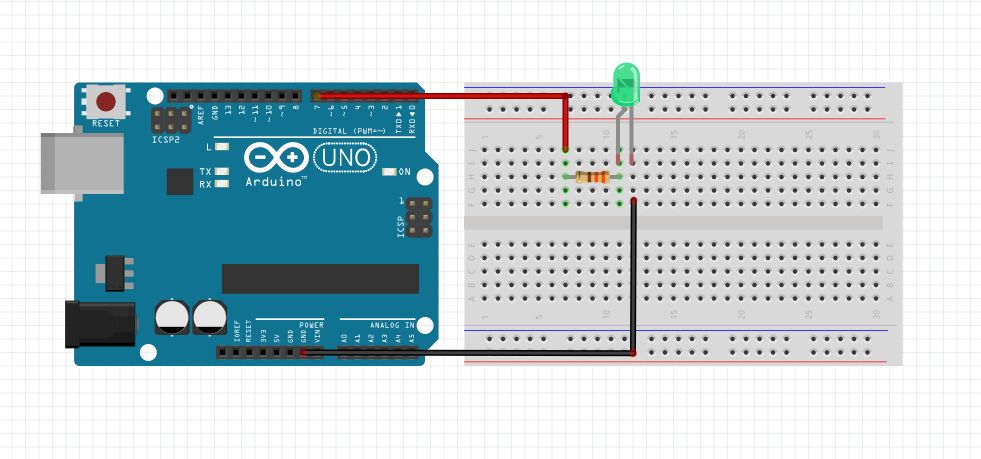
A) دست گرمی:

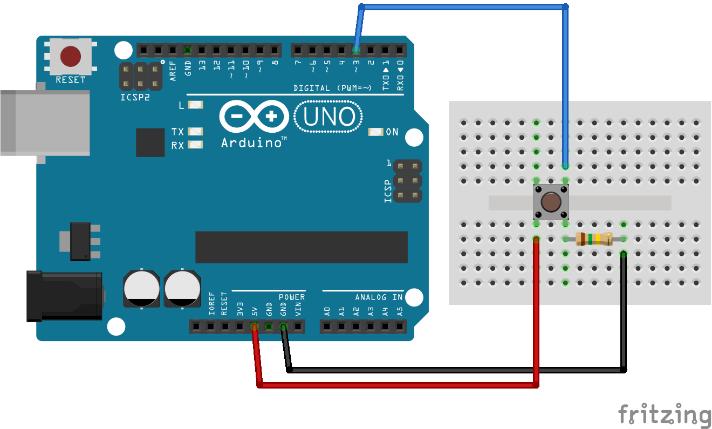
1. LED

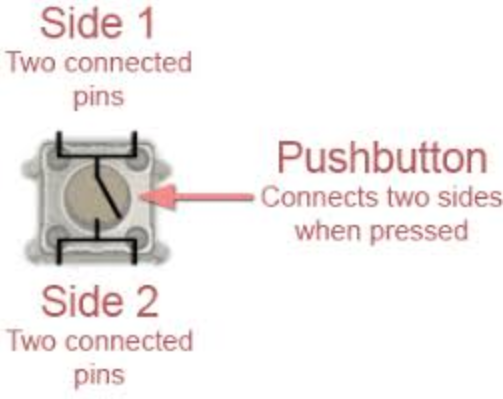
ابتدا یک مدار LED را طبق شکل زیر ببندید. سپس برنامه ای بنویسید که LED را روشن و خاموش کند:



مقدار مقاومت برابر 1K است

2. 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 10K یا 1K 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.

3. Buzzer

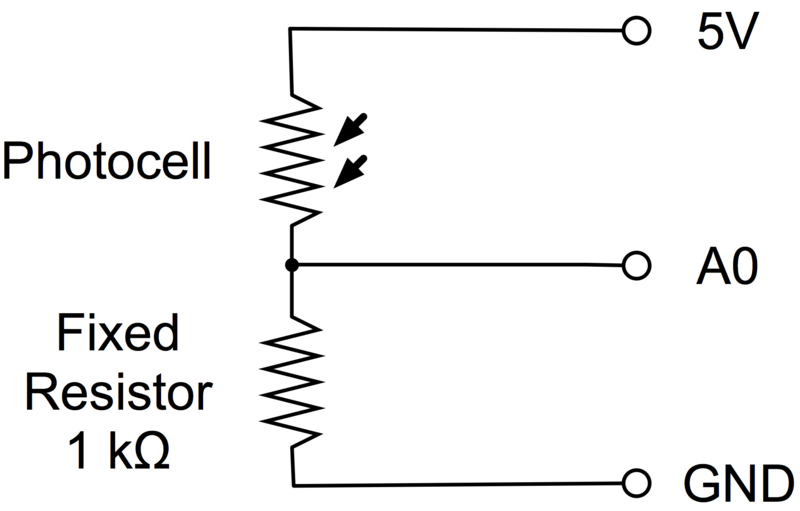
مدار Buzzer مانند LED بسته می‌شود(مقدار مقاومت ترجیها 1K).

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)

4. Photocell

سنسور شدت نور یا Photocell بصورت زیر بسته می‌شود که در این مدار سیگنال A0 شدت نور را بصورت یک سیگنال بین 0 تا 5 (analog) ولت نمایش می‌دهد:



B) کار با ماژول

5. JoyStick

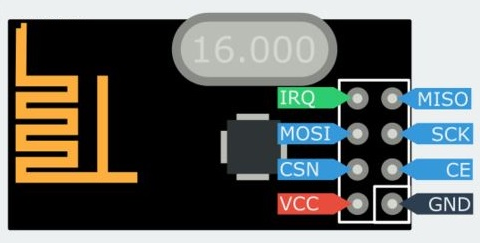
ماژول JoyStick دارای 5 پایه است که دو پایه آن برای ورودی ولتاژ (5v,GND) و دو پایه آن خروجی سیگنال‌های آنالوگ برای نمایش حالت JoyStick در راستای y و x است. یک پایه باقیمانده نیز برای کلیدی است که هنگام فشار دادن JoyStick فعال می‌شود.



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

6. ماژول بی سیم (NRF24L01)

ماژول NRF24L01 برای ایجاد ارتباط رادیویی در فرکانس های بین 2.4GHz تا 2.5GHz 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.





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.

در کد آردوینو کتاب خانه های لازم را include می‌کنیم:

#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 12345 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 0 تا 127 است و بعد از اجرای این دستور فرکانس برابر 2400 + channel 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 2.4GHz و آدرس “12345” ارسال کند.

راهنمایی: Arduino NRF24l01 را گوگل کنید :)

C) ساخت دسته بازی

حال به کمک ماژول‌ها و قطعات استفاده شده در بخش‌های قبلی مداری طراحی کنید که حداقل دارای دو دکمه، یک جوی استیک و یک ماژول NRF باشد. وظیفه این مدار این است که در فاصله های زمانی 50 میلی ثانیه داده‌های دکمه و جوی استیک را با پروتوکل زیر ارسال کند:

پروتوکل ارسال داده جوی استیک:

|  |  |  |  |
| --- | --- | --- | --- |
| کاراکتر ‘E’ | Y-AXIS | X-AXIS | کاراکتر‘J’ |
| 3 | 2 | 1 | 0 |

\* مقادیر X-AXIS و Y-AXIS باید در بازه 0تا 255 نسبت داده بشوند (برای مثال هنگامی که جوی استیک در حالت آزاد است این دو مقدار برابر حدود 127 است).

پروتوکل ارسال داده دکمه:

|  |  |  |  |
| --- | --- | --- | --- |
| کاراکتر ‘E’ | VAL | Button ID | کاراکتر‘B’ |
| 3 | 2 | 1 | 0 |

\*مقدار Button ID می‌تواند برابر 0 یا 1 شود و مقدار VAL در هنگام فشرده شدن دکمه برابر255 (0xFF HEX) و در هنگام آزاد بودن برابر 0 (0x00) شود.

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 0 – 1023)

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 9600 به عنوان مثال صدا بزنید. سپس برای گرفتن اطلاعات از کامپیوتر و یا نمایش آنها روی کامپیوتر از توابع read() و print() استفاده کنید.

