پروژه برنامه سازی پیشرفته

طراحی سیستم کنترل و نظارت بر روشنایی ساختمان

۵۰ درصد اول:

راه اندازی سنسور GY-۳۰ و موتور DC با درایور موتور L۲۹۸N

استاد: دکتر جهانشاهی

اعضای گروه: معصومه سجادی ۹۳۲۳۰۸۷

فهرست

۱- مقدمه

سیستم مدیریت ساختمان (BMS) یا سیستم اتوماسیون ساختمان (BAS) به سیستمی اطلاق می شود که در یک ساختمان نصب شده و از طریق اجزای خود کنترل قسمت های مختلف ساختمان و نمایش خروجی های مناسب را برای کاربر امکان پذیر می نماید. قسمت های مختلف تحت کنترل معمولا شامل تاسیسات مکانیکی و سیستم تهویه مطبوع و تجهیزات روشنایی بوده که می تواند به سیستم های ایمنی، آتش نشانی، تنظیم دسترسی، تامین برق اضطراری و ... نیز تسری یابد. به طور کلی هدف از استفاده از سیستم های BMS در یک ساختمان تطبیق شرایط کارکرد اجزای مختلف با توجه به شرایط محیطی و نیاز ساختمان در آن زمان است.

در ساختمان های هوشمند با استفاده از سیستم خودکار کنترل روشنایی ساختمان، کنترل سیستم سرمایش و گرمایش، کنترل دوربین های مدار بسته، کنترل در ها، کنترل وضعیت های اضطراری همچون آتش سوزی، زلزله و بسیاری کنترل های هوشمند دیگر، مصرف انرژی به نحو چشمگیری کاهش می یابد .ساختمان هوشمند، ساختمانی است که مجهز به یک زیرساختار ارتباطاتی قوی بود که می تواند به صورت مستمر نسبت به وضعیت های متغیر محیط عکس العمل نشان داده و خود را با آنها وفق دهد و همچنین به ساکنین ساختمان این اجازه را می دهد که از منابع موجود به صورت موثرتری استفاده کرده و امنیت و آسایش آنها را افزایش دهد .

در بیشتر ساختمان های هوشمند پرده برقی به عنوان یک المان لوکس نگاه می شود و در بسیاری از موارد به علت هزینه های این قابلیت از نصب آن خود داری می شود. اما در واقع پرده ی برقی از اهمیت بسیاری برخوردار است از جمله: بهینه سازی مصرف انرژی ، امنیت و حریم شخصی.

هدف از این پروژه، طراحی یک سیستم مبتنی بر اینترنت برای ارسال و دریافت مستقیم اطلاعات (در اینجا ارسال و دریافت روشنایی محیط مورد نظر در زمان های مشخص) به یک مدار کنترل کننده در ساختمان، به گونه ای که کنترل نور قسمت های مختلف ساختمان توسط کنترل میزان روشنایی چراغ ها و باز و بسته بودن پرده ها صورت می پذیرد.

۲- سیستم های مورد نیاز و راه اندازی آن ها

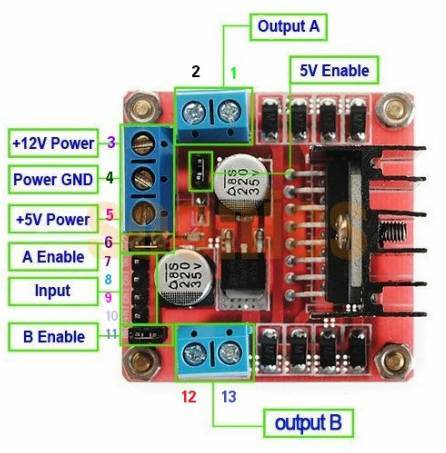
در خانه های هوشمند جهت هوشمند سازی نور و پرده ها می بایست از پرده اتوماتیک استفاده نمود. همچنین به برد و برنامه نویسی آردوینو، رزبری پای، موتور و راه انداز آن، جی کیوئری، پی سی و ساخت بستزی جهت نمایش دیتا و کار کاربر با برنامه نیازمندیم که هدف اصلی این پروژه ساخت برنامه ی روی دسکتاپ و اتصال به وب برای دیتای دریافتی و تغییر تنظیمات می باشد.

۲-۱- درایور موتور L۲۹۸N

ماژول درایور موتور L۲۹۸N، یکی از درایور موتورهای ساده و ارزان قیمت است که کار را برای کاربرانی که با موتور DC کار دارند ساده کرده است. برای کنترل سرعت و تغییر جهت انواع موتورهای الکتریکی، میکروکنترلرها به علت کم بودن جریان خروجی شان قادر به راه اندازی موتور الکتریکی نیستند و به همین دلیل از درایور موتورها استفاده می کنیم. یکی از درایور موتورهای پرکاربرد، درایور موتور L298N می باشد.

ماژول فوق با استفاده از آی سی درایور موتور L298 طراحی گردیده و توانایی کنترل همزمان ۲ موتور DC را دارد. این ماژول با توجه به اینکه از خنک کننده مناسب برخوردار است، توانایی جریان دهی بالا (۲ آمپر در هر کانال) را به صورت پیوسته دارد.

نحوه ی راه اندازی موتور:



Pin 1 - Green Wire (A+) Stepper Motor

Pin 2 - Black Wire (A-) Stepper Motor

Pin 3 - 12 V Adaptor (+ve lead)

Pin 4 - Adaptor Ground)

Pin 5 - (Nothing)

Pin 6 - Jumper Connected

Pin 7 - To Arduino Uno Pin 8

Pin 8 - To Arduino Uno Pin 9

Pin 9 - To Arduino Uno Pin 10

Pin 10 - To Arduino Uno Pin 11

Pin 11 - Jumper Connected

Pin 12 - Red Wire (B+) Stepper Motor

Pin 13 - Blue Wire (B-) Stepper Motor

Pin۱ : به یک سر موتور DC شماره۱ متصل می شود.

Pin۲ : به یک سر دیگر موتور DC شماره ۱متصل می شود.

Pin۳: جامپر ۱۲ ولت نامیده می شود. اگر ولتاژ ورودی از ۱۲ ولت بیشتر شود، باید جامپر قطع شود.

Pin۴: زمین ماژول است و به زمین منبع تغذیه وصل می شود. توجه شود که باید منفی منبع تغذیه با پایه شماره ۵ ماژول L298N و همچنین با GND برد آردوینو به هم وصل شوند.

Pin۵: در صورتی که جامپر ۱۲ ولت متصل باشد، این پایه دارای ولتاژ ۵ ولت است و می توان از این پایه برای تغذیه بردهای دیگر مثل برد آردوینو استفاده کرد.

Pin۶: جامپر فعال سازی موتور شماره ۱ است. زمانی که به جای موتور DC از Step motor استفاده می شود این جامپر را باید از جای خود خارج کرد. این پایه به پایه PWM برد آردوینو متصل می شود و بدین وسیله می توان سرعت موتور DC شماره (A)را کنترل کرد.

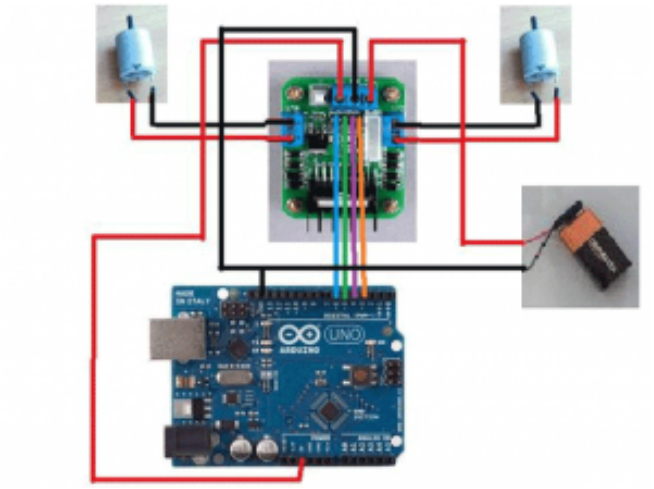
Pin۷،۸: IN1 و IN۲ برای تعیین جهت موتور شماره ۱

Pin۹،۱۰: IN۳ و IN۴ برای تعیین جهت موتور شماره ۲

Pin۱۱: جامپر فعال سازی موتور شماره ۲ است. زمانی که به جای موتور DC از Step motor استفاده می شود این جامپر را باید از جای خود خارج کرد.این پایه به پایه PWM بورد آردوینو متصل می شود و بدین وسیله می توان سرعت موتور DC شماره ۲ را کنترل کرد.

Pin۱۲: به یکی از پایه های موتور DC شماره۲ متصل می شود.

Pin۱۳: به پایه ی دیگر موتور DC شماره۲ متصل می شود.



برنامه راه اندازی موتور با آردوینو:

ابتدا پایه ها به صورت توضیح داده شده در زیر وصل خواهد شد.

VCC ماژول را به V5 آردوینو - GND ماژول به GND آردوینو - VSماژول را به منبع تغذیه خارجی

OUTPUT۱,۲ را به دو سر موتور شماره ۱ - OUTPUT۳,۴ به دو سر موتور شماره۲

INPUT۱ به پایه شماره ۵ آردوینو - INPUT۲ به پایه شماره ۶ آردوینو

INPUT3 به پایه شماره۹ آردوینو - INPUT4 به شماره ۱۰ آردوینو

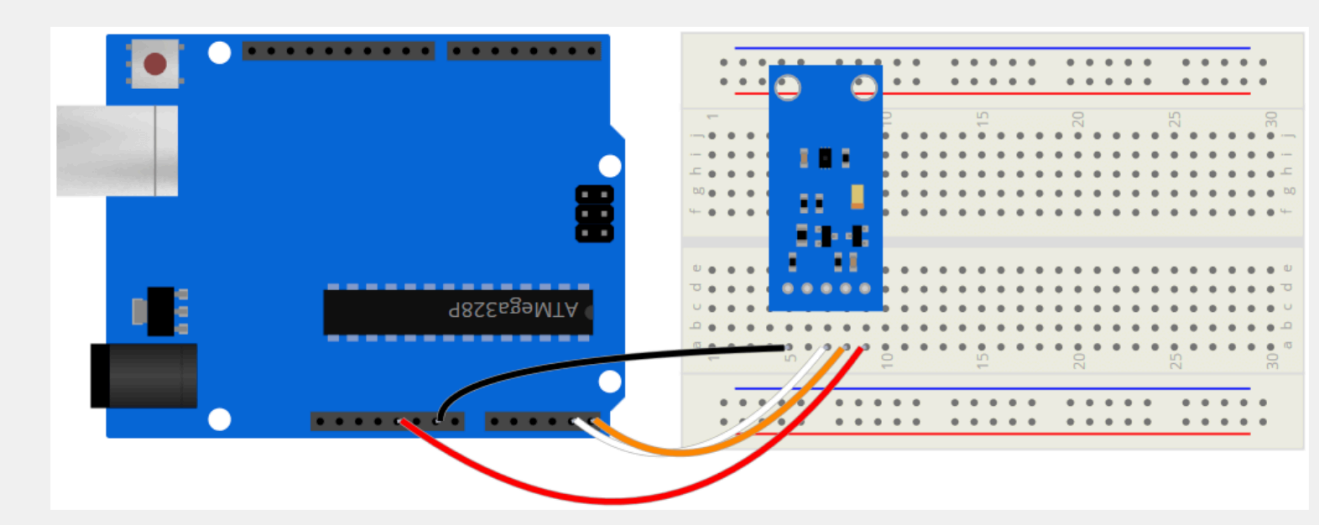
به دلیل استفاده از PWM برای کنترل سرعت موتورها لازم است از پایه های PWM استفاده گردد. برای منبع تغذیه می توان از باتری نیز استفاده کرد.

کد آردوینو به گزارش پیوست شده است.

۲-۲- سنسور نوری GY-۳۰

ماژول BH1750 یک برد مجهز به سنسور حساس به شدت نور است که دارای یک مبدل AD شانزده بیتی است. این ماژول می تواند مستقیما سیگنال دیجیتال در خروجی ایجاد کند. راه ارتباطی این ماژول رابط سریال I2C می باشد. این ماژول برای تشخیص میزان نور محیط با دقت و رزولوشن بالا مناسب بوده و داده های خروجی آن بصورت lx (لوکس متر) می باشد. همچنین این ماژول به راحتی به وسیله آردوینو قابل راه اندازی است.

نحوه ی راه اندازی سنسور:



برنامه ی راه اندازی سنسور با آردوینو :

کد آردوینو به گزارش پیوست شده است.

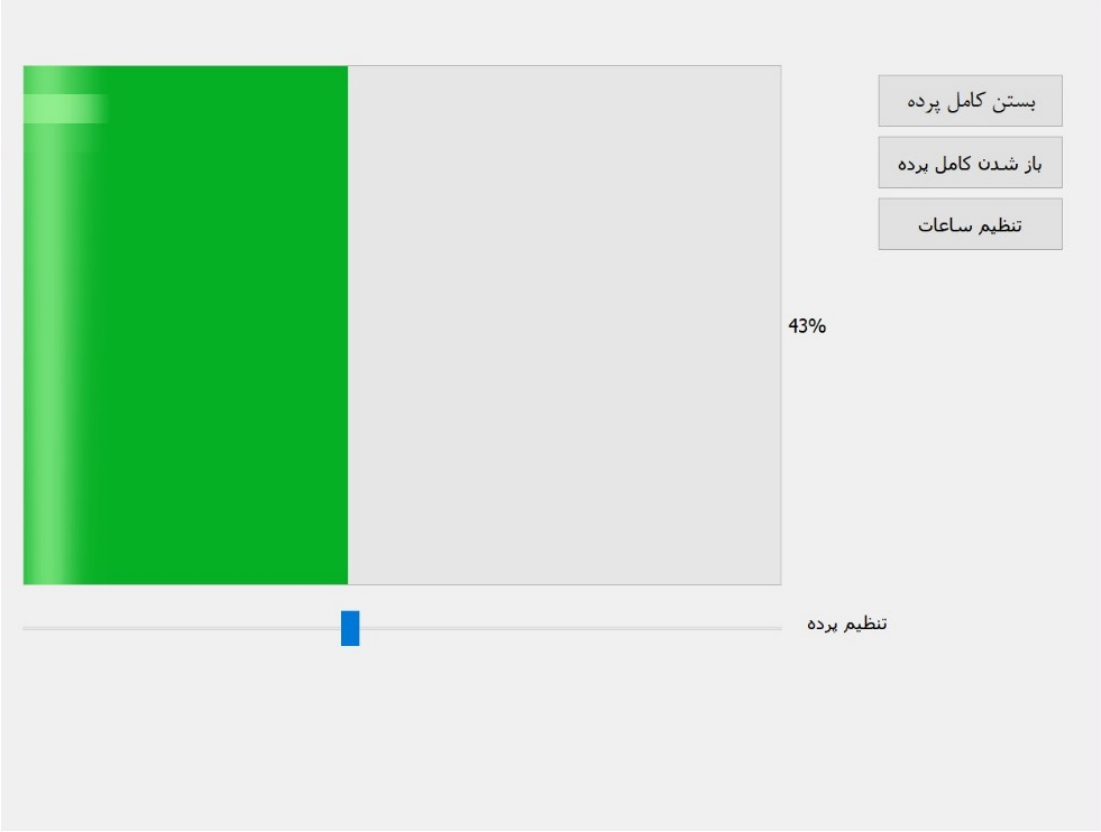
خروجی برنامه بر روی پورت سریال:

۳- پیاده سازی نرم افزاری

در این قسمت نحوه پیاده سازی پروژه توسط Qt Designer و Python توضیح داده می شود.

۳-۱- Qt Designer:

GUI برنامه توسط این نرم افزار طراحی شده است. در این GUI یک مستطیل بزرگ وجود دارد که موقعیت پرده در آن نمایش داده شده است. موقعیتی که پرده در آن قرار دارد توسط جعبه سبز رنگ نمایش داده می شود و مقدار آن به صورت درصدی نمایش داده می شود. در زیر آن نیز یک نوار تنظیم پرده وجود دارد که موقعیت پرده را تغییر می دهد. در سمت راست پرده سه دکمه وجود دارد. دو دکمه باز شدن و بستن کامل پرده هر کدام در صورت کلیک شدن پرده را به طور کامل باز یا بسته می کنند. در نهایت دکمه تنظیم ساعت برای تنظیم ساعت پیش فرض برای باز و بسته شدن پرده و سایر تنظیمات استفاده می شود.
این GUI در زیر نمایش داده شده است.



۲-۳- Python:

در این قسمت توسط کد Python اتفاقاتی که در صورت زدن هر کدام از دکمه ها رخ می دهد را handle می کنیم.
در ابتدا کتابخانه های مورد نظر را import می کنیم. سپس با استفاده از uic، GUI را به متغیری به اسم Form منتقل می کنیم از کلاسی به نام Mainclass برای handle کردن آن استفاده می کنیم. Form۲ و secondclass برای handle کردن پنجره جدیدی که با زدن دکمه تنظیمات باز می شود، مورد استفاده قرار می گیرند. در کلاس Mainclass ابتدا constructor فراخوانی می شود سپس متد های این کلاس تعریف می شوند. این متدها به ترتیب توضیح داده می شوند.

openbutton:
در این متد پس از کلیک شدن بر روی دکمه بازشدن پرده مقدار بازشدگی پرده بر اساس درصد برابر ۱ می شود و در بار زیر پرده ست می شود.

closebutton:
در این متد پس از کلیک شدن بر روی دکمه بسته شدن پرده مقدار بازشدگی پرده بر اساس درصد برابر ۱۰۰ می شود و در بار زیر پرده ست می شود.

settingbutton:
در این متد پس از کلیک شدن بر روی دکمه تنظیمات پرده یک صفحه جدید باز و نمایش داده می شود و آماده ایجاد تغییرات می شود.

valuechange:
در این متد پس از تغییر دادن موقعیت پرده در بار زیر پرده توسط کاربر مقدار بار در برنامه متناسب با آن تغییر می کند.

در نهایت از این کلاس یک نمومه ساخته می شود و در یک متغیر ریخته می شود. سپس صفحه برنامه نشان داده می شود و آماده استفاده برای کاربر می شود.

۴- اتصال سنسور و GUI

در ادامه سعی داریم که سنسور خوانده شده توسط برد آردوینو را به پایتون و از طریق آن به **GUI** متصل سازیم.برای این کار باید توجه داشت که نمیتوانیم با حالت قبلی به پاسخ دلخواد برسیم زیرا ک میخواهیم همواره مقدار سنسور را بخوانیم و هم زمان در فایل **GUI** نمایش دهیم که به معنای **multi Threading** می باشد. و باید یک **thread** جدید برایش تعریف کنیم تا جداگانه از ترد اصلی که همان نمایش دهنده ی **GUI** ما است، کار خواندن سنسور را از پورت سریال انجام دهد .

```
class serialupdate(QThread):
    update_trigger=QtCore.pyqtSignal(str)

    def __init__(self):
        QtCore.QThread.__init__(self)
        self.ser=serial.Serial()
        self.ser.port="COM3"
        self.ser.baudrate= 9600



    def run(self):
        self.ser.open()
        while 1:
            self.cc=str(self.ser.readline())
            self.cc=self.cc[2:][:-5]
            self.update_trigger.emit(self.cc)



    def stop(self):
        self.ser.close()


    def __del__(self):
        self.ser.close()
```

مقدار بادریت و همچنین پورت سریال را نیز به این ترد میدهیم و یک تابع ران جهت اجرای این ترد و شروع به کار آن در نظر میگیریم ک با **update trigger emit** در واقع هر بار که مقدار پورت سریال عوض شود این بخش اجرا و خوانده میشود .

با توجه به این ک در ترد های کاملا مجزا کار میکنیم گذاشتن دستور **while** برای ما در این بخش هیچ مشکلی ایجاد نمیکند.

تابع **stop** را برای آن در نظر میگیریم ک پورت سریال ما در در هنگامی ک کاری با آن نداریم آزاد سازد و دسترسی آن را به سایر برنامه ها یا به اجرای مجدد برنامه توسط پایتون بدهد.

حال بررسی میکنیم ک این **thread** چگونه به **thread** اصلی ما متصل میشود.برای این کار داریم :

```
self.SerialUpdate=serialupdate()
self.SerialUpdate.update_trigger.connect(self.TestBoxUpdate)
self.SerialUpdate.start()
```

در ای بخش کلاس **serialupdate** را که نوشته بودیم فراخوانی میکنیم و سپس در خط دوم با آمدم ترگیر نوشته ی درون باکس داخل **GUI** را عوض کرده و به مقدار خوانده شده از طریق سنسور تغییر میدهیم.

و سپس با آمدن خط استارت اجازه اتصال را به پورت میدهیم.

```
self.stopButton.clicked.connect(self.stopbutton)
self.startButton.clicked.connect(self.startbutton)
```

توابع بالا جهت کار راحتتر بوسیله ی سیگنال به یک دکمه در **GUI**متصل شده اند تا با آنها بتوانیم پورت سیگنال را آزاد و یا مشغول سازیم.

تعریف این توابع به صورت زیر است:

```
def TestBoxUpdate(self,cc):
    self.text.setText(cc)

def stopbutton(self):
    self.SerialUpdate.stop()


def startbutton(self):
    self.SerialUpdate.start()
```

از ترد سریال برای این توابع استفاده شده است و مقدار **CC** نیز ک همان مقدار نور تشخیص داده شده است را در تکست باکس ریخته ایم.

همچنین در صورتی که بخواهیم یک صفحه جدید در **GUI** جهت تنظیمات کلی سیستم باز کنیم میتوانیم از کد زیر بهره ببریم .البته این بخش ناقص میباشد اما در صورت نیاز کاربر برای پروژه های آتی میتوان تکمیلش کرد.

```
#==============================
class secendclass(Form1, QDialog):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
#==============================

Form = uic.loadUiType(os.path.join(os.getcwd(), 'gui.ui'))[0]
Form1 = uic.loadUiType(os.path.join(os.getcwd(), '2.ui'))[0]
```

برای باز کردن صفحات GUI طراحی شده از کد بالا بهره میبریم و با زدن دکمه ی setting Button صفحه جدید برای ما باز میشود:

```
def settingbutton(self):
    self.dialog1.show()
    self.dialog1=secondclass()
    self.dialog1.show()
```

حال نتایج را در بخش GUI مشاهده میکنیم؛با تغییر نور سنسور مقدار عدد داخل باکس نیز تغییر میکند: