Namatek
True Education

ESP8266

www.namatek.com

کنترل لامپ با ESP8266

فهرست مطالب

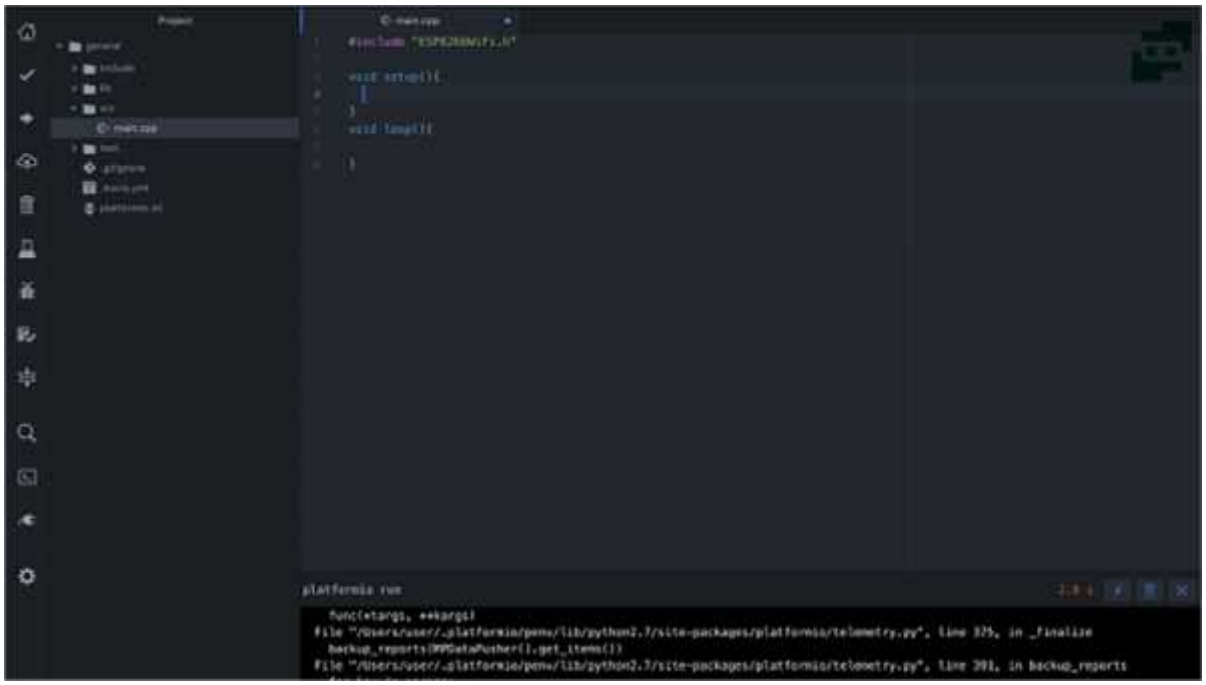
یکی از دغدغه های جدید بشر استفاده راحت از اشیا است، در همین راستا کنترل لامپ با ESP8266 از پروژه های ساده اما کاربردی شناخته می شود.

آیا شما هم مایل هستید لامپ های اتاق را با استفاده از یک ماژول کوچک و کامپیوتر خودتان کنترل کنید؟

در این مقاله روند اجرای پروژه آورده ایم تا شما به بهترین نحو آن را پیاده کنید.

کد نوشته شده برای کنترل لامپ با ESP8266

در محیط نرم افزار Atom کتابخانه ESP8266WiFi.h را اضافه کرده و در بخش setup ارتباط سریال را با baud rate مساوی ۱۱۵۲۰۰ راه اندازی می کنیم.



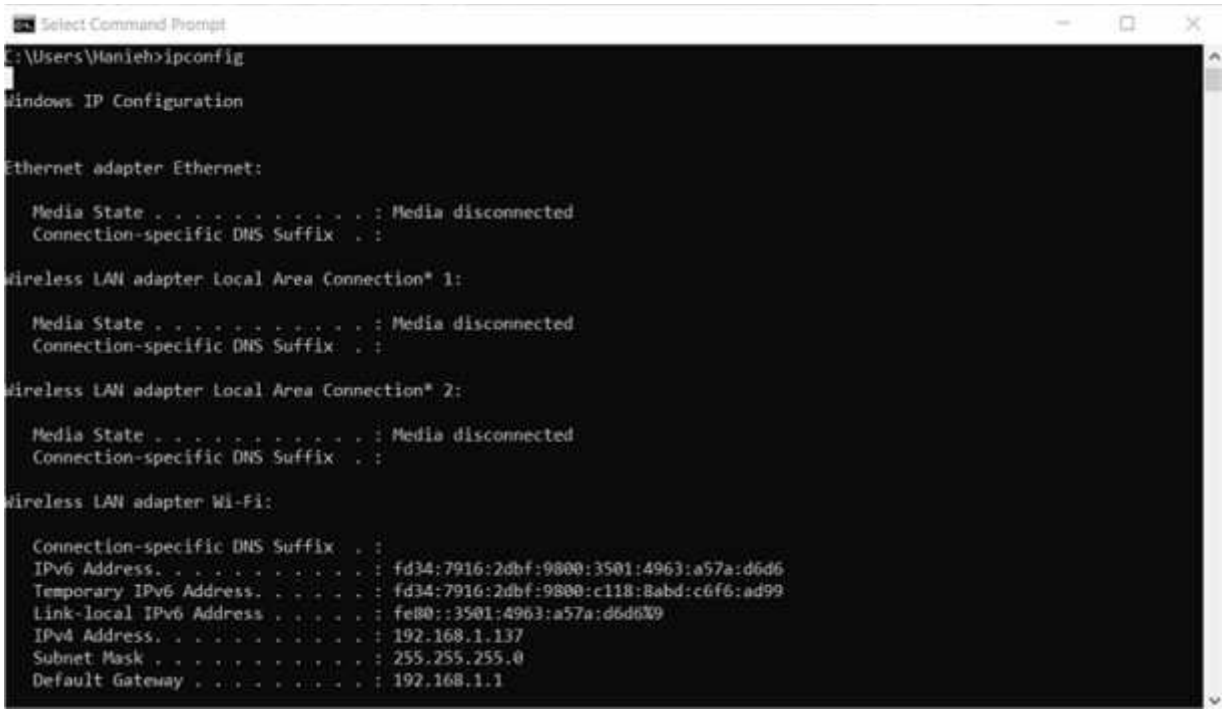
سپس ماژول ESP را به WiFi مورد نظر متصل می کنیم، برای این کار از دستور WiFi.begin(SSID, pass) استفاده می کنیم و برای نمایش برقراری اتصال از دستور Serial.print() استفاده کرده و Connecting را چاپ می کنیم.

در ادامه بررسی می کنیم که آیا مقدار WiFi.status() مساوی مقدار از پیش تعریف شده WL_CONNECTED شده است یا نه؟ و تا مادامی که این شرط برقرار نشده باشد با تاخیر ۵۰۰ میلی ثانیه ای، یک نقطه چاپ می کنیم و به محض برقرار شدن شرط، عبارت Connected را به همراه آدرس IP محلی اختصاص داده شده به آن نمایش می دهیم.

یک متغیر host که از نوع const char* مساوی مقدار IP کامپیوتر است و متغیر دیگری از نوع const uint_16 به اسم port و مقدار ۳۰۰۰ تعریف می کنیم.

پیدا کردن IP کامپیوتر

برای این که متوجه شویم IP کامپیوتر چند است در محیط cmd از دستور ipconfig استفاده می کنیم.



تعریف داده host از نوع const char* به این دلیل است که آن را یک string مخصوص C++ تعریف کرده باشیم.

در ابتدای بخش loop، در ترمینال سریال نمایش می دهیم که ماژول ESP به کدام host و با چه مقدار port متصل شده است.

کلاس WiFiClient در کد کنترل لامپ با ESP8266

در ادامه از کلاس WiFiClient استفاده می کنیم و یک object از آن با نام Client تعریف می کنیم. توجه داشته باشید که اسامی مورد استفاده برای object ها دلخواه هستند و لزومی در استفاده از یک اسم خاص نیست.

یکی از ابزارهایی که این کلاس در اختیار ما قرار می دهد، دستور وصل شدن به WiFi است که به صورت زیر نوشته می شود:

client.connect(host, port)

نوع ورودی های این تابع، string برای host و uint_16 برای port است.

این تابع بعد از صدا زده شدن مقدار true یا false برمی گرداند به این معنی که اتصال برقرار شده یا نشده است.

برای بررسی برقراری اتصال، نقیض این تابع را با دستور شرطی if بررسی می کنیم و در صورت عدم برقراری اتصال عبارت "connection failed" و

”waiting 5 sec...“ را نمایش داده و سپس تاخیر ۵۰۰۰ میلی ثانیه ای اعمال می کنیم تا زمان برای امتحان دوباره اتصال فراهم باشد.

در انتهای این بخش از دستور return استفاده می کنیم.

کاربرد دستور return

هرجایی از تابع return را بنویسیم، به محض رسیدن کامپایلر به آن، به خط اول تابع میرود و هرچیزی پایین آن نوشته شود اجرا نخواهد شد.

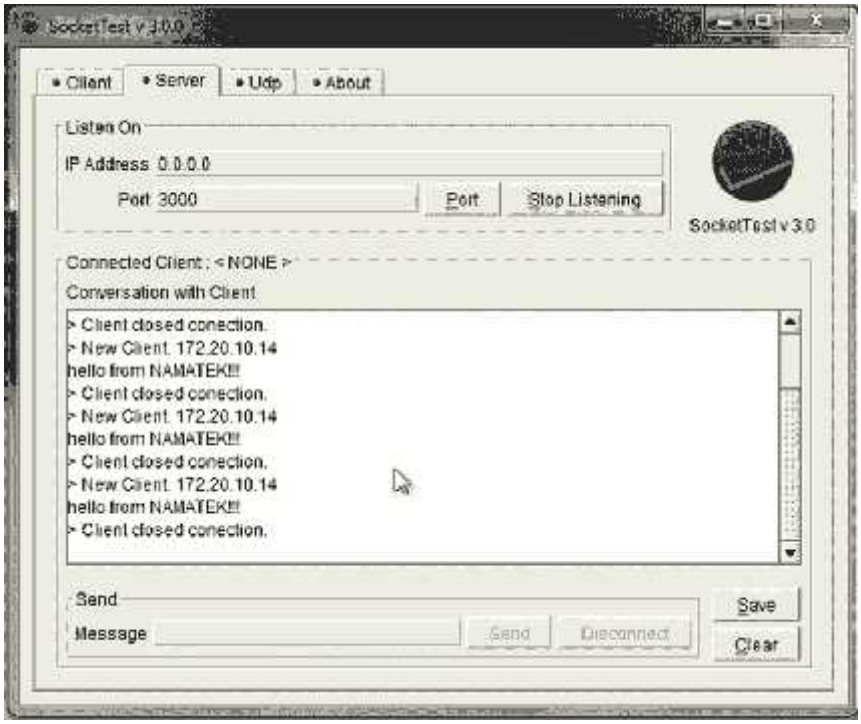
ادامه کد

در بیرون از if عبارت ”connected to host“ را نمایش می دهیم چون خروج از حلقه if به معنی برقراری اتصال است. بعد از متصل شدن روی سرور عبارت های مدنظرمان را می نویسیم برای این منظور از دستور client.println استفاده می کنیم.

این دستور عینا مشابه Serial.println عمل می کند و به جای نمایش داده ها روی ترمینال سریال، عبارت نوشته شده را روی TCP می فرستد.

استفاده از نرم افزار SocketTest

بر روی کامپیوتر نرم افزار SocketTest را باز کرده و IP Address را روی مقدار ۰/۰/۰/۰ و Port را روی مقدار ۳۰۰۰ تنظیم می کنیم و سپس start listening می کنیم.



در حین اجرای برنامه درون حلقه ممکن است هر از چندگاهی با مشکل اتصال رو به رو شده و پیغام connection failed نمایش داده شود. این مشکل به این دلیل پیش آمده است که بعد از هربار ارسال دیتا، ارتباط را غیرفعال نکرده ایم و به همین دلیل گاهی اوقات در وقفه ۵ ثانیه ای نوشته شده دوباره درخواست برقراری می دهد و باعث می شود که خط مشغول به نظر رسیده و ارتباط failed شود. برای رفع این مشکل کافی است در

خط نهایی از دستور client.stop استفاده کنیم و بعد از آن ۵ ثانیه تاخیر قرار می دهیم.

دریافت داده از کامپیوتر برای کنترل لامپ با ESP8266

دستور client.connected نمایش می دهد که آیا اتصالی برقرار است یا نه؟ دستور client.available برای این منظور استفاده می شود که نشان دهد آیا عبارتی برای خواندن در بافر وجود دارد یا خیر؟

در ادامه کد می خواهیم بررسی کنیم که آیا چیزی برای خواندن در بافر TCP موجود است؟ برای این منظور یک حلقه while روی OR دو شرط client.available و client.connected می نویسیم. درون این while می خواهیم به حرفی که سرور (کامپیوتر) می زند، گوش دهیم.

در ابتدا با دستور if بررسی می کنیم که داده ای برای خواندن هست یا نه؟ دستور readStringUntil برای خواندن داده تا یک جای مشخص از یک بافر استفاده می شود.

یک متغیر از نوع string با اسم line تعریف می کنیم و مساوی مقدار readStringUntil با ورودی 'r\' قرار می دهیم و در ترمینال سریال آن را نمایش می دهیم.

برای تست کردن این قسمت از کد کافیست در بخش message نرم افزار Socket، یک عبارت را نوشته و ارسال کنیم.

مشکل ایجاد شده در روند اجرای کد و رفع آن

در این حالت با یک مشکل برخورد می کنیم که بین هر عبارت نوشته شده و بعدی آن دو خط فاصله می افتد. علت این اتفاق این است که دکمه send را که می زنیم عبارت نوشته شده با r\n\ ارسال می شود و ما هم تا r\ آن را خواندیم و n\ درون بافر باقی مانده و بعد از سپری شدن یک زمان مشخصی آن هم ارسال می شود. برای رفع این مساله یک client.read بعد از تعریف مقدار line می نویسیم ولی با آن هیچ کاری نمی کنیم.

روشن و خاموش کردن لامپ با استفاده از ESP8266

با مقدار متغیر line کارهای مختلفی میتوان انجام داد، برای مثال اگر مقدار این متغیر ON بود عبارت turning on the lamp را نمایش داده و یا شرط مقدار OFF را بررسی کرد.

قطعات مورد نیاز مدار

- ترانزیستور BC338
- رله
- مقاومت K۱
- لامپ
- ماژول ESP8266

بستن مدار لامپ و ESP8266

با استفاده از یک ترانزیستور BC338 را طوری قرار می دهیم که کلکتور آن به یک پایه بوبین رله وصل شود و سر دیگر بوبین را به ولتاژ ورودی متصل می کنیم. بیس با یک مقاومت ۱ kبه یکی از پایه های GPIO مثل D0 و امیتر را هم به GND وصل می کنیم.

ادامه کد نوشته شده برای کنترل لامپ

در setup با استفاده از دستور digitalWrite، پین D0 را خروجی تعریف کرده و مقدار آن را هنگام بررسی شرط ON متغیر، HIGH کرده و در بررسی شرط OFF مقدار LOW به آن می دهیم. در نهایت با پروگرم کردن کد، با نوشتن مقدار ON در کادر message لامپ را روشن کرده و با نوشتن OFFآن را خاموش کنیم.

به راحتی می توان این کد را تعمیم داد و به جای یک لامپ تعداد زیادی لامپ را فعال کرد. در این پروژه برقراری ارتباط کاملا از طریق WiFi بوده و هیچ ارتباطی بین مدار بسته شده و کامپیوتر از طریق سیم کشی ها وجود ندارد. برای اطمینان می توانید به جای اتصال برد به پورت USB لپتاپ از یک پاور بانک استفاده کنید.