Python-Based Modem Communication Library Project Final Report

Introduction

General Purpose

This project aims to develop the ability to access embedded Linux systems, work with Raspberry Pi, and use basic Linux commands. It targets writing legible and maintainable code using the Object-Oriented Programming (OOP) approach in Python, version control with Git and GitHub, and effective project planning and management. Additionally, it involves creating documentation and understanding the fundamental operations of cellular network modems. At the heart of the project is the development of a Python-based modem communication library and creating tools that will allow data transmission and reception over fundamental network protocols such as HTTP, MQTT, PPP, QMI, and ECM. Accordingly, the project is designed to establish a competent infrastructure and the effective use of modern communication technologies.

Technical Details

Technologies and Methods Used

- **Python Programming Language:** Used as the development language in the project.
- **AT Commands**: Utilized for communication with the modem.
- **Serial Library**: Employed for serial communication between devices.
- **HTTP (Hypertext Transfer Protocol):** The basic web protocol used for data exchange.
- **MQTT (Message Queuing Telemetry Transport):** A lightweight and simple messaging protocol.
- **PPP (Point-to-Point Protocol):** A protocol used for establishing a point-to-point connection over the internet.
- **QMI (Qualcomm MSM Interface):** Used in mobile devices and data cards for 3G and 4G LTE connections.
- **ECM (Ethernet Control Model):** A protocol for sending control signals over Ethernet.

System Architecture

- ModemCommunicator Class: The class used for communicating with the modem.
- Examples of communication over HTTP and MQTT protocols.
- Establishing internet connection on Raspberry Pi using various network protocols.

Development Process

Development Stages

Stage 1: Development of the basic library for communication with the modem.

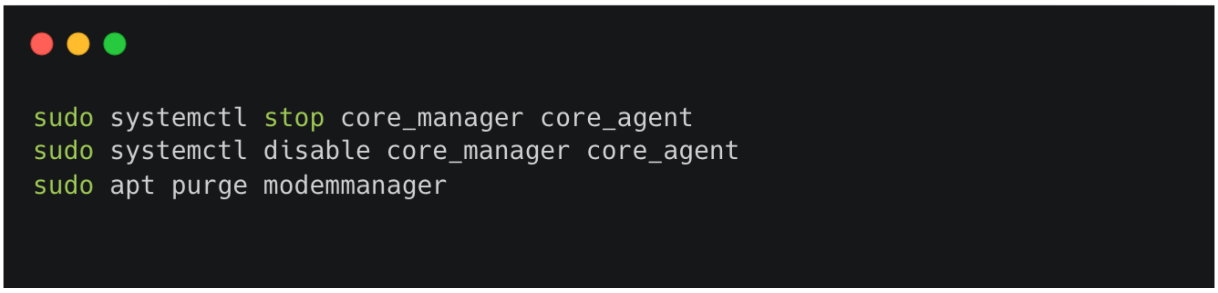
Stage 2: Addition of functions to the library for sending and receiving data using the HTTP protocol and the development of an example application that facilitates communication through these functions.

Stage 3: Incorporation of functions into the library for publishing and subscribing to messages using the MQTT protocol and the development of an example application that communicates through these functions.

Stage 4: Establishing internet connection using PPP, QMI, and ECM protocols, conducting speed tests, and comparing the protocols.

Challenges and Solutions

- Encountered '+CME ERROR 703' error messages during the HTTP GET operation. This problem was resolved by waiting longer for responses.
- After completing the PPP protocol setup for the modem, running the "sudo pon" command resulted in a 'Modem hangup Connection terminated.' error. The problem was corrected with the following commands:



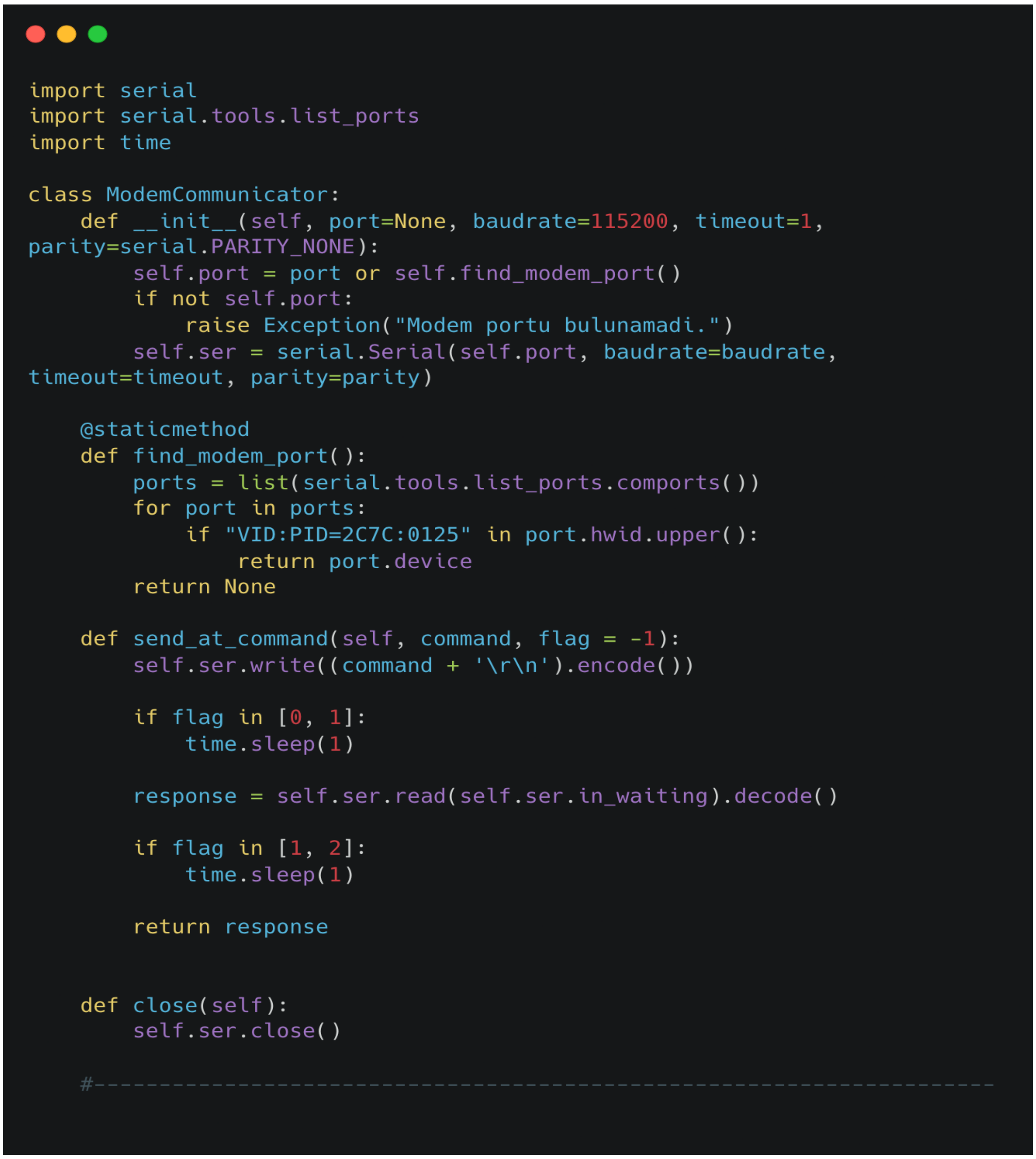
```
sudo systemctl stop core_manager core_agent
sudo systemctl disable core_manager core_agent
sudo apt purge modemmanager
```

Codes, Results and Learnings

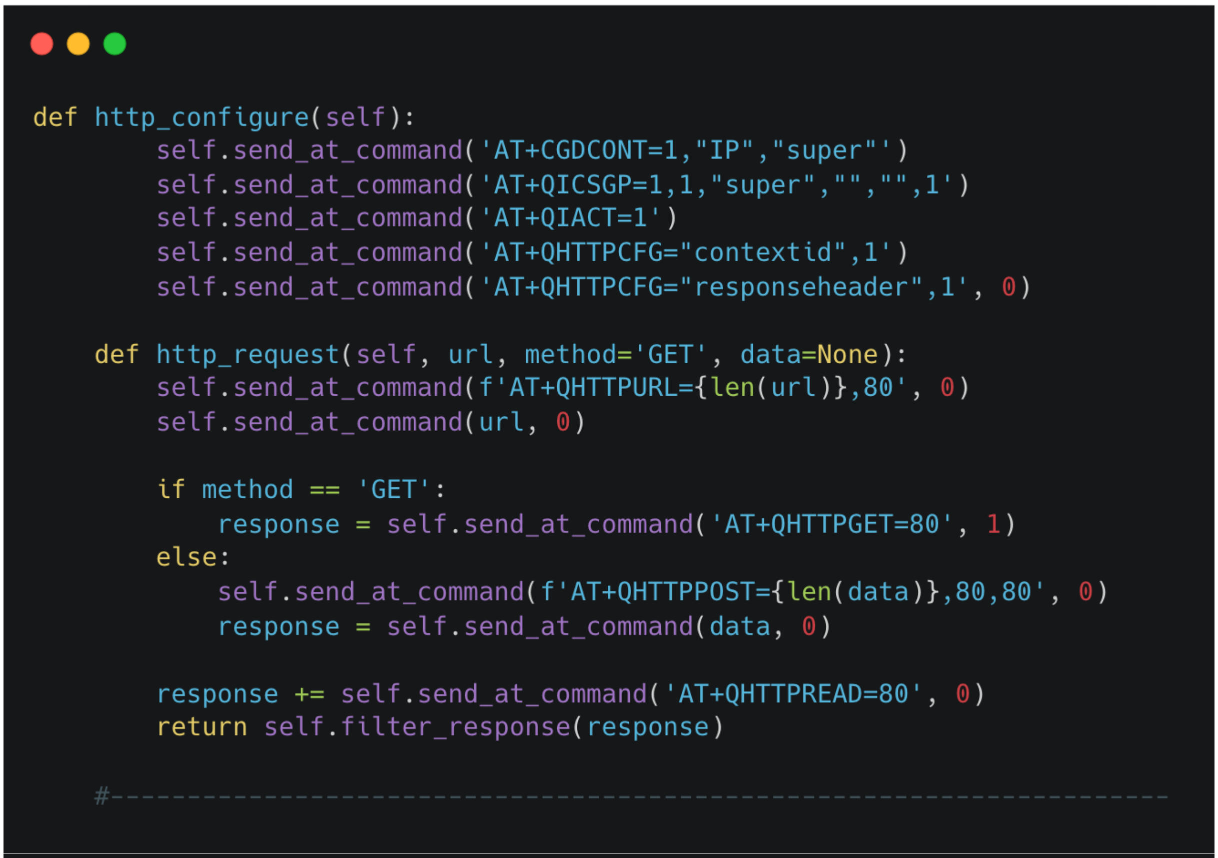
Achievements and Impact of the Project

- Successfully developed a class that communicates with the modem and includes various HTTP and MQTT functions.
- Communication was established over HTTP and MQTT protocols.
- Successfully connected to the internet using various network protocols on Raspberry Pi.

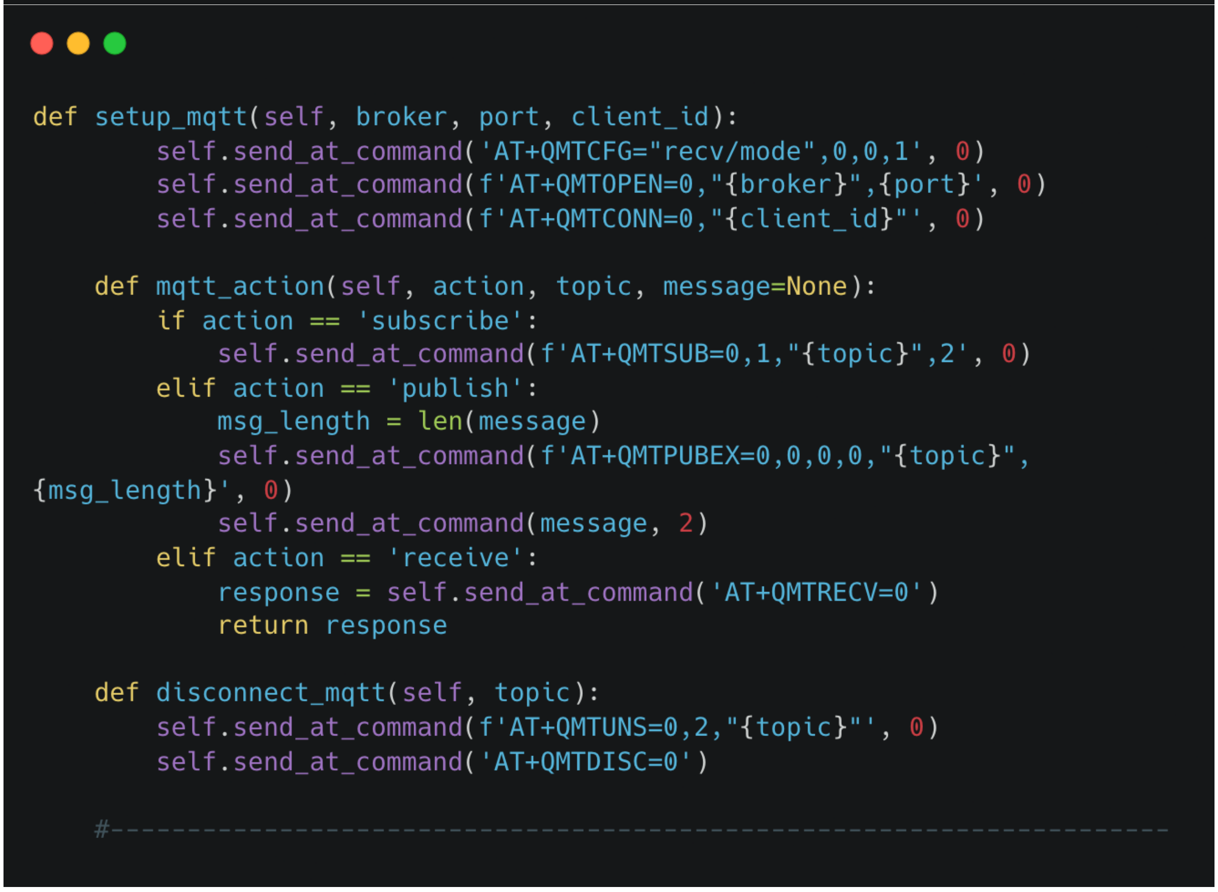
ModemComminator:



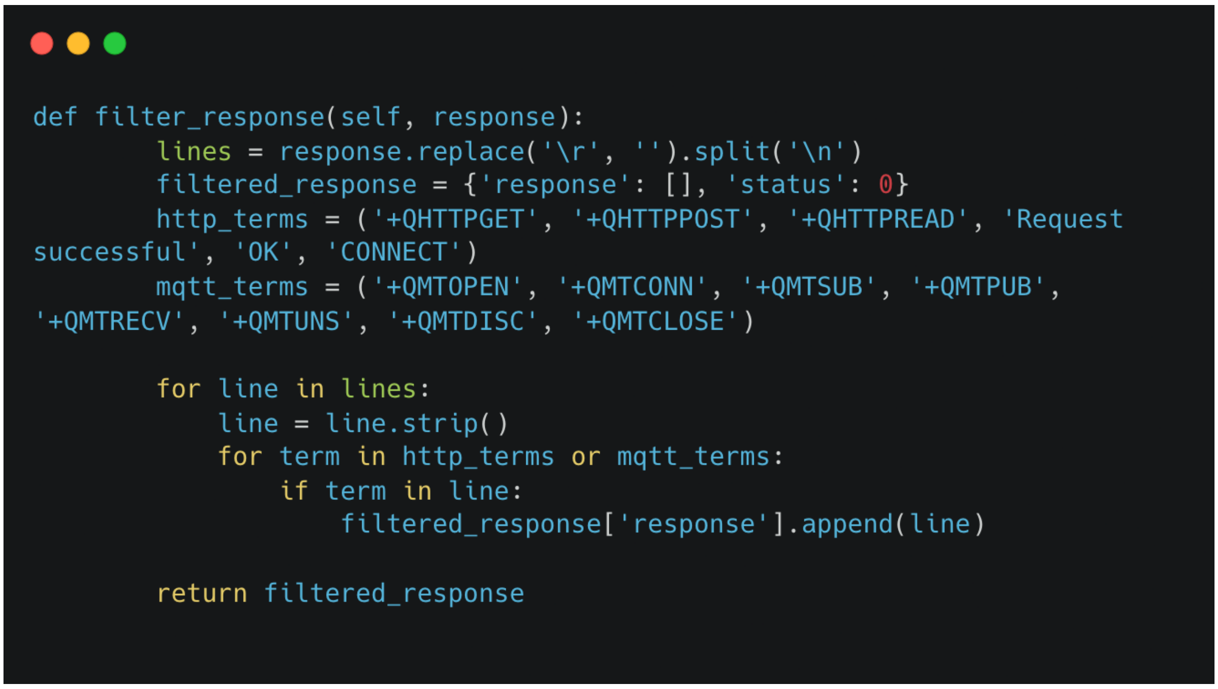
```python
import serial
import serial.tools.list_ports
import time

class ModemCommunicator:
    def __init__(self, port=None, baudrate=115200, timeout=1,
parity=serial.PARITY_NONE):
        self.port = port or self.find_modem_port()
        if not self.port:
            raise Exception("Modem portu bulunamadi.")
        self.ser = serial.Serial(self.port, baudrate=baudrate,
timeout=timeout, parity=parity)

    @staticmethod
    def find_modem_port():
        ports = list(serial.tools.list_ports.comports())
        for port in ports:
            if "VID:PID=2C7C:0125" in port.hwid.upper():
                return port.device
        return None

    def send_at_command(self, command, flag = -1):
        self.ser.write((command + '\r\n').encode())

        if flag in [0, 1]:
            time.sleep(1)

        response = self.ser.read(self.ser.in_waiting).decode()

        if flag in [1, 2]:
            time.sleep(1)

        return response


    def close(self):
        self.ser.close()

    #---------------------------------------------------------------------
```



```python
def http_configure(self):
        self.send_at_command('AT+CGDCONT=1,"IP","super"')
        self.send_at_command('AT+QICSGP=1,1,"super","","",1')
        self.send_at_command('AT+QIACT=1')
        self.send_at_command('AT+QHTTPCFG="contextid",1')
        self.send_at_command('AT+QHTTPCFG="responseheader",1', 0)

    def http_request(self, url, method='GET', data=None):
        self.send_at_command(f'AT+QHTTPURL={len(url)},80', 0)
        self.send_at_command(url, 0)

        if method == 'GET':
            response = self.send_at_command('AT+QHTTPGET=80', 1)
        else:
            self.send_at_command(f'AT+QHTTPPOST={len(data)},80,80', 0)
            response = self.send_at_command(data, 0)

        response += self.send_at_command('AT+QHTTPREAD=80', 0)
        return self.filter_response(response)

    #---------------------------------------------------------------------
```

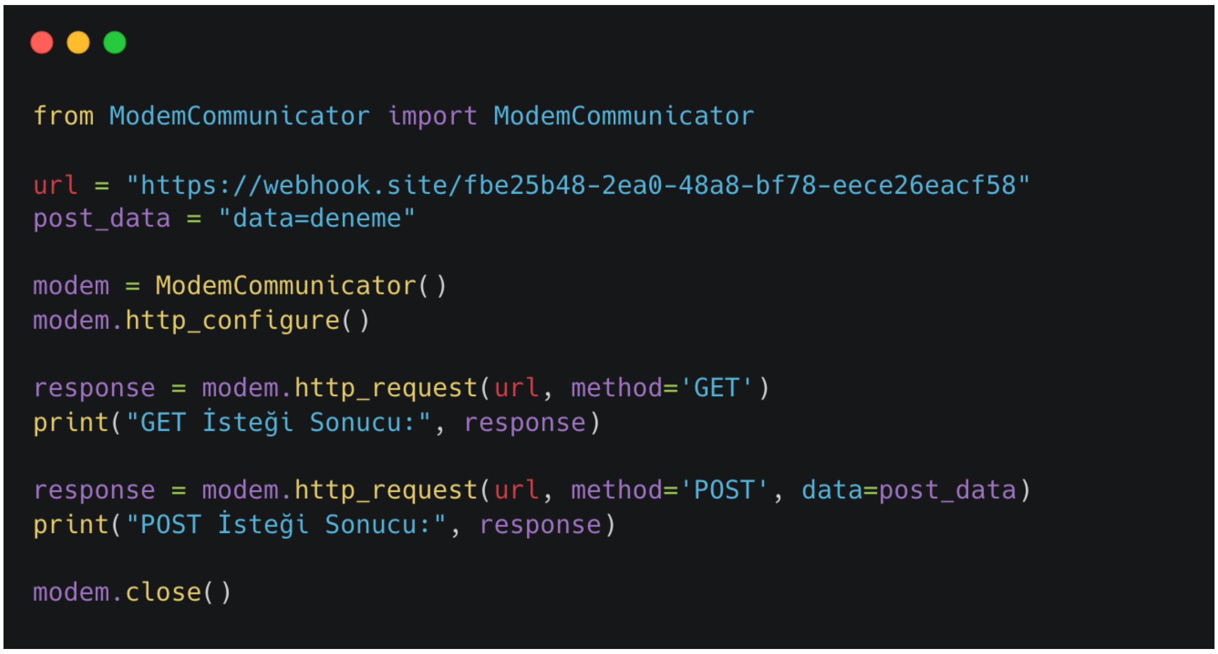


```python
def setup_mqtt(self, broker, port, client_id):
        self.send_at_command('AT+QMTCFG="recv/mode",0,0,1', 0)
        self.send_at_command(f'AT+QMTOPEN=0,"{broker}",{port}', 0)
        self.send_at_command(f'AT+QMTCONN=0,"{client_id}"', 0)

    def mqtt_action(self, action, topic, message=None):
        if action == 'subscribe':
            self.send_at_command(f'AT+QMTSUB=0,1,"{topic}",2', 0)
        elif action == 'publish':
            msg_length = len(message)
            self.send_at_command(f'AT+QMTPUBEX=0,0,0,0,"{topic}",
{msg_length}', 0)
            self.send_at_command(message, 2)
        elif action == 'receive':
            response = self.send_at_command('AT+QMTRECV=0')
            return response

    def disconnect_mqtt(self, topic):
        self.send_at_command(f'AT+QMTUNS=0,2,"{topic}"', 0)
        self.send_at_command('AT+QMTDISC=0')

    #---------------------------------------------------------------------
```

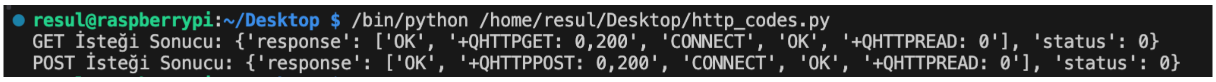


```
def filter_response(self, response):
        lines = response.replace('\r', '').split('\n')
        filtered_response = {'response': [], 'status': 0}
        http_terms = ('+QHTTPGET', '+QHTTPPOST', '+QHTTPREAD', 'Request
successful', 'OK', 'CONNECT')
        mqtt_terms = ('+QMTOPEN', '+QMTCONN', '+QMTSUB', '+QMTPUB',
'+QMTRECV', '+QMTUNS', '+QMTDISC', '+QMTCLOSE')

        for line in lines:
            line = line.strip()
            for term in http_terms or mqtt_terms:
                if term in line:
                    filtered_response['response'].append(line)

        return filtered_response
```

HTTP:

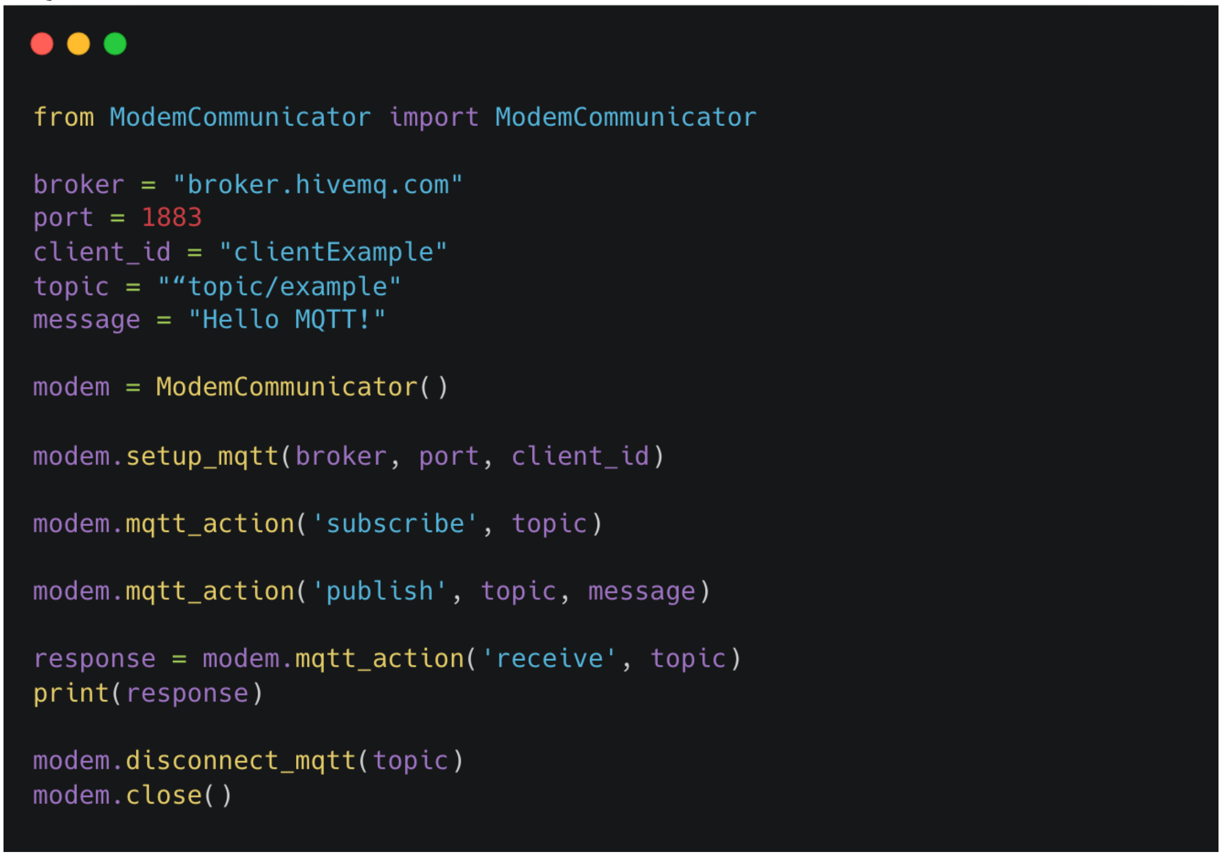


```
from ModemCommunicator import ModemCommunicator

url = "https://webhook.site/fbe25b48-2ea0-48a8-bf78-eece26eacf58"
post_data = "data=deneme"

modem = ModemCommunicator()
modem.http_configure()

response = modem.http_request(url, method='GET')
print("GET İsteği Sonucu:", response)

response = modem.http_request(url, method='POST', data=post_data)
print("POST İsteği Sonucu:", response)

modem.close()
```

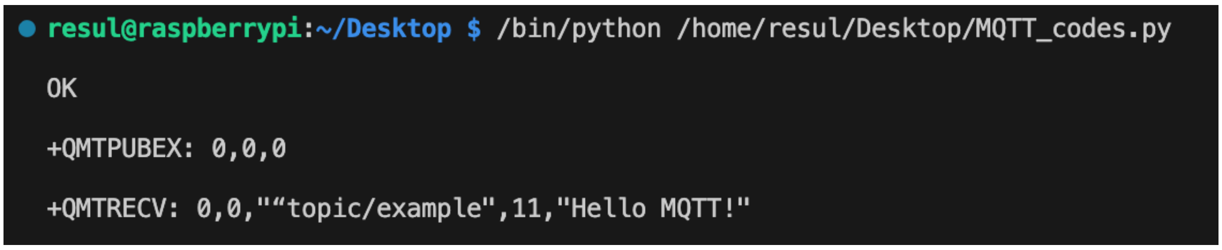


```
● resul@raspberrypi:~/Desktop $ /bin/python /home/resul/Desktop/http_codes.py
  GET İsteği Sonucu: {'response': ['OK', '+QHTTPGET: 0,200', 'CONNECT', 'OK', '+QHTTPREAD: 0'], 'status': 0}
  POST İsteği Sonucu: {'response': ['OK', '+QHTTPPOST: 0,200', 'CONNECT', 'OK', '+QHTTPREAD: 0'], 'status': 0}
```

MQTT:



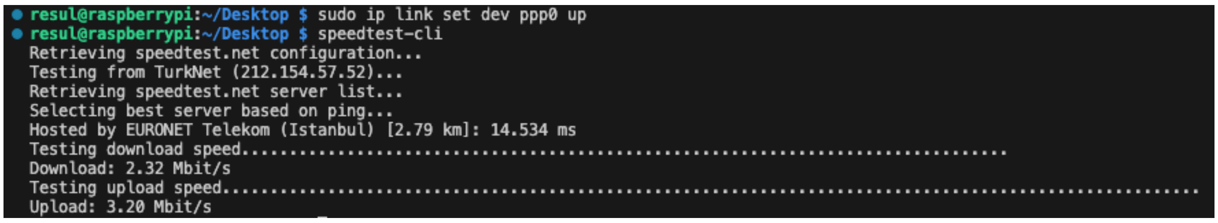
```
from ModemCommunicator import ModemCommunicator

broker = "broker.hivemq.com"
port = 1883
client_id = "clientExample"
topic = "“topic/example"
message = "Hello MQTT!"

modem = ModemCommunicator()

modem.setup_mqtt(broker, port, client_id)

modem.mqtt_action('subscribe', topic)

modem.mqtt_action('publish', topic, message)

response = modem.mqtt_action('receive', topic)
print(response)

modem.disconnect_mqtt(topic)
modem.close()
```



```
● resul@raspberrypi:~/Desktop $ /bin/python /home/resul/Desktop/MQTT_codes.py

  OK

  +QMTPUBEX: 0,0,0

  +QMTRECV: 0,0,"“topic/example",11,"Hello MQTT!"
```

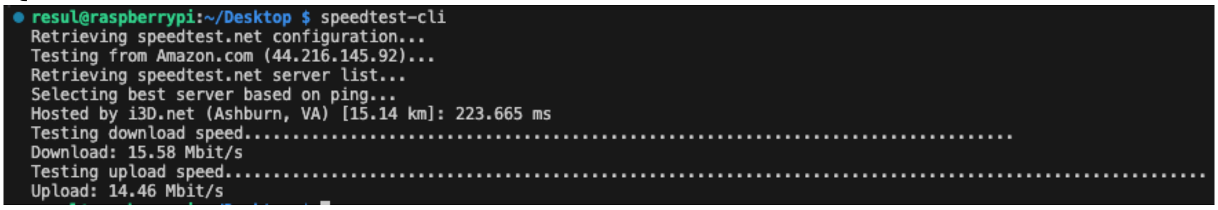
Speed Test Results:

PPP:



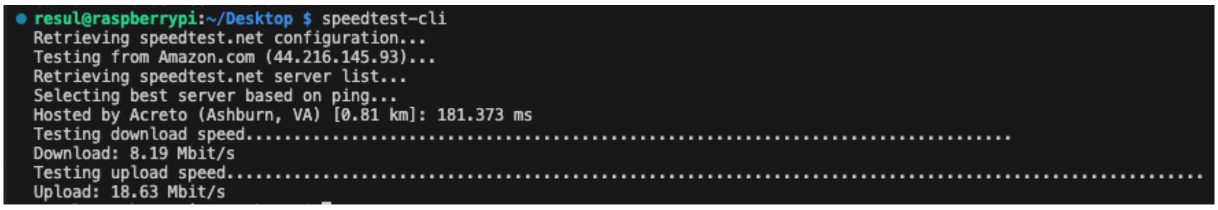
```
● resul@raspberrypi:~/Desktop $ sudo ip link set dev ppp0 up
● resul@raspberrypi:~/Desktop $ speedtest-cli
  Retrieving speedtest.net configuration...
  Testing from TurkNet (212.154.57.52)...
  Retrieving speedtest.net server list...
  Selecting best server based on ping...
  Hosted by EURONET Telekom (Istanbul) [2.79 km]: 14.534 ms
  Testing download speed................................................................................
  Download: 2.32 Mbit/s
  Testing upload speed......................................................................................................
  Upload: 3.20 Mbit/s
```

QMI:



```
● resul@raspberrypi:~/Desktop $ speedtest-cli
  Retrieving speedtest.net configuration...
  Testing from Amazon.com (44.216.145.92)...
  Retrieving speedtest.net server list...
  Selecting best server based on ping...
  Hosted by i3D.net (Ashburn, VA) [15.14 km]: 223.665 ms
  Testing download speed................................................................................
  Download: 15.58 Mbit/s
  Testing upload speed......................................................................................................
  Upload: 14.46 Mbit/s
```

ECM:



```
● resul@raspberrypi:~/Desktop $ speedtest-cli
  Retrieving speedtest.net configuration...
  Testing from Amazon.com (44.216.145.93)...
  Retrieving speedtest.net server list...
  Selecting best server based on ping...
  Hosted by Acreto (Ashburn, VA) [0.81 km]: 181.373 ms
  Testing download speed................................................................................
  Download: 8.19 Mbit/s
  Testing upload speed......................................................................................................
  Upload: 18.63 Mbit/s
```

Learning Points

Modem Communication: Acquired the ability to communicate effectively through AT commands and serial ports.

Protocol Use: Gained practical experience with the applied use of HTTP and MQTT protocols, facilitating data exchange.

Software Development: Developed applications in Python programming.

Problem-Solving: Generated solutions to technical issues.

Time Management: Acquired the skill to complete the project within the planned timeframe and manage tasks according to their priorities.

New Technologies: Attained technical knowledge and experience with embedded systems and cellular network modems.

Documentation Skills: Developed the ability to document technical topics, development processes, and code examples clearly and understandably.

Appendices

Source Codes

You can access the project's source codes through this GitHub link. This repository contains all the stages of development, the codes used, and related documentation.

References

- An Introduction to Cellular AT Commands - Cavli Wireless
- Sending AT Commands – Sixfab
- Modem AT Command - The Geek Stuff
- Read Response AT Command with PySerial - Stack Overflow
- Discussion on PySerial - Raspberry Pi Forum
- Discussion on PySerial - Python Forum
- Short Introduction – PySerial
- API Documentation – PySerial
- AT Commands Manual – Sixfab
- Solution of +CME ERROR 703
- PPP Guide – Sixfab
- QMI Huide – Sixfab
- ECM Guide - Sixfab

Resul Ekrem ÖZDEMİR
Sixfab
Intern Embedded Software Developer