Dokumentation: Projekt Wetterstation

1. Einleitung

Im Rahmen unseres Projekts haben wir eine eigene **Wetterstation** entwickelt. Ziel des Projekts war es, Wetter- und Umweltdaten wie **Temperatur**, **Luftfeuchtigkeit**, **Luftdruck** und **Luftqualität** zu messen, diese automatisch zu speichern und weiterzuverarbeiten. Zusätzlich wurde ein Gehäuse entworfen, das sich für den Außeneinsatz eignet und aufgehängt werden kann.

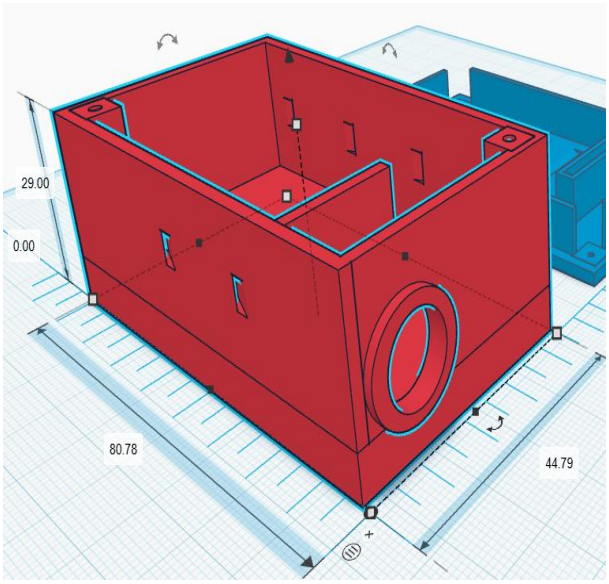
---

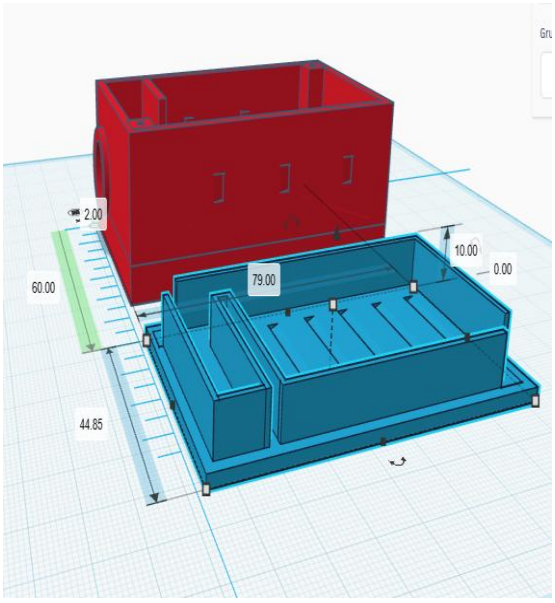
2. Planung und Entwurf

Zu Beginn wurde das Gehäuse der Wetterstation mit dem Online-Tool **Tinkercad** als 3D-Modell entworfen.

Dabei wurde darauf geachtet, dass:

- der **Raspberry Pi / Joy-Pi** und der **BME680-Sensor** genügend Platz haben,
- Öffnungen für Kabel, Sensoren und Belüftung vorhanden sind,
- und eine **Aufhängung** im Design integriert ist, um die Wetterstation befestigen zu können.





---

3. 3D-Druck und Nachbearbeitung

Nach dem Entwurf wurde das Modell mit einem **3D-Drucker** aus **schwarzem PLA-Filament** gedruckt.

Nach dem Druck wurden mehrere Nachbearbeitungsschritte durchgeführt:

- Die **Ecken wurden geschliffen**, um scharfe Kanten zu entfernen.
- Die **Löcher wurden nachgearbeitet**, damit Schrauben und Kabel besser passen.

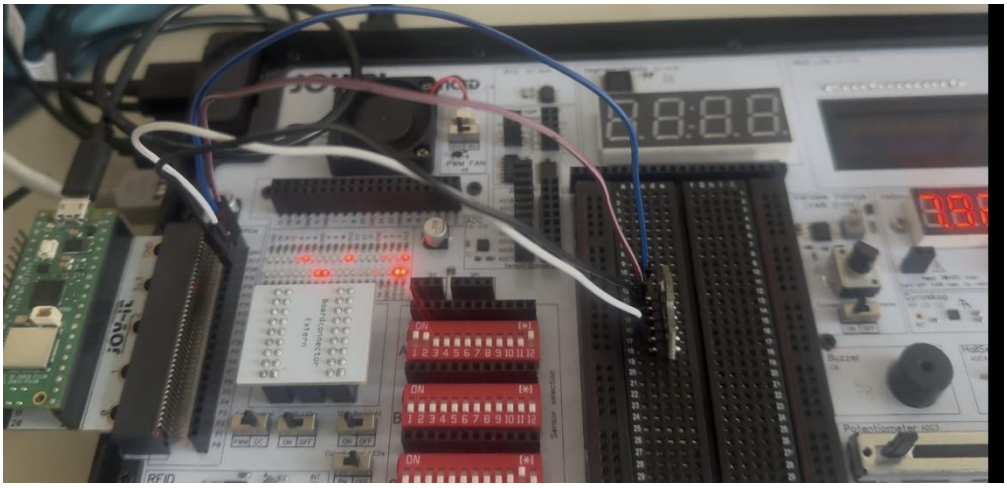
So entstand ein stabiles und funktionales Gehäuse für die Wetterstation.

4. Elektronik und Sensorik

Im Inneren der Wetterstation befindet sich ein **Raspberry Pi / Joy-Pi**, an den der **BME680-Umweltsensor** angeschlossen ist.
Der BME680 misst folgende Werte:

- **Temperatur**
- **Luftfeuchtigkeit**
- **Luftdruck**
- **Luftqualität (Gaswiderstand / VOC)**

Die Verbindung zwischen Mikrocontroller und Sensor erfolgt über den **I²C-Bus**.



5. Sensorprogramm (Messstation)

Das erste Programm läuft auf dem Joy-Pi bzw. Mikrocontroller, an dem der **BME680-Sensor** angeschlossen ist.
Das Programm:

- initialisiert den **I²C-Bus**,
- prüft, ob der Sensor korrekt erkannt wird,
- liest alle **2 Sekunden** die Messwerte aus,
- und gibt diese zur Weiterverarbeitung aus.

Dieses Programm stellt die eigentliche **Messstation** dar und sammelt zuverlässig alle Umweltdaten.

6. Datenübertragung über WLAN und MQTT

Die gemessenen Sensordaten werden über das **Schul-WLAN** an einen **MQTT-Broker** gesendet.
MQTT ist ein leichtes Kommunikationsprotokoll, das besonders für **IoT-Projekte** geeignet ist.
Die Daten werden als Nachrichten an ein bestimmtes **MQTT-Topic** übertragen und stehen dort für weitere Programme zur Verfügung.

Der Python-Code für den Raspberry Pi Pico W steuert die Wetterstation mit dem **BME680-Sensor**, liest die Messwerte aus und überträgt sie **automatisch an einen MQTT-Broker** sowie **per E-Mail**.

1. Aufbau & Verbindung

- **Sensor auf Breadboard** gesteckt und mit dem Pico W über **I2C** (SDA/GPIO4, SCL/GPIO5) verbunden
- WLAN-Verbindung über dd-wrt hergestellt
- MQTT-Verbindung zu einem Broker auf einem Raspberry Pi (10.118.28.14) aufgebaut
- E-Mail-Funktion über **Gmail SMTP** aktiviert

2. Funktionsweise

1. **WLAN-Verbindung:**
   - Der Pico W verbindet sich mit dem definierten WLAN und überprüft die Verbindung
   - IP-Adresse wird nach erfolgreichem Verbinden ausgegeben
2. **Sensorinitialisierung:**
   - Der BME680 wird über I2C initialisiert
   - Bereit für das Auslesen von:
     - Temperatur (°C)
     - Luftfeuchtigkeit (%)
     - Luftdruck (hPa)
     - Gaswiderstand (Ω)
3. **Messwerte auslesen:**
   - Alle 5 Sekunden werden die aktuellen Werte vom Sensor gelesen
4. **MQTT-Datenübertragung:**
   - Messwerte werden formatiert und an das MQTT-Topic sensor/bme680 gesendet
   - So können andere Geräte oder ein Dashboard die Daten empfangen
5. **E-Mail-Versand:**
   - Alle 30 Minuten wird eine E-Mail mit den aktuellen Messwerten verschickt
   - Die E-Mail enthält Temperatur, Luftfeuchtigkeit, Luftdruck und Gaswert
   - Versand erfolgt über Gmail SMTP mit App-Passwort
6. **Endlosschleife:**
   - Der Code läuft permanent
   - Prüft regelmäßig, ob eine E-Mail versendet werden muss
   - Sendet kontinuierlich die Messwerte über MQTT

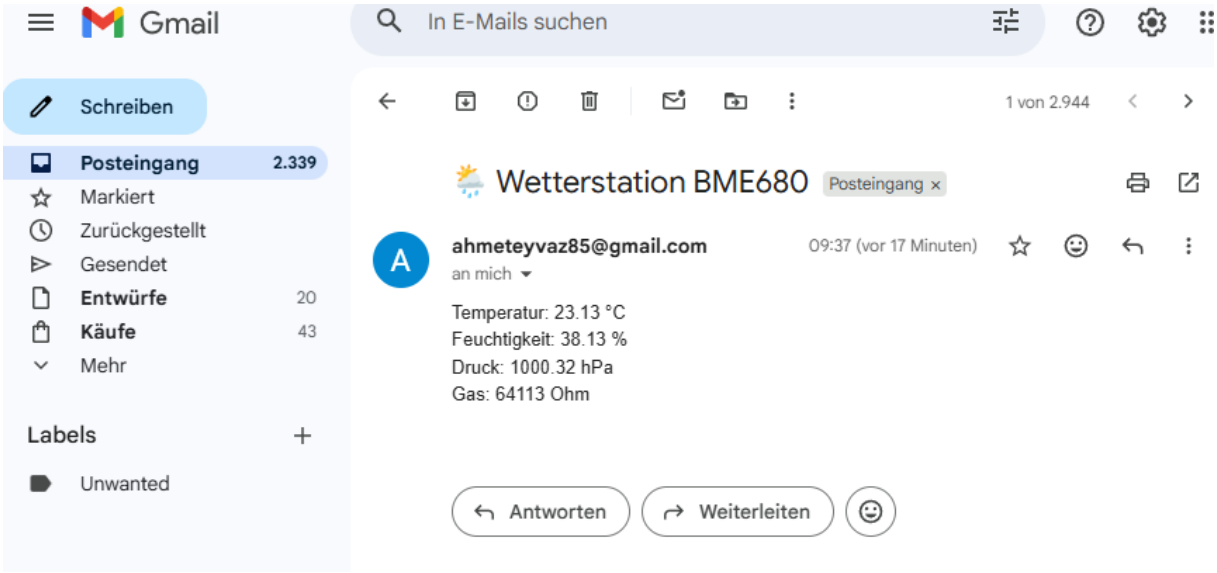
4. Ergebnis

- Der Code misst zuverlässig **Temperatur, Luftfeuchtigkeit, Luftdruck und Gaswerte**
- Daten werden **MQTT-kompatibel an den Broker gesendet**
- **Periodischer E-Mail-Versand** informiert über aktuelle Wetterwerte
- Vollständig automatisiertes System für Schul- oder Heimprojekte

```python
import time
import network
from machine import Pin, I2C
from bme680 import BME680_I2C
from simple import MQTTClient
from umail import SMTP

# ================================
# WLAN
# ================================
WIFI_SSID = "dd-wrt"
WIFI_PASSWORD = "54tzck23"

# ================================
# MQTT
# ================================
MQTT_BROKER = "10.118.28.14"   # IP vom Raspberry Pi
MQTT_PORT = 1883
MQTT_CLIENT_ID = "pico_w_bme680"
MQTT_TOPIC = b"sensor/bme680"

# ================================
# E-MAIL
# ================================
EMAIL_ENABLED = True
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 465
SMTP_SENDER_EMAIL = "ahmeteyvaz85@gmail.com"
SMTP_APP_PASSWORD = "defq bytz ghew gche"
EMAIL_RECIPIENT = "ahmeteyvaz85@gmail.com"
EMAIL_SUBJECT = "🌳 Wetterstation BME680"

EMAIL_INTERVAL = 1800  # 30 Minuten
last_email_time = 0

# ================================
# WLAN VERBINDEN
# ================================
def connect_wlan():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print("📡 WLAN verbinden...")
        wlan.connect(WIFI_SSID, WIFI_PASSWORD)
        while not wlan.isconnected():
            time.sleep(1)
    print("☑ WLAN verbunden:", wlan.ifconfig())
```

```
48
49  # ==============================
50  # E-MAIL SENDEN
51  # ==============================
52  def send_email(text):
53      global last_email_time
54      try:
55          smtp = SMTP(
56              SMTP_SERVER,
57              SMTP_PORT,
58              username=SMTP_SENDER_EMAIL,
59              password=SMTP_APP_PASSWORD,
60              ssl=True
61          )
62
63          msg = (
64              "Subject: {}\r\n"
65              "To: {}\r\n"
66              "From: {}\r\n\r\n{}"
67          ).format(EMAIL_SUBJECT, EMAIL_RECIPIENT, SMTP_SENDER_EMAIL, text)
68
69          smtp.to(EMAIL_RECIPIENT, mail_from=SMTP_SENDER_EMAIL)
70          smtp.send(msg.encode("utf-8"))
71          smtp.quit()
72
73          last_email_time = time.time()
74          print("✉ E-Mail gesendet")
75      except Exception as e:
76          print("❌ E-Mail Fehler:", e)
77
78  # ==============================
79  # SENSOR
80  # ==============================
81  i2c = I2C(0, sda=Pin(4), scl=Pin(5), freq=100000)
82  sensor = BME680_I2C(i2c)
83
84  # ==============================
85  # START
86  # ==============================
87  connect_wlan()
88
89  mqtt = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, port=MQTT_PORT)
90  mqtt.connect()
91  print("☑ MQTT verbunden")
92
93  print("\n▶ Starte Wetterstation...\n")
```

```
94
95  # ==============================
96  # HAUPTSCHLEIFE
97  # ==============================
98  while True:
99      temperature = sensor.temperature
100     humidity = sensor.humidity
101     pressure = sensor.pressure
102     gas = sensor.gas
103
104     message = (
105         f"Temperatur: {temperature:.2f} °C\n"
106         f"Feuchtigkeit: {humidity:.2f} %\n"
107         f"Druck: {pressure:.2f} hPa\n"
108         f"Gas: {gas} Ohm"
109     )
110
111     # MQTT senden
112     mqtt.publish(MQTT_TOPIC, message)
113     print("📤 MQTT gesendet")
114
115     # E-Mail senden (zeitgesteuert)
116     now = time.time()
117     if EMAIL_ENABLED and (now - last_email_time) >= EMAIL_INTERVAL:
118         send_email(message)
119
120     time.sleep(5)
121
```



Die Messdaten kommen auch an
Für den Passwort habe ich ein App Passwort erstellt über

https://myaccount.google.com/ > Sicherheit
Dann bei der Suchleiste App Passwörter und dann auf erstellen und dann bekommt man ein Passwort

Dieser Python-Code läuft auf dem Raspberry Pi 400 und übernimmt die Funktion eines MQTT-Servers, der eingehende Daten empfängt und in einer MariaDB-Datenbank speichert.

Funktionsweise:

1. **Datenbank-Verbindung:**
   - **Verbindung zur MariaDB (mqtt_data) wird hergestellt**
   - **Benutzername, Passwort und Host werden über Variablen definiert**
2. **MQTT-Setup:**
   - **MQTT-Client wird erstellt**
   - **Definierte Topics:**
     - **test/topic/in → zum Empfangen von Nachrichten**
     - **test/topic/out → für Testnachrichten oder Rückmeldungen**
3. **MQTT-Callbacks:**
   - **on_connect: Meldet erfolgreiche Verbindung und abonniert das Topic**
   - **on_message:**
     - **Empfängt eingehende Nachrichten**
     - **Speichert die Nachricht mit Topic in der MariaDB**
4. **Testnachricht:**
   - **Beim Start sendet der Pi eine Testnachricht an test/topic/out**
   - **So kann überprüft werden, ob die Kommunikation funktioniert**
5. **Endlosschleife:**
   - **Das Programm bleibt aktiv, empfängt kontinuierlich neue Nachrichten**
   - **Daten werden automatisch in der Datenbank protokolliert**

Ergebnis:

- **Alle vom Pico W gesendeten MQTT-Daten werden zuverlässig empfangen**
- **Nachrichten werden in MariaDB gespeichert**
- **Ermöglicht langfristige Speicherung und spätere Auswertung der Wetterdaten**

```python
import paho.mqtt.client as mqtt
import mariadb
import sys

# --- MariaDB-Verbindungsdaten anpassen ---
DB_HOST = "localhost"
DB_USER = "root"
DB_PASS = "54tzck23"
DB_NAME = "mqtt_data"

# --- MQTT-Broker Daten ---
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC_SUB = "test/topic/in"
MQTT_TOPIC_PUB = "test/topic/out"

# --- Verbindung zur MariaDB herstellen ---
def connect_db():
    try:
        conn = mariadb.connect(
            user=DB_USER,
            password=DB_PASS,
            host=DB_HOST,
            database=DB_NAME
        )
        return conn
    except mariadb.Error as e:
        print(f"Fehler bei DB-Verbindung: {e}")
        sys.exit(1)

# --- MQTT Callbacks ---
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("☑ MQTT Verbindung erfolgreich")
        client.subscribe(MQTT_TOPIC_SUB)
        print(f"Subscribed auf Topic: {MQTT_TOPIC_SUB}")
    else:
        print(f"❌ Verbindung fehlgeschlagen mit Code {rc}")

def on_message(client, userdata, msg):
    print(f"Nachricht empfangen auf {msg.topic}: {msg.payload.decode()}")
    # In DB speichern
    try:
        conn = userdata['db_conn']
        cursor = conn.cursor()
        sql = "INSERT INTO mqtt_messages (topic, message) VALUES (?, ?)"
        cursor.execute(sql, (msg.topic, msg.payload.decode()))
        conn.commit()
        print("☑ Nachricht in MariaDB gespeichert.")
        print("----------------------------------------")
    except mariadb.Error as e:
        print(f"❌ DB Fehler: {e}")
```

```
53
54  # --- Hauptprogramm ---
55  def main():
56      # DB-Verbindung aufbauen
57      db_conn = connect_db()
58
59      # MQTT Client einrichten
60      client = mqtt.Client(userdata={'db_conn': db_conn})
61      client.on_connect = on_connect
62      client.on_message = on_message
63      client.connect(MQTT_BROKER, MQTT_PORT, 60)
64
65      # Nachrichten veröffentlichen (z.B. Testnachricht)
66      client.loop_start()
67      client.publish(MQTT_TOPIC_PUB, "Hallo von Raspberry Pi!")
68      print(f"📤 Nachricht gesendet auf {MQTT_TOPIC_PUB}")
69
70      # Endlosschleife, um Nachrichten zu empfangen
71      try:
72          while True:
73              pass
74      except KeyboardInterrupt:
75          print("Programm beendet")
76      finally:
77          client.loop_stop()
78          db_conn.close()
79
80  if __name__ == "__main__":
81      main()
```

Mit der Eingabe bei Terminal
mosquitto_sub -h 10.118.28.14 -t "sensor/bme680" -v
-h → IP-Adresse des MQTT-Brokers (Raspberry Pi)

-t → Topic, das du abonnieren willst (sensor/bme680)

-v → verbose, zeigt Topic + Nachricht
Wir die Nachricht angezeigt die angekommen ist bzw die Messdaten werden angezeigt

10. Fazit

In diesem Projekt wurde erfolgreich eine eigene **Wetterstation** entwickelt, die **3D-Design**, **3D-Druck**, **Elektronik** und **Programmierung** miteinander verbindet. Besonders wichtig war die automatische Übertragung und Speicherung der Messdaten. Das Projekt zeigt, wie moderne IoT-Technologien eingesetzt werden können und bietet viele Möglichkeiten für zukünftige Erweiterungen.