Microcontrolling

Material:

- 1x Arduino Nano
- 1x ESP8266
- 3x LED (Aktor)
- 1x BME680 (Sensor)
- 1x LCD 1602 Modul(Aktor)
- 2x Widerstände (220Ω & 470Ω)

Grobes Konzept:

Messung und Monitoring der Luftqualität und anderen Umgebungsdaten. Ausgabe der Messung wird auf einem LCD-Display ausgegeben. Sobald ein bestimmter Grenzwert über- oder unterschritten werden, leuchten eine oder mehr LEDs auf.

Zeitplan:

1. Doppelstunde: Brainstorming was wir machen wollen.

- Alarmanlage mit Bewegungssensor
- MQTT-Luftqualitätsmesser
- MQTT-Lichtsteuerung

2. Doppelstunde: Alle benötigten Materialien aufgezeichnet und rausgesucht. Begonnen die Dokumentation zu verfassen. Installieren der Arduino IDE.

Verbindung zwischen den beiden Arduino aufgebaut:

Serielle Kommunikation (UART):

1. **Hardware**: Die jeweiligen RX-Pins mit den TX-Pins des anderen Arduino verbunden.
2. **Gemeinsame Masse (GND)**: Verbinde die GND-Pins beider Arduinos.
3. **Programmierung**:
   - Verwenden der `Serial`-Bibliothek in der Arduino-IDE.
   - Beispielcode (Sender):

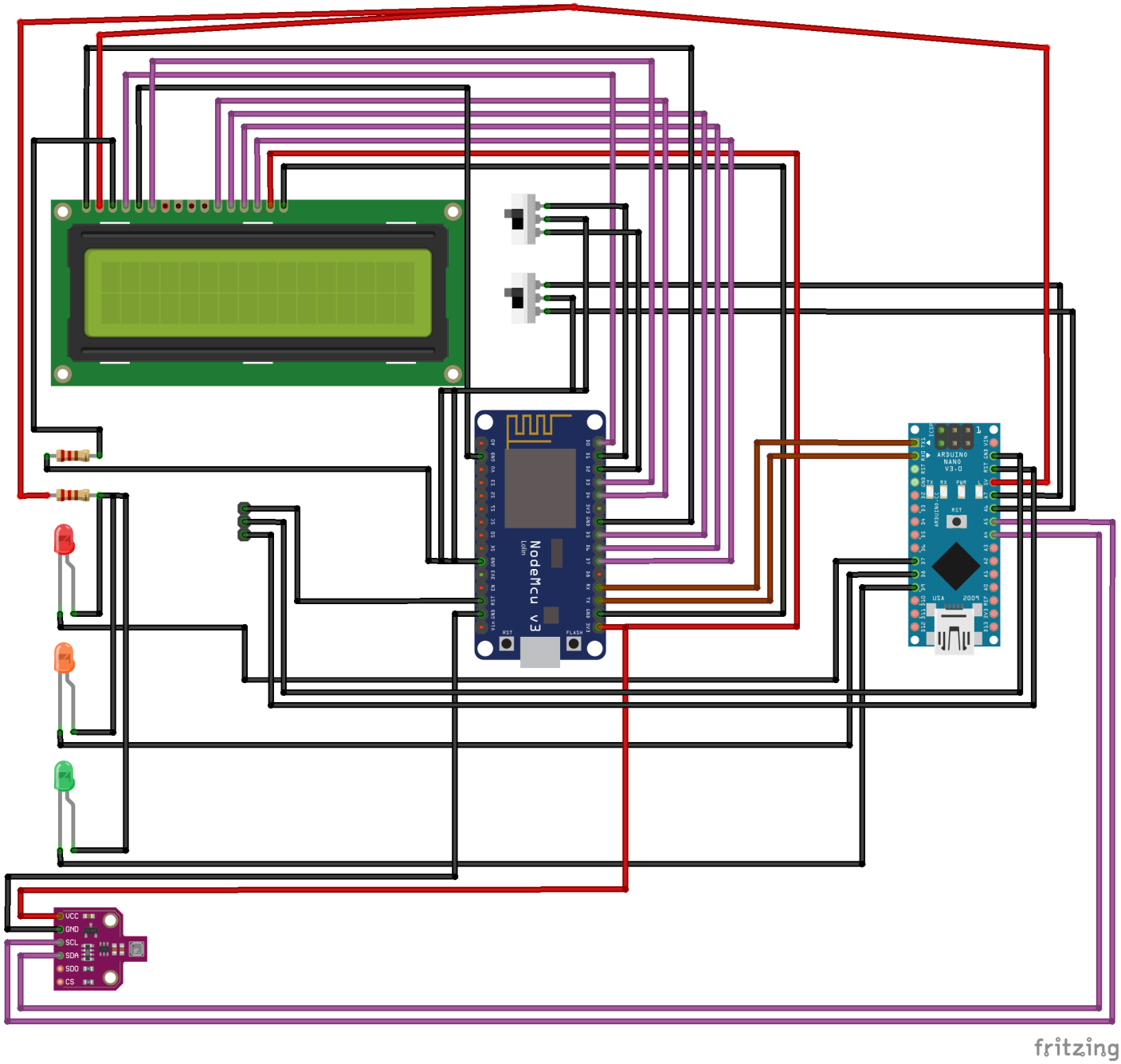
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("Hallo von Arduino 1");
  delay(1000);
}
```

- Beispielcode (Empfänger):

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String message = Serial.readString();
    Serial.println("Empfangene Nachricht: " + message);
  }
}
```



Code für den ESP8266:

```
#include <ArduinoMqttClient.h>
#include <Adafruit_NeoPixel.h>

#include <ESP8266WiFi.h>
#include <Wire.h>
#include <LiquidCrystal.h>

#define wifi_ssid "Bunker"
#define wifi_password "DasIstNichtDasPasswort"

WiFiClient espClient;
MqttClient mqttClient(espClient);
```

```
const char broker[] = "nioy.de";
int        port        = 1883;
const char willTopic[] = "arduino/error";

#define MQTT_Topic_Status "JJ-Test/Wetterstation/ESP8266_Status"
#define MQTT_Topic_LCD "JJ-Lab/Wetterstation/LCD"

const long interval = 2000;
unsigned long previousMillis = 0;

bool debug = false;

unsigned long delayTime = 20;
unsigned long lastHeapCheck = 0;
const unsigned long heapCheckInterval = 60000;

unsigned long lastMessageTime = 0;
const unsigned long timeout = 10000;

int cnt=129600;

const int rs = 16, en = 0, d4 = 2, d5 = 14, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Starte nun...");
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(60);

  Serial.println(" ");
  Serial.println("Moin Moin ....");
  Serial.println("Der Arduino Startet nun     //JJ");

  lcd.setCursor(0, 0);
  lcd.print("Attempting to");
  lcd.setCursor(0, 1);
  lcd.print("connect to WPA");

  Serial.print("Attempting to connect to WPA SSID: ");
  Serial.println(wifi_ssid);
  while (WiFi.begin(wifi_ssid, wifi_password) != WL_CONNECTED) {
    Serial.print(".");
    delay(5000);
  }

  digitalWrite(LED_BUILTIN, LOW);

  Serial.print("Ich habe die IP Addresse : ");
  Serial.print(WiFi.localIP());
  lcd.begin(0, 2);
  lcd.print(WiFi.localIP());
  delay(60);
```

```
  Serial.println("You're connected to the network");
  Serial.println();
  /*
  String willPayload = "oh no!";
  bool willRetain = true;
  int willQos = 1;

  mqttClient.beginWill(willTopic, willPayload.length(), willRetain, willQos);
  mqttClient.print(willPayload);
  mqttClient.endWill();
  */

  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();

  mqttClient.onMessage(onMqttMessage);

  Serial.print("Subscribing to topic: ");
  Serial.println(MQTT_Topic_LCD);

  int subscribeQos = 1;
  mqttClient.subscribe(MQTT_Topic_LCD, subscribeQos);


  // topics can be unsubscribed using:
  // mqttClient.unsubscribe(MQTT_Topic_LED);

  Serial.print("Waiting for messages on topic: ");
  Serial.println(MQTT_Topic_LCD);
  Serial.println();

  mqttClient.beginMessage(MQTT_Topic_Status);
  mqttClient.print("Ready");
  mqttClient.endMessage();

  /*
  if ( hello_start )
  {
  }
  */
  ESP.wdtEnable(WDTO_8S);
}

void loop() {
  Serial.println(cnt);
  if(cnt==20){
```

```
    mqttClient.beginMessage(MQTT_Topic_Status);
    mqttClient.print("Reset ESP8266 equal");
    mqttClient.endMessage();
    delay(120);
  }
  if(cnt==0){
    Serial.println("Reset..");
    mqttClient.beginMessage(MQTT_Topic_Status);
    mqttClient.print("Reset ESP8266 now....");
    mqttClient.endMessage();
    delay(120);
    ESP.restart();
  }
  cnt--;
  delay(50);

  ESP.wdtFeed();
  mqttClient.poll();

  unsigned long currentMillis = millis();

  if (currentMillis - lastHeapCheck >= heapCheckInterval) {
    lastHeapCheck = currentMillis;
    Serial.print("Free Heap: ");
    Serial.println(ESP.getFreeHeap());

    mqttClient.beginMessage(MQTT_Topic_Status);
    mqttClient.print(ESP.getFreeHeap());
    mqttClient.endMessage();
  }

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    bool retained = false;
    int qos = 1;
    bool dup = false;

    mqttClient.beginMessage(MQTT_Topic_Status);
    mqttClient.print("Online");
    mqttClient.endMessage();
  }
  mqttClient.poll();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    bool retained = false;
    int qos = 1;
    bool dup = false;

    mqttClient.beginMessage(MQTT_Topic_Status);
    mqttClient.print("Online");
    mqttClient.endMessage();
  }
  mqttClient.poll();
```

```
}

void onMqttMessage(int messageSize) {
  if (debug) {
    Serial.print("Received a message with topic '");
    Serial.print(mqttClient.messageTopic());
    Serial.print("', duplicate = ");
    Serial.print(mqttClient.messageDup() ? "true" : "false");
    Serial.print(", QoS = ");
    Serial.print(mqttClient.messageQoS());
    Serial.print(", retained = ");
    Serial.print(mqttClient.messageRetain() ? "true" : "false");
    Serial.print("', length ");
    Serial.print(messageSize);
    Serial.println(" bytes:");
  }

  if (strcmp(mqttClient.messageTopic().c_str(), MQTT_Topic_LCD) == 0)
  {
    lastMessageTime = millis();
    Serial.println("Neue Nachricht empfangen:");
    Serial.println("Topic: " + mqttClient.messageTopic());
    Serial.print("Payload: ");

    // Konvertiere das Payload-Array in einen String
    String payloadString;
    for (int i = 0; i < messageSize; i++) {
      payloadString += (char)mqttClient.read();
    }
    Serial.println(payloadString);

    int maxLineLength = 16;
    String line1 = payloadString.substring(0, maxLineLength);
    String line2 = payloadString.length() > maxLineLength ? payloadString.substring(maxLineLength) : "";
    lcd.setCursor(0, 0);
    lcd.print(line1);

    lcd.setCursor(0, 1);
    lcd.print(line2);
  }
}
```

Revision #20
Created 28 November 2023 20:38:04 by Julian
Updated 18 December 2023 08:02:50 by Guest