Sensoren en interfacing – Evaluatie week 5

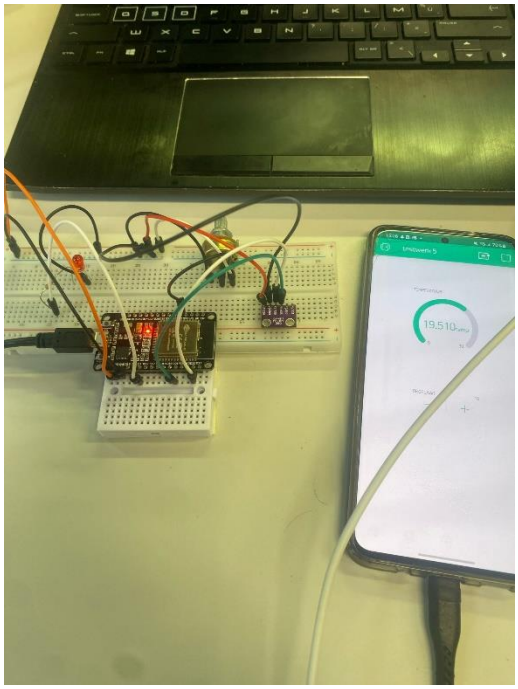
Opdracht: ontwerp (op basis van een microcontroller naar keuze en sensoren naar keuze) een IoT thermostaat voor een centraal verwarmingssysteem:

1. Temperatuur meting (0-30 graden op basis van een digitale sensor naar keuze).
2. Gewenste temperatuur instelling (0-30 graden met knopjes of potentiometer)
3. Lokale weergave van de gemeten en gewenste temperatuur en de toestand van de verwarmingsketel (aan/uit) naar keuze via serial monitor, LED display, LCD, OLED,… Ketel zelf wordt weergegeven door een LED.
4. Via de Blynk applicatie volledige bediening vanaf je telefoon (weergave van actuele temperatuur, ingestelde temperatuur (met instelknop) en status van de verwarmingsketel.

Al je antwoorden komen op dit document. **Je plakt telkens onder de vraag je antwoord**, mag via knipprogramma een schemaatje zijn of foto van je schets op papier.

Als je klaar bent **sla je dit document op als .pdf** en laad je het (tijdig) op in de uploadzone.

1. Teken het schema van je temperatuursensor en hoe hij verbonden is met je controller. (mag eventueel samen met vraag 2)
   (3 punten)
2. Maak het **basis programma** dat lokaal de functie van thermostaat uitoefent. Simuleer je verwarmingsketel met een LED (aan/uit). Knip en plak het programma hieronder. Instellen van de temperatuur mag met drukknopjes of potentiometer naar keuze.
   (5 punten)
3. Maak een foto van je schak eling en plak deze hieronder



(3 punten)

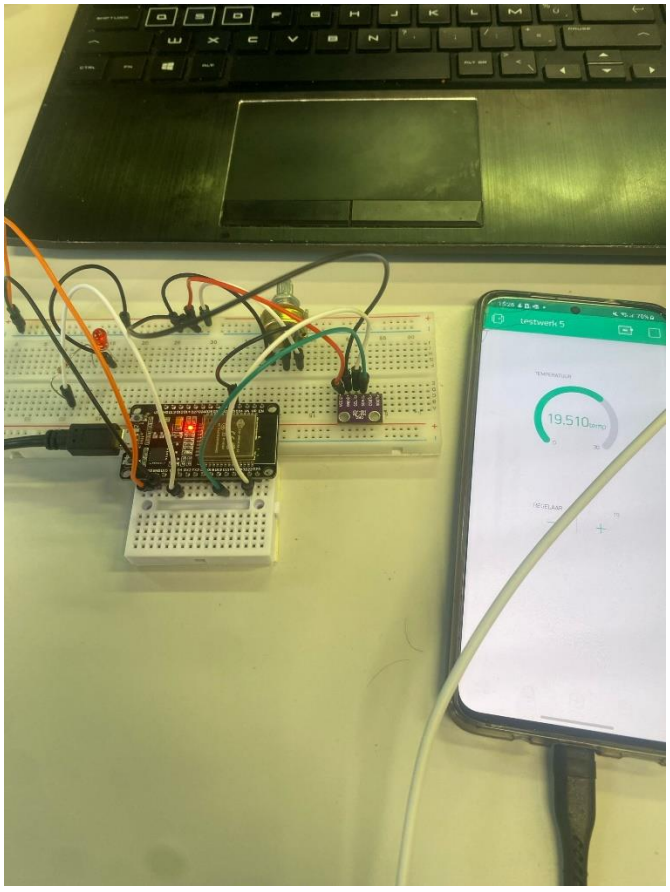
4. Breid je schakeling uit met je afstandsbediening via telefoon. Zorg ervoor dat je
   - De huiskamertemperatuur kan zien op je telefoon

- De status van de verwarmingsketel kan zien op je telefoon
- De gewenste temperatuur kan instellen

5. Publiceer alle files op je github pagina in een nieuwe repository “IoT thermostaat”.
   (4 punten)
6. Knip en plak het **volledige programma** hieronder.
   (5 punten)
7. Toon de werking aan de docent.
   *Maak eventueel later met de webcam van je PC (of een andere telefoon) een filmpje waarin je zelf de werking van je IoT thermostaat uitlegt (aan iemand die geen specialist is in de materie) met demo waarop duidelijk de werking ervan te zien is en plak de link ernaar hieronder. Zet het filmpje bij voorkeur op je GitHub pagina. Noteer hieronder ook **de link naar je Github project**. Als om de een of de andere reden iets in je programma niet naar behoren werkt leg je dit in dit filmpje uit waar je tegenaan gelopen bent.*
   (10 punten)

Zoals altijd is dit een individuele opdracht. Overleg over oplossingsmethodes kan en mag zolang je maar een **eigen originele realisatie** oplevert, geen kopieerwerk toegelaten, niet naar hardware noch software. Deadline voor indienen van je .pdf document is 17:45.

Succes !



```
#define BLYNK_TEMPLATE_ID "user11"
#define BLYNK_TEMPLATE_NAME "user11@server.wyns.it"
#define BLYNK_PRINT Serial

int pin = 4;
const int potPin = 35;
int potVal = 0;
int stepHValue = 0;
int lastChangedValue = 0;  // Houdt de laatst gewijzigde waarde bij
unsigned long lastUpdateTime = 0;  // Houdt de tijd van de laatste update bij

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>

#define BMP_SCK (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS (10)

#define VIRTUAL_PIN_TEMPERATURE V0
#define VIRTUAL_PIN_STEP_H V2

Adafruit_BMP280 bmp;  // I2C

char auth[] = "tvT2lhZZp5DGrswafZQlCbykCpasbL4k";
char ssid[] = "embed";
char pass[] = "weareincontrol";

BLYNK_WRITE(VIRTUAL_PIN_STEP_H) {
  // Get the value from the Step H widget
  stepHValue = param.asInt();
  lastChangedValue = stepHValue;  // Update lastChangedValue
  lastUpdateTime = millis();  // Update lastUpdateTime
}

void updatePotValue() {
  // Lees de waarde van de fysieke potentiometer
  int currentPotVal = map(analogRead(potPin), 0, 1023, 10, 15);

  // Controleer of de waarde is gewijzigd en of het al meer dan 2 seconden
geleden is
  if (currentPotVal != potVal && (millis() - lastUpdateTime) > 2000) {
    potVal = currentPotVal;
    lastChangedValue = potVal;  // Update lastChangedValue
```

```
    lastUpdateTime = millis();  // Update lastUpdateTime
  }
}

void setup() {
  pinMode(pin, OUTPUT);
  digitalWrite(pin, HIGH);
  Serial.begin(115200);

  updatePotValue();  // Initialisatie van potVal

  Blynk.virtualWrite(VIRTUAL_PIN_TEMPERATURE, bmp.readTemperature());

  delay(10);
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, pass);
  int wifi_ctr = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi connected");

  Blynk.begin(auth, ssid, pass, "server.wyns.it", 8081);

  while (!Serial)
    delay(100);
  Serial.println(F("BMP280 test"));
  unsigned status;
  status = bmp.begin(0x76);
  if (!status) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                     "try a different address!"));
    while (1)
      delay(10);
  }

  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                  Adafruit_BMP280::SAMPLING_X2,
                  Adafruit_BMP280::SAMPLING_X16,
                  Adafruit_BMP280::FILTER_X16,
                  Adafruit_BMP280::STANDBY_MS_500);
}

void loop() {
```

```
  updatePotValue();  // Update potVal

  Blynk.run();

  Serial.print("gewenste temperatuur:  ");
  Serial.print(lastChangedValue);
  Serial.println("*C");

  Serial.print(F("Temperature = "));
  Serial.print(bmp.readTemperature());
  Serial.println(" *C");

  if (lastChangedValue >= bmp.readTemperature()) {
    digitalWrite(pin, HIGH);
  } else {
    digitalWrite(pin, LOW);
  }

  Blynk.virtualWrite(VIRTUAL_PIN_TEMPERATURE, bmp.readTemperature());

  delay(2000);
}
```

LINK GITHUB

https://github.com/RobinDeVoecht/TestWeek5RobinDeVoecht