



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

گزارش کار آزمایشگاه سخت افزار: پایش وضعیت بیمار

اعضای گروه

رضا صومی، علیرضا نوروزی، مجید طاهرخانی

استاد

دکتر اجلالی

مشاور راهنمای

جناب آقای بحرینی

۱۴۰۲ بهمن

چکیده

هدف اصلی این پروژه، طراحی یک دستگاه قابل حمل با منبع تغذیه از باطری است که قادر به ارسال اطلاعات از مژول‌های مختلف از طریق واای فای به یک سرور می‌باشد. این سرور یک وب‌سایت را میزبانی می‌کند که می‌توان اطلاعات ارسالی را نظارت کرد. مژول اصلی از برد NodeMCU استفاده می‌کند و برای پایش در زمان واقعی مکان بیمار، از یک مژول GPS برای ارسال اطلاعات مکانی به سرور وب استفاده می‌شود.

زمانبندی تسک‌های مختلف پروژه به شرح زیر صورت گرفت.

تسک ۱ : نوشتن پروپوزال ۱۰ آبان

تسک ۲ : مطالعه مفاهیم ذکر شده، خرید قطعات ۱۷ آبان

تسک ۳ : اتصال سنسور‌ها به برد و آزمایش آفلاین سیستم‌ها ۱ آذر

تسک ۴ : اتصال برد به سرور و نمایش در کامپیوتر ۱۵ آذر

تسک ۵ : طراحی نهایی سایت ۲۹ آذر

تسک ۶ : گزارش نهایی ۶ دی

کلیدواژه‌ها: IOMT

فهرست مطالب

- | | |
|----|---|
| ۱ | ۱ مژول‌های استفاده شده |
| ۳ | ۲ قسمت سخت‌افزاری پروژه و کدهای مربوط به آن |
| ۷ | ۳ سرور و انتقال ماشین به ماشین |
| ۱۱ | ۴ وبسایت |
| ۱۴ | ۵ نحوه اجرا |
| ۱۷ | ۶ نتایج |

فهرست شکل‌ها

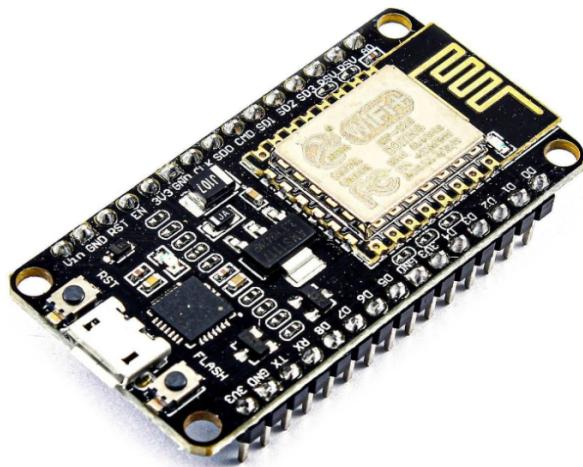
۱۱	ثبت کردن آدرس سرور	۴-۱
۱۲		۲-۴
۱۳	دربیافت اطلاعات از سرور و پردازش آن در وبسایت	۴-۳
۱۴	ایجاد مدل و ثبت آن در دیتابیس	۴-۴

فصل ۱

ماژول های استفاده شده

:Nodemcu

ویژگی های این برد که برای این کار مناسب است وجود ماژول وایفای ESP-۳۲ روی برد میباشد که هم مصرف کمی دارد هم با استفاده از زبان برنامه نویسی C به سادگی قابل برنامه ریزی میباشد.



سنسور های بایومتریک: ۱. پالس اکسیمتر: ماژول اوکسیمتر ضربان قلب PULSE OXIMETER که نیاز است روی انگشت بیمار قرار گیرد. و ضربان قلب بیمار را اندازه گیری میکند. MAX۳۰۱۰۰ با این وجود، اطلاعات معمولاً مرتبط با اسپو ۲ (اشباع اکسیژن خون) و ضربان قلب (BPM) به شرح زیر است:

۱. اشباع اکسیژن خون: (SPO₂)

- توضیح: اندازه گیری میزان اشباع اکسیژن در خون، که نسبت اکسیژن متصل به هموگلوبین به کل هموگلوبین را نشان می دهد.

- واحد اندازه‌گیری: درصد (%)
- معمولاً محدوده طبیعی: بین ۹۵ تا ۱۰۰ درصد است.

۲. ضربان قلب (BPM)

- توضیح: اندازه‌گیری تعداد ضربان قلب در یک دقیقه.
- واحد اندازه‌گیری: ضربان در هر دقیقه (bpm)
- معمولاً محدوده طبیعی: بین ۶۰ تا ۱۰۰ ضربان در هر دقیقه.

در دستگاه‌های پزشکی مدرن، سنسورها و تکنولوژی‌های مختلف برای اندازه‌گیری این مقادیر به کار می‌روند. برای اطلاعات دقیق‌تر و بروزتر، به منابع مرتبط و نوآورانه در حوزه تکنولوژی پزشکی مراجعه کنید.



۲. مازول GPS: مازول GPS موقعیت یاب جغرافیایی NEO 6M محصول GY



فصل ۲

قسمت سخت‌افزاری پروژه و کدهای مربوط به آن

یکی از مسایل قابل تحقیق در پروژه، طراحی با در نظر گرفتن توان مصرفی بود. طراحی کم توان سیستمی است که از مجموعه‌ای از تکنیک‌ها و متداول‌ترین‌ها به منظور بهینه‌سازی عمر باتری و کاهش اتلاف انرژی کلی سیستم استفاده می‌کند. برای بهینه‌سازی توان، تکنیک‌های کم مصرف زیادی وجود دارد که به سطح طراحی انتخاب شده بستگی دارد، از فناوری نیمه‌هادی تا سطوح بالاتر انتزاع.

چرا به طراحی کم توان نیاز داریم؟ سیستم‌های تعبیه‌شده باید در حین کار از انرژی کارآمد باشند تا از عمر باتری طولانی اطمینان حاصل شود، مصرف برق شهری کاهش یابد و از تولید گرمای اضافی جلوگیری شود. عمر طولانی‌تر باتری یک محصول همچنین می‌تواند منجر به کاهش هزینه‌های نگهداری شود، زیرا بازدیدهای پرهزینه برای تعویض باتری‌ها کمتر اتفاق می‌افتد. علاوه بر این، دستگاه‌های قابل حمل مانند تلفن‌های همراه، کنسول‌های بازی، و سیستم‌های الکترونیکی با باتری، نیازمند مدارهای میکروالکترونیکی هستند که با اتلاف انرژی بسیار کم طراحی شده‌اند.

برای این کار از ۳ مولفه استفاده شد.

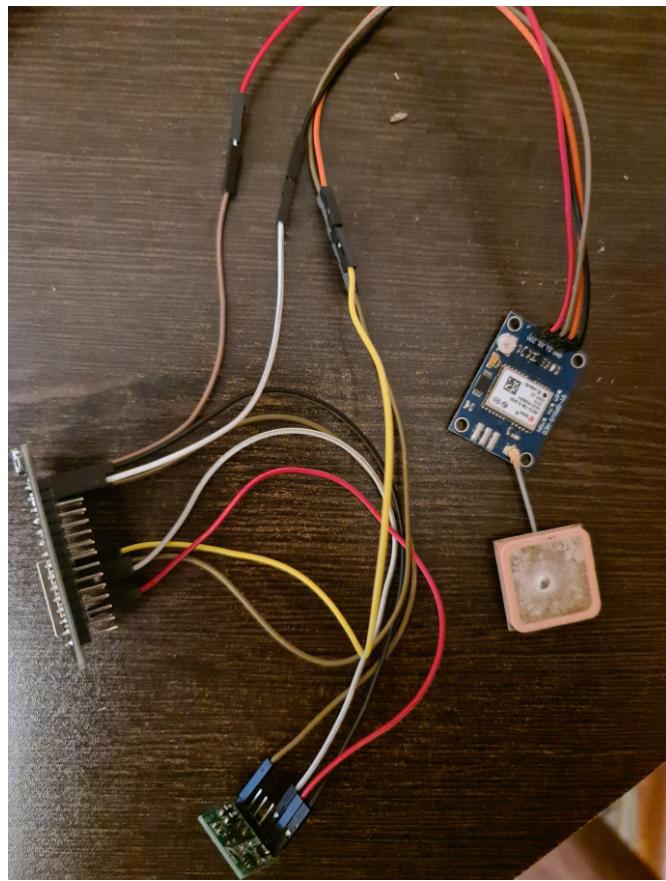
مورد اول در کد بود و کد طراحی شده کمترین میزان محاسبات را در برداشت. برای مثال برای انتقال محتوا از قسمت سخت‌افزاری به سرور از روش Comma Separated استفاده شد. در حالی که می‌توانستیم یک json ساخته و آن را ارسال کنیم که در این صورت حجم فایل ارسالی و محاسبات برای آن بیشتر می‌شد.

مورد دوم استفاده از پاوربانک بود که کارکردن پروژه با پاوربانک نیز تست شد.

و مورد اصلی و مهم، خاموش‌کردن وای‌فای هنگام ارسال نکردن داده است.

میتوانید از لینک گیت هاب زیر به پروژه دسترسی پیدا کنید:

https://github.com/Alirzeanoroozi/Hardware_lab/tree/main



توضیح کد : در اینجا یک برنامه نمونه برای استفاده از مژوول‌های Wire، WiFi، TinyGPS++، MAX30102_PulseOximeter، و PubSubClient بر روی NodeMCU نشان داده شده است. این برنامه داده‌های موقعیت جغرافیایی از مژوول GPS و داده‌های ضربان قلب و اکسیژن خون از مژوول Pulse Oximeter را جمع‌آوری کرده و از طریق اتصال به یک سرور MQTT ارسال می‌کند.

تعاریف و تنظیمات : در این بخش، تعدادی از تعریفات اولیه و تنظیمات مشخص شده‌اند. به عنوان مثال، نام شبکه WiFi، رمز عبور WiFi، آدرس MQTT Broker و... تنظیم شده‌اند.

تابع اصلی : در اینجا توابع اصلی برنامه مانند تنظیم WiFi، اتصال به MQTT Broker، تابع بازیابی اتصال، و توابع کمکی مانند تولید عدد تصادفی آورده شده‌اند.

تابع setup

این تابع در ابتدای اجرای برنامه فراخوانی می‌شود و تنظیمات اولیه را انجام می‌دهد. به عنوان مثال، اتصال به WiFi، تنظیمات مربوط به MQTT Broker، و شروع مژوول Pulse Oximeter انجام می‌شود.

تابع loop : این تابع به صورت مداوم اجرا می‌شود و در آن عملیات اصلی برنامه انجام می‌شود. این شامل جمع‌آوری داده‌های GPS و Pulse Oximeter، ارسال این داده‌ها به سرور MQTT، و کنترل زمانی

برای ارسال داده‌ها است.

تعریف و ابتدایی‌سازی کتابخانه‌ها

```
#edulcni <TinyGPS++.h>
#edulcni <WiFi.h>
#edulcni <PubSubClient.h>
#edulcni <SoftwareSerial.h>
 
TinyGPSPlus gps;
WiFiClient espClient;
PubSubClient client(espClient);
SoftwareSerial ss(RXPin, TXPin);
```

تعریف متغیرها

```
#enifed GPSBaud 9600
#enifed WLAN_SSID "      _iFiW_"
#enifed WLAN_PASS "      _iFiW_"
#enifed AIO_UPDATE_RATE_SEC 10
#enifed MQTT_BROKER "      TTQM_"
#enifed MQTT_PORT 1883
#enifed PUBLISH_TOPIC "          SPG_____"
```

تعریف توابع

```
diov setup_wifi();
diov callback(rahc* topic, byte* payload, dengisnu tni length);
diov reconnect();
diov setup();
diov loop();
```

تعریف تابع اتصال به WiFi

```
diov setup_wifi() {
    //      iFiW
}
```

توابع فراخوانی و اتصال مجدد

```
diov callback(rahc* topic, byte* payload, dengisnu tni length) {  
    // TTQM  
}  
  
{  
  
    diov reconnect() {  
        // TTQM  
    }  
}
```

تابع اولیه (setup)

```
diov setup() {  
    // TTQM  
}  
  
{
```

تابع حلقه اصلی (loop)

```
diov loop() {  
    : SPG TTQM SPG  
}  
}
```

نکات مهم

- تابع callback در حال حاضر برای پردازش پیام‌های دریافتی از سرور MQTT استفاده نمی‌شود (بخشی از کد کامنت شده است).
- اطلاعات GPS به سرور MQTT به فرمت Date;speed,altitude,longitude,latitude ارسال می‌شود.
- برنامه هر ۱۰ ثانیه یکبار اطلاعات GPS را به سرور MQTT ارسال می‌کند.
- نام کاربری، گذر واژه، و جزئیات اتصال MQTT (مانند نام کاربری و کلید) در کد وجود ندارند و باید تنظیم شوند.
- از ماژول SoftwareSerial برای ارتباط با GPS استفاده شده است و ورودی و خروجی آن به ترتیب در پایه‌های RXPin و TXPin قرار دارند.

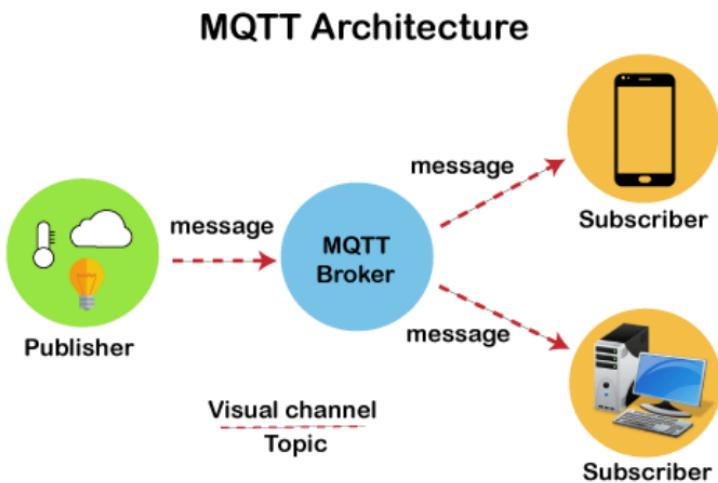
فصل ۳

سرور و انتقال ماشین به ماشین

پروتکل MQTT یک پروتکل پیام رسانی مبتنی بر استاندارد یا مجموعه‌ای از قوانین است که برای ارتباط ماشین به ماشین استفاده می‌شود. سنسورهای هوشمند، ابزارهای پوشیدنی و سایر دستگاه‌های اینترنت اشیا (IoT) معمولاً باید داده‌ها را از طریق شبکه‌ای با محدودیت منابع با پهنای باند محدود ارسال و دریافت کنند. این دستگاه‌های اینترنت اشیا از MQTT برای انتقال داده‌ها استفاده می‌کنند، زیرا پیاده‌سازی آن آسان است و می‌تواند داده‌های اینترنت اشیا را به طور کارآمدی برقرار کند. MQTT از پیام رسانی بین دستگاه‌ها به ابر (cloud) و ابر به دستگاه پشتیبانی می‌کند. مزایای استفاده از این پروتکل: سبک و کارآمد مقیاس پذیر قابل اعتماد امن سازگاری با تمام دستگاه‌ها (اندروید، لینوکس) پشتیبانی قوی

پروتکل MQTT بر اساس اصول مدل انتشار/اشتراک کار می‌کند. در ارتباطات سنتی شبکه، کلاینت‌ها و سرورها مستقیماً با یکدیگر ارتباط برقرار می‌کنند. کلاینت‌ها منابع یا داده‌ها را از سرور درخواست می‌کنند، سپس سرور پاسخ را پردازش کرده و پس می‌فرستد. با این حال، MQTT از یک الگوی انتشار/اشتراک برای جدا کردن فرستنده پیام (publisher) از گیرنده پیام (subscriber) استفاده می‌کند. در عوض، جزء سوم به نام واسطه پیام، ارتباط بین ناشران و مشترکین را مدیریت می‌کند. وظیفه کارگزار این است که تمام پیام‌های دریافتی ناشران را فیلتر کرده و آنها را به درستی بین مشترکین توزیع کند.

ناشران و مشترکین را به شرح زیر جدا می‌کند: جداسازی فضا: ناشر و مشترک از موقعیت شبکه یکدیگر آگاه نیستند و اطلاعاتی مانند آدرس IP یا شماره پورت را رد و بدل نمی‌کنند. جدا شدن زمان: ناشر و مشترک همزمان اجرا یا اتصال شبکه ندارند. جداسازی همگام سازی: هم ناشران و هم مشترکین می‌توانند بدون ایجاد وقفه در یکدیگر پیام ارسال یا دریافت کنند. برای مثال، مشترک مجبور نیست منتظر ارسال پیام توسط ناشر باشد.



نصب Mosquitto بر روی Ubuntu

تیم ما با اجرای دستورات زیر، Mosquitto را بر روی سرور Ubuntu نصب کرده است:

```

sudo apt update
sudo apt install mosquitto
sudo systemctl start mosquitto
sudo systemctl enable mosquitto

```

آزمون Mosquitto

برای اطمینان از صحت نصب و اجرای Mosquitto تیم ما از دستورات زیر برای تست استفاده کرده است:

```

mosquitto_sub -h localhost -t testTopic
mosquitto_pub -h localhost -t testTopic -m "Hello, MQTT!"

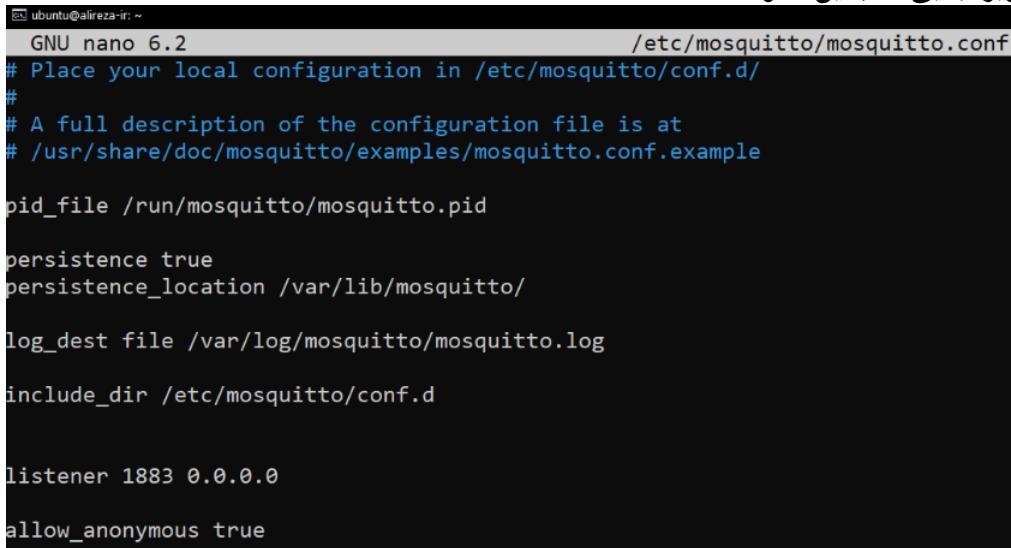
```

نتایج این تست نشان می‌دهند که Mosquitto به درستی نصب شده و اطلاعات بین Publisher و Subscriber به درستی تبادل شده‌اند.

لازم به ذکر است برای اینکه از بیرون بتوانیم به سرور درخواست بدھیم باید در فایل کانفیگ ماسکیتو دو دستور زیر را وارد کنیم:

```
sudo nano /etc/mosquitto/mosquitto.conf
listener 1883 0.0.0.0
true allow_anonymous
```

تصویر نهایی آن بدین صورت است:



```
ubuntu@alireza-ir: ~
GNU nano 6.2                               /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883 0.0.0.0

allow_anonymous true
```

حال با استفاده از کتابخانه *paho_mqtt* یک اسکریپت پایتونی نوشته تا بتوانیم داده های با یک Topic خاص را از سمت NodeMCU مورد نظر دریافت کنیم و پس از آن با یک تاپیک مشخص به سمت بکند بفرستیم تا در نهایت در وبسایت نمایش دهیم. کد مورد نظر که در سرور اجرا می شود بدین شرح است:

```

GNU nano 6.2                                     mqtt_listener.py

import paho.mqtt.client as mqtt

# MQTT broker information
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
SUBSCRIBE_TOPIC = "NodeMCU_Data"
PUBLISH_TOPIC = "Backend_Data"

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe(SUBSCRIBE_TOPIC)
#    new_message = "1,2,3,4"
#    client.publish(PUBLISH_TOPIC, new_message)
#    print(f"Published message on {PUBLISH_TOPIC}: {new_message}")

def on_message(client, userdata, msg):
    received_message = msg.payload.decode("utf-8")
    print(f"Received message on {msg.topic}: {received_message}")

    # Process the received message as needed
    # For example, you can perform some logic and create a new message
    new_message = f"Processed: {received_message}"

    # Publish the new message to a different topic
    client.publish(PUBLISH_TOPIC, new_message)
    print(f"Published message on {PUBLISH_TOPIC}: {new_message}")

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Blocking call that processes network traffic, dispatches callbacks, and handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a manual interface.
client.loop_forever()

```

با تاپیک *BackendData* اطلاعات از سخت افزار دریافت می شود و با تاپیک *NodeMCUData* به سمت بکند (که خود یک سابسکرایبر است) ارسال می شود.

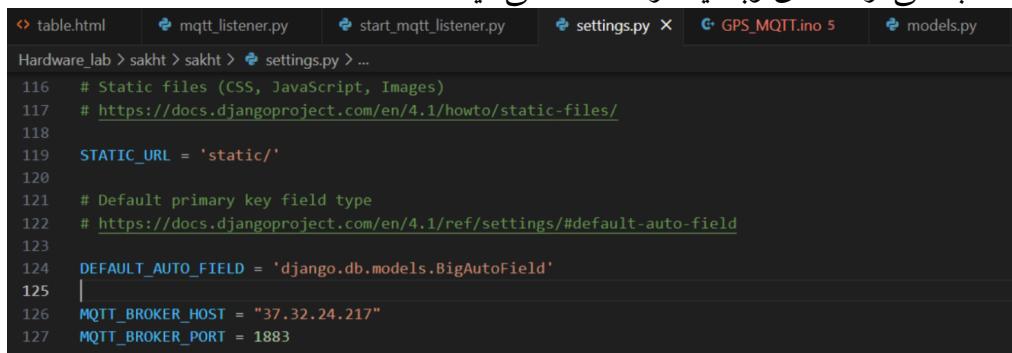
برای راه اندازی سرور کافیست اسکریپت پایتونی موجود را اجرا کنید.

فصل ۴

وبسایت

یک پروژه Django ساخته شد. دیتابیس مورد نظر با اطلاعات دریافتی ساخته شد. یک اسکریپت پایتونی برای ارتباط با سرور با استفاده از کتابخانه paho mqtt ساخته شد. و در نهایت یک صفحه فرانت نیز برای نمایش داده ها ساخته شده است. لازم به ذکر است بلافاصله به صورت hot reload هنگامی که داده ها به بکند ارسال می شود در صفحه فرانت آن اطلاعات در جدول آپدیت می شود.

در ادامه بخشی از کل های وبسایت را مشاهده می کنید.



```
table.html mqtt_listener.py start_mqtt_listener.py settings.py GPS_MQTT.ino 5 models.py

Hardware_lab > sakht > sakht > settings.py > ...
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.1/howto/static-files/
118
119 STATIC_URL = 'static/'
120
121 # Default primary key field type
122 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
123
124 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
125
126 MQTT_BROKER_HOST = "37.32.24.217"
127 MQTT_BROKER_PORT = 1883
```

شکل ۴-۱: ثبت کردن آدرس سرور

```
table.html mqtt_listener.py start_mqtt_listener.py settings.py GPS_MQTT.ino 5 models.py
Hardware_lab > sakht > app > mqtt_listener.py ...
1 # python manage.py start_mqtt_listener
2
3 import time
4 import paho.mqtt.client as mqtt
5 from django.conf import settings
6 from app.models import HealthHistory
7
8 def on_connect(client, userdata, flags, rc):
9     print("Connected with result code " + str(rc))
10    client.subscribe("Backend_Data") # The MQTT topic we want to subscribe to
11
12 def on_message(client, userdata, msg):
13     payload = msg.payload.decode("utf-8")
14     print(f"Received message: {payload}")
15
16     # Parse the payload and create a new instance of HealthHistory
17     data = payload.split(',')
18
19     if len(data) == 4:
20         location_x, location_y, SPO2, BPM = data
21
22         HealthHistory.objects.create(
23             location_x=int(location_x),
24             location_y=int(location_y),
25             SPO2=int(SPO2),
26             BPM=int(BPM)
27         )
28     else:
29         print("Invalid payload format")
30
31 def start_mqtt_listener():
32     client = mqtt.Client()
33     client.on_connect = on_connect
34     client.on_message = on_message
```

شكل ٤-٢

```

table.html mqtt_listener.py start_mqtt_listener.py settings.py GPS_MQTT.ino 5 models.py
Hardware_lab > sakht > app > mqtt_listener.py ...
1 # python manage.py start_mqtt_listener
2
3 import time
4 import paho.mqtt.client as mqtt
5 from django.conf import settings
6 from app.models import HealthHistory
7
8 def on_connect(client, userdata, flags, rc):
9     print("Connected with result code " + str(rc))
10    client.subscribe("Backend_Data") # The MQTT topic we want to subscribe to
11
12 def on_message(client, userdata, msg):
13     payload = msg.payload.decode("utf-8")
14     print(f"Received message: {payload}")
15
16     # Parse the payload and create a new instance of HealthHistory
17     data = payload.split(',')
18
19     if len(data) == 4:
20         location_x, location_y, SPO2, BPM = data
21
22         HealthHistory.objects.create(
23             location_x=int(location_x),
24             location_y=int(location_y),
25             SPO2=int(SPO2),
26             BPM=int(BPM)
27         )
28     else:
29         print("Invalid payload format")
30
31 def start_mqtt_listener():
32     client = mqtt.Client()
33     client.on_connect = on_connect
34     client.on_message = on_message

```

شکل ۴-۳: دریافت اطلاعات از سرور و پردازش آن در وبسایت

```

table.html mqtt_listener.py start_mqtt_listener.py settings.py GPS_MQTT.ino 5 models.py
Hardware_lab > sakht > app > models.py ...
1 from django.db import models
2
3
4 class HealthHistory(models.Model):
5     location_x = models.IntegerField()
6     location_y = models.IntegerField()
7     SPO2 = models.IntegerField(default=0)
8     BPM = models.IntegerField(default=0)
9     created_at = models.DateTimeField(auto_now_add=True, auto_now=False, null=True)
10
11 @staticmethod
12 def add_health_history(location_x, location_y, SPO2, BPM):
13     health_history = HealthHistory()
14     health_history.location_x = location_x
15     health_history.location_y = location_y
16     health_history.SPO2 = SPO2
17     health_history.BPM = BPM
18     health_history.save()
19     return health_history
20

```

شکل ۴-۴: ایجاد مدل و ثبت آن در دیتابیس

فصل ۵

نحوه اجرا

اتصال سخت افزار به سرور تیم ما با موفقیت برد میکروکنترلر مبتنی بر ESP8266 یا NodeMCU را به بروکر MQTT خود سرور Mosquitto متصل کرده است. این اتصال از طریق استفاده از کتابخانه PubSubClient صورت گرفته است. در زیر، مراحل انجام شده با جزئیات بیشتر آورده شده است:

۱. نصب Arduino IDE: ابتدا، Arduino IDE از وب سایت رسمی Arduino دانلود و نصب شد.
۲. نصب پشتیبانی از برد ESP8266: با افزودن پشتیبانی از برد ESP8266 به Arduino IDE و نصب کتابخانه های مربوطه، محیط برنامه نویسی آماده سازی شد.
۳. نصب کتابخانه PubSubClient: کتابخانه PubSubClient برای ارتباط با بروکر MQTT نصب و تنظیم شد.
۴. نوشتن کد NodeMCU برای WiFi8266 و MQTT: یک کد نمونه با استفاده از کتابخانه های WiFi8266 و PubSubClient برای اتصال به وای فای و بروکر MQTT نوشته شد. مقادیر placeholder با اطلاعات واقعی جایگزین شدند.
۵. آپلود کد بر روی NodeMCU: با اتصال NodeMCU به کامپیوتر و آپلود کد از طریق IDE، نرم افزارهای مورد نیاز بر روی NodeMCU نصب شدند.
۶. نظارت بر خروجی Serial: با استفاده از Serial Monitor در Arduino IDE، تیم ما موفق به مشاهده خروجی های NodeMCU شدند که نشان دهنده مراحل اتصال به وای فای و MQTT بود.

در این اقدامات، NodeMCU به سرور Mosquitto متصل شده و توانسته ایم پیامها را با

موفقیت به تاپیک "testTopic" ارسال و از همان تاپیک مشترک شده را دریافت کنیم. هر تغییر یا تنظیمات اضافی می‌تواند بر اساس نیازها و موارد خاص پروژه تغییر یابد.

جزئیات دستورات: برای اتصال یک برد میکروکنترلر مبتنی بر ESP8266 به بروکر MQTT Mosquitto نیاز به استفاده از یک کتابخانه NodeMCU برای MQTT دارد. یکی از کتابخانه‌های محبوب برای این منظور، کتابخانه PubSubClient است. در ادامه، راهنمای گام به گام برای اتصال NodeMCU به بروکر MQTT Mosquitto آورده شده است:

۱. نصب Arduino IDE: در صورتی که هنوز Arduino IDE را نصب نکرده‌اید، آن را از وبسایت [رسمی Arduino](https://www.arduino.cc/en/software) دانلود و نصب کنید (https://www.arduino.cc/en/software).

۲. نصب پشتیبانی از برد ESP8266:

- Arduino IDE را اجرا کنید.
- به "File" > "Preferences" بروید.
- آدرس زیر را به "Additional Boards Manager URLs" اضافه کنید:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

- بر روی "OK" کلیک کنید تا پنجره Preferences بسته شود.
- به "Tools" > "Board" > "Boards Manager" بروید.
- برای جستجوی "ESP8266" و نصب "ESP8266 package board" اقدام کنید.

۳. نصب کتابخانه PubSubClient:

- به "Sketch" > "Include Library" > "Manage Libraries" بروید.
- در Library Manager برای جستجوی "PubSubClient" اقدام کنید و آن را نصب کنید.

۴. نوشتن کد برای MQTT NodeMCU: در اینجا یک کد نمونه برای اتصال یک NodeMCU به بروکر MQTT آورده شده است. با جایگزینی مقادیر مورد نیاز با اطلاعات واقعی شما (نام وای‌فایی، رمز وای‌فایی، IP سرور MQTT و ...)، این کد را بر روی NodeMCU خود اجرا کنید.

آپلود کد بر روی NodeMCU:

- NodeMCU خود را از طریق یک کابل USB به کامپیوتر متصل کنید.

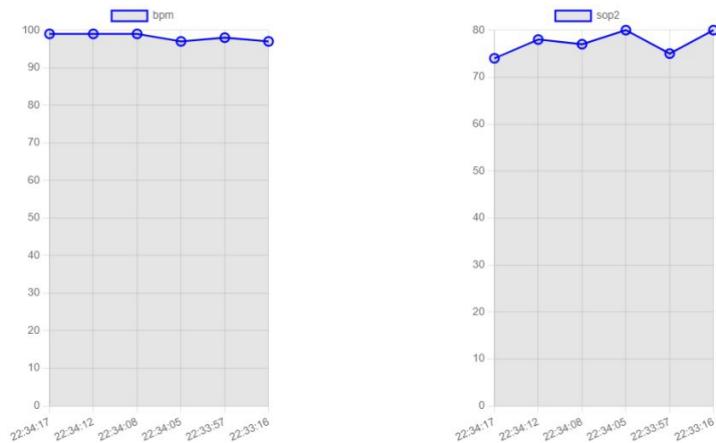
- برد انتخاب برای NodeMCU استفاده کنید.
- صحیح را از "Tools" > "Port" انتخاب کنید.
- با کلیک بر روی دکمه "Upload" کد را به NodeMCU آپلود کنید.

نظرارت بر خروجی Serial: پس از آپلود، Serial Monitor از "Tools" > "Serial Monitor" را باز کنید تا خروجی NodeMCU را مشاهده کنید. باید پیام‌هایی را که اتصال به وای‌فای و MQTT را نشان می‌دهند، همچنین هر پیام دریافتی، مشاهده کنید. حالا باید NodeMCU شما به سرور Mosquitto MQTT متصل شده باشد و پیام‌ها را به تاپیک "test" ارسال و از همان تاپیک مشترک شده باشد. کد را بر اساس نیازها و تنظیمات شبکه‌ای خود تنظیم کنید.

فصل ۶

نتایج

در تصویر زیر اطلاعات بیمار را در وبسایت مشاهده می‌کنید. نمودار سمت راست برای spo₂ و نمودار سمت چپ برای bpm می‌باشد. در طرحی وبسایت سعی شده است تا حد امکان از اصل سادگی استفاده شود.



همچنین در صفحه دیگر اطلاعات را به صورت جدول می‌توانید مشاهده کنید. این اطلاعات شامل طول و عرض جغرافیای و ساعت ذخیره شدن اطلاعات در دیتابیس نیز است و همچنین یک لینک برای هر رکورد جدول وجود دارد که در صورت کلیک بر حسب طول و عرض جغرافیای موجود آن نقطه در

وبسایت google maps نمایش داده می شود.

time	location_x	location_y	SPO2	BPM
22:33:16	35	51	80	97
22:33:57	35	51	75	98
22:34:05	35	51	80	97
22:34:08	35	51	77	99
22:34:12	35	51	78	99
22:34:17	35	51	74	99
22:52:59	35	51	79	96