

نام استاد: دکتر فصحتی

نام درس: مدارهای واسط

شماره دانشجویی: ۴۰۱۱۰۶۳۶۳

نام دانشجو: میترا قلی پور

شماره دانشجویی: ۴۰۱۱۰۶۲۵۵

نام دانشجو: ملیکا علیزاده

شماره دانشجویی: ۴۰۱۱۷۰۶۶۱

نام دانشجو: الینا هژبری

۱- مقدمه

در بسیاری از پروژه‌های صنعتی و تحقیقاتی، نظارت بر دما از اهمیت بالایی برخوردار است. به‌ویژه در سیستم‌های باتری، دما نقش مهمی در عملکرد و عمر باتری‌ها ایفا می‌کند. اما در بسیاری از موارد، استفاده از سنسور دما به دلایل مختلفی از جمله محدودیت‌های اندازه، هزینه‌های بالا، و مشکلات طراحی به‌صرفه نیست. بنابراین، استفاده از روش‌های پیش‌بینی دما بر اساس داده‌های موجود، یک راه‌حل جایگزین و موثر به‌شمار می‌آید.

در این پروژه، با استفاده از داده‌های ولتاژ، جریان، وضعیت و زمان باتری‌ها، هدف پیش‌بینی دمای آن‌ها است. با طراحی یک مدل یادگیری ماشین مبتنی بر شبکه عصبی، سعی شده است تا از ویژگی‌های مختلف داده‌ها برای پیش‌بینی دما به‌طور دقیق و با استفاده از داده‌های تاریخی استفاده شود. این روش نه تنها هزینه‌های مرتبط با استفاده از سنسورهای فیزیکی را کاهش می‌دهد، بلکه به دقت بالا و قابلیت توسعه در سیستم‌های پیچیده‌تر کمک می‌کند.

۲- شرح پروژه

در این پروژه، هدف این است که یک مدل شبکه عصبی روی بورد (Arduino) با استفاده از مدل (ESP32 WROOM) پیاده‌سازی شود تا دمای باتری پیش‌بینی شود. داده‌های ورودی به مدل از طریق دو پروتکل مختلف WiFi و سریال منتقل می‌شوند. برای پردازش و پیش‌بینی دما، از یک مدل یادگیری ماشین مبتنی بر شبکه عصبی استفاده شده است که ابتدا با استفاده از TensorFlow طراحی و آموزش داده شده است.

پس از آموزش مدل، آن را به یک کتابخانه C تبدیل کرده‌ایم تا بتوان آن را روی برد ESP32 WROOM بارگذاری کرد. با استفاده از این کتابخانه، داده‌های ورودی به برد ارسال شده و مدل پیش‌بینی دمای باتری را محاسبه کرده و خروجی آن از طریق پروتکل‌های WiFi یا سریال به سیستم‌های دیگر ارسال می‌شود. این روش شبیه سازی از طراحی برد هایی که می‌توانند جایگزین سنسورهای دما در باتری شوند.

۳- آموزش مدل شبکه عصبی

نحوه اجرای یادگیری شبکه عصبی را می‌توانید در آدرس `"make_model/train.ipynb"` ببینید. در این فایل، مدل پیش‌بینی دما بر اساس ولتاژ و جریان باتری طراحی شده است. ابتدا داده‌های موجود در فایل CSV بارگذاری شده و پیش‌پردازش‌هایی روی آن‌ها انجام شد. این پیش‌پردازش‌ها شامل حذف ستون‌های غیر ضروری و تبدیل ویژگی state به فرمت one-hot encoding بود. سپس، داده‌ها به ویژگی‌ها (X) و هدف (y) تقسیم شده و داده‌های ویژگی مقیاس‌بندی نشده برای آموزش و ارزیابی مدل آماده شد. مدل شبکه عصبی با استفاده از TensorFlow ساخته شده است. مدل دارای دو لایه مخفی با تعداد نرون‌های ۳۲ و ۱۶ و یک لایه خروجی برای پیش‌بینی دما است. مدل با استفاده از داده‌های آموزشی آموزش داده شده و سپس بر روی داده‌های آزمایشی ارزیابی شد.

بعد از آموزش مدل، پیش‌بینی‌هایی بر اساس داده‌های آزمایشی انجام شد و نمودار مقایسه‌ای بین مقادیر پیش‌بینی شده و مقادیر واقعی رسم شد. سپس، مدل به فرمت TensorFlow Lite تبدیل شد تا بتوان از آن در برد آردوینو استفاده کرد.

در نهایت، مدل TensorFlow Lite بر روی داده‌های تست اجرا شده و پیش‌بینی دما برای هر نمونه از داده‌ها نمایش داده شد. برای دیتاست نیز از دیتاست ناسا استفاده شده است.

۴- پیاده‌سازی پروژه با استفاده از پروتکل سریال

برای انجام این پروژه، ما از پروتکل سریال برای انتقال داده‌ها بین کامپیوتر و برد ESP32 استفاده کردیم. در ابتدا، مدل شبکه عصبی که پیش‌بینی دمای باتری را انجام می‌دهد، به برد منتقل شد و سپس داده‌ها از کامپیوتر به برد ارسال می‌شود تا مدل بتواند آنها را پردازش کرده و دمای پیش‌بینی شده را برگرداند.

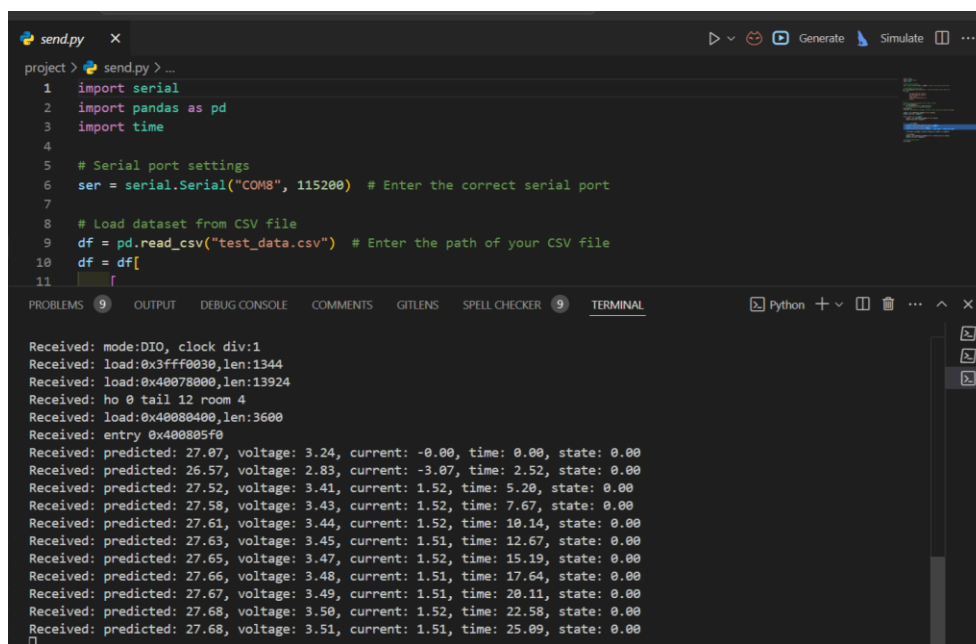
مراحل انجام پروژه:

- ۱- بارگذاری مدل شبکه عصبی: مدل شبکه عصبی که برای پیش‌بینی دما طراحی شده است، به صورت یک فایل آرایه‌ای تبدیل شده و در برد ESP32 بارگذاری می‌شود. این کار با استفاده از توابع کتابخانه TensorFlowLite_ESP32 انجام می‌شود.
- ۲- مقداردهی اولیه: ابتدا متغیرهای لازم برای ذخیره داده‌ها (ولتاژ، جریان، زمان و وضعیت باتری) تعریف شده است. این داده‌ها به عنوان ورودی به مدل شبکه عصبی ارسال می‌شوند. در این مرحله، حافظه برای مدل تخصیص داده می‌شود و از توابع `AllocateTensors()` و `Interpreter` برای آماده‌سازی محیط اجرا استفاده می‌شود.

۳- دریافت داده‌ها از پورت سریال: در حلقه اصلی، هر بار که داده‌ای از کامپیوتر از طریق پروتکل سریال دریافت می‌شود، این داده‌ها به فرمت مناسب پردازش می‌شوند. داده‌ها شامل ولتاژ، جریان، زمان و وضعیت باتری هستند که در قالب یک رشته متنی از هم جدا شده‌اند. سپس این داده‌ها استخراج شده و به متغیرهای مختلف اختصاص داده می‌شوند.

۴- پردازش داده‌ها و پیش‌بینی دما: داده‌ها پس از استخراج، در آرایه‌ای به نام `input_data` قرار می‌گیرند. این آرایه به ورودی مدل ارسال می‌شود. سپس مدل با استفاده از تابع `Invoke()` اجرا می‌شود تا پیش‌بینی دمای باتری انجام شود. نتیجه پیش‌بینی شده در متغیر `predictedTemperature` ذخیره می‌شود.

۵- ارسال خروجی به کامپیوتر: پس از پردازش داده‌ها، دمای پیش‌بینی شده به همراه مقادیر ورودی (ولتاژ، جریان، زمان و وضعیت) به صورت یک رشته متنی به کامپیوتر ارسال می‌شود. این داده‌ها از طریق پورت سریال ارسال و در کامپیوتر نمایش داده می‌شوند.



```
send.py X
project > send.py > ...
1 import serial
2 import pandas as pd
3 import time
4
5 # Serial port settings
6 ser = serial.Serial("COM8", 115200) # Enter the correct serial port
7
8 # Load dataset from CSV file
9 df = pd.read_csv("test_data.csv") # Enter the path of your CSV file
10 df = df[
11
Received: mode:DIO, clock div:1
Received: load:0x3fff0030,len:1344
Received: load:0x40078000,len:13924
Received: ho 0 tail 12 room 4
Received: load:0x40080400,len:3600
Received: entry 0x400805f0
Received: predicted: 27.07, voltage: 3.24, current: -0.00, time: 0.00, state: 0.00
Received: predicted: 26.57, voltage: 2.83, current: -3.07, time: 2.52, state: 0.00
Received: predicted: 27.52, voltage: 3.41, current: 1.52, time: 5.20, state: 0.00
Received: predicted: 27.58, voltage: 3.43, current: 1.52, time: 7.67, state: 0.00
Received: predicted: 27.61, voltage: 3.44, current: 1.52, time: 10.14, state: 0.00
Received: predicted: 27.63, voltage: 3.45, current: 1.51, time: 12.67, state: 0.00
Received: predicted: 27.65, voltage: 3.47, current: 1.52, time: 15.19, state: 0.00
Received: predicted: 27.66, voltage: 3.48, current: 1.51, time: 17.64, state: 0.00
Received: predicted: 27.67, voltage: 3.49, current: 1.51, time: 20.11, state: 0.00
Received: predicted: 27.68, voltage: 3.50, current: 1.52, time: 22.58, state: 0.00
Received: predicted: 27.68, voltage: 3.51, current: 1.51, time: 25.09, state: 0.00
```

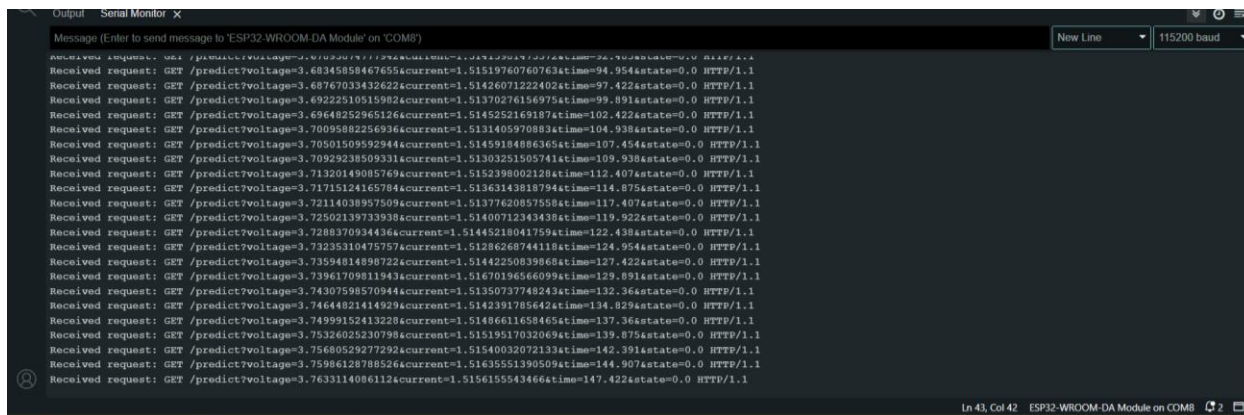
شکل ۱ خروجی حاصل از انتقال با پروتکل serial سمت ارسال کننده داده

۵- پیاده‌سازی پروژه با استفاده از پروتکل WiFi

برای پیاده‌سازی این پروژه با استفاده از پروتکل WiFi، ابتدا دستگاه ESP32 به شبکه WiFi متصل می‌شود. اطلاعات مربوط به SSID و رمز عبور شبکه در کد تنظیم شده و سپس از تابع `WiFi.begin(ssid, password)` برای شروع اتصال استفاده می‌شود. پس از اتصال به شبکه، آدرس IP دستگاه بر روی سریال نمایش داده می‌شود. بعد از این مرحله، یک سرور HTTP در پورت ۸۰ راه‌اندازی می‌شود که برای دریافت درخواست‌های ورودی از کلاینت‌ها آماده است.

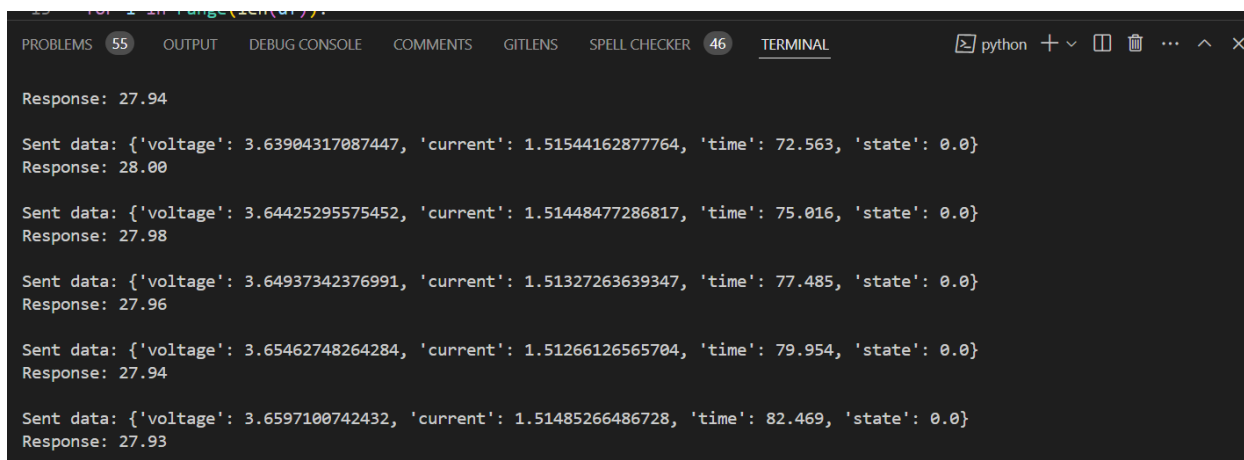
در قسمت بعدی، زمانی که یک درخواست HTTP از کلاینت دریافت می‌شود، پارامترهای مربوط به ولتاژ، جریان، زمان و وضعیت به سرور ارسال می‌شود. این پارامترها به مدل ورودی ارسال شده و پیش‌بینی دما از طریق مدل انجام می‌شود. پس از اجرای مدل، دما

پیش‌بینی‌شده به صورت یک پاسخ HTTP به کلاینت ارسال می‌شود. این فرایند برای هر درخواست به صورت تکراری انجام می‌شود و تأخیر ۱ ثانیه بین درخواست‌ها اعمال می‌شود تا بار اضافی به سرور وارد نشود.



```
Message (Enter to send message to 'ESP32-WROOM-DA Module' on COM8)
New Line 115200 baud
Received request: GET /predict?voltage=3.6767033432622&current=1.51426071222402&time=97.422&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.68245858467625&current=1.51519740760763&time=94.954&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.68767033432622&current=1.51426071222402&time=97.422&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.69222510515982&current=1.51370276156975&time=99.891&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.69648252965126&current=1.5145252169187&time=102.422&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.70095882256936&current=1.5131405970883&time=104.938&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.70501509592944&current=1.51459184886365&time=107.454&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.70929238509331&current=1.51303251505741&time=109.938&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.71320149085769&current=1.5152398002128&time=112.407&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.71715124165784&current=1.51363143818794&time=114.875&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.72114038957509&current=1.51377620857558&time=117.407&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.72502139733938&current=1.51400712343438&time=119.922&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.7288370934436&current=1.51445218041959&time=122.438&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.73235310475757&current=1.51286269744118&time=124.954&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.73594814898722&current=1.5144225083968&time=127.422&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.73961709811943&current=1.51470196566099&time=129.891&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.74307598570944&current=1.51350737748243&time=132.36&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.74644821414929&current=1.5142391785642&time=134.829&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.74999152413228&current=1.51486611658465&time=137.36&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.75326025230798&current=1.51519517032069&time=139.875&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.75680529277292&current=1.51540032072133&time=142.391&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.75986128788526&current=1.51635551390509&time=144.907&state=0.0 HTTP/1.1
Received request: GET /predict?voltage=3.7633114086112&current=1.515615543466&time=147.422&state=0.0 HTTP/1.1
```

شکل ۲ خروجی حاصل از انتقال داده با پروتکل WiFi سمت ESP32



```
for i in range(10):
    # Send data
    data = {'voltage': voltage, 'current': current, 'time': time, 'state': state}
    send_data(data)
    # Receive response
    response = receive_data()
    print(f"Response: {response}")
```

```
Response: 27.94

Sent data: {'voltage': 3.63904317087447, 'current': 1.51544162877764, 'time': 72.563, 'state': 0.0}
Response: 28.00

Sent data: {'voltage': 3.64425295575452, 'current': 1.51448477286817, 'time': 75.016, 'state': 0.0}
Response: 27.98

Sent data: {'voltage': 3.64937342376991, 'current': 1.51327263639347, 'time': 77.485, 'state': 0.0}
Response: 27.96

Sent data: {'voltage': 3.65462748264284, 'current': 1.51266126565704, 'time': 79.954, 'state': 0.0}
Response: 27.94

Sent data: {'voltage': 3.6597108742432, 'current': 1.51485266486728, 'time': 82.469, 'state': 0.0}
Response: 27.93
```

شکل ۳ خروجی حاصل از انتقال داده با پروتکل WiFi سمت ارسال کننده داده

کدهای مربوط به یادگیری شبکه عصبی در پوشه `make_model`

کدهای مربوط به انتقال داده با پروتکل سریال در پوشه `Serial`:

- کد ارسال کننده: `send.py`
- کد آردوینو در پوشه `main`

کدهای مربوط به انتقال داده با پروتکل WiFi در پوشه `WiFi`:

- همانند بخش سریال