



لوستر هوشمند

آزمایشگاه سخت افزار

دانشکده مهندسی کامپیوتر

۹۷۱۱۰۲۸۵
۹۷۱۱۰۱۷۷
۹۷۱۰۶۲۱۶
محمد رضا عبدی
حمید رضا کامکاری
یگانه قره داغی

خرداد ۱۴۰۱

چکیده

روند خودکار سازی فعلی خانه‌ها قدمی مهم برای کاهش مصرف بی‌اندازه برق و تسهیل زندگی افراد خانه است. انواع لوسترها و تجهیزات روشناکی از جدیدترین وسایل اضافه شده به لوازم خانگی هوشمند هستند. هدف از این پروژه طراحی یک لوستر هوشمند است که بتواند به کمک یک برنامه به لوازم جانبی هوشمند دیگر (مانند تلفن‌های همراه) متصل شود و توسط آن به صورت خودکار یا دستی تنظیم شود. این دستگاه جزو یک شبکه اینترنت اشیا (Internet of Things) است و با استفاده از یک برنامه موبایل قابل مدیریت است.

در این گزارش به توضیح قابلیت‌ها، محدودیت‌ها، قطعات و جزئیات پیاده‌سازی لوستر هوشمند در ۶ بخش می‌پردازیم.

فهرست مطالب

۴	۱ قابلیت‌های لوستر هوشمند
۵	۲ محدودیت‌های اولیه لوستر هوشمند
۶	۳ قطعات مورد استفاده
۱۱	۴ طراحی مدار
۱۴	RemoteXY ۵
۱۶	۶ برنامه موبایل
۱۸	۷ توضیحات کد

۱ قابلیت‌های لوستر هوشمند

با توجه به اهداف پروژه در خودکار سازی روشنایی خانه^۱، قابلیت‌های زیر برای محصول در نظر گرفته شده‌اند^۲:

- در اولین مرحله این لوستر می‌تواند به کمک سنسوری بر اساس شرایط محیطی مانند وضعیت پرده‌ها یا روشنایی طبیعی بازخورد بدهد؛ در صورت زیاد بودن شدت نور محیط، روشنایی لوستر کاهش و در صورت کم بودن شدت نور افزایش می‌یابد. میزان حساسیت نسبت به روشنایی و میزان روشنایی مورد نیاز می‌تواند بر اساس نیاز کاربر تغییر کند. به عنوان مثال پارامتر اندازه اتاق می‌تواند در تنظیمات تاثیر داده شود. یعنی برای اتاق‌های بزرگتر شدت نور به هنگام روشن بودن بیشتر باشد.
- می‌توان تنظیمات روشنایی لوستر را به صورت خودکار یا دستی تنظیم کرد. در صورت انتخاب حالت دستی، کاربر می‌تواند میزان روشنایی ثابتی را انتخاب کند.
- کاربر می‌تواند شاخه (اتاق‌های مختلف خانه) دلخواه خود را انتخاب کند و تنظیمات هر کدام از قسمت‌ها را به صورت جداگانه انجام دهد.
- کاربر می‌تواند از میان حالت‌های^۳ مختلف ارائه شده برای زیبایی یا رقص نور استفاده نماید.
- تمامی تنظیمات مذکور می‌توانند توسط یه برنامه موبایل سازگار با دستگاه‌های مبتنی بر سیستم عامل‌های مختلف همانند Andriod و IOS در لوستر اعمال شوند.

^۱ کاهش مصرف برق و تسهیل استفاده نسبت به لوسترها مرسوم

^۲ جزئیات قابلیت‌ها در توصیف برنامه موبایل به صورت کامل توضیح داده می‌شود.
^۳ Mode

۲ محدودیت‌های اولیه لوستر هوشمند

در هنگام پیاده‌سازی و استفاده از پروژه با محدودیت‌هایی مواجه می‌شویم که در ادامه آن‌ها را تشریح می‌کنیم:

- چالش اصلی اتصال تعداد زیادی دیود ساطع نور⁴ LED با نورهای متغیر به برد برای شبیه‌سازی یک لوستر واقعی است. در نهایت با اتصال ۴۰ قطعه LED به یک منبع خارجی ۵ ولت و کنترل آن بویسله خروجی PWM و دو ترانزیستور (برای دو شاخه – که این پروژه قابلیت پشتیانی از تعداد شاخه‌های بیشتر را نیز دارد) لوستر را شبیه‌سازی کردیم.
- در برخی موارد، برنامه‌های گوشی‌های هوشمند با اتصال خود به سیستم روشنایی سازگار نیستند. در این پروژه سعی شده‌است که یک برنامه موبایل سازگار با سیستم عامل‌های مانند iOS و Android مختلف برای برطرف شدن این مشکل ارائه شود.
- با وجود اینکه این موضوع کمتر و کمتر اتفاق می‌افتد، اما هر اتصال WiFi ای گاهی اوقات دچار اختلال می‌شود. با توجه به اینکه لوستر هوشمند بستری بر پایه IoT است، بدون اتصال WiFi نمی‌توان از تنظیمات لوستر بهره برد. بنابراین پیشنهاد می‌شود که دکمه‌ها و یا کلیدهای سخت‌افزاری همچنان برای تنظیمات پایه موجود باشد.

Light-emitting diode⁴

۳ قطعات مورد استفاده

لوستر از دو شاخه LED تشکیل شده و میزان ولتاژ ورودی هر کدام از این شاخه‌ها از طریق پورت مربوطه روی بورد Arduino و ترانزیستور (MOSFET IRF640) کنترل می‌شود. برای پیاده‌سازی منطق لوستر از بورد Arduino Mega استفاده می‌کنیم که از طریق سنسورهای تشخیص نور (BH1750FVI) نور محیط را تشخیص می‌دهد و بر اساس پیش‌فرض هاییکه در سرور IOT چیده شده میزان ولتاژ خروجی‌های آنالوگ را تنظیم می‌کند. از طرفی برای اتصال به اینترنت اشیا یک سرور کوچک خانگی را روی ماژول ESP-01S ESP8266 اجرا می‌کنیم. این سرور تنظیمات کنترل لوستر را در خود دارد و با استفاده از گوشی همراه و اتصال به آن سرور می‌توانیم این تنظیمات را کنترل کنیم. می‌توان در این پروژه از برد Arduino Uno نیز استفاده کرد.

در جدول زیر می‌توان هزینه برآورد شده قطعات و هزینه‌های پروژه را مشاهده کرد (هزینه کل معادل با ۷,۵۸۰,۰۰۰ ریال است).

نام قطعه	تعداد	هزینه
ماژول سنجش شدت نور GY-30 با سنسور BH1750FVI	۱	۷۰۰,۰۰۰
ماژول وای فای ESP8266	۱	۴۵۰,۰۰۰
آردوینو مگا R3 ۲۵۶۰	۱	۴,۵۰۰,۰۰۰
کابل USB به Type-B USB مخصوص آردوینو	۱	۲۰۰,۰۰۰
اورل LED در دو رنگ اصلی	۴۰	۲۴۰,۰۰۰
کابل ۳۰ سانتی نر به ماده (۱ بسته ۴۰ عددی)	۱	۷۰,۰۰۰
کابل جامپر مخصوص برد بورد (بسته ۶۰ عددی)	۲	۷۰۰,۰۰۰
برد بورد مدل MB-1۰۲ بدون ماژول تغذیه	۲	۷۰۰,۰۰۰
مقاومت	۱۰	۲۰,۰۰۰

جدول ۱: برآورد هزینه قطعات

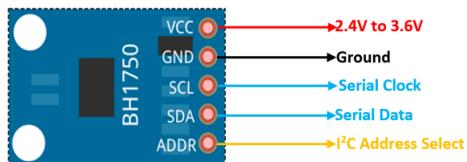
لیست قطعات و شرح پیش‌های آن‌ها به شرح زیر است:

۱. سنسور روشنایی : BH1750FVI ^۵



شکل ۱: سنسور روشنایی BH1750FVI

^۵ ما مقادیر lux را از BH1750 از طریق بوسیله I2C دریافت می‌کنیم. ADC در IC روشنایی آنالوگ را به مقدار لوکس دیجیتال تبدیل می‌کند. سپس این داده‌ها با کمک پیش‌های I2C یعنی SCL و SDA به میکروکنترلر منتقل می‌شوند. برای ارائه پالس ساعت و SDA برای انتقال مقدار lux استفاده می‌شود.

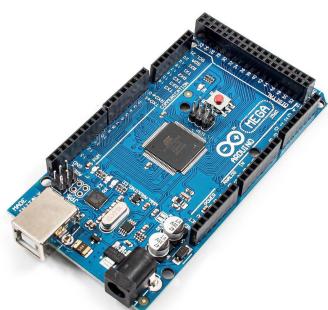


شکل ۲: شرح پین‌های BH1750FVI

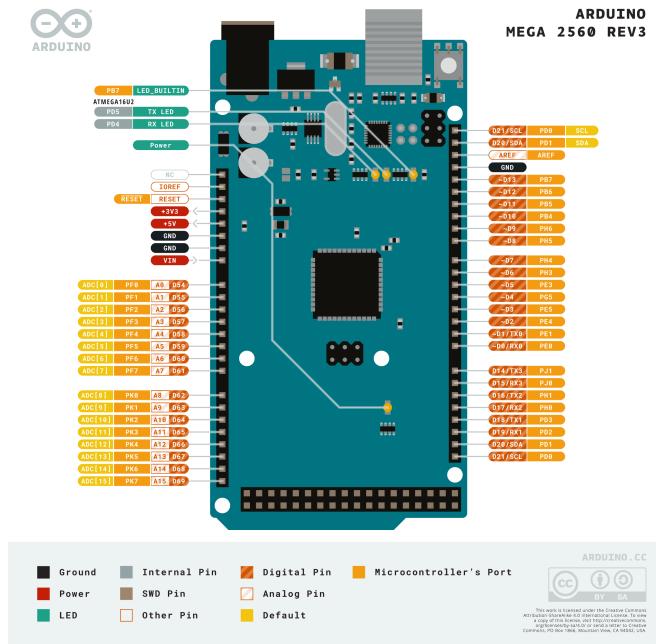
Pin Number	Pin Name	Description
1	VCC	Power supply for the module can be 2.4V to 3.6V, typically 3.0V is used
2	GND	Ground of the module, connected to ground of the circuit
3	SCL	Serial Clock Line, used to provide clock pulse for I2C communication
4	SDA	Serial Data Address, used to transfer the data through I2C communication
5	ADDR	Device address pin, used to select the address when more than two modules are connected

شکل ۳: تنظیمات پین‌های BH1750FVI

۲. برد آردوینو مگا : Arduino Mega 2560 R3

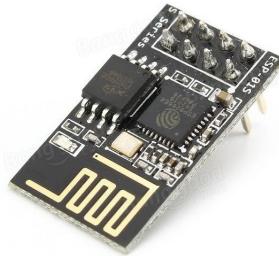


شکل ۴: برد Arduino Mega 2560 R3



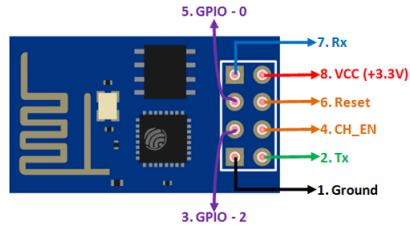
شکل ۵: شرح پین‌های Arduino Mega 2560 R3

۳. مژول وای‌فای : ESP8266 ESP-01S



شکل ۶: مژول وای‌فای ESP8266 ESP-01S

^۶ این مژول می‌تواند هم به عنوان یک نقطه دسترسی و هم به عنوان یک ایستگاه متصل به وای‌فای کار کند، بنابراین به راحتی داده‌ها را واکشی کرده و در اینترنت آپلود کند. همچنین می‌تواند با استفاده از API، داده‌ها را از اینترنت واکشی کند و به هر اطلاعاتی که در اینترنت موجود است دسترسی داشته باشد. این مژول فقط با ولتاژ ۲.۳ ولت کار می‌کند و هر ولتاژی بیش از ۷.۰۳ ولت باعث از بین رفتن مژول می‌شود.



شکل ۷: شرح پین‌های ESP8266 ESP-01S

Pin Number	Pin Name	Pin Function
1	Ground	Ground
2	GPIO1	General purpose IO, Serial Tx1
3	GPIO2	General purpose IO
4	CH_PD	Active High Chip Enable
5	GPIO0	General purpose IO, Launch Serial Programming Mode if Low while Reset or Power ON
6	RESET	Active Low External Reset Signal
7	GPIO3	General purpose IO, Serial Rx
8	VCC	Power Supply

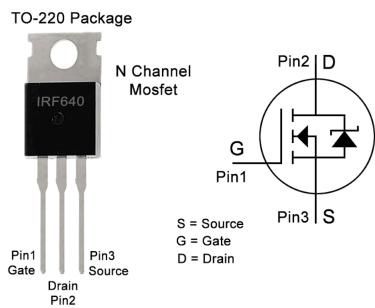
شکل ۸: تنظیمات پین‌های ESP8266 ESP-01S (در اینجا پین GPIO1 همان TX و GPIO3 همان RX است. همچنین CH_EN معادل CH_PD در شرح پین است.)

۴. ترانزیستور : ^v MOSFET IRF640



شکل ۹: ترانزیستور MOSFET IRF640

ماژول IRF640 یک ماسفت با N کانال است که برای اهداف سوئیچینگ با سرعت بالا طراحی شده است. این قابلیت سوئیچینگ با سرعت بالا می‌تواند در برنامه‌هایی که سرعت سوئیچینگ در آنها بسیار مهم است بسیار مفید باشد. در لوستر هوشمند، روشنایی LED ها به سرعت توسط PWM تغییر کند. در اینجا با توجه به اینکه منبع ولتاژ خارجی است (باتری)، باید از یک ماسفت استفاده کنیم.

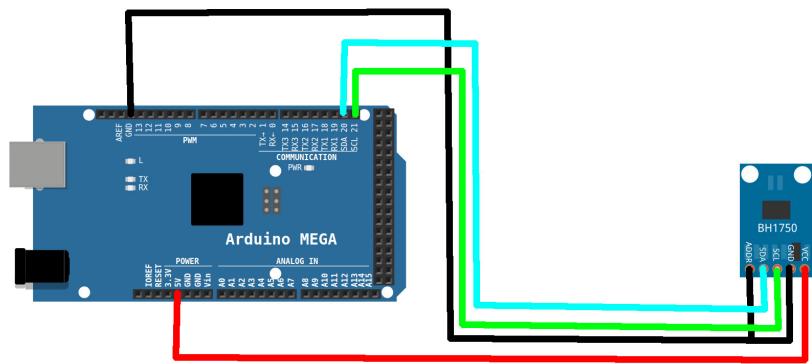


شكل ١٠ : ترانزیستور MOSFET IRF640

۴ طراحی مدار

مدار نهایی لوستر هوشمند در طی سه مرحله طراحی شده است. در مرحله اول مدار سنسور روشنایی بسته شد تا بتوانیم روشنایی تعداد کمی LED را تحت تاثیر نور محیط تغییر دهیم. در مرحله دوم دو شاخه ۲۰ تایی از LED ها را متصل کرده و به کمک منبع خارجی روشن می کنیم. در آخرين مرحله ماژول واي فاي را برای برقاري ارتباط ميان برنامه موبایل و آردوینو وصل می کنیم. هر کدام از مراحل به تفصیل در ادامه این بخش تشریح می شوند.

۱. اتصال سنسور روشنایی: در این مرحله ماژول BH1750 برای تشخیص نور را بورد آردوینو طبق شکل زیر وصل می کنیم. همانطور که مشاهده می شود، پورت های SDA و SCL و به پورت مربوطه با همان اسم در آردوینو متصل شده اند. VCC را به ۵ ولت و ADO و GND را به زمین متصل کردیم.

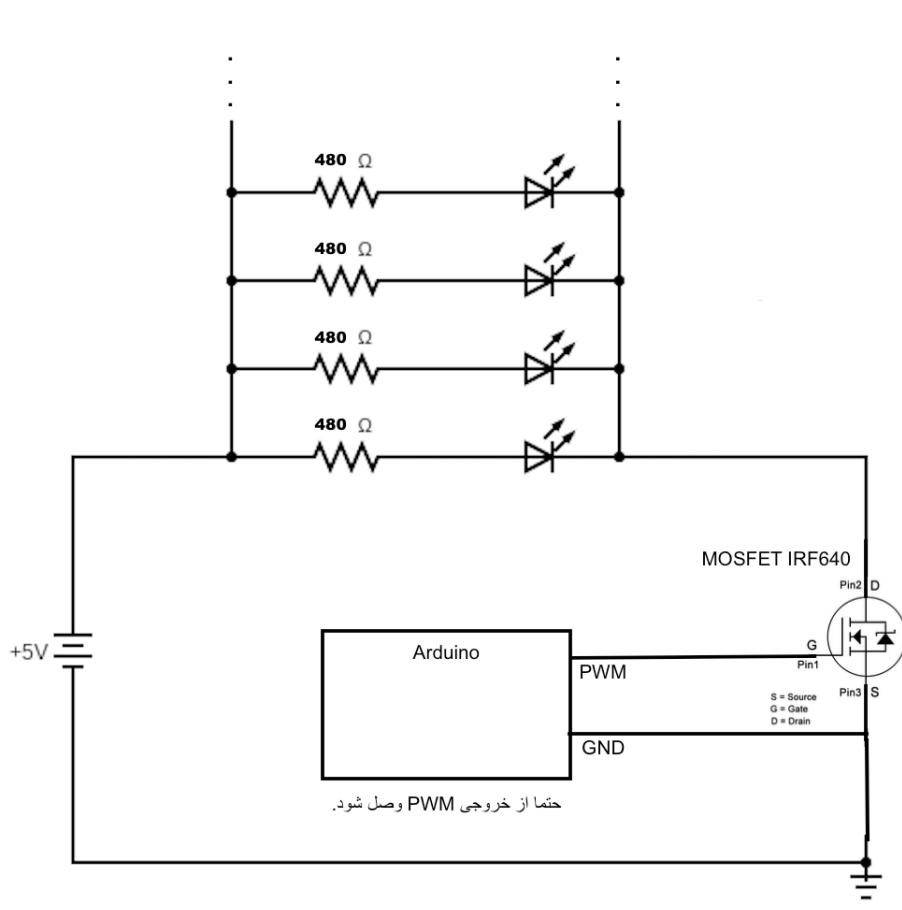


شکل ۱۱: شماتیک مدار تشخیص نور.

با استفاده از رابطه ای ساده ورودی به دست آمده از سنسور را با فرمولی تبدیل به ای می کنیم که از طریق PWM قابل کنترل است. مقدار ورودی سنسور را می توان با عددی ممیز شناور در بازه ۰ تا 2^{16} مدل کرد. اما به علت کاربرد ما که نور محیطی است، این مقدار خروجی با استفاده از آزمایش مقداری بین ۰ و ۵۰۰ به دست آمد. پس از scale کردن این مقدار بین صفر و یک مقدار روشنایی خروجی را به صورت عددی اعشاری به دست آوردم که پس از ضرب شدن در ۲۵۵ به ما عدد روشنایی LED ها را می دهد.

۲. اتصال ۴۰ LED برای شبیه سازی عملکرد لوستر: هدف از این آزمایش این مرحله، بستن یک لوستر شامل ۴۰ قطعه LED، اتصال آن به منبع خارجی، همچنین کنترل آن با ترانزیستور و در نهایت تقسیم LED ۴۰ به دو شاخه مستقل از هم است.

ابتدا، مدار آردوینو شامل سنسور مرحله قبل را بر اساس شماتیک زیر تکمیل می‌کنیم. از ترانزیستور MOSFET IRF640 برای کنترل و یک باتری ۵ ولتی به عنوان منبع خارجی استفاده می‌کنیم.



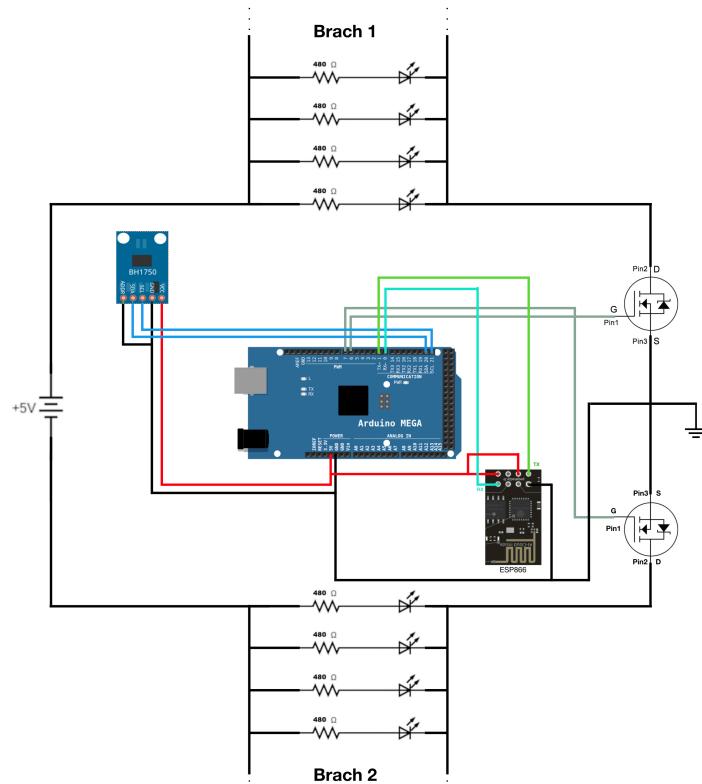
شکل ۱۲: شماتیک مدار اتصال ۲۰ LED موجود در یک شاخه. هر کدام از شاخه به یکی از پین‌های PWM شماره ۶ یا ۷ آردوینو متصل می‌شوند.

در این مدار، LED ها به باتری وصل هستند و خروجی PWM به گیت کنترل ترانزیستور MOSFET IRF640 متصل است. می‌دانیم که PWM به کمک روشن و خاموش کردن سریع می‌تواند روش‌نایی را کنترل کند. در اینجا، بجای اتصال مستقیم به LED ها، PWM به گیت ترانزیستور متصل شده و آن را به سرعت قطع و وصل می‌کند. یعنی ترانزیستوری میان باتری و LED است که سرعت قطع و وصل کردن آن با PWM تنظیم می‌شود.

۳. اتصال مازول وای‌فای: برای ارتباط میان موبایل اپ و آردوینو به کمک مازول وای‌فای ESP8266 ESP-01S کردن کد باید پین‌های RX و TX قطع شوند:

- (آ) پین RX در مژول ESP را به پین آردوینو متصل می‌کنیم.
- (ب) پین TX در مژول ESP را به پین آردوینو متصل می‌کنیم.
- (ج) پین CH_PD یا Enable در مژول ESP را به پین 3V+ آردوینو متصل می‌کنیم.
- (د) پین VCC در مژول ESP را به پین 3V+ آردوینو متصل می‌کنیم.

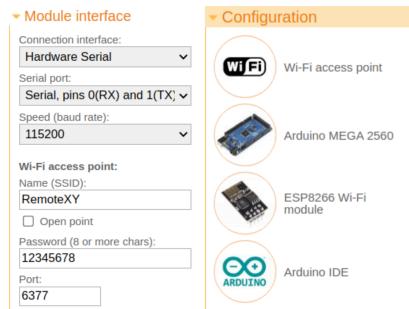
در نهایت می‌توان مدار کلی را در شکل زیر مشاهده کرد:



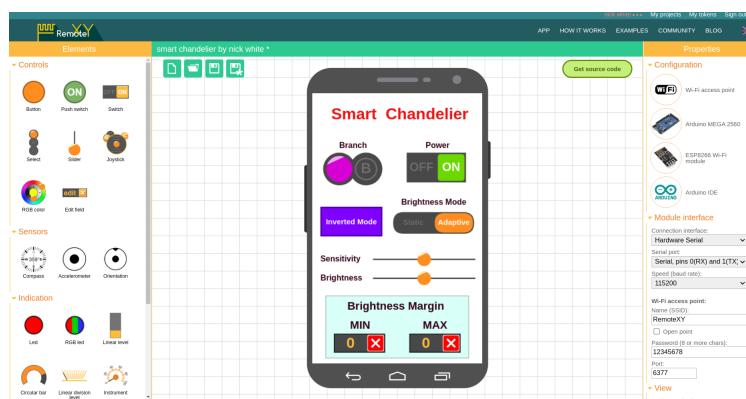
شکل ۱۳: مدار نهایی

RemoteXY ۵

یک ابزار کنترلی برای کنترل مدارها با برد های اصلی آردوینو از طریق موبایل با کامپیوتر می باشد. به طور خلاصه در این ابزار ابتدا یک رابط کاربری به صورت گرافیکی طراحی می شود، سپس تعیین می گردد که این رابط برای چه کنترل چه بردی طراحی شده و با چه ماژولی به برد متصل می گردد. (از طریق ماژول های بلوتوث، وای فای یا OTG) در ادامه پارامترهای کلی مدار شامل نحوه اتصال ماژول ارتباطی به برد، سرعت پورت های سریال، نام کاربری و پسورد مربوط به نقطه اتصال وای فای و... تعیین می شوند. در نهایت این ابزار یک فایل project.ino به ما می دهد که حاوی RemoteXY struct و پارامترهای مورد نیاز می باشد. سپس برنامه ای اصلی در توابع loop و setup نوشته می شود که توضیحات آن در بخش توضیحات کد قابل مشاهده است. دقت کنید که قبل از کامپایل کردن کد در Arduino IDE ابتدا می بایست کتابخانه RemoteXY را از [این لینک](#) دانلود کرده و به IDE اضافه کنیم.



شکل ۱۴: تصاویر مربوط به تنظیمات اولیه



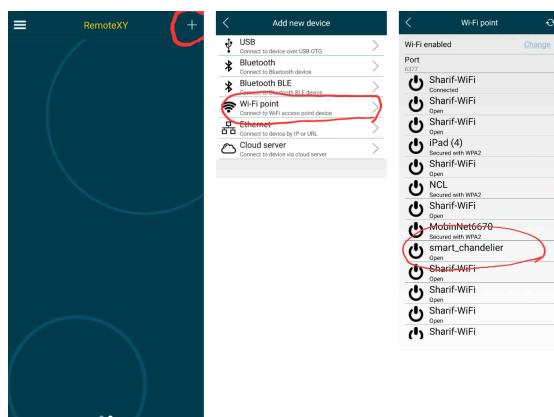
شکل ۱۵: نمای کلی از محیط طراحی رابط کاربری در برنامه تحت وب RemoteXY

برای اجرای مدار:

۱. دو شاخه LED و برد آردوینو را به برق متصل می کنیم.

۲. اپلیکیشن موبایل RemoteXY را دانلود کرده و نصب می‌کنیم. سپس از منوی سمت چپ گزینه WiFi point را انتخاب کرده و نقطه smart_chandelier را انتخاب می‌کنیم.
(دقت کنید که پورت اتصال روی ۶۳۷۷ تنظیم شده باشد.)

۳. سپس کلمه عبور ۱۲۳۴۵۶۷۸ را وارد کرده تا اتصال از طریق ماثول وای‌فای برقرار گردد.

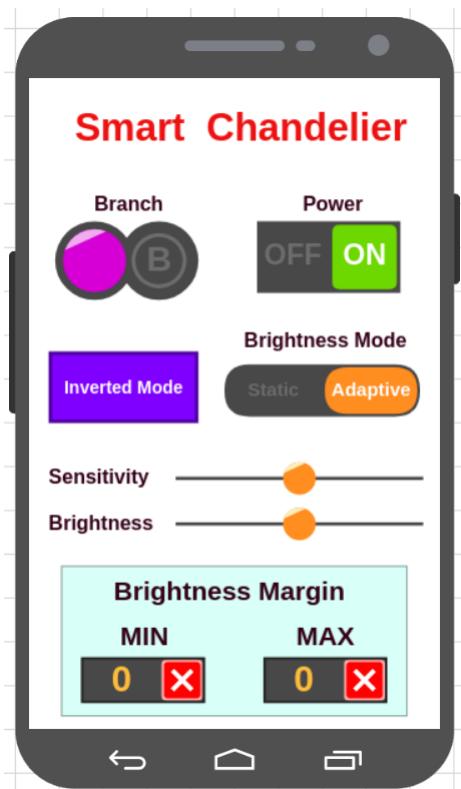


شکل ۱۶: نمای کلی از محیط طراحی رابط کاربری در برنامه تحت وب RemoteXY

۴. بعد از اتصال به مدار، رابط کاربری طراحی شده پدیدار می‌گردد و می‌توان روشنایی شاخه‌ها را با آن کنترل کرد.

۶ برنامه موبایل

متناسب به قابلیت‌های لوستر، یک برنامه موبایل طراحی کرده‌ایم که بتوان در آن تنظیمات نامبرده مربوط به لوستر هوشمند را انجام داد. شماتیک برنامه به شکل زیر است:



شکل ۱۷: شماتیک اپلیکیشن موبایل

در نهایت می‌توان با نصب کردن برنامه موبایل و اتصال آن به ESP و آردوبینو، تنظیمات موردنظر را انجام داد. این تنظیمات و گزینه‌های موجود در این موبایل اپ به شرح زیر است:

- تنظیم حساسیت سنسور نوری: کاربر می‌تواند با انتخاب عددی میان ۱۰ الی ۱۵۰ (توسط slider)، میزان حساسیت سنسور نور را تنظیم کند (عدد کمتر معادل حساسیت کمتر است؛ یعنی میزان نور با تغییرات بیشتری نسبت به عددی بیشتر تغییر می‌کند).
- دکمه روشن و خاموش: کاربر می‌تواند تمامی چراغ‌های لوستر (شاخه مورد نظر) را خاموش یا روشن کند.
- انتخاب شاخه: کاربر می‌تواند انتخاب کند که متغیرهای تغییر داده شده، مربوط به کدام شاخه باشند (هر شاخه، مستقل از شاخه دیگر حالت‌های مختلف و دکمه‌های متفاوت دارد).

- حالت استاتیک یا داینامیک (Adaptive): کاربر می‌تواند با حالت استاتیک یک مقدار خاص را برای روشنایی انتخاب کرده و تمامی LED های لوستر با آن مقدار تنظیم می‌شوند. در حالت داینامیک نیز مقدار روشنایی لوستر با سنسورهای تنظیم می‌شود.
- تنظیم حداکثر و حداقل میزان روشنایی: کاربر می‌تواند با انتخاب عددی میان ۰ تا ۲۵۵ (توسط slidebar)، حداقل و حداکثر میزان روشنایی یک شاخه را تعیین کند. بنابراین روشنایی یک شاخه، محدود به این دو عدد می‌شود و نمی‌تواند مقداری خارج از این بازه بگیرد.
- مودهای مختلف لوستر: مودهای متفاوت که می‌توانند شامل لوستر را در حالت تنظیم داینامیک عادی یا رقص نور (همانند گزارش اول) تنظیم کنند (این حالت‌ها در قسمت‌های بعدی تکمیل می‌شوند). یکی از مودهای در نظر گرفته شده برای این قسمت حالت In-Inverted min to max زیاد می‌شود.

۷ توضیحات کد

بخش اولیه کد شامل ورودی‌هایی است که کاربر از طریق UI وارد می‌کند و همانطور که در شکل زیر می‌بینید در قالب فرمت struct آمده است:

```
1 // this structure defines all the variables
2 // and events of the control interface
3 struct {
4
5     // input variables
6     int8_t sensitivity;
7     // =-100..100 slider position
8     uint8_t power;
9     // =1 if switch ON and =0 if OFF
10    uint8_t branch;
11    // =0 if select position A, =1 if position B, =2
12    // if position C, ...
13    int8_t brightness;
14    // =-100..100 slider position
15    uint8_t inverted_mode;
16    // =1 if button pressed, else =0
17    int16_t min_brightness;
18    // 32767.. +32767
19    int16_t max_brightness;
20    // 32767.. +32767
21    uint8_t brightness_mode;
22    // =1 if switch ON and =0 if OFF
23
24    // other variable
25    uint8_t connect_flag;
26    // =1 if wire connected, else =0
27
28 } RemoteXY;
```

این ورودی در بخش حلقه اصلی کد به ازای شاخه انتخابی در یک ساختار Config به ازای هر شاخه ذخیره می‌شوند. ساختار Config به ازای هر شاخه یک زیر نوع به فرم زیر است.

```
1
2 // This struct contains configurations
3 // for each of the branches
4 struct Config{
5     int8_t sensitivity;
6     // =-100..100 slider position
7     uint8_t power;
8     // =1 if switch ON and =0 if OFF
```

```

9     int8_t brightness;
10    // ==-100..100 slider position
11    uint8_t inverted_mode;
12    // =1 if button pressed, else =0
13    int16_t min_brightness;
14    // 32767.. +32767
15    int16_t max_brightness;
16    // 32767.. +32767
17    uint8_t brightness_mode;
18    // =1 if switch ON and =0 if OFF
19 };
20 Config branch_conf[branch_num];

```

در ابتدای کار این مقادیر به صورت زیر مقداردهی اولیه می‌شوند.

```

1 // Activate all the configurations
2 // and setup default configurations per each branch
3 for (int i = 0; i < branch_num; i++) {
4     conf_activated[i] = 1;
5     branch_conf[i].power = 0;
6     branch_conf[i].sensitivity = 0;
7     branch_conf[i].brightness = 0;
8     branch_conf[i].inverted_mode = 0;
9     branch_conf[i].min_brightness = 0;
10    branch_conf[i].max_brightness = 255;
11 }

```

و به صورت زیر در هر مرحله مقادیر ورودی شاخه از طریق UI به ازای شاخه مورد نظر وارد می‌شود. لازم به ذکر است که برنامه به نحوی زده شده که اگر بیشتر از دو شاخه نیز داشته باشیم باز هم بدون هیچ مشکلی کار کند.

```

1 RemoteXY_Handler ();
2 int setup_branch_index = RemoteXY.branch;
3
4 // Setup the current branch according
5 // to the current input of the UI
6 branch_conf[setup_branch_index].sensitivity = \
7     RemoteXY.sensitivity;
8 branch_conf[setup_branch_index].power = \
9     RemoteXY.power;
10 branch_conf[setup_branch_index].brightness = \
11     RemoteXY.brightness;
12 branch_conf[setup_branch_index].inverted_mode = \
13     RemoteXY.inverted_mode;
14 branch_conf[setup_branch_index].min_brightness = \

```

```

15           RemoteXY.min_brightness;
16 branch_conf[setup_branch_index].max_brightness = \
17             RemoteXY.max_brightness;
18 branch_conf[setup_branch_index].brightness_mode = \
19             RemoteXY.brightness_mode;
20
21 // Setup whether the config is activated or not for the pins
22 // (This comes to use when we set a pin to inverted mode)
23 for (int i = 0; i < branch_num; i++) {
24     int prv = (i - 1 + branch_num) % branch_num;
25     if (branch_conf[prv].inverted_mode == 1) {
26         conf_activated[i] = 0;
27     } else {
28         conf_activated[i] = 1;
29     }
30 }

```

توجه کنید اگر مقدار conf-activated به ازای یکی از شاخه‌ها غیر فعال باشد یعنی در حلقه اصلی مقدار خروجی این شاخه از روی شاخه‌های دیگر محاسبه می‌شود و لازم به مقدار دهی آن نیستیم. توجه کنید اگر شاخه‌ای شاخه قبلش روی حالت inverted تنظیم شده باشد یعنی این شاخه و قبلی به صورت رقص نور و سنکرون باهم تغییر می‌کنند و شاخه قبلی مقدار خروجی این شاخه را تنظیم می‌کند.

در بخش بعدی از حلقه اصلی روی تمامی شاخه‌ها پیمایش می‌کنیم و به ازای شاخه‌هایی که باید مقدار خروجی‌شان را تعیین کنیم بر اساس مقادیر داده شده در branch-conf مقدار خروجی این پین را تنظیم می‌کنیم.

مقدار خروجی بر اساس ورودی بر اساس چهار تا از پارامترهای کانفیگ هر شاخه تنظیم می‌شود:

- اگر مقدار conf.power برابر با صفر باشد یعنی این شاخه غیر فعال شده و در نتیجه مقدار خروجی آن مستقل از هر چیز دیگری برابر صفر است.
- مقدار conf.brightness-mode به شاخه می‌گوید که ورودی سنسور را در این شاخه لحظه کنیم یا نه. در صورتیکه این مقدار صفر باشد، مقدار خروجی برابر conf.min خواهد بود و در صورتیکه برابر یک باشد به صورت adaptive از روی ورودی سنسور خروجی اش تنظیم می‌شود. مقدار خروجی بر اساس ورودی سنسور یکتابع خطی است. ابتدا تابع زیر را در نظر بگیرید که اسکیل کردن یک مقدار ratio بین دو بازه استفاده می‌شود.

$$scale(l, x, r) = (1 - x) \times l + x \times r$$

و مطابق با فرمول زیر مقدار ورودی سنسور را تبدیل به عددی بین صفر و یک می‌کنیم

$$b = 1 - \min(1, \frac{x}{scale(s_{\min}, s, s_{\max})})$$

این خروجی باید بین brightness-min و brightness-max تنظیم شود و به صورت زیر این کار انجام می‌شود.

$$out = scale(brightness_{\min}, b, brightness_{\max})$$

توجه کنید به ازای این کد خاص مقدار s_{\min} و s_{\max} به ترتیب برابر ۱۰ و ۱۵۰ شده‌اند که محدوده بیشترین نور و کمترین نور در اتفاق محل آزمایش است. همچنین مقدار s نیز یک پارامتر قابل تنظیم است که از طریق ورودی UI و مقدار conf.sensitivity تنظیم می‌شود. ابتدا مقدار روشنایی اولیه هر کدام از LED‌ها را برابر صفر تنظیم می‌کنیم و در ادامه این حلقه مقدار اصلی آن‌ها مشخص می‌شود.

```

1 // For each branch we will setup their brightnesses according
2 // to their configuration
3
4 for (int branch_pin = branch_base; branch_pin <
5     branch_base + branch_num; branch_pin++) {
6     int index = branch_pin - branch_base;
7
8     // This means that the brightness for this branch
9     // is set by another pin
10    // for example in the inverted mode
11    if (conf_activated[index] == 0) {
12        continue;
13    }
14
15    int brightness = 0;
16    float brightness_ratio = 0.0;
17    .....
18 }
```

در کد زیر نیز روشن بودن یا نبودن هر شاخه را مشخص می‌کیم.

```

1 if (branch_conf[index].power == 1){
2     // branch power switch is ON
3     ...
4 }
```

در تکه کد زیر، روشنایی هر شاخه با توجه به تنظیمات خودکار (Adaptive) و با توجه به سنسور تنظیم می‌شود.

```

1 if (branch_conf[index].brightness_mode == 1){
2     // adaptive brightness mode
3     // In this mode the brightness is adaptively set
4     // according to the value received from the sensor
```

```

5
6     float sensitivity_level_ratio =
7         (- branch_conf[index].sensitivity + 100) * 1.0 / 200;
8     int sensitivity_level =
9         sensitivity_level_min + sensitivity_level_ratio *
10        (sensitivity_level_max - sensitivity_level_min);
11     brightness_ratio =
12        1 - min(1, lightMeter.readLightLevel() / sensitivity_level);
13 }

```

در تکه کد زیر، روشنایی هر شاخه در حالت تنظیمات ثابت تنظیم می‌شود.

```

1 else {
2     // static brightness mode
3     // In this mode the brightness ratio is set only according
4     // to the input provided in the brightness meter of the UI.
5     // This value is static and independent of the changes in
6     // the environment's lighting.
7     brightness_ratio =
8         (branch_conf[index].brightness + 100) * 1.0 / 200;
9 }

```

- مقدار conf.inverted-mode برابر صفر یا یک است که اگر برابر یک باشد یعنی این شاخه و شاخه بعدی رقص نور انجام می‌دهند. برای پیاده‌سازی این رقص نور مقدار خروجی آیتم بعدی که برابر با *out* بود را در نظر می‌گیریم و به ترتیب مقدار این شاخه و بعدی اش را مطابق دنباله زیر تنظیم می‌کنیم:

$$< \left(\frac{out}{2}, \frac{out}{2}\right), \left(\frac{out}{2} + 1, \frac{out}{2} - 1\right), \left(\frac{out}{2} + 2, \frac{out}{2} - 2\right) \dots >$$

برای اینکه فرمت خروجی پویا باشد از یک پارامتر شمارنده به اسم nxt-branch-counter بهره بردیم و این مقدار هر سری در بین بازه‌ای معقول قرار می‌گیرد و به شاخه اول اضافه می‌شود و از شاخه دوم کم می‌شود. به این ترتیب همیشه جمع اندازه خروجی این دو شاخه متوالی برابر با مقدار خروجی‌ای است که به ازای شاخه اولی تنظیم شده‌بود.

```

1 if (branch_conf[index].inverted_mode == 1) {
2     // If the inverted mode is set to on then a form
3     // of light dance is performed between the branch and the next one
4     // the dance is performed as follows:
5     //     with the base_brightness obtained from before,
6     //     the branch and the next one change their brightness
7     //     values such that the sum of their brightnesses are
8     //     always equal to base_brightness. e.g.
9     //     (base_brightness / 2, base_brightness / 2)

```

```

10      //      (base_brightness / 2 + 1, base_brightness / 2 - 1)
11      //      (base_brightness / 2 + 2, base_brightness / 2 - 2)
12      //      ...
13
14      int range = min(base_brightness / 2, 255 - base_brightness / 2);
15      int nxt_branch_pin = get_next_branch(branch_pin);
16
17      // A dynamic parameter that changes each time the loop runs
18      // we truncate the parameter in [-range, +range].
19      int i = inverted_mode_counter % (range + range + 1) - range;
20
21      // The values are set such that their sum is equal to
22      // `base_brightness`
23      analogWrite(branch_pin, base_brightness / 2 + i);
24      analogWrite(nxt_branch_pin, base_brightness / 2 - i);
25      inverted_mode_counter++;
26  }

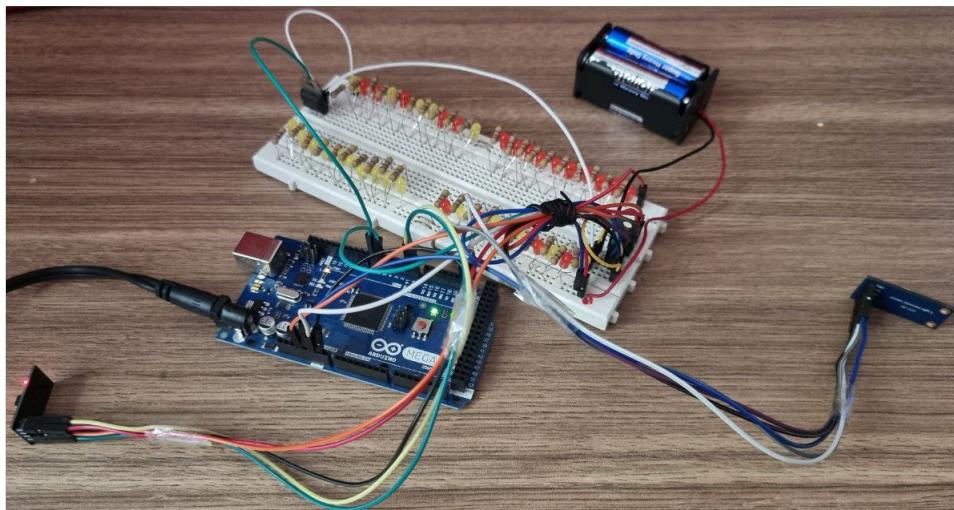
```

برای اطلاعات بیشتر از بخش‌های کد می‌توانید سورس کامل را در پیوست‌های پروژه به همراه مستندات کامل مشاهده نمایید.

جمع‌بندی

در این پروژه قصد داشتیم که یک لوستر هوشمند طراحی کنیم که بتواند با توجه به نیازمندی‌های کاربر روشنایی دو اتاق در خانه را شبیه‌سازی کند. کاربر می‌تواند بوسیله برنامه موبایل و به طریق بی‌سیم از میان حالت‌های ارائه شده انتخاب کند یا تنظیمات خودش را اعمال نماید. برای ادامه دادن این کار می‌توان تنظیمات ساعت‌های روشنایی را طبق برنامه کاربر نیز به برنامه اضافه کرد. مدار نهایی در شکل زیر قابل مشاهده است:

([لینک ویدیو عملکرد لوستر هوشمند](#))



شکل ۱۸: مدار نهایی لوستر هوشمند