**IOT Projekt**

**Opis:**  
Projekt IoT ma na celu monitorowanie temperatury za pomocą modułu ESP8266 wyposażonego w czujnik DS18B20. Odczyty temperatury są wyświetlane w terminalu Arduino oraz na wyświetlaczu LCD. Dane temperatury są wysyłane do zdalnego brokera MQTT z zastosowaniem autoryzacji oraz szyfrowania. Aplikacja Python subscriber subskrybuje te dane, zapisuje je do bazy danych InfluxDB, a wizualizację danych zapewnia Grafana. Dodatkowo, projekt zawiera funkcje zamiany mowy na tekst (speech-to-text) oraz detekcji obiektów na obrazie (video-to-text) przy użyciu modeli Whisper i YOLO, które również wysyłają dane przez MQTT do brokera. Cały system jest uruchamiany za pomocą Docker, co umożliwia łatwe zarządzanie kontenerami.  
  
**Autorzy:**

Wojciech Biziuk, Mateusz Karolewski

**Wymagania sprzętowe:**

Sprzęt: komputer x2, moduł ESP8266, Czujnik temperatury DS18B20, Wyświetlacz LCD

Oprogramowanie: Arduino IDE, Docker, Python 3.x, InfluxDB, Grafana

**Opis Komponentów:**

ESP8266 + Subscriber  
  
MQTT configuration

MQTT\_BROKER = 'broker.emqx.io' # Replace with your broker's address MQTT\_PORT = 1883 MQTT\_TOPIC = 'emqx/esp8266\_laby'

InfluxDB configuration

token = "JvNlsVx9-TNoat\_Op6d0GeiQZezk7DnAWhYQj7AdB9sSv65ypZtQYd3qDxbT3aytzTEfxWFLPxjVz71bn4uSNw==" org = "IOT" url = "http://localhost:8086/" bucket = "new" # Replace with the name of your InfluxDB bucket write\_client = InfluxDBClient(url=url, token=token, org=org) write\_api = write\_client.write\_api(write\_options=SYNCHRONOUS)

Define the MQTT client callbacks

def on\_connect(client, userdata, flags, rc): print("Connected with result code " + str(rc)) client.subscribe(MQTT\_TOPIC) def on\_message(client, userdata, msg): try: temperature = float(msg.payload.decode()) print(f"Received temperature: {temperature}") # Create a data point and write it to InfluxDB point = ( Point("measurement1") .tag("tagname1", "tagvalue1") .field("field1", temperature) ) write\_api.write(bucket=bucket, org=org, record=point) print("Data written to InfluxDB.") except ValueError as e: print("Failed to decode or write data:", e)

Initialize and run the MQTT client

client\_mqtt = mqtt.Client() client\_mqtt.on\_connect = on\_connect client\_mqtt.on\_message = on\_message client\_mqtt.connect(MQTT\_BROKER, MQTT\_PORT, 60) client\_mqtt.loop\_forever()

Speech to text:

logging.basicConfig(level=logging.DEBUG) print("Starting application...") print("MQTT polaczenie")

MQTT configuration

MQTTBROKER = 'broker.emqx.io' MQTTPORT = 1883 MQTTTOPIC = 'emqx/esp8266laby' def on\_connect(client, userdata, flags, rc): print("Connected with result code " + str(rc)) client.subscribe(MQTT\_TOPIC)

Inicjalizacja modelu Whisper

print("Loading Whisper model...") whisper\_asr = pipeline("automatic-speech-recognition", model="openai/whisper-small") def record\_audio(duration=5, samplerate=16000): """ Rejestracja dźwięku z mikrofonu. """ print(f"Recording audio for {duration} seconds...") audio = sd.rec(int(samplerate \* duration), samplerate=samplerate, channels=1, dtype=np.float32) sd.wait() # Oczekiwanie na zakończenie nagrywania print("Recording complete.") return audio.flatten() def transcribe\_audio(audio): """ Zamiana dźwięku na tekst przy użyciu modelu Whisper. """ print("Transcribing audio...") result = whisper\_asr(audio) # Usunięto sampling\_rate return result["text"] if \_\_name == "\_\_main": try: duration = 5 # Czas nagrywania w sekundach samplerate = 16000 # Częstotliwość próbkowania audio\_data = record\_audio(duration=duration, samplerate=samplerate) transcription = transcribe\_audio(audio\_data) client = mqtt.Client() client.connect(MQTT\_BROKER,1883,60) client.publish(MQTT\_TOPIC, transcription); client.disconnect(); print(f"Transcribed text: {transcription}") except Exception as e: print(f"An error occurred: {e}") finally: print("Press Ctrl+C to exit...") while True: time.sleep(1) client\_mqtt = mqtt.Client() client\_mqtt.on\_connect = on\_connect  
  
  
  
Video to text: