



دانشگاه صنعتی شریف  
دانشکده‌ی مهندسی کامپیوتر

پروژه درس  
آزمایشگاه سخت افزار

عنوان:

## کلید هوشمند با قابلیت دریافت فرمان صوتی

نگارش:

سید علی مرعشیان سرائی - ۹۷۱۰۲۴۴۱

محمدجواد حمزه - ۹۷۱۰۱۵۵۳

آرین اعتمادی حقیقی - ۹۷۱۱۰۰۰۳

گروه ۹

استاد:

دکتر اجلالی

نیم سال دوم ۱۴۰۱ - ۱۴۰۰

سلام

## چکیده

در دهه‌های اخیر شاهد رشد روزافزون شبکه اینترنت در تمام زمینه‌های زندگی بشر بودیم. این تغییرات محدود به ارتباط با بکدیگر و یا کنترل پروژه‌های عظیم نبوده، و به کنترل اشیا روزمره که با آنها سر و کار داریم نیز راه پیدا کرده است. در این پروژه، ما قصد داریم که دو لامپ را که توسط یک کلید دو پل کنترل می‌شوند، به صورت هوشمند کنترل کنیم. این کنترل کردن در بستر وب انجام می‌شود، می‌توان به لامپ‌های فرمان‌های خاص صوتی داد و یا برنامه‌ای برای روشن و خاموش شدنشان ریخت. کاربران به این صورت نیاز به حضور فیزیکی در محیط برای کنترل لامپ‌ها ندارند.

**کلیدواژه‌ها:** لامپ هوشمند، اینترنت اشیا، خانه هوشمند

# فهرست مطالب

۹	۱ مقدمه
۹	۱-۱ تعریف مسئله
۱۰	۲-۱ اهمیت مسئله
۱۱	۲ معماری سیستم
۱۱	۱-۲ بلوک دیاگرام سیستم
۱۲	۲-۲ میکروکنترلر
۱۲	۳-۲ کلیدها
۱۲	۴-۲ رله
۱۳	۵-۲ لامپ
۱۴	۳ مشخصات سخت افزاری
۱۴	۱-۳ ماژول NodeMCU
۱۴	۱-۱-۳ مشخصات تکنیکی
۱۷	۴ رابط کاربری
۱۷	۱-۴ وبسایت
۱۷	۱-۱-۴ کنترل از راه دور

۱۸	۲-۱-۴ برنامه ریزی
۱۸	۳-۱-۴ برنامه
۱۹	۴-۱-۴ استفاده از صوت
۲۱	۵ پیاده‌سازی نرم افزار
۲۱	۱-۵ راه‌اندازی ماژول NodeMCU
۲۲	۲-۵ برنامه نویسی ماژول NodeMCU
۲۲	۱-۲-۵ کتابخانه‌های استفاده شده
۲۳	۲-۲-۵ ساختار کلی کد
۲۴	۳-۵ راه‌اندازی سرور
۲۴	۱-۳-۵ ارتباط ماژول NodeMCU با سرور
۲۵	۴-۵ تشخیص صوت
۲۸	۶ بسته‌بندی
۲۸	۱-۶ کلید دویل
۲۹	۲-۶ باتری
۳۱	۳-۶ مدار نهایی
۳۲	۷ نتیجه‌گیری
۳۲	۱-۷ مرور کلی
۳۳	۲-۷ هزینه‌های صورت گرفته

## فهرست شکل‌ها

۱-۲	بلوک دیاگرام سیستم	۱۲
۲-۲	معماری سیستم	۱۳
۱-۳	ماژول NodeMCU	۱۵
۲-۳	پین‌های ماژول NodeMCU	۱۶
۳-۳	یک رله	۱۶
۱-۴	قسمت‌های مختلف صفحه اصلی وبسایت	۱۷
۲-۴	قسمت کنترل از راه دور	۱۸
۳-۴	قسمت زمان بندی لامپ‌ها	۱۹
۴-۴	برنامه زمان بندی لامپ‌ها	۱۹
۵-۴	قسمت استفاده از صوت	۲۰
۱-۵	نصب پکیج مخصوص ESP8266	۲۲
۲-۵	حلقه اصلی برنامه	۲۳
۳-۵	بخشی از کد فایل views.py	۲۵
۴-۵	نمونه ای از ارتباط ماژول با سرور	۲۶
۵-۵	نمونه دیگری از کد ارتباط ماژول با سرور	۲۶

۶-۵	بخشی از کد تشخیص صوت سمت سرور	۲۷
۱-۶	کلید روکار	۲۹
۲-۶	کلید توکار	۳۰
۳-۶	باتری LIR2450	۳۰
۴-۶	مدار با لامپ‌های خاموش	۳۱
۵-۶	مدار با لامپ‌های روشن	۳۱

## فهرست جدول‌ها

۱-۷ هزینه قطعات پروژه	۳۳
-----------------------	----



# فصل ۱

## مقدمه

### ۱-۱ تعریف مسئله

بعد از انقلاب صنعتی، دنیا وارد مرحله جدیدی از تاریخ شد. انسان‌ها بیشتر و بیشتر به سمت خودکارسازی فعالیت‌ها پیش رفتند. این تغییرات ابتدا در ابعاد بزرگ‌تر مانند تولیدات کارخانه‌ای خود را نشان داد و کم‌کم به محصولات شخصی‌تری راه یافت: کامپیوترهای شخصی، تلفن و ساعت‌های هوشمند که ابعادشان محدودتر و عملکردشان خاص‌منظوره‌تر است. با خودکارسازی و اتوماسیون، نیروی انسانی کمتر شده و در نتیجه خطای انسانی نیز کمتر می‌شود. گسترش شبکه‌ی اینترنت در چند دهه اخیر بستر آن را فراهم کرده که گستره این اتوماسیون به کوچکترین بخش‌های زندگی روزمره ما وارد شود، مخصوصاً با انقلابی که اینترنت اشیا<sup>۱</sup> در حوزه محصولات خانگی به پا کرده است. با به کارگیری بسترهای موجود و پیشرفت در حوزه اینترنت اشیا، این پروژه را برای اتوماسیون فرآیند کنترل لامپ‌ها تعریف می‌کنیم. به طور خاص، در این پروژه می‌خواهیم یک کلید هوشمند با قابلیت دریافت فرمان صوتی طراحی کنیم. این کلید قرار است بدون ایجاد اختلال در عملکرد یک کلید دوپل معمولی درون چنین کلیدی قرار گیرد. کنترل این کلید هوشمند از طریق یک سرور بر روی بستر وب صورت می‌گیرد. در این طرح می‌توان علاوه بر دادن فرمان صوتی به سرور، سناریوهای مختلفی را برای زمان بندی کنترل وضعیت لامپ‌ها در نظر گرفت، بدون اینکه کاربر بخواهد از طریق تغییر وضعیت کلید به صورت فیزیکی این کار را انجام دهد.

---

<sup>1</sup>Internet of Things (IoT)

## ۱-۲ اهمیت مسئله

احتمالا بسیار این اتفاق برای شما افتاده باشد که بعد از خروج از خانه یادتان بیافتد که لامپها را خاموش نکرده اید. این که کاربر بتواند اسباب خانه را در هر مکان و هر زمانی کنترل کند به وضوح انعطاف پذیری زیادی به او می دهد.

از طرفی این قابلیت ها زمینه استفاده را برای کم توانان و ناتوانان نیز فراهم می کند تا بتوانند بدون نیاز به کمک فرد دیگری از لامپها استفاده کنند. اینکه فرمان های صوتی پردازش می شوند نیز این امر را ساده تر می کند.

## فصل ۲

# معماری سیستم

در این بخش توضیحی اجمالی در مورد نحوه اتصال قطعات مختلف به کار رفته در محصول و سیم کشی آنها می دهیم.

ابتدا توجه کنید که چهار ماجول یا قطعه اصلی در طرح ما وجود دارد:

- میکروکنترلر NodeMCU

- کلید (دو پل)

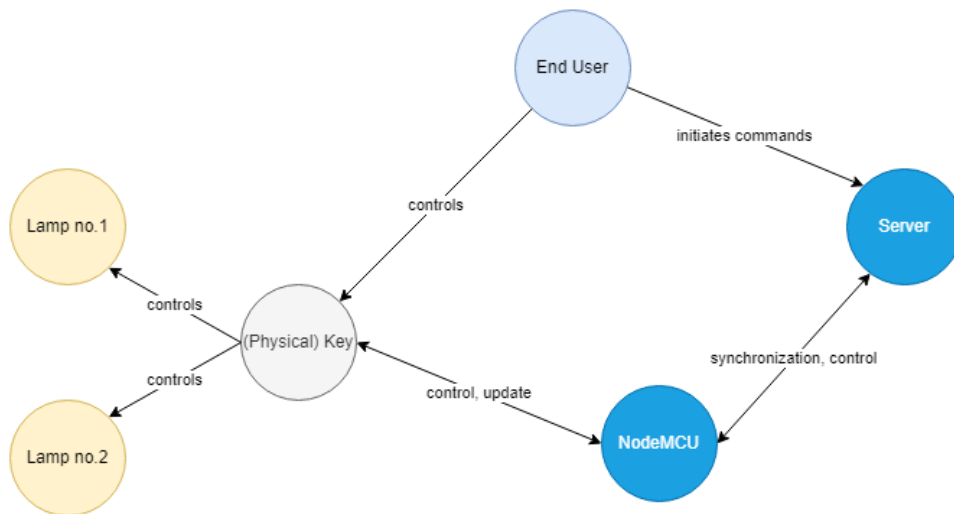
- رله (دو عدد)

- لامپ (دو عدد)

نحوه سیم کشی و طرح کلی اتصالات در سیستم را می توانید در شکل ۲-۲ مشاهده کنید.

## ۱-۲ بلوک دیاگرام سیستم

در شکل ۱-۲ بلوک دیاگرام سیستم را مشاهده می کنید.



شکل ۲-۱: بلوک دیاگرام سیستم

## ۲-۲ میکروکنترلر

میکروکنترلر NodeMCU هسته محاسباتی سیستم است و در مجموع از ۷ پین آن استفاده می‌شود. دو پین به کلیدها، دو پین برای کنترل رله‌ها، و سه پین استفاده شده پین‌های تغذیه و زمین هستند.

## ۳-۲ کلیدها

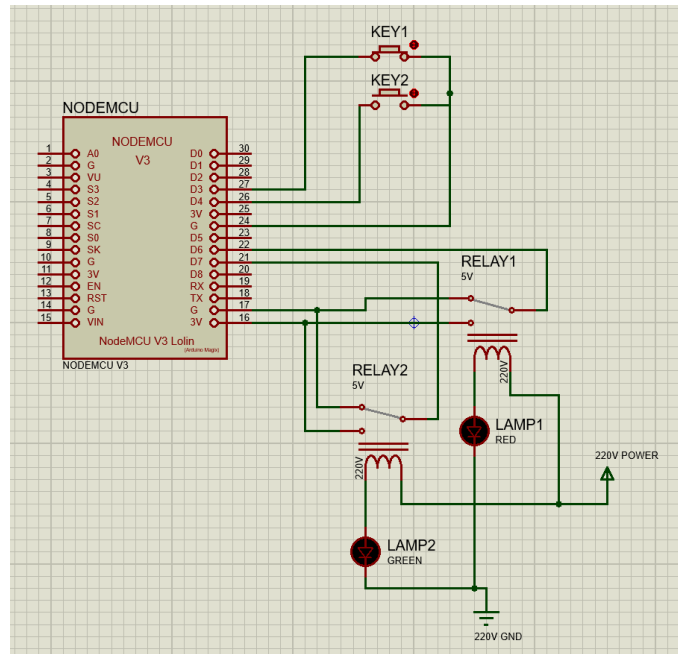
همان طور که می‌بینید، کلیدها به دو ورودی  $D^3$  و  $D^4$  میکروکنترلر وصل هستند. این دو ورودی در حالت عادی مقدار HIGH دارند و پس از اتصال کلید، به پین G میکروکنترلر وصل شده و مقدار صفر پیدا می‌کنند. بدین ترتیب میکروکنترلر می‌تواند از فشرده شدن کلید اطلاع پیدا کند.

## ۴-۲ رله

رله‌ها از یک سمت با جریان ۳.۳ ولتی که از میکروکنترلر می‌آید، کنترل می‌شوند و از سمت دیگر با جریان ۲۲۰V در مسیر سیم‌های لامپ‌ها قرار گرفته‌اند تا اینگونه روشن و خاموش کردن لامپ‌ها ممکن شود.

## ۵-۲ لامپ

لامپ‌ها از یک سمت به یک خروجی رله، و از سمت دیگر به زمین متصل هستند.



شکل ۲-۲: معماری سیستم

## فصل ۳

# مشخصات سخت افزاری

### ۱-۳ ماژول NodeMCU

#### ۱-۱-۳ مشخصات تکنیکی

ماژولی که در این پروژه استفاده کردیم مشخصات زیر را دارد:

- ESP-8266 32-bit :Microcontroller

- 58mm x 32mm :Size

- فاصله پین ها: 27.94mm

- فرکانس کلاک: 80 MHz

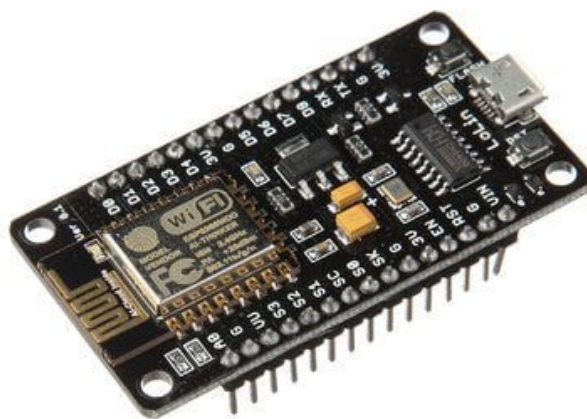
- ولتاژ کاری: 3.3V

- محدوده ولتاژ ورودی: 4.5V-10V

- حافظه: 4MB

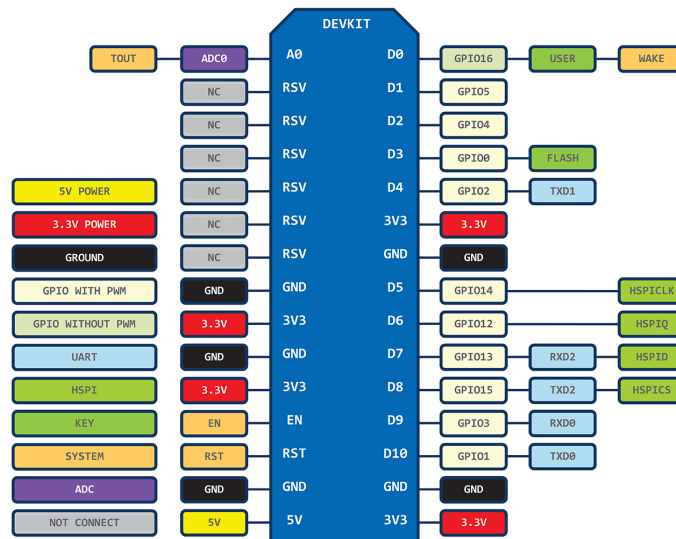
شکل ۱-۳ ماژول را نشان می دهد. این ماژول ۱۱ تا پین ورودی/خروجی دیجیتال دارد. پین های آن

در شکل ۲-۳ مشخص شده است. همچنین، به دو رله نیاز خواهیم داشت تا مانند کلید مکانیکی عمل کنند. شکل ۳-۳ یک رله را نشان می دهد.



شکل ۳-۱: ماژول NodeMCU

در بخش بسته بندی به نحوه کنار هم قرار گرفتن اجزای مختلف اشاره خواهیم کرد.



شکل ۳-۲: پین‌های ماژول NodeMCU



شکل ۳-۳: یک رله



## فصل ۴

# رابط کاربری

### ۱-۴ وبسایت

صفحه اصلی وبسایت در URL به نشانی <https://my-smartlamp.fandogh.cloud/control/> قرار دارد که تصویر آن در شکل ۴-۱ قابل مشاهده است. پس از مراجعه به این آدرس، با صفحه‌ای مواجه می‌شویم که دارای چهار بخش است:

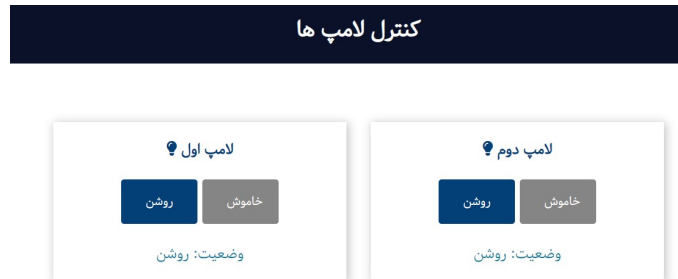


شکل ۴-۱: قسمت‌های مختلف صفحه اصلی وبسایت

#### ۱-۱-۴ کنترل از راه دور

این بخش در قسمت بالای سمت چپ صفحه اصلی وبسایت قرار دارد. در این بخش، کاربر با کلیک بر روی دکمه واحد کنترل، وارد صفحه جدیدی می‌شود که تصویر آن در شکل ۴-۲ قابل مشاهده است.

در این صفحه، کاربر می‌تواند وضعیت هریک از دو لامپ موجود را با کلیک بر روی دکمه روشن یا خاموش به صورت دستی کنترل کند.



شکل ۴-۲: قسمت کنترل از راه دور

#### ۴-۱-۲ برنامه ریزی

این بخش در قسمت بالای سمت راست صفحه اصلی وبسایت قرار دارد. در این بخش کاربر با کلیک بر روی دکمه انتخاب وقت، وارد صفحه جدیدی می‌شود که تصویر آن در شکل ۴-۳ قابل مشاهده است. در این صفحه، کاربر ابتدا با کلیک بر روی تقویم موجود در جلوی برچسب تاریخ و زمان، تاریخ میلادی مدنظر را به همراه ساعت مدنظر برای انجام شدن تغییر در آن تاریخ، با یک فرمت ۱۲ ساعته که AM قبل از ظهر و PM بعد از ظهر را تعیین می‌کند، مشخص کنیم و در فیلد لامپ، لامپی که می‌خواهیم وضعیتش را تعیین کنیم (لامپ ۱ یا لامپ ۲) و در فیلد وضعیت، وضعیت آن لامپ (روشن یا خاموش) آن را تعیین می‌کنیم. در صورتی که بخواهیم این وضعیت به صورت هفتگی تکرار شود - یعنی مثلاً روزی که تعیین کردیم شنبه است و می‌خواهیم از آن تاریخ به بعد در روزهای شنبه این کنترل وضعیت تکرار شود - تیک تکرار هفتگی را می‌زنیم.

#### ۴-۱-۳ برنامه

این بخش در قسمت پایین سمت چپ صفحه اصلی وبسایت قرار دارد. در این بخش می‌توانیم با کلیک بر روی صفحه وارد صفحه جدیدی شویم که در آن تمامی زمان‌بندی‌هایی را که در قسمت برنامه ریزی تنظیم کرده بودیم را به ما در قالب یک جدول نشان می‌دهد. تصویری از این صفحه در شکل ۴-۴ قابل مشاهده است. این جدول دارای ستون‌های زمان، لامپ، وضعیت، تکرار هفتگی می‌باشد که به ترتیب زمان تعیین شده برای تغییر وضعیت، لامپی که قرار است وضعیت آن تغییر یابد، وضعیت آن لامپ

## زمان بندی لامپ ها

🔗 زمان بندی لامپ ها

تاریخ و زمان:  ☐ تکرار هفتگی ☐

لامپ:

وضعیت:

شکل ۴-۳: قسمت زمان بندی لامپ ها

(اینکه قرار است روشن شود یا خاموش) و اینکه قرار است هر هفته در آن روز و ساعت آن تغییر به طور مکرر رخ دهد را نشان می دهد. توجه داریم که در ستون وضعیت، True به معنای روشن و False به معنای خاموش شدن لامپ در زمان تعیین شده و در ستون تکرار هفتگی، True به معنای رخ دادن تکرار هفتگی و False به معنای رخ ندادن آن است.

## زمان بندی لامپ ها

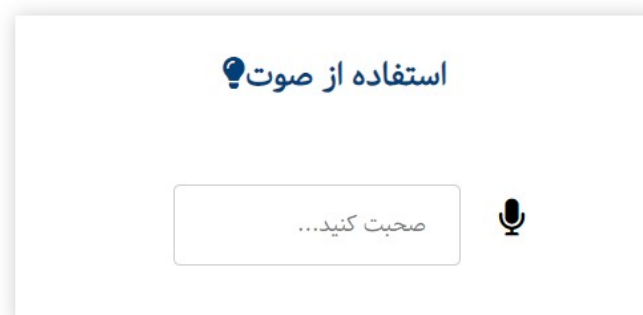
🔗 برنامه تنظیم شده

زمان	لامپ	وضعیت	تکرار هفتگی
May 29, 2022, 10:43 a.m	1	True	False
			<a href="#">delete</a>

شکل ۴-۴: برنامه زمان بندی لامپ ها

## ۴-۱-۴ استفاده از صوت

این بخش در قسمت پایین سمت راست صفحه اصلی وبسایت قرار دارد که تصویر آن در شکل ۴-۵ قابل مشاهده است. در این قسمت کاربر در ابتدا بر روی آیکون میکروفون موجود در سمت راست فیلد متن خروجی صوت کلیک می کند تا میکروفن برای او فعال شود، سپس می تواند دستور مدنظرش (مثلاً لامپ یک روشن یا هر دو خاموش) را به صورت صوتی بیان کند تا از طریق API تشخیص صوت دستور تشخیص داده شود و در فیلد موجود در این قسمت پردازش شود تا پس از انجام شدن یک سری پردازش ها، در نهایت دستور صوتی داده شده بر روی لامپ های مدنظر کاربر اعمال شود.



شکل ۴-۵: قسمت استفاده از صوت

## فصل ۵

# پیاده‌سازی نرم افزار

### ۵-۱ راه‌اندازی ماژول NodeMCU

به جای اینکه از زبان Lua برای نوشتن کد استفاده کنیم، از نرم افزار Arduino IDE (به طور خاص نسخه 1.8.19) استفاده می‌کنیم. و به این ترتیب، همان کد آردوینو برای ماژول کار خواهد کرد. البته که به پکیج دیگری برای این انطباق نیاز خواهد بود. برای اینکه با پورت USB به ماژول NodeMCU متصل شویم، به پورت سریال نیاز داریم. نسخه ما از سریال چیپ CH340 استفاده می‌کند. درایور مخصوص به آن را از [اینجا](#)<sup>۱</sup> دانلود و نصب می‌کنیم. در ادامه، در محیط کاری Arduino IDE به File، Preferences، Settings می‌رویم و در قسمت Addtional Board Manager URLs عبارت زیر را وارد می‌کنیم:

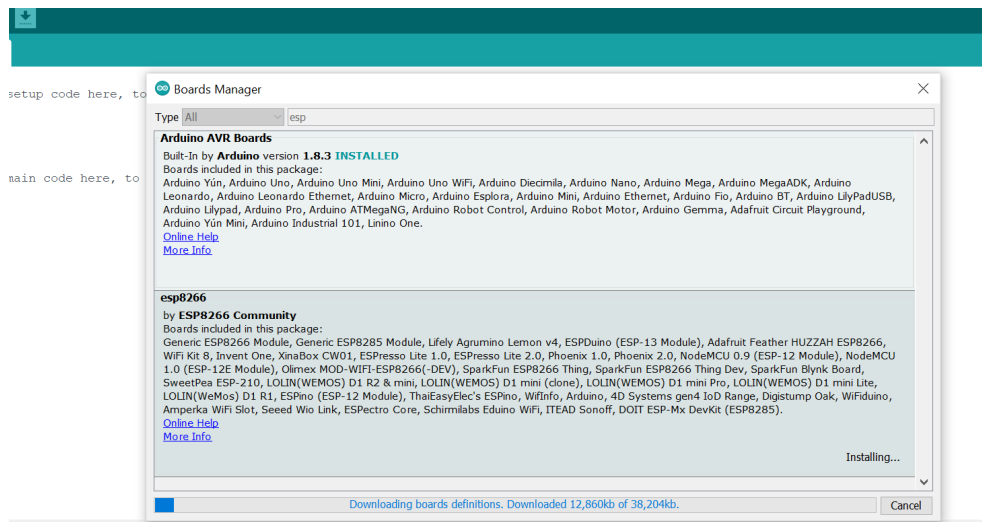
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

حال، OK را انتخاب می‌کنیم. بعد به Tools، Board (Arduino ...)، Board Manager رفته و esp8266 by ESP8266 Community را دانلود و نصب می‌کنیم. شکل ۵-۱ این فرآیند را نشان می‌دهد.

بعد از نصب پکیج، از قسمت Tools پورت سریال مورد نظر را که ماژول با آن به کامپیوتر وصل

---

<sup>1</sup><https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers>



شکل ۵-۱: نصب پکیج مخصوص ESP8266

شده است را انتخاب می‌کنیم. حال می‌توانیم هر تکه کد آردوینویی را روی برد آپلود کنیم. کد مورد نظر که باید روی برد آپلود شود در **ریپوی پروژه** قابل دسترسی است.

## ۲-۵ برنامه نویسی مازول NodeMCU

برنامه‌ی کامل نوشته شده برای میکروکنترلر NodeMCU را می‌توانید در صفحه گیتهاب پروژه، در بخش کدها، در بخش نسخه نهایی مشاهده کنید. در اینجا به توضیحی اجمالی بسنده خواهیم کرد.

### ۱-۲-۵ کتابخانه‌های استفاده شده

در برنامه نوشته شده از تابع‌های موجود در چهار کتابخانه کمک گرفته ایم.

- کتابخانه‌های ESP8266WiFi، WiFiClient، و ESP8266HTTPClient: این سه کتابخانه ما را در اتصال دستگاه به اینترنت یاری می‌کنند.
- کتابخانه LittleFS: این کتابخانه به ما ابزاری پایه‌ای و ساده برای مدیریت فایل‌ها بر روی حافظه داخلی میکروکنترلر می‌دهد. از توابع آن می‌توان برای نوشتن و خواندن اطلاعات زمان‌بندی دریافتی از سرور استفاده کرد.

## ۲-۲-۵ ساختار کلی کد

کد نوشته شده دو تابع اصلی دارد:

۱. تابع `setup`: در این تابع راه‌اندازی‌های اولیه صورت می‌گیرد، مانند اتصال به وای‌فای.
  ۲. تابع `loop`: پس از راه‌اندازی‌های اولیه در تابع قبلی، این تابع متوالیا تا زمان بی‌نهایت از بیرون صدا زده می‌شود.
- توجه کنید که سیستم ما در نهایت یک سیستم بی‌درنگ است و در سیستم‌های بی‌درنگ، تسک‌ها مهم‌ترین مفهوم هستند. در اینجا نیز ما چهار تسک اصلی داریم که باید به صورت دوره‌ای در همان حلقه اصلی اجرا شوند. این چهار وظیفه و بدنه حلقه را می‌توانید در شکل ۲-۵ مشاهده کنید.

```
/** execute four periodic tasks */
void loop() {

    read_lamps_status_from_server();

    read_schedule_from_server();

    monitor_physical_keys();

    put_schedule_in_action();

}
```

شکل ۲-۵: حلقه اصلی برنامه

در ادامه هر کدام از این چهار تسک را به طور مختصر توضیح می‌دهیم:

- خواندن وضعیت لامپ‌ها از سرور: کاربر می‌تواند از وب‌سایت سیستم لامپ‌ها را خاموش/روشن کند. در نتیجه، سیستم باید بتواند مرتباً وضعیت لامپ‌ها را از سرور وب‌سایت جویا شده و تغییرات لازم را بر روی لامپ‌ها اعمال کند. دوره‌ی اجرای این تسک یک ثانیه است.
- خواندن زمان‌بندی از سرور: کاربران می‌توانند در وب‌سایت سیستم خاموش/روشن شدن لامپ‌های خود را برنامه‌ریزی کنند. در نتیجه، دستگاه باید هر از گاهی، این اطلاعات را از سرور دریافت

کند. دوره‌ی اجرای این تسک ۳۰ ثانیه است.

- مشاهده دائم وضعیت کلیدها: کاربر ممکن است کلیدها را فشار داده و به صورت فیزیکی (عادی) وضعیت لامپ‌ها را بخواهد تغییر دهد. در نتیجه، دستگاه باید چندین بار در هر ثانیه تغییر وضعیت در کلیدها را رصد کند. دوره‌ی اجرای این تسک، ۱۵ ثانیه است.
- اجرای زمان‌بندی دریافت شده از سرور: دستگاه باید هر دقیقه زمان‌بندی‌های دریافت شده از سرور را بررسی کرده، و هر کدام که زمانش رسیده بود را اجرا کند. این تسک هر ثانیه ۱۰ بار اجرا می‌شود.

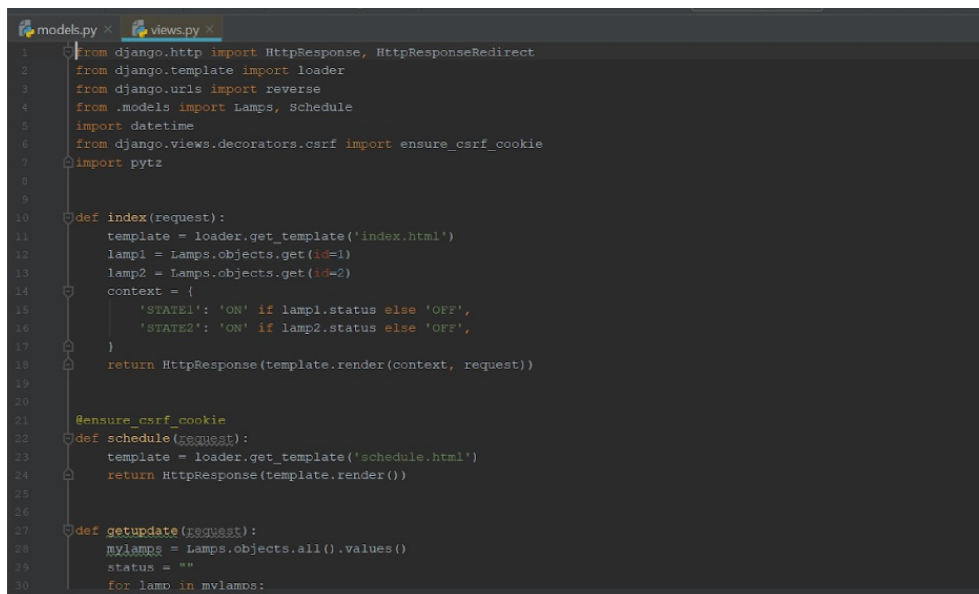
### ۳-۵ راه‌اندازی سرور

برای این بخش باید کمی Django یاد می‌گرفتیم. راه‌اندازی اولیه برای جنگو سمت سرور را از اینجا یاد گرفتیم، همچنین به دانش اولیه‌ی ای از HTML و CSS نیاز داشتیم. روی سرویس **فندق** سرورمان را بالا می‌آوریم. نام فضا را smartlamp انتخاب کردیم و نام پروژه را my گذاشتیم. به این ترتیب سرور در آدرس <http://my-smartlamp.fandogh.cloud/control> قابل دسترسی خواهد بود. همان‌طور که در ادامه توضیح می‌دهیم، امکان آن وجود دارد که برای درخواست‌های خاصی، جواب‌های خاصی فرستاد و به این ترتیب خود ماژول NodeMCU به سرور وصل شده، اطلاعات لازم را دریافت می‌کند. حال با تغییر وضعیت، و یا برنامه‌ریزی، لامپ‌ها نیز روی برد مورد تغییر وضعیت می‌دهند. چند نوع مدل روی سرور تعریف شده، از جمله اطلاعات مربوط به تغییر وضعیت (در مدل Schedule). حال هر وقت که NodeMCU آپدیت زمان‌بندی‌ها را بخواهد، سرور نوع تغییر به همراه میلی‌ثانیه‌های باقی‌مانده تا تغییر را به ماژول اعلام می‌کند. ابتدای کد مربوط به فایل views.py را که بخشی از کد سمت سرور است، در شکل ۳-۵ قابل مشاهده است.

### ۱-۳-۵ ارتباط ماژول NodeMCU با سرور

پس از بالا آوردن سرور، باید کدی در NodeMCU بنویسیم که با سرور ارتباط برقرار کند، ریکوئست‌های لازم را بدهد و داده‌های لازم را دریافت کند. برای این کار، ابتدا باید دو کتابخانه ESP8266HTTPClient و WiFiClient را include کنیم. سپس





```

1 from django.http import HttpResponse, HttpResponseRedirect
2 from django.template import loader
3 from django.urls import reverse
4 from .models import Lamps, Schedule
5 import datetime
6 from django.views.decorators.csrf import ensure_csrf_cookie
7 import pytz
8
9
10 def index(request):
11     template = loader.get_template('index.html')
12     lamp1 = Lamps.objects.get(id=1)
13     lamp2 = Lamps.objects.get(id=2)
14     context = {
15         'STATE1': 'ON' if lamp1.status else 'OFF',
16         'STATE2': 'ON' if lamp2.status else 'OFF',
17     }
18     return HttpResponse(template.render(context, request))
19
20
21 @ensure_csrf_cookie
22 def schedule(request):
23     template = loader.get_template('schedule.html')
24     return HttpResponse(template.render())
25
26
27 def getupdate(request):
28     mylamps = Lamps.objects.all().values()
29     status = ""
30     for lamp in mylamps:

```

شکل ۵-۳: بخشی از کد فایل views.py

با استفاده از قطعه کدی مانند کد موجود در شکل ۵-۴ می‌توان یک ریکوئست HTTP GET به سرور فرستاد.

در ادامه، تنها لازم است که به صورت متناوب، برای مثال هر یک ثانیه، یک ریکوئست مانند بالا به سرور بدهیم و وضعیت لامپ‌ها را از سرور بخوانیم و با توجه به آن، لامپ‌ها را خاموش/روشن کنیم. دقت کنید که ماژول می‌تواند ریکوئست‌های مختلفی به سرور بدهد و سرور نیز چون به ماژول جواب می‌دهد، لزومی ندارد که جوابش در قالب اسکلت HTTP باشد، کافی است فرمت جواب از قبل توسط دو طرف تعیین شده باشد. برای مثال ممکن است جواب یک رشته باشد که اطلاعات مورد نیاز ماژول در آن به فرمت خاصی نوشته شده باشد. این ارتباط فقط به گرفتن اطلاعات از سرور خلاصه نمی‌شود، گاهی لازم داریم وضعیت لامپ‌ها را به سرور اطلاع دهیم. در این مواقع باید با پروتکل POST با سرور ارتباط برقرار کنیم. بخشی از کد نوشته شده به این منظور را در شکل ۵-۵ مشاهده می‌کنید.

## ۴-۵ تشخیص صوت

برای کنترل لامپ‌ها با استفاده از فرمان‌های صوتی، نیاز به پیاده‌سازی یک روش تشخیص صوت (Speech Recognition) است. مروری بر روش‌های تشخیص صوت برتر که به صورت بی‌درنگ

```

HTTPClient http; // object of the class HTTPClient.
http.begin(wifiClient,
    "http://my-smartlamp.fandogh.cloud/control/"); // request dest
int httpCode = http.GET(); // send the request.
if (httpCode > 0) { // check the returning code
    String payload = http.getString(); // get the text from server
    Serial.println(payload); // print the text.
} else {
    Serial.println("HTTP Connection Failed!");
}
http.end(); // close connection

```

شکل ۵-۴: نمونه ای از ارتباط مازول با سرور

```

HTTPClient http;
http.begin(wifiClient,
    "http://my-smartlamp.fandogh.cloud/control/report/");
http.addHeader("Content-Type", "text/plain");
String lamps_status = String(digitalRead(lamp1)) +
    String(digitalRead(lamp2));
int httpResponseCode = http.POST(lamps_status);
Serial.print("HTTP Response code: ");
Serial.println(httpResponseCode);
String payload = http.getString();
Serial.println(payload);
http.end();

```

شکل ۵-۵: نمونه دیگری از کد ارتباط مازول با سرور

برای استفاده بر روی پردازنده‌های گوشی‌های همراه مناسب هستند، در اینجا<sup>۲</sup> آمده است. این روش‌ها که همه از شبکه‌های عصبی مدرن استفاده می‌کنند، توسط گوگل در یک صفحه جمع آوری شده اند. در نتیجه، یک راه حل ممکن، استفاده از یکی از این شبکه‌ها و یادگیری (Train) آن بر روی داده‌های فارسی است. اما متأسفانه، این کار نیازمند تولید تعداد قابل توجهی داده صوت فارسی برای کلمه‌های به کار رفته در فرمان‌های صوتی ما است که کار بسیار دشواری است و در مقیاس پروژه ما نمی‌گنجد. گرچه خوشبختانه، نیازی به اختراع دوباره چرخ توسط ما نیست و خود گوگل سرویس‌های تشخیص گفتارش را به رایگان در اختیار توسعه‌دهندگان می‌گذارد. این سرویس‌ها هم بهترین کیفیت موجود در بازار را دارند و هم از زبان فارسی پشتیبانی می‌کنند که برای پروژه ما کلیدی است. توجه کنید که سرویس Google Voice Typing هم به هنگام استفاده از مرورگر گوگل یعنی گوگل کروم و هم بر روی سیستم

<sup>2</sup>[https://github.com/google-research/google-research/tree/master/kws\\_streaming](https://github.com/google-research/google-research/tree/master/kws_streaming)

```
def process_voice(request):
    string = request.POST['command']
    if string in first_on_commands or string in both_on_commands or string in all_on_commands:
        change_lamp(1, True)
    if string in second_on_commands or string in both_on_commands or string in all_on_commands:
        change_lamp(2, True)
    if string in first_off_commands or string in both_off_commands or string in all_off_commands:
        change_lamp(1, False)
    if string in second_off_commands or string in both_off_commands or string in all_off_commands:
        change_lamp(2, False)
```

#### شکل ۵-۶: بخشی از کد تشخیص صوت سمت سرور

عامل های اندروید فعال است. اما توجه کنید که ما به دنبال استفاده از این ویژگی در برنامه وب (یا موبایل) خود هستیم. به همین هدف، یک پیاده‌سازی از نحوه به کارگیری **API Speech Web** در برنامه‌های وب انجام دادیم. با استفاده از این API می‌توان بر روی مرورگرهای Google Chrome و Samsung Internet از سرویس تشخیص گفتار گوگل به رایگان استفاده کرد. تنها نیاز است که کاربر اجازه دسترسی صفحه به میکروفون خود را بدهد تا صوت او برای چند ثانیه ضبط شده، به سرورهای گوگل ارسال شده و متن تشخیص داده شده آن به سرور وب سایت ما برگردد. بخشی از کدی که در سرور، برای پردازش دستور مورد نظر اجرا می‌شود را در شکل ۵-۶ مشاهده می‌کنید.

## فصل ۶

### بسته‌بندی

#### ۱-۶ کلید دوپل

در این مورد دو رویکرد می‌توانیم داشته باشیم:

۱. استفاده از کلید دو پل توکار

۲. استفاده از کلید دوپل روکار

در شکل‌های ۱-۶ و ۲-۶ می‌توانید این دو نوع کلید را مشاهده و مقایسه کنید.

تا اواسط مراحل انجام پروژه تصمیم بر آن بود که در بسته‌بندی از کلید توکار استفاده شود. در این صورت، میکروکنترلر، رله‌ها، باتری و سیم‌ها در پشت کلید و در داخل دیوار قرار می‌گیرند که منجر می‌شود ظاهر کلید ما با هر کلید عادی غیروشمند دیگری تفاوت نکند. با وجود این مزیت، به سه دلیل بهتر است که کلید روکار را برای پیاده‌سازی بر توکار ترجیح دهیم:

۱. با قرار گیری قطعات در پشت کلید، عمق کلید بیشتر از حالت عادی می‌شود. در نتیجه، این احتمال وجود دارد که حفره موجود در دیوار کاربر به اندازه کافی عمیق نباشد و کاربر برای نصب کلید ناچار به افزایش عمق حفره و تخریب بیشتر دیوار خود بشود.

۲. در این حالت قطعات سیستم از جمله خود میکروکنترلر در مجاورت حفره دیوار و بدون محافظت



شکل ۶-۱: کلید روکار

قرار می‌گیرند. این موضوع باعث افزایش احتمال آسیب‌دیدگی و کاهش طول عمر سیستم می‌گردد.

۳. نصب کلید روکار برای کاربر آسانتر می‌باشد.

در نتیجه، برای بسته‌بندی از کلید روکار استفاده می‌کنیم. اگر این کلید کمی جادار باشد، به راحتی می‌توان درون آن تمامی قطعات پروژه شامل میکروکنترلر، دو رله، باتری، و سیم‌ها را به خوبی جای داد.

## ۶-۲ باتری

انتخاب‌های مختلفی برای باتری داریم. مثلاً می‌توانیم از باتری لیتیومی  $\text{LiFePO}_4$  استفاده کنیم که برای ماژول NodeMCU ESP8266 جز بهترین انتخاب هاست. ما اینجا طبق توضیحاتی که پیدا کردیم، استفاده از باتری LIR2450 را پیشنهاد می‌کنیم. از آنجا که حداکثر ولتاژ کاری این باتری، ۲.۴ ولت است. (و Nominal Voltage اش ۶.۳ ولت است.) باید از یک regulator استفاده کنیم تا ولتاژ تا پایین بیاوریم. می‌توان از HT7333 linear voltage regulator استفاده کرد. نسبت به regulator های دیگر مانند LM1117 و L78L33C، ولتاژ Dropout کمتری دارد که برای ما مناسب است. در حالت



شکل ۶-۲: کلید توکار

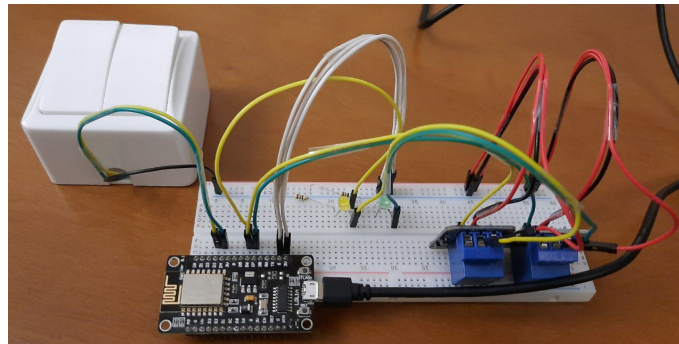
عادی بین ۳ هفته تا یک ماه کار می‌کند. توصیه می‌گردد که طول عمر بیشتری دارد. شکل ۶-۳ این باتری را نشان می‌دهد. اگر فضای کمی بزرگ‌تری داشته باشیم، استفاده از  $\text{LiFePO}_4$  توصیه می‌گردد.



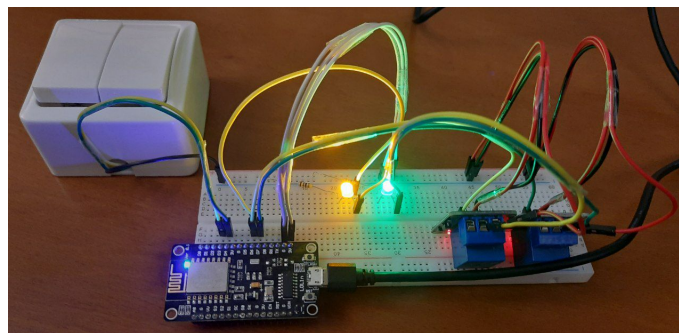
شکل ۶-۳: باتری LIR2450

## ۳-۶ مدار نهایی

تصاویری از مدار نهایی در شکل‌های ۴-۶ و ۵-۶ قابل مشاهده است:



شکل ۴-۶: مدار با لامپ‌های خاموش



شکل ۵-۶: مدار با لامپ‌های روشن

## فصل ۷

### نتیجه گیری

#### ۷-۱ مرور کلی

در این پروژه، هدف نهایی کنترل کردن دو لامپ توسط یک کلید دویپل به صورت هوشمند بر روی بستر وب، به همراه تعدادی قابلیت بود. ابتدا در فصل اول، مسئله را تعریف کردیم و علت اهمیت آن را بیان کردیم. در فصل دوم که فصل معماری سیستم بود، بیان کردیم که برای این پروژه، از میکروکنترلر NodeMCU به عنوان هسته محاسباتی سیستم، به همراه دو عدد کلید، دو عدد لامپ و دو عدد رله بهره گرفتیم که جزئیات آن در فصل مذکور قابل مطالعه است. در فصل سوم مشخصات تکنیکی ماژول NodeMCU را برای آشنایی بیشتر خوانندگان با این ماژول بیان کردیم.

در فصل چهارم، قابلیت‌هایی را که کاربر می‌تواند هنگام استفاده از وبسایت از آن‌ها بهره گیرد، شرح دادیم.

در فصل پنجم ابتدا درباره راه‌اندازی ماژول NodeMCU صحبت کردیم. در ادامه، توضیحاتی مختصر درباره برنامه نویسی ماژول NodeMCU ارائه دادیم. در ادامه مطالبی را درباره راه‌اندازی سرور بیان کردیم. در واقع در این قسمت درباره نحوه بالا آوردن سرور با استفاده از Django، چگونگی ارتباط ماژول NodeMCU با سرور و تشخیص صوت سمت سرور مطالبی را ارائه دادیم. و در انتها، در فصل ششم درباره بسته‌بندی محصول صحبت کردیم و گفتیم که چرا رویکرد روکار را برای کلید دویپل انتخاب کردیم.



## ۲-۷ هزینه‌های صورت گرفته

در این پروژه به بردبورد، تعدادی رشته کابل بردبورد، تعدادی LED، دو ماژول رله تک کانال و یک کابل USB به microUSB نیاز داشتیم که خوشبختانه نیازی به تهیه آن‌ها نبود. هزینه مربوط به سایر قطعات مورد نیاز در جدول ۷-۱ قابل مشاهده است.

نام قطعه	قیمت (تومان)
ماژول NodeMCU	۱۱۹۰۰۰
کلید دو پل	۲۳۰۰۰
تعدادی مقاومت ۱۸ اهم	۲۹۰۰۰
هزینه شارژ سرویس سرور	۱۴۰۰۰۰
مجموع هزینه ها	۳۱۱۰۰۰

جدول ۷-۱: هزینه قطعات پروژه