
```

JS-Object



Note: In NodeRed you need to convert the string to a float using the JS Function Number():

Example:
msg.payload = Number(msg.payload.Temp_R);



```
▾object
  SensorID: "BME680-Office"
  Temp_R: 22.83
  Pres_R: 99131.16
  Humid_R: 50.37
  GasResi_R: 810241
  IAQ: 187.7
  IAQ_A: 1
  Temp: 22.78
  Humid: 50.58
  IAQ_S: 110.15
  CO2: 1101.52
  VOC: 1.96
```

JS-Object with float insted of string values