# به نام خدا

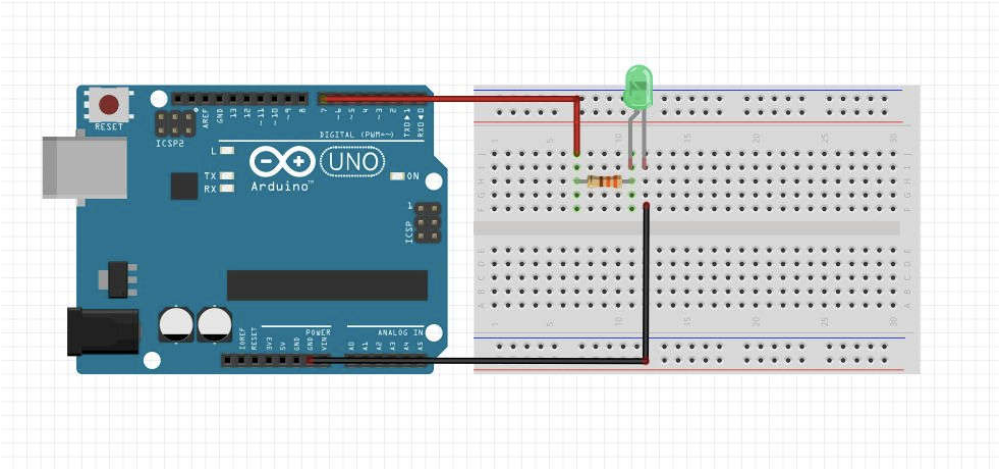
## مسابقه ی HardWar

## مستند قسمت Arduino

# A) دست گرمی:

## ۱. LED

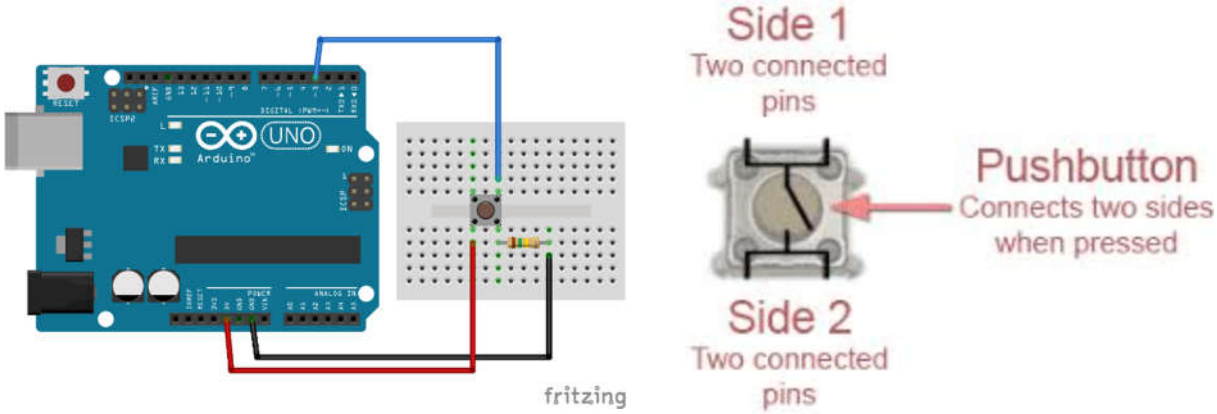
ابتدا یک مدار LED را طبق شکل زیر ببندید. سپس برنامه ای بنویسید که LED را روشن و خاموش کند:




مقدار مقاومت برابر K ۱ است

## ۲. push button

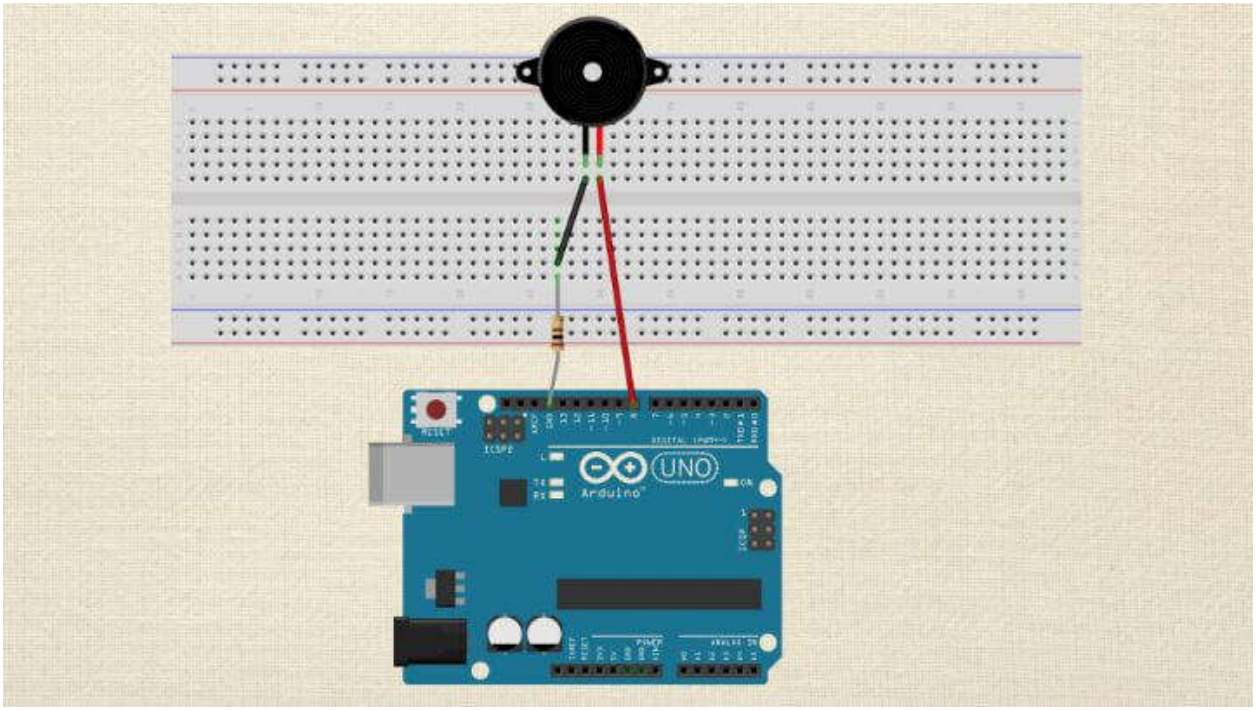
نحوه کار push button در شکل زیرتوضیح داده شده است:




در شکل سمت چپ نحوه بسته شدن مدار push button آمده است(مقدار مقاومت میتواند ۱۰ K یا ۱ K باشد).

به مدار سوال قبل یک دکمه اضافه کنید و آردوینو را طوری برنامه ریزی کنید تا با فشردن دکمه، LED روشن و با آزاد کردن آن LED خاموش شود.

## ۳. Buzzer



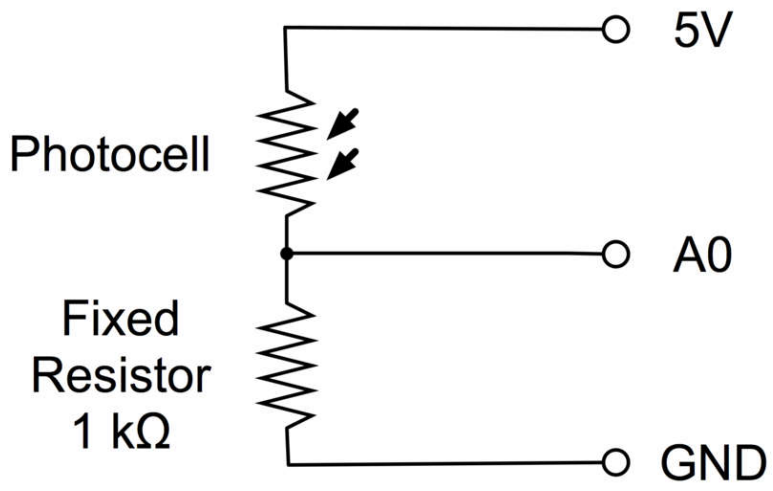

(مقدار مقاومت ترجیها K ۱).

سیستمی طراحی کنید که با دکمه های L تا A بر روی کیبورد و به کمک Serial Monitor نوت های پیانو بر روی بازر اجرا شود.

(راهنمایی: از تابع tone استفاده کنید)

## ٤. Photocell

سنسور شدت نور یا Photocell بصورت زیر بسته می‌شود که در این مدار سیگنال A0 شدت نور را بصورت یک سیگنال بین ۰ تا ۵ (analog) ولت نمایش می‌دهد:



# B) کار با ماژول

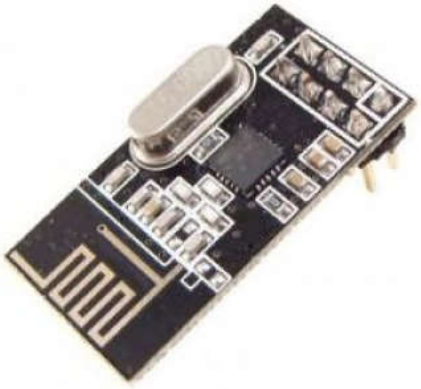
## ۵. JoyStick

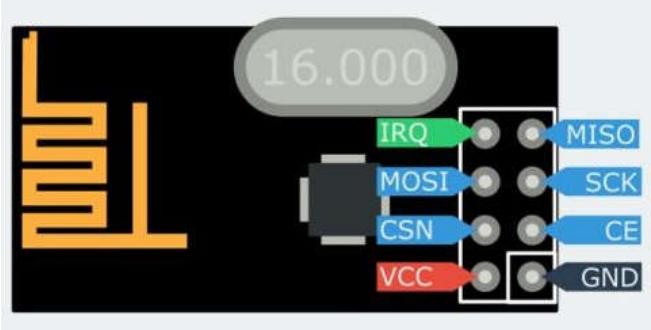
ماژول JoyStick دارای ۵ پایه است که دو پایه آن برای ورودی ولتاژ (v,GND ۵) و دو پایه آن خروجی سیگنال‌های آنالوگ برای نمایش حالت JoyStick در راستای y و x است. یک پایه باقیمانده نیز برای کلیدی است که هنگام فشار دادن JoyStick فعال می‌شود.

برنامه‌ای بنویسید که مقدارهای آنالوگ جوی استیک را خوانده و در Serial Monitor نمایش دهد.

## ٦. ماژول بی سیم (NRF24L01)

ماژول NRF24L01 برای ایجاد ارتباط رادیویی در فرکانس های بین ۲,۴ GHz تا ۲,۵ GHz طراحی شده است. این ماژول می‌تواند در یکی از دو حالت فرستنده یا گیرنده قرار بگیرد. برای ایجاد ارتباط باید فرکانس (Channel) و آدرس (Address) فرستنده و گیرنده یکسان باشد.






3.3V - VCC
GND - GND
8 - CSN
7 - CE
13 - SCK
11 - MOSI
12 - MISO

نحوه ارتباط این ماژول به وسیله پروتوکل SPI است. برای اینکه درگیر راه اندازی این پروتوکل نشویم می‌توانیم از کتابخانه‌های موجود استفاده کنیم. فایل zip. کتابخانه RF24 را از لینک
https://github.com/nRF24/RF24/archive/master.zip

دانلود کنید یا از برگزار کننده‌ها دریافت کنید.

سپس در برنامه آردوینو از **Sketch -> Include Library -> Add .ZIP Library...** فایل ZIP. دریافت شده را انتخاب کنید.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
```

سپس یک شی از کلاس radio ایجاد می‌کنیم و پایه‌هایی از آردوینو که می‌خواهیم به پین‌های CE و CSN وصل کنیم را مشخص می‌کنیم:

```
RF24 radio(7, 8); // CE, CSN
```

در تابع ()setup ارتباط SPI میان آردوینو و ماژول را با فراخوانی متد زیر شروع می‌کنیم:

```
radio.begin();
```

سپس به وسیله متد openWritingPipe آدرس ارسال را تنظیم می‌کنیم (در اینجا آدرس را ۱۲۳۴۵ قرار می‌دهیم):

```
const byte address[6] = "12345";
radio.openWritingPipe(address);
```

سپس قدرت ارسال را تنظیم می‌کنیم (در اینجا ماکسیمم می‌گذاریم):

```
radio.setPALevel(RF24_PA_HIGH);
```

به کمک متد setChannel(uint8_t channel) می‌توان فرکانس ماژول را تنظیم کرد، باید توجه کرد که ورودی تابع عددی بین ۰ تا ۱۲۷ است و بعد از اجرای این دستور فرکانس برابر channel + ۲۴۰۰ خواهد شد:

```
radio.setChannel(0);
```

با فراخوانی متد زیر، ماژول از حالت پیش‌فرض خود که گیرنده است به حالت فرستنده می‌رود:

```
radio.stopListening();
```

در ادامه برنامه هر کجا که متد write(const void* buf, uint8_t len) فراخوانی می‌شود، پارامترهای داده شده به تابع از طریق بی سیم ارسال می‌شود، برای مثال:

```
const char text[] = "GATUINO ROCKS!!!";
radio.write(&text, sizeof(text));
```

برنامه ای بنویسید که هر داده‌ای که از Serial کامپیوتر دریافت می‌شود را از طریق بی سیم به فرکانس ۲,۴□□□ و آدرس "۱۲۳۴۵" ارسال کند.

راهنمایی: Arduino NRF24l01 را گوگل کنید :)

## ۷. دات ماتریس :

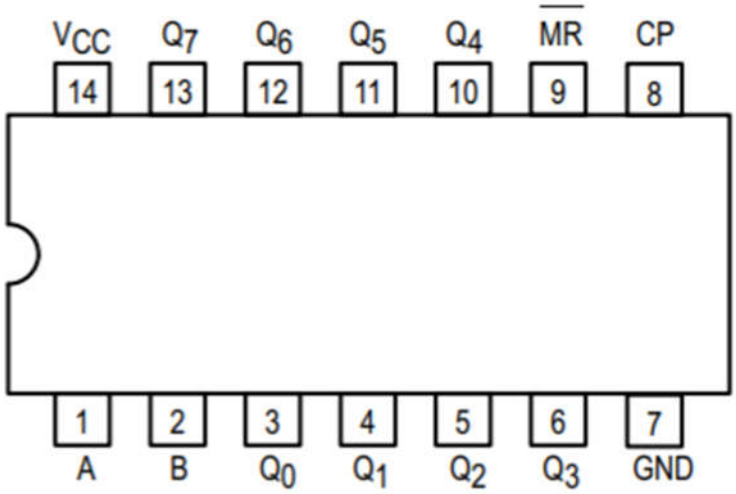
دات ماتریس مجموعه ای از LED هایی است که به صورت منظم در یک قالب چیده شده اند . دات ماتریس هایی که در اختیار دارید ۸ x ۸ اند . نحوه ی بررسی و اتصال پین های دات ماتریس به شکل زیر است . توجه داشته باشید که برای روشن کردن ستون ها پین مربوط به آن باید به GND و برای روشن کردن سطر ها به VCC باید وصل باشد .

دات ماتریس ۲۴ پین دارد که این پین ها به صورتی که در شکل نشان داده شده تقسیم شده اند و هر LED در این ماژول ۴ حالت دارد ؛ خاموش ، سبز ، قرمز و نارنجی ( ترکیب قرمز و سبز )

برای اینکه با نحوه کار پین ها آشنا شوید می توانید آنها را با توجه به اطلاعات گفته شده ببندید و آن را تست کنید . ( این قسمت امتیازی ندارد )

## ۸. شیفت رجیستر ( ۷۴۱۶۴) :

مجموعه ای از فلیپ فلاپ ها است که به صورت همگام و در یک کلاک مشترک کار می کنند و می تواند آرایه ی باینری ورودی را ۱ بیت به راست یا چپ انتقال دهد . توجه داشته باشید ( شکل پایین ) که ورودی های A و B به هم متصل باید باشند و MR ( Clear ) دائما به Vcc وصل شود .




در این قسمت شما با استفاده از شیفت رجیستر باید سه مرحله از کار های زیر را انجام دهید :

- ابتدا باید بتوانید با استفاده از شیفت رجیستر ردیف های دات ماتریس را کنترل نمایید ؛ یعنی باید یکی از ردیف ها را به یکی از رنگ ها ( به انتخاب خودتان ) روشن کنید .
- ردیف ها را باید یکی در میان قرمز و سبز کنید . ( کل یک ردیف به رنگ قرمز ، کل ردیف بعدی به رنگ سبز و کل ردیف بعدی به رنگ قرمز و کل ردیف بعد به رنگ سبز و ... )
- دات ماتریس را باید به صورت یک صفحه ی شطرنجی از رنگ های سبز و قرمز در آورید ؛ در واقع باید هر کدام از سطر ها و ستون ها توسط آردوییینو کنترل شوند .

# داستان بازی اصلی مسابقه :

شما و رقیبان شما در یک نقشه ای قرار دارید که حاوی سکه است و شما در این نقشه و با استفاده از حرکت دادن شخصیت خودتون باید این سکه ها را گرفته و امتیاز کسب کنید . در این نقشه هر کاراکتر رنگ مربوط به خود را دارد :

سکه ها زرد رنگ ،

دیوار ها قرمز رنگ ،

زمین بایر بازی سیاه ( LED خاموش ) ،

بازیکن ها سبز رنگ هستند .

اگر دو بازیکن در یک خانه قرار بگیرند بین آنها یک دوئل از جنس بازی Connect4 شروع می شود .

( بازی Connect4 به صورتی است که هر بازیکن یک مهره را در یک ستون از ستون های موجود می اندازد ، اگر هر بازیکن توانست ۴ مهره بصورت افقی یا عمودی یا مورّب پشت هم بچیند برنده ی بازی است )

این بازی توسط دکمه و JoyStick انجام می شود .

بازیکنی که دوئل را ببرد ۵ سکه می گیرد و بازیکن بازنده نه چیزی از دست می دهد نه چیزی بدست می آورد و اگر بازی مساوی شود به هر کدام یک سکه می رسد و پس از اتمام بازی بازیکن ها به خانه های تصادفی در نقشه فرستاده می شوند .

*** توجه کنید که هیچ بازیکنی نمی تواند بیشتر از ۵ سکه متوالی بگیرد مگر این که حداقل یک دوئل ( با هر نتیجه ای ) انجام داده باشد .

## چالش :

شما از طریق کلاینتی که در اختیارتون قرار داده شده با سرور ارتباط برقرار می کنید به این نحو که در هر بار سرور یک پک ۱۶ بایتی برای شما ارسال می کند که در واقع نقشه ی بازی است ؛ هر ۲ تا بایت نشان دهنده ی یک ردیف است و هر ۲ بیت نشان دهنده ی یک خانه در ردیف است که ممکن است یکی از اعداد ۰۰ ، ۰۱ ، ۱۰ و ۱۱ را داشته باشند که به ترتیب به معنی رنگ های سیاه ، قرمز ، سبز و نارنجی است .

شما در خروجی یک پک ۵ بایتی به سرور برمیگردانید که بایت اول آن عدد ۵۵ در مبنا ی ۱۶ و بایت آخر آن عدد AA در مبنای ۱۶ باید باشد و بایت دوم آن مولفه ی افقی بازیکن ( X ) و بایت سوم مولفه ی عمودی آن ( Y ) می باشد و در بایت چهارم  در صورت پایین بودن دکمه FF در مبنای ۱۶ و در صورت بالا بودن دکمه ، ۰۰ را می فرستید .

در واقع این چالش استفاده ی ترکیبی از Serial ، Button ، JoyStick ، Dot Matrix و Shift Register می باشد که در قسمت های قبلی مسابقه با آنها کار کرده اید .

## توابع کاربردی:

pinMode(pinNumber, pinMode)

تعیین نوع خروجی (INPUT یا OUTPUT)

digitalWrite(pin, value)

نوشتن مقدار HIGH یا LOW روی پین دیجیتال

digitalRead(pin)

خواندن مقدار پین دیجیتال (خروجی HIGH یا LOW)

analogRead(pin)

خواندن مقدار پین آنالوگ(خروجی بین ۰ – ۱۰۲۳)

analogWrite(pin,value)

نوشتن مقدار روی پین آنالوگ(تولید موج PWM)

tone(pin, frequency, duration)

یک موج به اندازه فرکانس گفته شده روی پین تولید می‌کند. مدت زمان تولید موح را نیز می‌توان مشخص کرد(دلخواه).

noTone(pin)

در صورتی که مدت زمان برای تابع tone() تعریف نشده باشد با فراخوانی این تابع تولید موج متوقف می‌شود.

millis()

مدت زمان گذشته از لحظه شروع به کار آردوینو را به میلی ثانیه به عنوان خروجی می‌دهد.

delay(ms)

ایجاد تاخیر به میلی‌ثانیه

delayMicroseconds(us)

ایجاد تاخیر به میکرو ثانیه

map(value, fromLow, fromHigh, toLow, toHigh)

مقدار value که بین fromLow تا fromHigh می‌باشد را به عددی بین toLow تا toHigh مپ می‌کند و به عنوان خروجی برمی‌گرداند.

Serial.begin(BuadRate)

Serial.print()

Serial.read()

Serial.readBytes(buffer, length)

برای برقراری ارتباط سریال بین آردوینو و کامپیوتر از این توابع استفاده می‌شود (توسط کابل USB). ابتدا تابع begin() را با مقدار ۹۶۰۰ به عنوان مثال صدا بزنید. سپس برای گرفتن اطلاعات از کامپیوتر و یا نمایش آنها روی کامپیوتر از توابع read() و print() استفاده کنید.

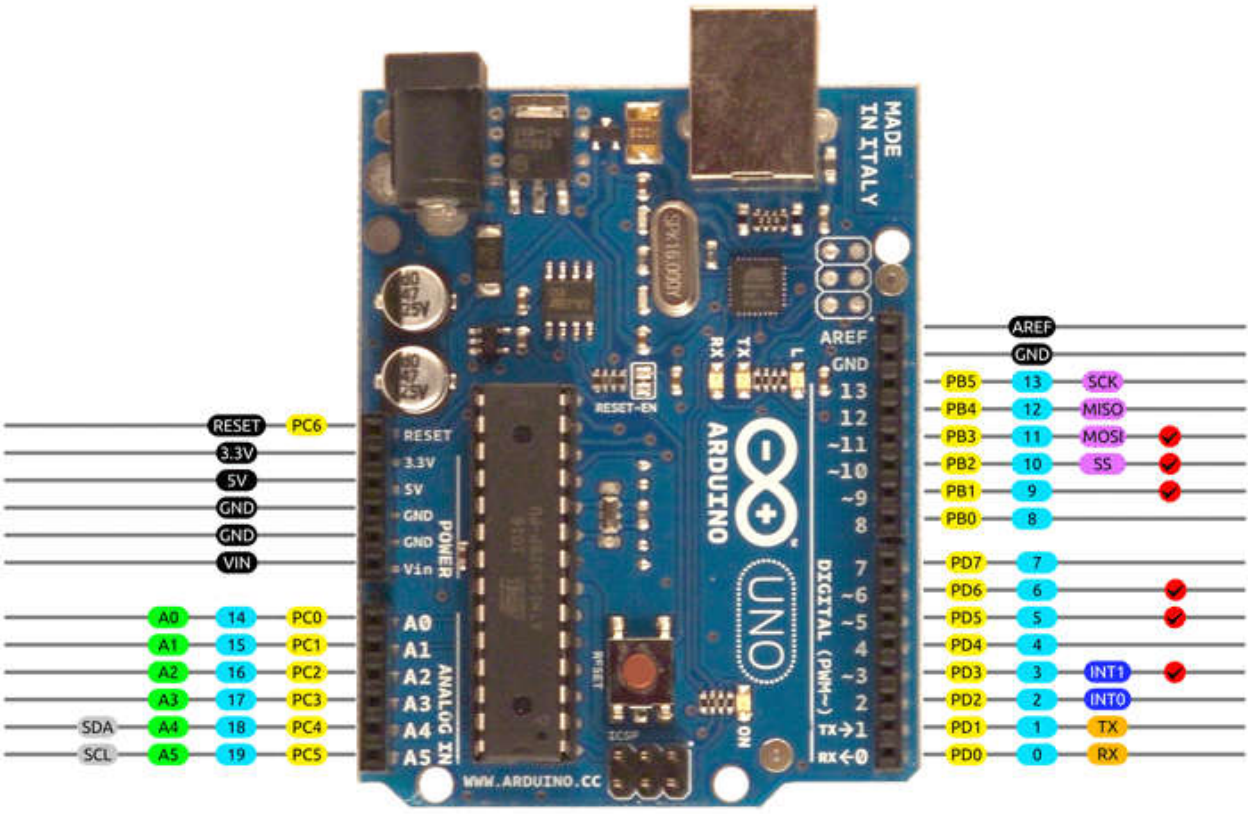
shiftOut( dataPin , clockPin , bitOrder , Value )

برای مقدار دهی شیف رجیستر از آن استفاده می شود .

clockPin و dataPin به ترتیب برای پایه های کلاک و داده ی شیفت رجیستر هستند .

Value مقدار بایتی است که میخواهیم به شیف رجیستر بدهیم .

bitOrder برای مشخص کردن این است که بایت داده شده از کدام طرف خوانده شود . ( یعنی بیت با ارزش بیشتر در طرف راست است یا چپ )




AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT