ملینا طاهری ۴۰۲۱۰۶۱۶۷

کیمیا قدیر ۴۰۲۱۰۶۳۲۶

نام پروژه: طبقه بند رنگ تطبیقی

معرفی پروژه

این پروژه با هدف پیاده‌سازی یک سامانه طبقه‌بندی رنگ قابل یادگیری طراحی و اجرا شده است. ایده اصلی پروژه بر پایه این مفهوم است که دستگاه در ابتدا تنها تعداد محدودی رنگ (به‌صورت پیش‌فرض) را تشخیص می‌دهد و سپس با تعامل کاربر و استفاده از دکمه «یادگیری»، امکان اضافه کردن رنگ‌های جدید فراهم می‌شود. بخش نرم‌افزاری نیز نقش یک سرور را دارد که داده‌ها را دریافت کرده، مدل را آموزش می‌دهد و نتیجه تشخیص را برای نمایش به دستگاه ارسال می‌کند.

طبق مستند پروژه، ساختار کلی شامل:

- یک بخش سخت‌افزاری (میکروکنترلر + سنسور رنگ + کلیدهای آموزشی LED + نمایشگر)
- یک بخش نرم‌افزاری (Python Server + TensorFlow Lite + OTA / HTTP)

در نسخه پیاده‌سازی‌شده، به دلیل اجرای پروژه در محیط شبیه‌ساز، برخی اجزای سخت‌افزار واقعی با معادل‌های ساده‌تر جایگزین شده‌اند؛ با این وجود، منطق اصلی پروژه و جریان داده کاملاً مطابق هدف پروژه حفظ شده است.

معماری سیستم و جریان عملکرد

این سیستم از دو بخش اصلی تشکیل شده است:

کلاینت (Arduino UNOدرProteus)

وظایف کلاینت عبارت‌اند از:

- تولید و خواندن داده‌های رنگی R،G، B(در نسخه واقعی از سنسور رنگ؛ در نسخه شبیه‌سازی از ورودی‌های آنالوگ)
- ارسال داده‌ها به سرور به صورت پیام‌های JSON روی ارتباط سریال
- دریافت فرمان LED از سرور و اعمال PWM برای نمایش رنگ تشخیص داده‌شده
- دریافت ورودی کاربر با یک دکمه و ارسال دستورهایی مثل یادگیری/حذف/تعویض اسلات به سرور

سرور(Python)

وظایف سرور عبارت‌اند از:

- دریافت RGB از سریال
- پیش‌بینی رنگ با مدل یادگیری ماشین
- تصمیم‌گیری برای خروجی LED و ارسال آن به دستگاه
- ذخیره رنگ‌های جدید در اسلات‌های آموزشی و بازآموزی مدل) شبیه‌سازی(OTA

تشریح کامل سخت افزار در پروتئوس

Arduino UNO (ARD1)

در شماتیکProteus ، برد Arduino UNO نقش اصلی را دارد. پایه‌های مهم مطابق کد و مدار:

ورودی‌ها (شبیه‌سازی سنسور رنگ):

- A0 : کانال قرمز(R)
- A1 : کانال سبز(G)
- A2 : کانال آبی(B)

ورودی دکمه:

- D2 : کلید کنترل (BTN) با Pull-up داخلی

خروجی‌های نمایش رنگ:(PWM)

- D9 : خروجی LED قرمز
- D10 : خروجی LED سبز
- D11 : خروجی LED آبی

ارتباط با سرور:

- ارتباط سریال UART از طریق TX/RX برقرار می‌شود و پیام‌ها به صورت JSON رد و بدل می‌شوند.

شبیه‌سازی سنسور رنگ با پتانسیومترها

در سمت چپ مدار سه پتانسیومتر مشاهده می‌شود:

- RV-RED
- RV-GREEN
- RV-BLUE

هر پتانسیومتر یک ولتاژ متغیر (۰ تا ۵ ولت) تولید می‌کند و به یکی از پایه‌های آنالوگ متصل است. این ولتاژ متغیر نقش شدت هر کانال رنگ را ایفا می‌کند و عملاً جایگزین خروجی سنسور رنگ واقعی شده است.

در کد Arduino مقدار analogRead در بازه ۰..۱۰۲۳ خوانده شده و سپس به بازه ۰..۲۵۵ نگاشت می‌شود تا با استاندارد PWM و پیام‌های سرور هم‌مقیاس شود.

LEDها و مقاومت‌های محدودکننده جریان

در سمت راست مدار سه LED برای نمایش خروجی رنگی وجود دارد:

- LEDقرمز(D1)
- LEDسبز(D3)
- LEDآبی(D2)

برای هر LED یک مقاومت سری 220Ωقرار داده شده است .(R1, R2, R3) این مقاومت‌ها برای محدود کردن جریان و جلوگیری از آسیب دیدن LED استفاده می‌شوند و در شبیه‌سازی نیز رفتار مدار را واقعی‌تر می‌کنند.

دکمه فشاری و منطق چندفرمانی

یک دکمه روی پایه D2 قرار دارد و با INPUT_PULLUP تنظیم شده است. بنابراین:

- حالت عادی : HIGH
- هنگام فشرده شدن : LOW

با تشخیص مدت زمان نگه داشتن همین دکمه، سه فرمان مختلف ایجاد می‌شود:

- حدود ۱ ثانیه: تعویض اسلات
- حدود ۲ ثانیه: یادگیری رنگ

- حدود ۳ ثانیه: حذف رنگ ذخیره‌شده

برای جلوگیری از خطای نویز مکانیکی کلید از Debounce استفاده شده است.

پروتکل ارتباطی (JSON روی Serial)

پیام‌های ارسالی از Arduino به سرور

پیام داده(Data) :

```
{"type":"data","r":120,"g":30,"b":220}
```

این پیام به صورت دوره‌ای (هر ۲۵۰ ms) ارسال می‌شود.

پیام فرمان(Command) :

```
{"type":"cmd","action":"learn","r":80,"g":200,"b":60}
```

این پیام هنگام رها شدن دکمه و بر اساس مدت فشار ارسال می‌شود.

پیام‌های ارسالی از سرور بهArduino

فرمانLED :

```
{"cmd":"led","R":255,"G":0,"B":0}
```

Arduino این مقدارها را دریافت کرده و روی خروجی PWM اعمال می‌کند.

تشریح کد پایتون(سرور)

تنظیمات، پارامترها و دیتای اولیه

در کد سرور، ارتباط سریال روی COM8 با نرخ 9600 تنظیم شده است. همچنین:

- مدل از ابتدا دو رنگ پایه red و blue دارد.
- یک دیکشنری برای رنگ‌های یاد گرفته شده توسط کاربر (user_slots) وجود دارد.
- تعداد اسلات‌ها ۵ عدد تعریف شده است.

مدل یادگیری ماشین(KNN)

مدل استفاده‌شده:

KNeighborsClassifier(n_neighbors=1)

دلیل انتخاب:

- مناسب برای دیتاست‌های بسیار کوچک
- آموزش سریع
- امکان اضافه کردن نمونه جدید و تغییر رفتار سیستم بلافاصله بعد از آموزش مجدد

دلیل انتخاب:

- مناسب برای دیتاست‌های بسیار کوچک
- آموزش سریع
- امکان اضافه کردن نمونه جدید و تغییر رفتار سیستم بلافاصله بعد از آموزش مجدد

تابع train_model و مفهوم OTA

تابع train_model() دیتای آموزشی را از دو بخش می‌سازد:

- رنگ‌های پایه
- رنگ‌های ذخیره‌شده در اسلات‌ها

سپس مدل آموزش داده می‌شود و نسخه مدل (model_version) افزایش پیدا می‌کند. پیام چاپ‌شده با عنوان OTA نشان‌دهنده شبیه‌سازی مفهوم «به‌روزرسانی از راه دور» است.

تصمیم‌گیری برای خروجی LED

تابع decide_led_logic برای نمایش خروجی دو حالت دارد:

۱. اگر پیش‌بینی دقیقاً یکی از رنگ‌های اصلی باشد ← همان رنگ کاملاً روشن شود.

۲. اگر رنگ ترکیبی یا ناشناخته باشد:

- در صورت نزدیک بودن ← R,G,B سفید
- در غیر این صورت دو کانال بزرگ‌تر روشن می‌شوند (نمایش ترکیب رنگ)

حلقه اصلی دریافت و پاسخ

در حلقه اصلی:

- پیام خوانده می‌شود.
- JSON decode انجام می‌شود.
- اگر پیام data باشد:
  - پیش‌بینی انجام می‌شود.
  - نتیجه در قالب لاگ HTTP شبیه‌سازی می‌شود.
  - فرمان LED ارسال می‌شود.
- اگر پیام cmd باشد:
  - next_slot: تعویض اسلات و یک چشمک سفید
  - learn: ذخیره نمونه جدید، بازآموزی مدل، تایید با LED سبز
  - delete: حذف اسلات، بازآموزی مدل، تایید با LED قرمز

تشریح کد آردوینو

خواندن ورودی رنگ

هر حلقه مقادیر سه کانال از A0/A1/A2 خوانده شده و از ۰..۱۰۲۳ به ۰..۲۵۵ تبدیل می‌شوند.

تشخیص فرمان با مدت نگه داشتن دکمه

با Debounce ابتدا پایداری دکمه تضمین می‌شود، سپس با اندازه‌گیری مدت فشار و مقایسه با پنجره خطا:

- ۱ ثانیه : next_slot

- ۲ ثانیه : learn
- ۳ ثانیه : delete

در صورت تشخیص فرمان، پیام JSON شامل RGB همان لحظه ارسال می‌شود.

ارسال داده دوره‌ای

هر ۲۵۰ ms یک پیام data ارسال می‌شود، مشروط بر این‌که دکمه در حال نگه داشتن نباشد.

دریافت فرمان LED و اعمالPWM

Arduino پیام دریافتی را JSON decode می‌کند و اگر "cmd="led باشد:

- سه خروجی PWM (D9/D10/D11) با analogWrite تنظیم می‌شوند.

دلیل عملکرد صحیح سیستم و منطق کلی

این سیستم به دلیل وجود یک چرخه کامل و هماهنگ بین کلاینت و سرور به درستی کار می‌کند:

۱. کلاینت RGB را تولید و ارسال می‌کند.

۲. سرور داده را دریافت و با مدل ML رنگ را طبقه‌بندی می‌کند.

۳. سرور نتیجه را به فرمان LED تبدیل می‌کند.

۴. کلاینت فرمان را دریافت کرده و نمایش می‌دهد.

۵. در حالت یادگیری، سرور نمونه جدید را ذخیره کرده و مدل را مجدداً آموزش می‌دهد؛ بنابراین از همان لحظه، تشخیص رنگ جدید امکان‌پذیر می‌شود.

مفهوم OTA در این شبیه‌سازی به صورت «بازآموزی مدل و تغییر فوری رفتار سیستم در سمت سرور» پیاده‌سازی شده و در عمل همان نتیجه مورد انتظار (یعنی به‌روزرسانی هوشمندی دستگاه بعد از آموزش جدید) را ایجاد می‌کند.

جمع بندی

در این پروژه یک سامانه تشخیص رنگ قابل یادگیری پیاده‌سازی شد که قابلیت‌های زیر را ارائه می‌دهد:

- تشخیص رنگ‌های پایه
- یادگیری رنگ‌های جدید توسط کاربر
- مدیریت چند اسلات آموزشی با یک دکمه
- ارتباط دوطرفه کلاینت/سرور با JSON
- بازآموزی سریع مدل و اعمال اثر یادگیری به صورت آنی (شبیه OTA)

نسخه شبیه‌سازی‌شده در Proteus با جایگزینی سنسور واقعی و شبکه Wi-Fi ، یک نمونه کامل و قابل اجرا از منطق اصلی پروژه ارائه می‌دهد و تمامی مراحل تشخیص، یادگیری و نمایش نتیجه را به صورت دقیق پیاده‌سازی می‌کند.