

# Web-based Remote Control System for a Car using Python and MQTT

by:

Eddy Zhang

Tufts University

EE129 - Computer Networks

April 28 2024

# Introduction

- This project creates a web-based remote control system for a car using Python for the server-side code, MQTT for message communication, and HTML/CSS/JavaScript for the client-side interface.
- The system will allow users to control the car's movement (forward, backward, left, right) using a virtual joystick on a webpage.
- This project can also be used for any vehicle or devices including drones or robots that requires remote control.
- The communication MQTT and web server is fast-responsive and reliable, and also the simple webpage design makes it easy to get access to.
- The data format is set to

(x-value, y-value)

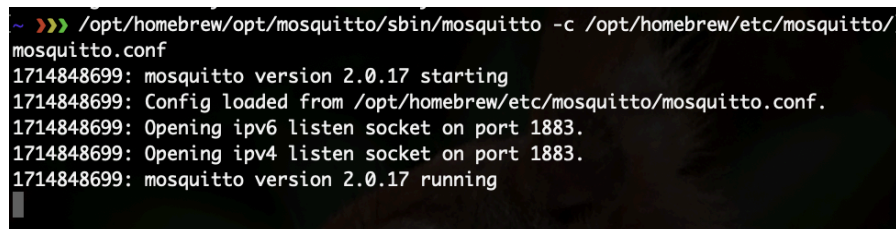
and will be parsed and processed by the receiver. The ranges of x and y are both from -100 to 100, same as the coordinate plane.

# Experimental Setup

- Download the project files here: <https://github.com/Eddyzzzzz/Project/tree/main/Project3>
- Install mqtt here: <https://mqtt.org/getting-started/>
- Set up the broker on your computer by running

...(location)/mosquitto.conf

E.g.

A terminal window with a dark background showing the command to start Mosquitto and its output. The command is `/opt/homebrew/opt/mosquitto/sbin/mosquitto -c /opt/homebrew/etc/mosquitto/mosquitto.conf`. The output shows the version (2.0.17), config loading, and opening of listen sockets on port 1883 for both IPv6 and IPv4.

```
~ >>> /opt/homebrew/opt/mosquitto/sbin/mosquitto -c /opt/homebrew/etc/mosquitto/mosquitto.conf
1714848699: mosquitto version 2.0.17 starting
1714848699: Config loaded from /opt/homebrew/etc/mosquitto/mosquitto.conf.
1714848699: Opening ipv6 listen socket on port 1883.
1714848699: Opening ipv4 listen socket on port 1883.
1714848699: mosquitto version 2.0.17 running
```

- You should be able to see the broker starting to accept clients
- Replace the "broker\_address" in the main.py file with your own IP address
- In another terminal go to the folder containing all the files, run

`python3 main.py`

to start publishing to the mqtt topic "Remote" as client "Eddy"

- Open a browser and go to 127.0.0.1/8000 to access the joystick
- Open another terminal (the 3rd one), run

`mosquitto_sub -t "Remote" -h your_IP -p 1883`

to subscribe to the mqtt topic "Remote"

E.g.

### Subscribe

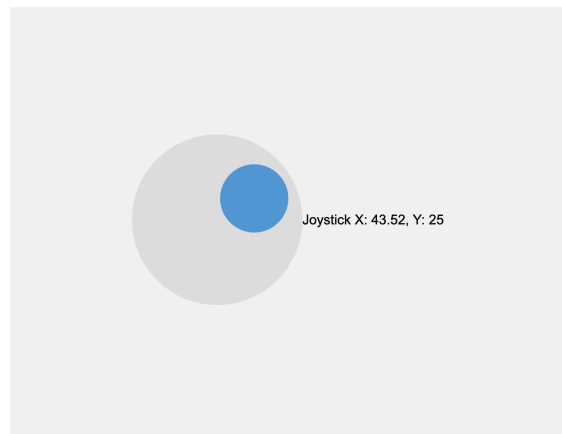
```
Last login: Sat May  4 14:51:27 on ttys045
~ >>> mosquitto_sub -t "Eddy" -h 192.168.1.176 -p 1883
```

### Confirm of subscription

```
1714866588: New connection from 192.168.1.176:57511 on port 1883.
1714866588: New client connected from 192.168.1.176:57511 as auto-83D8B33B-14ED-
953B-E3D2-53A832614014 (p2, c1, k60).
```

- Play around with the joystick and see what happens to the subscriber!

### Joystick



### Subscriber getting coordinates

```
[~ >>> mosquitto_sub -t 'Remote' -h 192.168.1.176 -p 1883
2,1
4,1
3,1
1,1
-10,1
-12,-1
-23,-5
-28,-10
-26,-13
-31,-20
-31,-25
-32,-29
-31,-32
-30,-35
-27,-37
-20,-39
-13,-40
-4,-40
-3,-40
0,-40
5,-37
7,-33
```

- Then you can use mqtt to subscribe to the topic on your Arduino, RaspberryPi, ESP32, or other microcomputers to receive the joystick values and use it to control DC motors to drive a little car remotely through your webpage! It also supports multiple subscribers.

# How it works

- Import necessary libraries including flask and mqtt

```
#!/usr/bin/env python3
import time
from http.server import HTTPServer
import paho.mqtt.client as mqtt
import time
from flask import Flask, request, jsonify, Response
import os
from http.server import BaseHTTPRequestHandler
from routes.main import routes
from response.staticHandler import StaticHandler
from response.templateHandler import TemplateHandler
from response.badRequestHandler import BadRequestHandler
import threading
```

- Set up network info including ip address, initiate client, and define the host and port

```
# Replace this with your own IP address
broker_address='192.168.1.176'

client = mqtt.Client("Eddy")
client.connect(broker_address)

HOST_NAME = '127.0.0.1'
PORT_NUMBER = 8000
```

- Write the html, css, and javascript for the webpage

```
page = '''
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PMM Remote Control System</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    .container {
      display: flex;
      align-items: center;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="text">
      <h1>PMM Remote Control System</h1>
    </div>
  </div>
</body>
</html>
'''
```

- Define the method for parsing and sending the mqtt message

```
def send_mqtt_message(x, y):  
    message = f"{x},{y}"  
    client.publish("Remote", message)
```

- Server class, implementation for GET to get information from html

```
class Server(BaseHTTPRequestHandler):  
  
    # def infinite_loop():  
    #     print('Some magic')  
    #     threading.Timer(60, infinite_loop()).start()  
  
    def do_HEAD(self):  
        return  
  
    def do_GET(self):  
        split_path = os.path.splitext(self.path)  
        request_extension = split_path[1]  
        self.send_response(200)  
        self.send_header("Content-type", "text/html")  
        self.end_headers()  
        # if request_extension == "" or request_extension == ".html":  
        #     if self.path in routes:  
        #         handler = TemplateHandler()  
        #         handler.find(routes[self.path])  
        #     else:  
        #         handler = BadRequestHandler()  
        # elif request_extension == ".py":  
        #     handler = BadRequestHandler()  
        # else:  
        #     handler = StaticHandler()  
        #     handler.find(self.path)  
        # self.respond({  
        #     'handler': handler  
        # })  
  
        self.wfile.write(bytes(page, "utf-8"))  
        if '/send' in self.path:  
            x = self.path.split('=')[1].split(',')[0]  
            y = self.path.split('=')[1].split(',')[1]  
            print("Send: " + x + ', ' + y)  
            send_mqtt_message(x, y)
```

- Main method to keep the server alive forever

```
if __name__ == '__main__':  
    httpd = HTTPServer((HOST_NAME, PORT_NUMBER), Server)  
    print(time.asctime(), 'Server UP - %s:%s' % (HOST_NAME, PORT_NUMBER))  
  
    try:  
        httpd.serve_forever()  
    except KeyboardInterrupt:  
        pass  
  
    httpd.server_close()  
    print(time.asctime(), 'Server DOWN - %s:%s' % (HOST_NAME, PORT_NUMBER))
```

# Results

Full demonstration can be viewed here: <https://imgur.com/dnXsYx9>

# Discussion

The result indicates that the server displayed the webpage, updated the html based on the joystick position, and responded to the client's GET requests. It successfully showed the client-server interaction through HTTP protocol.

There are more 200 lines of code written for this project, but roughly 70% of them are html and css codes for the additional decoration for the webpage, the main body of the code is quite simple and concise.

The biggest challenge I faced in this project was updating the coordinates of the joystick and triggering the sending of mqtt message.

# Conclusion

In this project I used python, html, css, and javascript to build a server that could interact with clients using a joystick through HTTP protocol.

The server updates the data, receives and accepts client's GET requests and sent mqtt messages to the corresponding topic on the mqtt server.

# Appendix

- Source Code: <https://github.com/Eddyzzzzz/Project/blob/main/Project3>