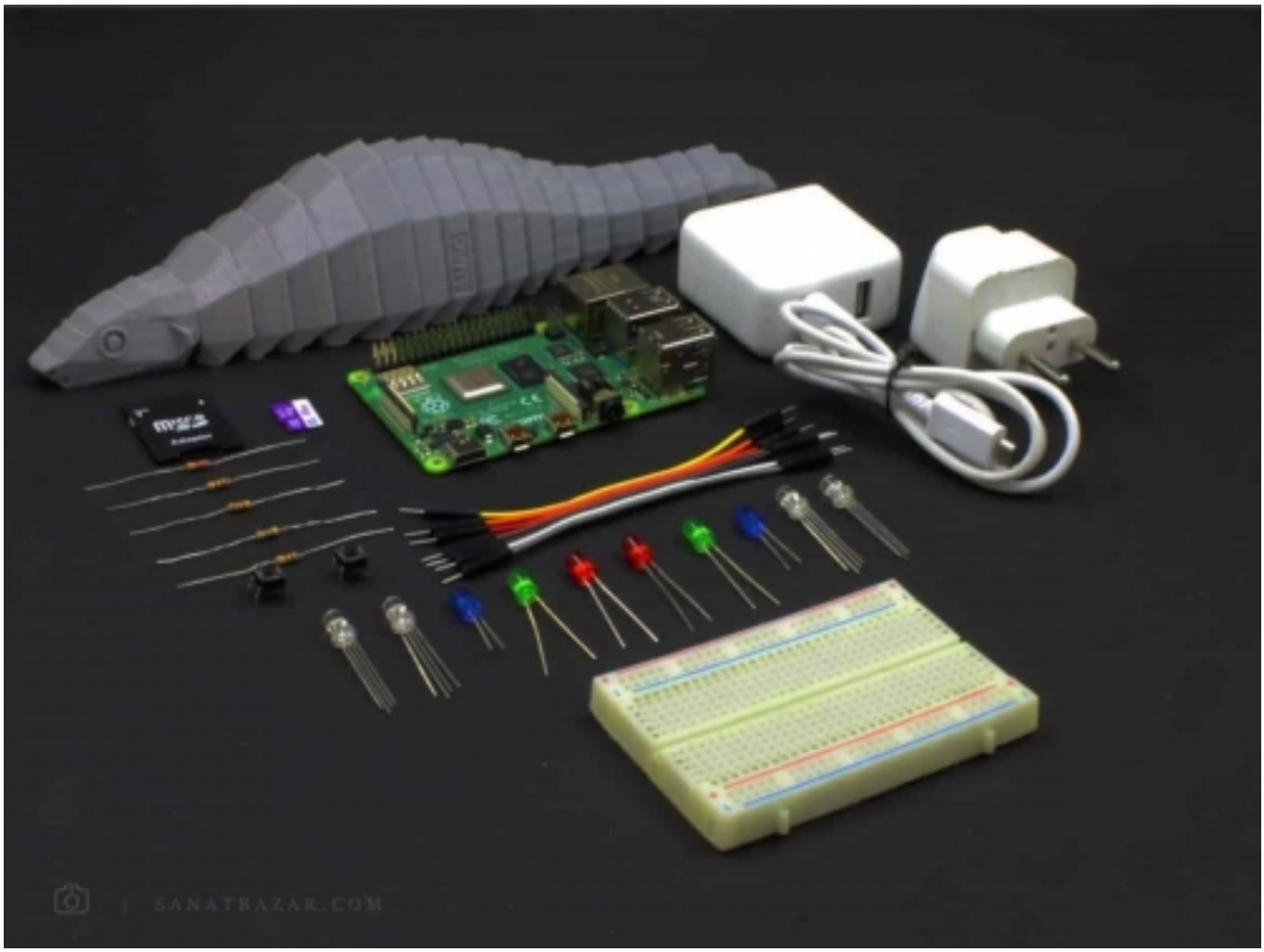
آموزش جامع راه‌اندازی GPIO رزبری‌پای



شاید در نگاه اول، عنوان این آموزش ساده و پیش پا افتاده به نظر بیاد. اما بهترین کار برای یادگیری و انجام پروژه در زمینه‌های مختلف، شروع کردن از پایه و ابتدا است. در غیر این صورت بعداً برای انجام یک دستور و پروژه‌ی ساده، باید کلی وقت توی اینترنت تلف کنید. حتی بدتر! (بارها شده خودم چند روز درگیر کدنویسی یه برنامه بودم، آخرشم اونطوری که می‌خواستم انجامش ندادم. اما بعداً دیدم برای اون کار یه دستور ساده وجود داشته!). در این بخش از آموزش مقدماتی رزبری‌پای قصد داریم اولین و اساسی ترین قسمت انجام یک پروژه‌ی عملی با رزبری‌پای، یعنی کار با پایه‌های ورودی خروجی و وقفه‌ها را باهم بررسی کنیم. در این پروژه راه‌اندازی LED اهمیتی ندارد، بلکه هدف آشنا کردن شما با دستورات مهم و کاربردی پایتون برای کار با GPIOها در قالب یک مثال جذاب است. پس با همراه باشید تا موارد زیر را خیلی ساده و روان و فقط در طول چند دقیقه به شما معرفی کنم:

- آشنایی با پین‌های رزبری‌پای و عملکرد آن‌ها
- آموزش خواندن و نوشتن روی پین‌های برد با پایتون
- آموزش نحوه‌ی استفاده از تاخیر و حلقه‌های بی‌نهایت در برنامه‌نویسی پایتون
- آموزش کار با وقفه‌ها توسط پایتون (مناسب برای تشخیص لبه‌های پالس ورودی)
- آموزش کار با کتابخانه‌ی WiringPi (C++)

قطعات مورد نیاز

رزبری‌پای 4

کارت حافظه با ظرفیت 8 GB یا بیشتر

LED

RGB LED

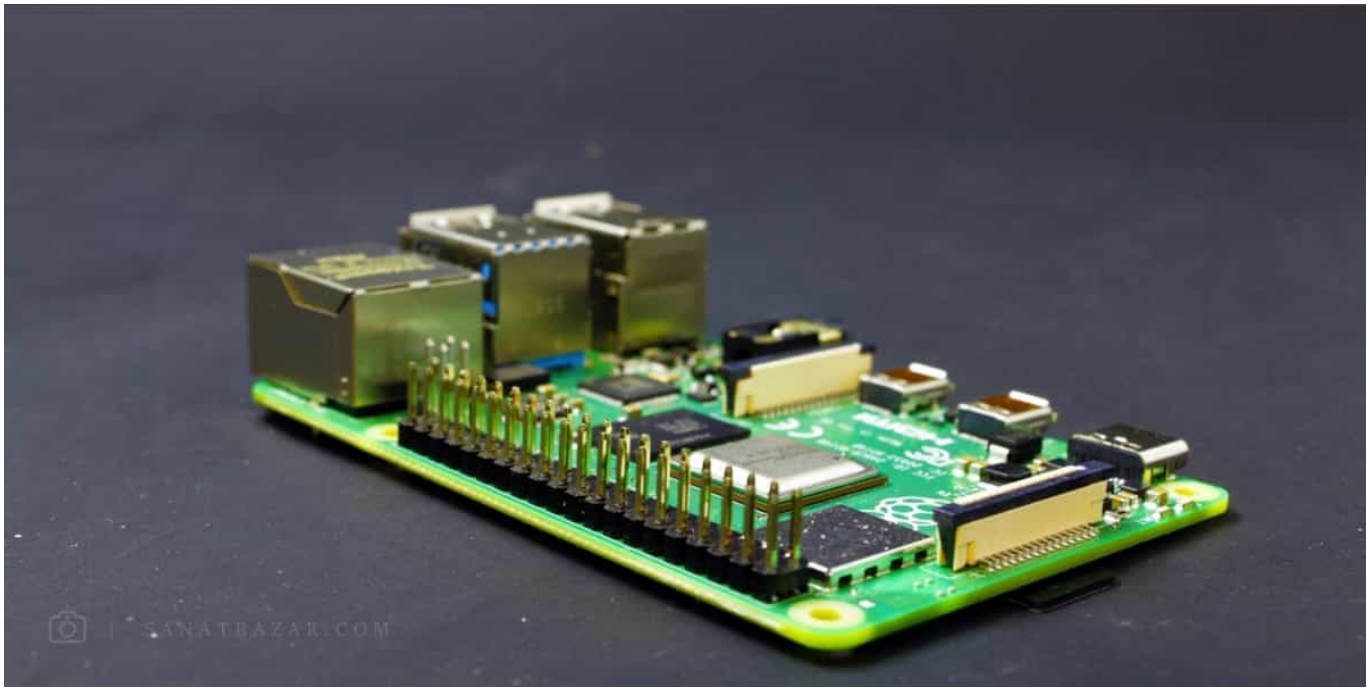
Breadboard

مقاومت 470Ω

سیم

کلید

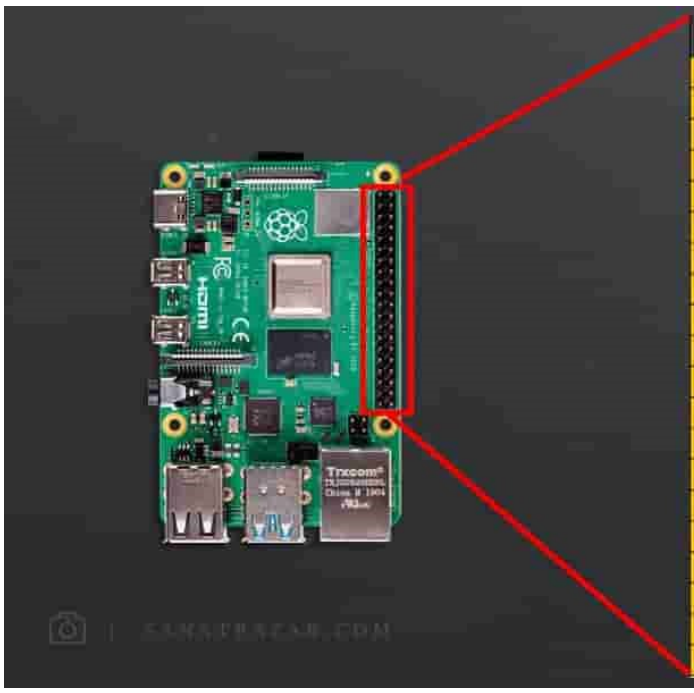
هر آنچه درباره‌ی GPIOهای رزبری‌پای لازم است بدانید !



یکی از عوامل فروش فوق‌العاده‌ی بردهای رزبری‌پای در سراسر جهان، برخورداری این برد از پایه‌های ورودی خروجی برای ارتباط با قطعات الکترونیکی است. با داشتن دانش کمی از برنامه‌نویسی، به راحتی می‌توانید به این پین ها دستور داده و ماژول‌های مختلفی را راه‌اندازی کنید. همانطور که می‌دانید، بردهای رزبری‌پای همگی دارای ۴۰ پایه‌ی ورودی خروجی‌اند که از این ۴۰ پایه، ۲۳ پایه برای خواندن و نوشتن داده‌ها استفاده می‌شود (GPIO). نکته‌ی جالب درباره‌ی بردهای رزبری‌پای، شباهت کامل پین‌ها و شماره‌گذاری آن‌ها در تمامی مدل‌ها است. به‌طوری که مثلاً شماره و عملکرد پین‌ها در رزبری‌پای ۴، دقیقاً مشابه رزبری‌پای 3B+ و سایر مدل‌های پیشین است. همانطور که در بخش معرفی و مشخصات برد رزبری‌پای ۴ اشاره کردیم، برای شماره‌گذاری و کار با پین‌های رزبری‌پای از دو استاندارد BCM و BOARD استفاده می‌شود:

BOARD: در این استاندارد شماره‌گذاری پین‌ها دقیقاً مطابق محل و ترتیب قرارگیری آن‌ها روی برد از ۱ تا ۴۰ انجام می‌شود. استفاده از این استاندارد برای کسانی که خیلی با برد و پین‌های آن آشنایی ندارند راحت‌تر است.

BCM: در این روش، شماره‌گذاری بر اساس استاندارد شرکت Broadcom انجام می‌شود. از مزیت BCM نسبت به BOARD می‌توان به این نکته اشاره کرد که در BCM، تنها پایه‌های GPIO دارای شماره هستند. در این صورت امکان اشتباه در تعیین سایر پایه‌ها (مانند GND و VCC) به عنوان ورودی خروجی وجود ندارد. اما در طرف مقابل، در این روش پایه‌ها به ترتیب شماره‌گذاری نشده‌اند. در تصویر زیر شماره‌ی پین‌ها برای برد رزبری‌پای ۴ نمایش داده شده است. البته این شماره پین‌ها برای سایر بردهای زبری‌پای نیز قابل استفاده‌اند.



شماره پین BCM (عملکرد)	شماره پین BOARD	شماره پین BOARD	شماره پین BCM (عملکرد)
3V3	1	2	5V
GPIO 2 (SDA1,I2C)	3	4	5V
GPIO 3 (SCL1,I2C)	5	6	GND
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD0,UART)
GND	9	10	GPIO 15(RXD0,UART)
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3V3	17	18	GPIO 24
GPIO 10 (SPI0_MOSI)	19	20	GND
GPIO 9 (SPI0_MISO)	21	22	GPIO 25
GPIO 11 (SPI0_CLK)	23	24	GPIO 8 (SPI0_CE0 N)
GND	25	26	GPIO 7 (SPI0_CE1 N)
(SDA0,I2C)	27	28	(SCL0,I2C)
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

برای مشاهده‌ی شماره پین‌ها می‌توانید از دستور $ pinout در Command Line رزبری‌پای استفاده کنید.

اجازه بدید قبل از شروع برنامه‌نویسی، مرور خیلی کوتاهی داشته باشیم بر عملکرد هریک از پایه‌های این برد. در اینجا پایه‌ها را بر اساس استاندارد BOARD توضیح می‌دهیم.

پین‌های تغذیه: از این پین‌ها برای تغذیه‌ی سایر بردها و قطعات الکترونیکی استفاده می‌شود. پین‌های ۲ و ۴، ۷۵ و پین‌های ۱ و ۱۷، ۷۳.۳ می‌باشند. حداکثر جریان قابل تحمل برای این پین‌ها به‌صورت غیر رسمی ۵۰mA تعیین شده است.

با توجه به این که پین‌های تغذیه به مدار داخلی برد متصل‌اند، در هنگام استفاده مراقب اتصال کوتاه و حداکثر جریانی که از این پین‌ها می‌کشید باشید!

GND: زمین مداز (پین‌های ۶، ۹، ۱۴، ۲۰، ۲۵، ۳۰، ۳۴ و ۳۹)

SDA1 و SCL1: از این پایه‌ها برای برقراری ارتباط I2C استفاده می‌شود. I2C یک پروتکل ارتباط سریال همزمان می‌باشد که در آن انتقال داده توسط دو سیم انجام می‌شود. SDA (پین ۳) وظیفه‌ی انتقال داده و SCL (پین ۵) وظیفه‌ی ایجاد کلاک پالس برای انتقال آن را دارد. انتقال توسط I2C نیمه دوطرفه و ارزان است. برای اطلاعات بیشتر می‌توانید به صفحه‌ی آشنایی با I2C مراجعه کنید.

GPCLK0: به کمک پایه‌ی ۷ می‌توانید کلاک با دقت بالا ایجاد کنید.

SPI0_CE0 N، SPI0CLK، SPI0_MISO، SPI0_MOSI و SPI0CE1 N، به‌منظور برقراری ارتباط سریال با پروتکل SPI، باید از این پایه‌ها استفاده کنید. SPI نوعی رابط سریال سنکرون دو طرفه است که برای فاصله‌های کم و انتقال با سرعت بالا مناسب است. برای انتقال داده با این روش به ۴ سیم SPI0_MOSI (پایه‌ی ۱۹) ، SPI0_MISO (پایه‌ی ۲۱)، SPI0_CLK (پایه‌ی ۲۳) و SPI0_CE0 N (پایه‌ی ۲۴) یا SPI0_CE1 N (پایه‌ی ۲۶) برای انتخاب Slave، نیاز دارید. برای اطلاعات بیشتر می‌توانید به صفحه‌ی آشنایی با SPI مراجعه کنید.

TXD0 و RXD0: برای استفاده از رابط سریال UART هم می‌توانید از پین‌های ۸ (TXD0) و ۱۰ (RXD0) استفاده کنید. UART یک مدار فیزیکی برای انتقال داده‌ها به‌صورت آسنکرون و سریال است. این روش انتقال به دلیل سادگی، کاربرد زیادی دارد. برای اطلاعات بیشتر می‌توانید به صفحه‌ی آشنایی با UART مراجعه کنید.

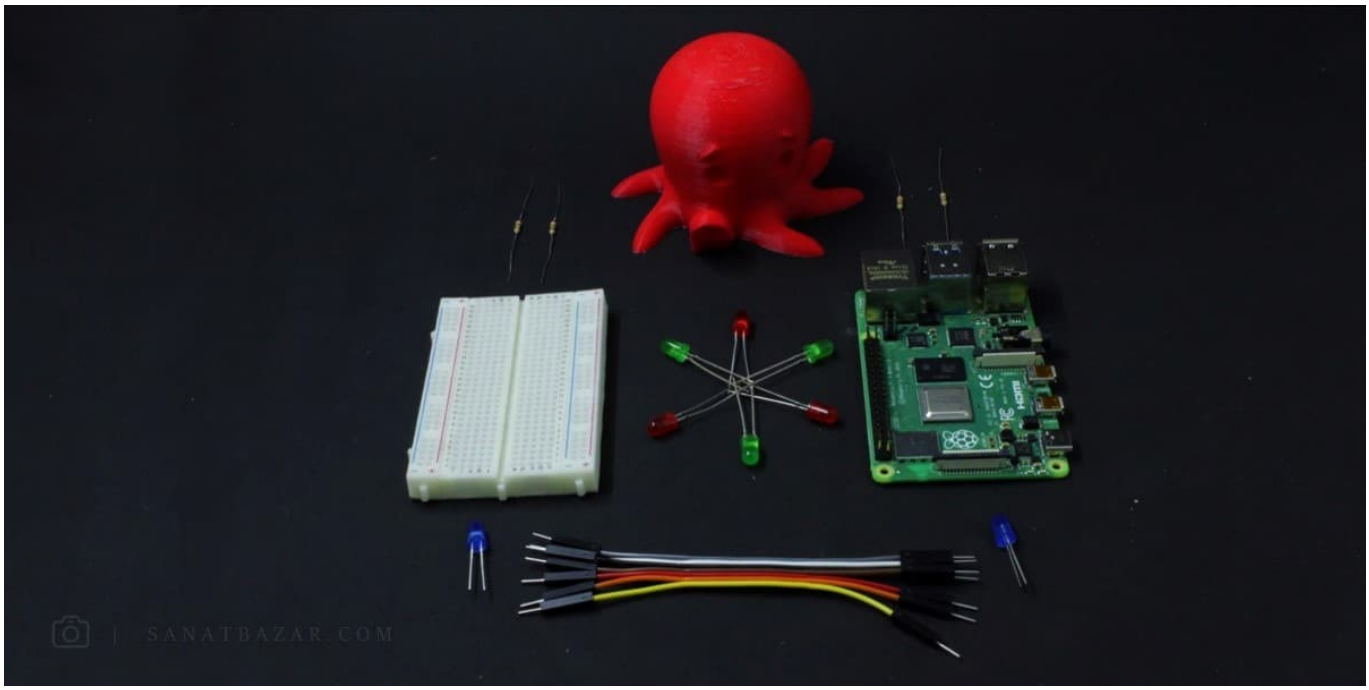
برای فعال کردن SPI، I2C و UART می‌توانید با دستور $ sudo raspi-config وارد Configuration برد شده و در بخش Interfaces، رابط‌های فوق را فعال کنید.

رابط سریال UART به‌صورت پیش‌فرض برای دسترسی سریال به سیستم‌عامل از طریق لپتاپ استفاده می‌شود (Console Login) بنابراین برای استفاده از این رابط در مدارهای الکتریکی ابتدا باید در Interfaces، Login Shell را غیرفعال و سپس Serial Port Hardware را فعال کنید.

SDA0 و SDA1: از این دو پین تنها برای ارتباط I2C با حافظه‌های EEPROM در HATهای رزبری‌پای استفاده می‌شود. بنابراین از پین‌های ۲۷ و ۲۸ نمی‌توانید به عنوان GPIO استفاده کنید. برای اطلاعات بیشتر در مورد HAT ها می‌توانید به معرفی و مشخصات رزبری‌پای ۴ مراجعه کنید.

GPIO: تمامی پایه‌هایی که با نام GPIO (General Purpose Input/Output) مشخص شده‌اند (از جمله پایه‌های UART، SPI، I2C و GPCLK0) می‌توانند به عنوان ورودی خروجی استفاده شوند. حداکثر ولتاژ و جریان قابل تحمل برای این پایه‌ها به ترتیب ۳.۳ و ۱۶mA می‌باشد.

اولین پروژه: روشن کردن LED



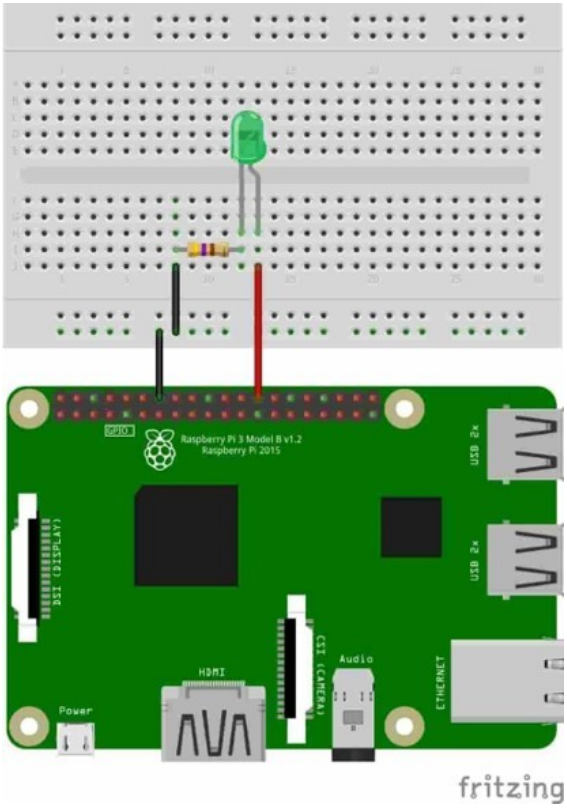
خب اولین پروژه‌ای که برای آموزش کار با GPIOها در نظر گرفتیم، روشن و خاموش کردن LED است. در این بخش قصد داریم یک LED را برای ۵ ثانیه روشن و سپس آن را خاموش کنیم. برای این کار از زبان برنامه‌نویسی پایتون و کتابخانه‌های RPi.GPIO و time استفاده می‌کنیم. در قدم اول اگر هنوز سیستم‌عاملی را روی رزبری‌پای خود نصب نکردید، با مطالعه قسمت آموزشی نصب و راه‌اندازی رزبری‌پای با سیستم‌عامل رزبین خیلی سریع، این کار را انجام بدید.

اگر با پایتون آشنا نیستید، نگران نباشید. برای یادگیری پایتون کافیست به بخش آموزش پایتون برای کار با رزبری‌پای مراجعه کنید.

پس از نصب سیسم‌عامل، اتصالات سخت‌افزاری را مطابق تصویر زیر انجام دهید. ما در اینجا از Breadboard، یک LED سبز و یک مقاومت 470Ω استفاده کردیم. شما می‌توانید به‌جای 470Ω از یک مقاومت دیگر بین 300Ω تا 1kΩ استفاده کنید. طبق تصویر:

- زمین مدار را به GND رزبری‌پای (پین ۱۴)
- مقاومت 470Ω را به زمین و کاتد LED
- آند LED را به GPIO07 رزبری‌پای (پین ۲۶)

وصل می‌کنیم.



در این آموزش فرض شده که رزبری‌پای شما از قبل دارای سیستم‌عامل است. در غیر این صورت برای نصب سیستم‌عامل می‌توانید به آموزش راه‌اندازی رزبری‌پای ۴ با نصب سیستم‌عامل رزبین مراجعه کنید.

حالا برد خود را روشن کرده و به آن متصل شوید. من ابتدا برای کدنویسی در Command Line، یک دایرکتوری به نام Project در Home Directory می‌سازم برای این کار دستورات زیر را وارد کنید:

```
$ mkdir Project
$ cd Project
$ nano led.py
```

اگر با لینوکس آشنا نیستید، نگران نباشید. برای یادگیری پایتون کافیست به بخش آموزش لینوکس برای کار با رزبری‌پای مراجعه کنید.

سپس اولین برنامه را در led.py به‌صورت زیر می‌نویسم:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com

import RPi.GPIO as GPIO
import time

LED=7
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)
```

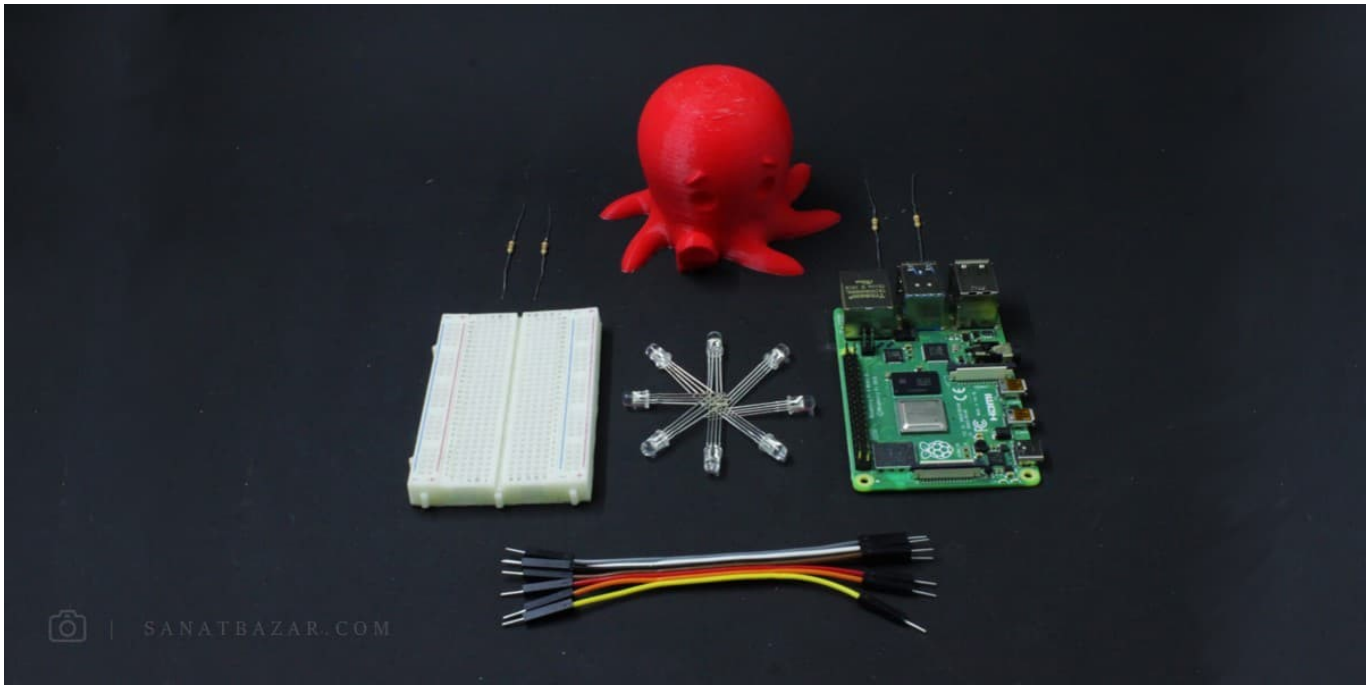
 |

```
GPIO.output(LED,GPIO.HIGH)
time.sleep(5)
GPIO.output(LED,GPIO.LOW)


GPIO.cleanup()
```

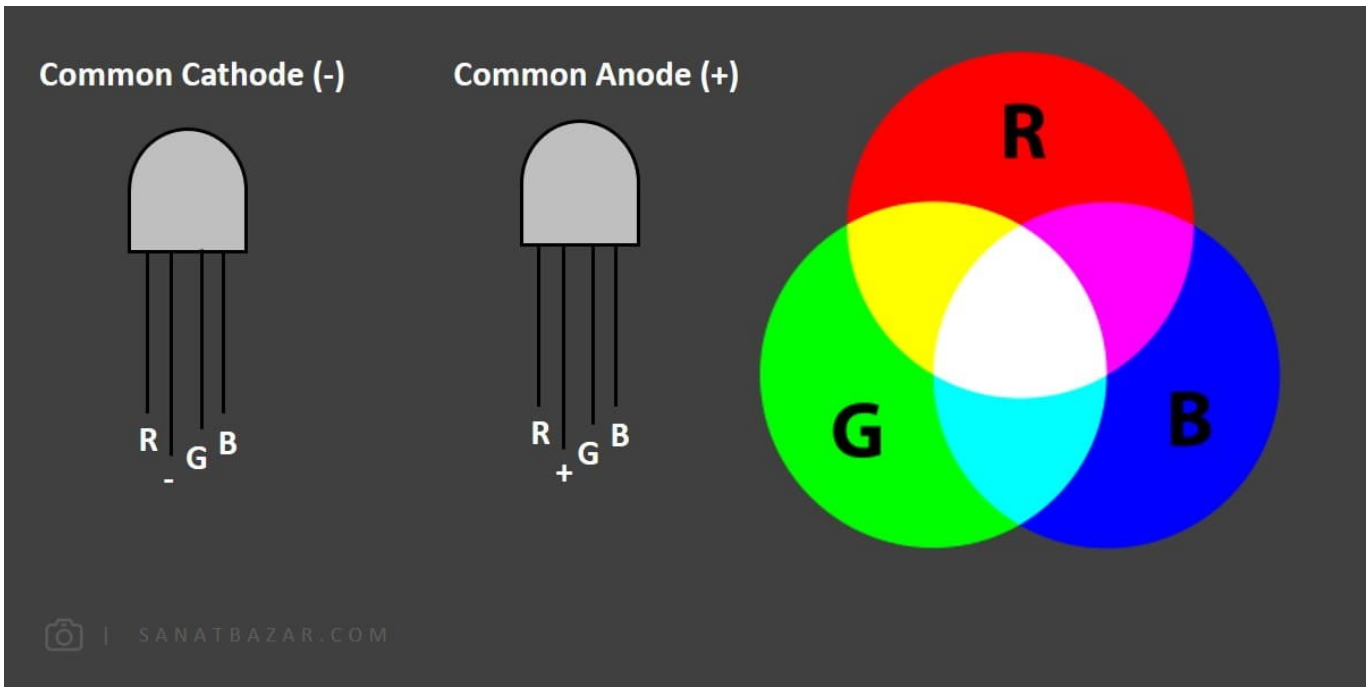
در کد بالا ابتدا کتابخانه‌ی RPi.GPIO برای دسترسی به پین‌های برد و کتابخانه‌ی time را برای محاسبه‌ی تاخیر زمانی فراخوانی می‌کنم. سپس متغیری به نام LED تعریف و شماره‌ی پایه‌ی 7 (BCM) را به آن مقداردهی می‌کنم. در ادامه استاندارد BCM را تعریف و پایه‌ی متصل به LED را به‌عنوان خروجی تعریف می‌کنیم. تا اینجا پیکربندی مدار را انجام دادیم. سپس LED را ابتدا HIGH و پس از ۵ ثانیه LOW می‌کنیم. در نهایت برای این که پیکربندی پایه‌ها برای برنامه‌ی بعدی شما Reset شود، دستور GPIO.cleanup() را وارد کنید. برای ذخیره‌ی برنامه CTRL+X را فشار داده و با وارد کردن Y و زدن Enter برنامه را ذخیره کنید. برای اجرا کافیست دستور زیر را وارد کنید:

```
$ python led.py
```

روشن کردن RGB LED: کار با پایه‌های بیشتر، آشنایی با حلقه‌های بی‌نهایت

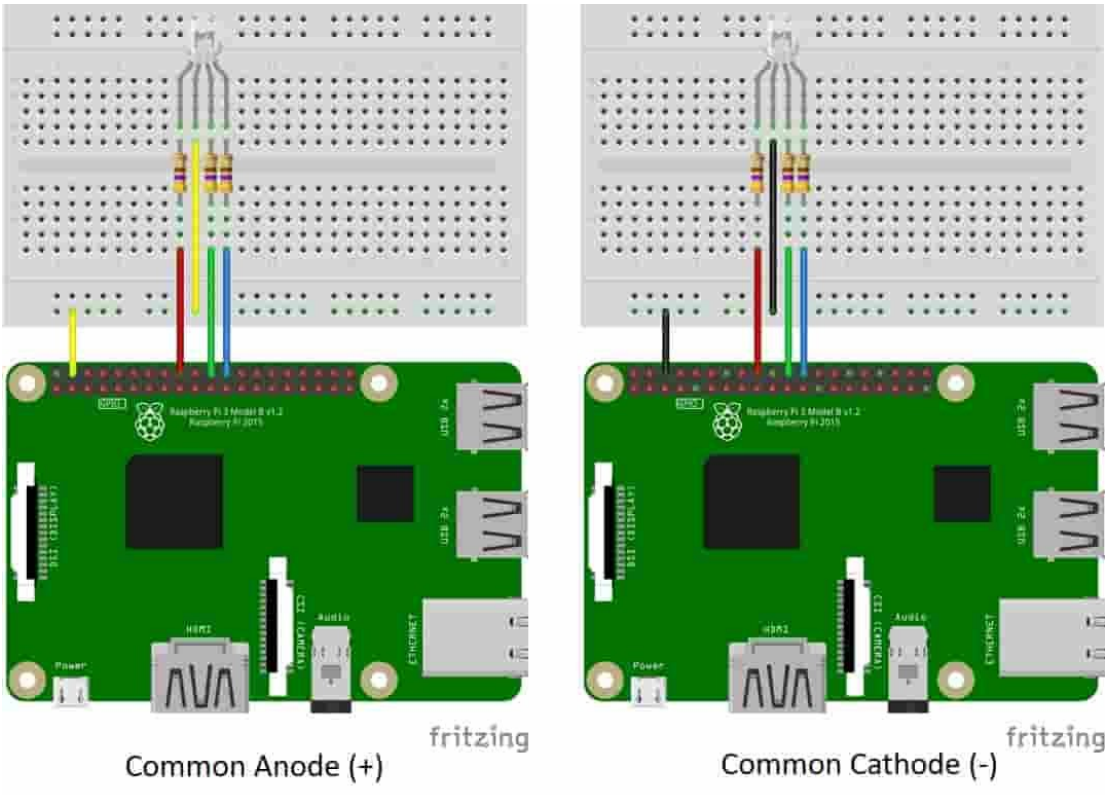


در گام بعدی می‌خواهیم برنامه و پروژه‌ی آموزشی را کمی پیچیده‌تر کنیم. هدف از انجام این پروژه آشنا کردن شما با حلقه‌های بی‌نهایت و دستورهای مهم پایتون است. بنابراین پروژه‌ی بعدی ما، روشن خاموش کردن یک LED RGB با فاصله‌ی زمانی معین است. مطابق شکل زیر، این قطعه دارای ۴ پایه است که از آن‌ها یکی مشترک و سه تای دیگر هر کدام مربوط به یک رنگ LED هستند. LED RGBها کاتد مشترکند یا آند مشترک. در حالت کاتد مشترک، پایه‌ی بلندتر زمین و برای هر سه رنگ مشترک است در حالی که در آند مشترک، این پایه VCC می‌باشد و برای روشن کردن هر رنگ باید پایه‌ی مربوط به آن را زمین کنید. رنگ‌ها و نام پایه‌ها در تصویر زیر نمایش داده شده است.



اما از کجا بفهمیم RGB LED ما آند مشترک است یا کاتد مشترک؟

برای این کار به یک مولتی‌متر نیاز دارید. عملگر آن را روی حالت Continuity یا دیود قرار دهید. سپس ابتدا سیم زمین (مشکی) مولتی متررا به پایه‌ی بلند و سیم قرمز را به سایر پایه‌ها متصل کنید. اگر LED روشن شود، کاتد مشترک است. اگر برعکس این حالت، سیم قرمز را به پایه‌ی بلند و مشکی را به سایر پایه‌ها وصل کردید و روشن شد، آند مشترک است. LED من کاتد مشترک بود اما کد و نحوه‌ی اتصال هر دو حالت را برای شما قرار می‌دهد تا با توجه به قطعه‌ی خود، از آن‌ها استفاده کنید. اکر هم اصلاً RGB LED ندارید، می‌توانید از سه LED، جداگانه استفاده کنید.



مطابق تصویر:

- زمین LED را به پایه‌ی GND رزبری‌پای (پین ۶)
- R را با مقاومت 470Ω به GPIO 24 رزبری‌پای (پین ۱۸)
- G را با مقاومت 470Ω به GPIO 25 رزبری‌پای (پین ۲۲)

- B را با مقاومت 470Ω به GPIO 8 رزبری‌پای (پین ۲۴)

وصل کردیم. در گام بعدی، دوباره برد خود را روشن و تغییرات زیر را در برنامه‌ی قبلی اعمال می‌کنیم. سپس کد جدید را در فایلی به نام RGB_LED.py ذخیره کنید:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com

import RPi.GPIO as GPIO
import time

red=18
green=22
blue=24

GPIO.setmode(GPIO.BOARD)
GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)

while True:
    try:
          # RED
          GPIO.output(red,1)
          GPIO.output(green,0)
          GPIO.output(blue,0)
          time.sleep(3)
          # GREEN
          GPIO.output(red,0)
          GPIO.output(green,1)
          GPIO.output(blue,0)
          time.sleep(3)
          # BLUE
          GPIO.output(red,0)
          GPIO.output(green,0)
          GPIO.output(blue,1)
          time.sleep(3)
          # YELLOW
          GPIO.output(red,1)
          GPIO.output(green,1)
          GPIO.output(blue,0)
```



```
            time.sleep(3)
            # MAGNETA
            GPIO.output(red,1)
            GPIO.output(green,0)
            GPIO.output(blue,1)
            time.sleep(3)
            # CYAN
            GPIO.output(red,0)
            GPIO.output(green,1)
            GPIO.output(blue,1)
            time.sleep(3)
            # WHITE
            GPIO.output(red,1)
            GPIO.output(green,1)
            GPIO.output(blue,1)
            time.sleep(3)

    except KeyboardInterrupt:
            print('Exit')
            break
GPIO.cleanup()
```

برای آشنایی بیشتر با دستورات استفاده شده در کد پایتون بالا می‌توانید به آموزش پایتون برای کار با رزبری‌پای مراجعه کنید.

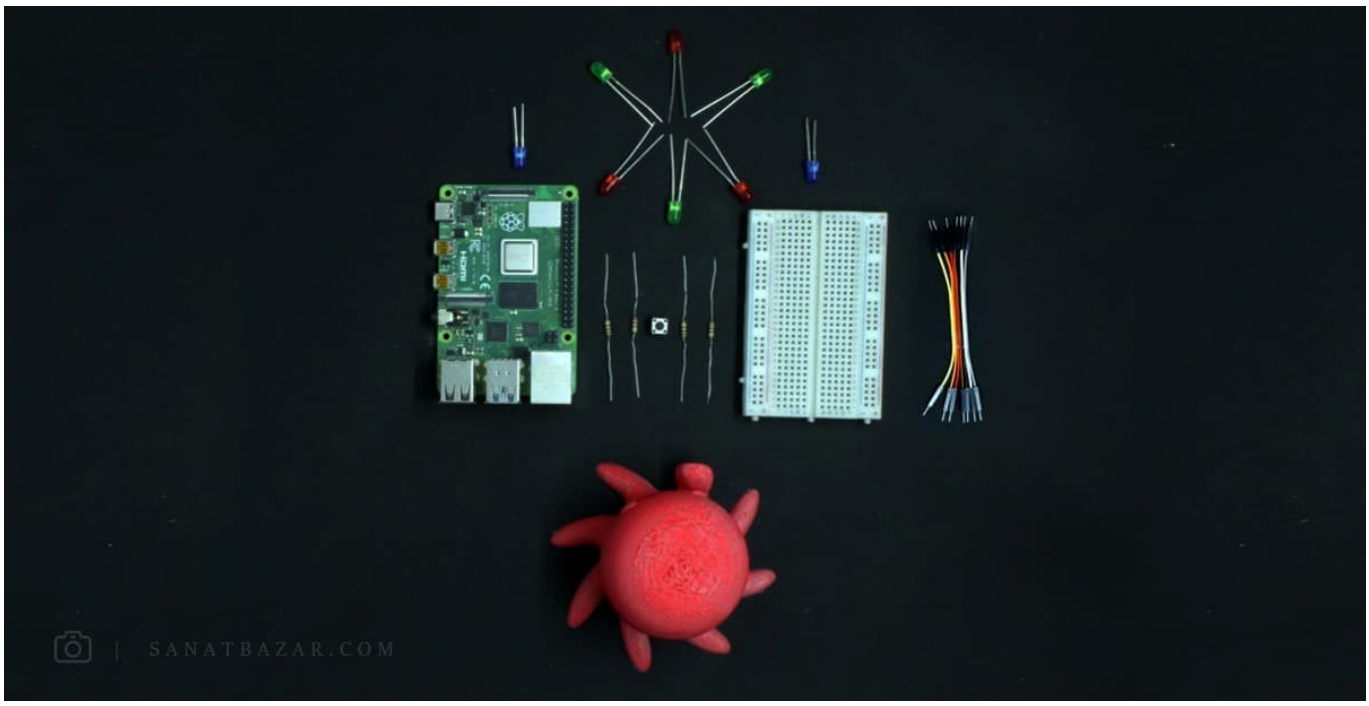
پس از ذخیره‌ی کد کافیست دستور زیر را اجرا کنید:

```
$ python RGB_LED.py
```

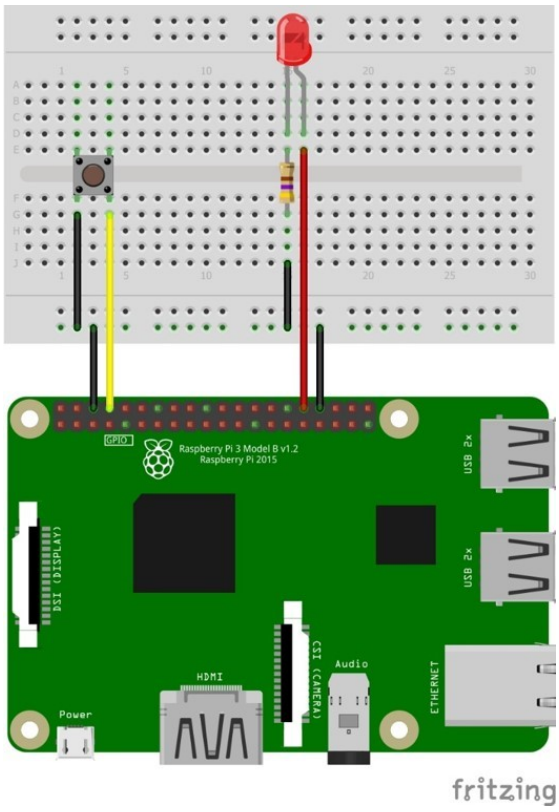
در اینجا برای آشنایی شما با انواع شیوه‌های کار با GPIO، به جای HIGH و LOW از 0 و 1 و به‌جای BCM از BOARD استفاده کردیم. در برنامه‌ی بالا، به منظور اجرای همیشگی دستورات، کد را در حلقه‌ی بی‌نهایت while True قرار دادیم و در انتها تعیین کردیم که در صورت وارد کردن وقفه‌ی کیبورد (CTRL+C) برنامه از حلقه خارج و پیکربندی پایه‌ها Reset شود.

برای RGB LED آند مشترک کافیست صفرها را به یک و یک‌ها را به صفر تغییر دهید.

چطور LED را با کلید روشن و خاموش کنم؟



ار آنجایی که کلیدها یکی از قطعات بسیار مهم در پروژه‌های الکترونیکی و IoT محسوب می‌شوند، یادگرفتن نحوه‌ی کار با آن‌ها و به‌طور کلی نحوه‌ی خواندن ورودی‌ها در رزبری‌پای می‌تواند بسیار مفید باشد.به همین دلیل می‌خواهیم این کار در قالب یک پروژه‌ی ساده و عملی باهم انجام دهیم. در این قسمت قصد داریم برنامه‌ای بنویسیم که مقدار یک کلید را به‌عنوان ورودی با رزبری‌پای خوانده و با توجه به آن، LED را روشن یا خاموش کنیم. در اینجا ما از کلیدهای فشاری استفاده می‌کنیم. خب مثل پروژه‌های قبلی رزبری‌پای را خاموش و مدار خود را به شکل زیر ببندید:



این بار مطابق تصویر بالا:

- پایه‌ی کاتد LED را با مقاومت 470Ω به پین GND رزبری‌پای (پین ۳۴)
- پایه‌ی آند LED را به GPIO 12 رزبری‌پای (پین ۳۲)
- یکی از پایه‌های کلید را به GND رزبری‌پای
- پایه‌ی دیگر کلید را به GPIO 14 رزبری‌پای (پین ۸)

 |

وصل کنید. پس از این کار فایل جدیدی به نام button.py ایجاد و کد زیر را در آن وارد کنید:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com

import RPi.GPIO as GPIO
import time

button=14
led=12

GPIO.setmode(GPIO.BCM)
GPIO.setup(led,GPIO.OUT)
GPIO.setup(button,GPIO.IN,pull_up_down=GPIO.PUD_UP)

flag=0

while True:
    try:
            state=GPIO.input(button)
            if state==0:
                time.sleep(0.5)
                if flag==1:
                    flag=0
                else:
                    flag=1
            if flag==1:
                GPIO.output(led,1)
            if flag==0:
                GPIO.output(led,0)
    except KeyboardInterrupt:
            print('Exit')
            break
GPIO.cleanup()
```

در نهایت برای اجرا دستور زیر را وارد می‌کنیم:

```
$ python button.py
```

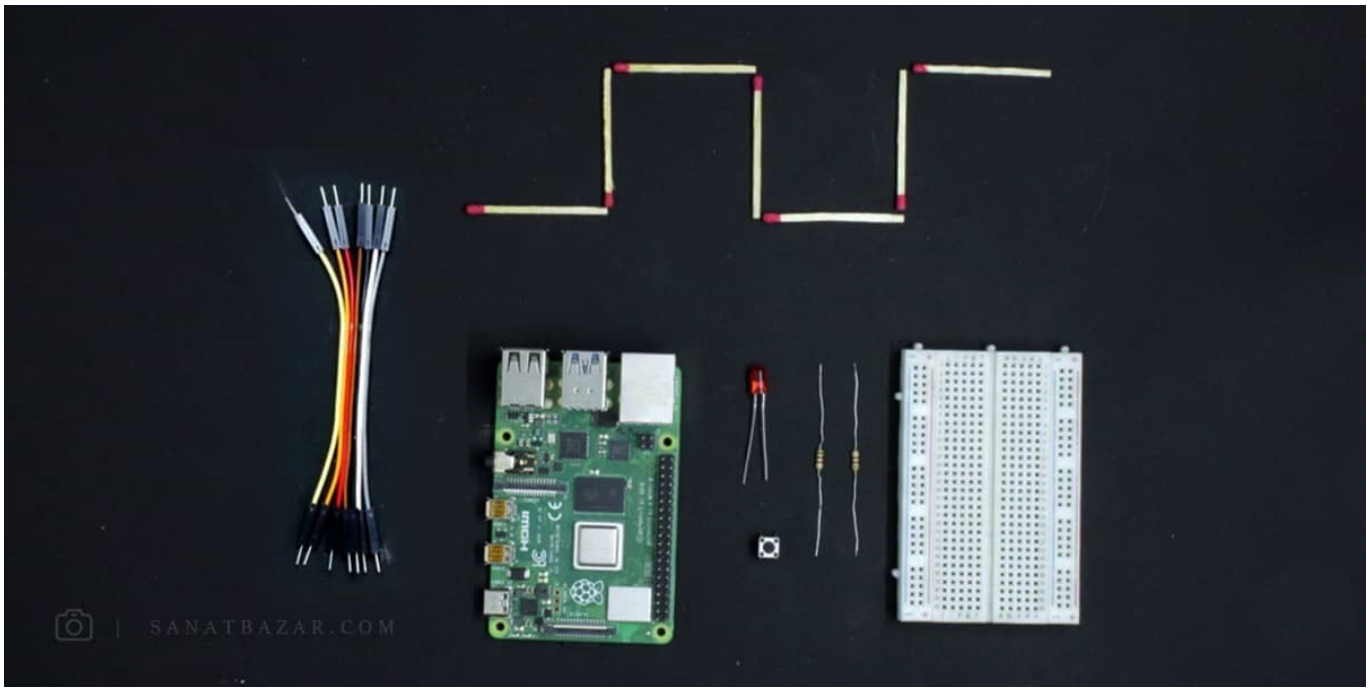
با توجه به این که فشرده شدن کلید، ورودی صفر را به رزبری‌پای ارسال می‌کند، پایه‌ی متصل به کلید را Pull up می‌کنیم، در اینجا با دستور GPIO.setup(button,GPIO.IN,pull_up_down=GPIO.PUD_UP) از پول‌آپ داخلی رزبری‌پای استفاده کردیم.

طبق این برنامه، LED بر اساس مقدار flag، روشن و خاموش می‌شود. با توجه به این که ما می‌خواهیم هر بار با فشار دادن کلید LED را خاموش یا روشن کنیم، پس هر بار با تشخیص فشرده شدن باید مقدار flag تغییر کند. نکته‌ی مهم درباره‌ی کلیدهای فشاری، bounce آن‌ها است. اما این Bounce چیه؟

زماین که دکمه‌ی کلید را فشار می‌دهید، در واقع دو سطح فلزی که هر کدام به یکی از پای‌های کلید وصل شده را به هم متصل می‌کنید. اما این اتفاق به همین سادگی و ناگهانی نمی‌افتد، بلکه این دو سطح فلزی در کسری از ثانیه چندین بار به هم برخورد می‌کنند. در مدارهای الکترونیکی ایم اتفاق ناخواسته باعث ایجاد خطا می‌شود. به طوری که کنترل کننده و حسگر شما (در اینجا رزبری‌پای) به اشتباه فکر می‌کند چندین بار کلید را فشار داده‌اید. برای حل این مشکل بعد از تشخیص اول، تاخیر در نظر گرفته می‌شود. یعنی هنگامی که اولین برخورد تشخیص داده شد، تا زمان کوتاهی سایر برخوردها را در نظر نگیر. ما هم در اینجا برای حل این مشکل از تاخیر ۵۰۰ ثانیه‌ای استفاده کردیم.

آموزش پیشرفته: وقفه‌ها و تشخیص لبه‌های پالس (خیلی کاربردی !!)

رزبری پای و کلید و پالس



حتماً با شکل موج پالس و کاربردهای آشنا هستید. از این کل موج مهم در پروژه‌های حرفه‌ای و دانشگاهی مثل پردازش سیگنال یا پروژه‌های عملی مثل راه‌اندازی موتورها و اندازه‌گیری سرعت و موقعیت با شفت انکودرها بسیار استفاده می‌شود. از این رو کتابخانه‌ی RPi.GPIO دستور آماده و خیلی ساده‌ای را برای کار با این شکل موج به کاربران ارائه می‌کند که در ادامه قصد داریم آن را در قالب یک مثال ساده‌ی آموزشی به کار بگیریم. این دستور به‌صورت وقفه عمل می‌کند. یعنی فارغ از این که کجای برنامه‌ی شما نوشته شود، در صورت وقوع پالس، اجرا خواهد شد. در ادامه با شکل‌های مختلف این وقفه آشنا می‌شویم:

```
add_event_detect(نام پایه , نوع لبه , نام تابع)
```

با استفاده از دستور فوق می‌توانید لبه‌ی پالس ورودی به پایه‌ی مورد نظر را تشخیص داده و در صورت وقوع، تابع خاصی را جرا کنید. این لبه می‌تواند پایین رونده، بالا رونده یا هر دو باشد. به مثال‌های زیر توجه کنید:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com
```

 |

```
import RPi.GPIO as GPIO
pulse=2
GPIO.setmode(GPIO.BCM)
GPIO.setup(pulse,GPIO.IN)

def my_fun (pulse):
       print('Pulse detected!')

GPIO.add_event_detect(pulse,GPIO.RISING,callback=my_fun)

while True:
       print('No pulse!')
```

در مثال بالا پین شماره‌ی 3 رزبری‌پای (GPIO 2) را با نام pulse به عنوان ورودی تعریف کردیم. بنابراین منبع پالس را باید به این پورت متصل کنید. سپس در دستور GPIO.add_event_deect()، تشخیص لبه‌ی بالا رونده تعیین و در صورت وقوع تابع my_fun() اجرا خواهد شد. همانطر که گفتیم، این دستور به‌صورت وقفه اجرا می‌شود. پس در حالتی که لبه‌ی بالا رونده‌ی پالس تشخیص داده نشود، مقدار No pulse! و در صورت وقوع لبه، Pulse detected! چاپ می‌شود.

می‌توانید به جای GPIO.RISING از GPIO.FALLING برای لبه‌ی پایین رونده و GPIO.BOTH برای تشخیص هر دو لبه استفاده کنید.

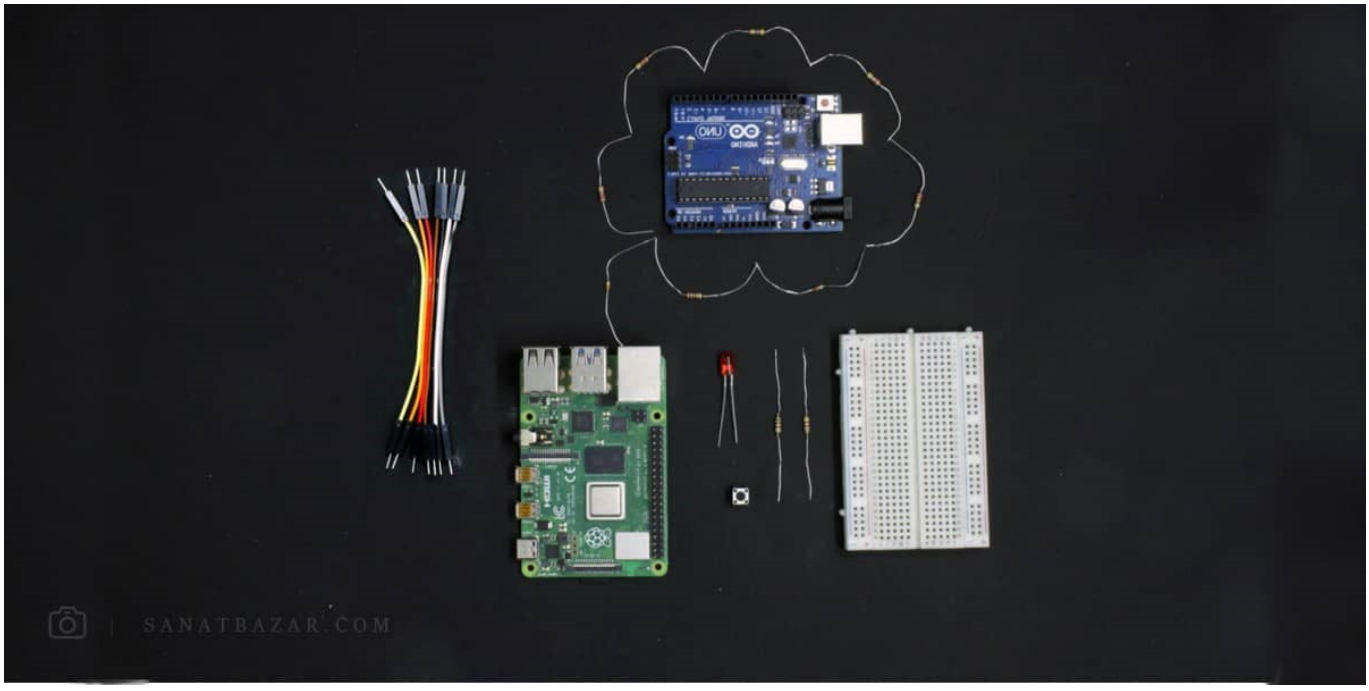
کتابخانه‌ی RPi.GPIO در دپاین دستور هم تاخیر لازم برای مقابله با Bounce را در نظر گرفته است. برای این کار کافیست مقدار bouncetime را در این دستور بر حسب میلی ثانیه تعریف کنید. خب حالا که با این دستور به اندازه‌ی کافی آشنا شدید، می‌خواهیم همان پروژه‌ی قبلی (روشن و خاموش کردن LED با کلید) را این بار با وقفه‌ها اجرا کنیم. بنابراین کد زیر را در فایل جدیدی به نام button_interrup.py ذخیره و اجرا می‌کنیم:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com

import RPi.GPIO as GPIO
import time

button=14
led=12

GPIO.setmode(GPIO.BCM)
GPIO.setup(led,GPIO.OUT)
GPIO.setup(button,GPIO.IN,pull_up_down=GPIO.PUD_UP)
flag=0

def change(button):
    global flag
    print('Buttton Pushed!')
    if flag==0:
```

```
            flag=1
        else:
            flag=0
    GPIO.add_event_detect(button,GPIO.BOTH,callback=change,bouncetime=200)


    while True:
        try:
            if flag==0:
                GPIO.output(led,0)
            if flag==1:
                GPIO.output(led,1)
        except KeyboardInterrupt:
            print('Exit')
            break
    GPIO.cleanup()
```

تفاوت این برنامه با برنامه‌ی قبلی در این است که در اینجا با علاوه‌بر این که هر بار با زدن کلید، LED خاموش و روشن می‌شود، با نگه داشتن آن نیز روشن و با رها کردن، خاموش می‌شود. این عمل به خاطر استفاده از GPIO.BOTH در وقفه است. چرا که با فشار دادن کلید؛ لبه بالا رونده تشخیص و LED روشن می‌شود و با رها کردن آن نیز لبه پایین رونده تشخیص و LED خاموش می‌شود. همچنین برای مقابله با Bounce هم از bouncetime با مقدار 200ms استفاده کرده‌ایم.

WiringPi برای کسانی که قبلا با آردوینو کار کرده اند!



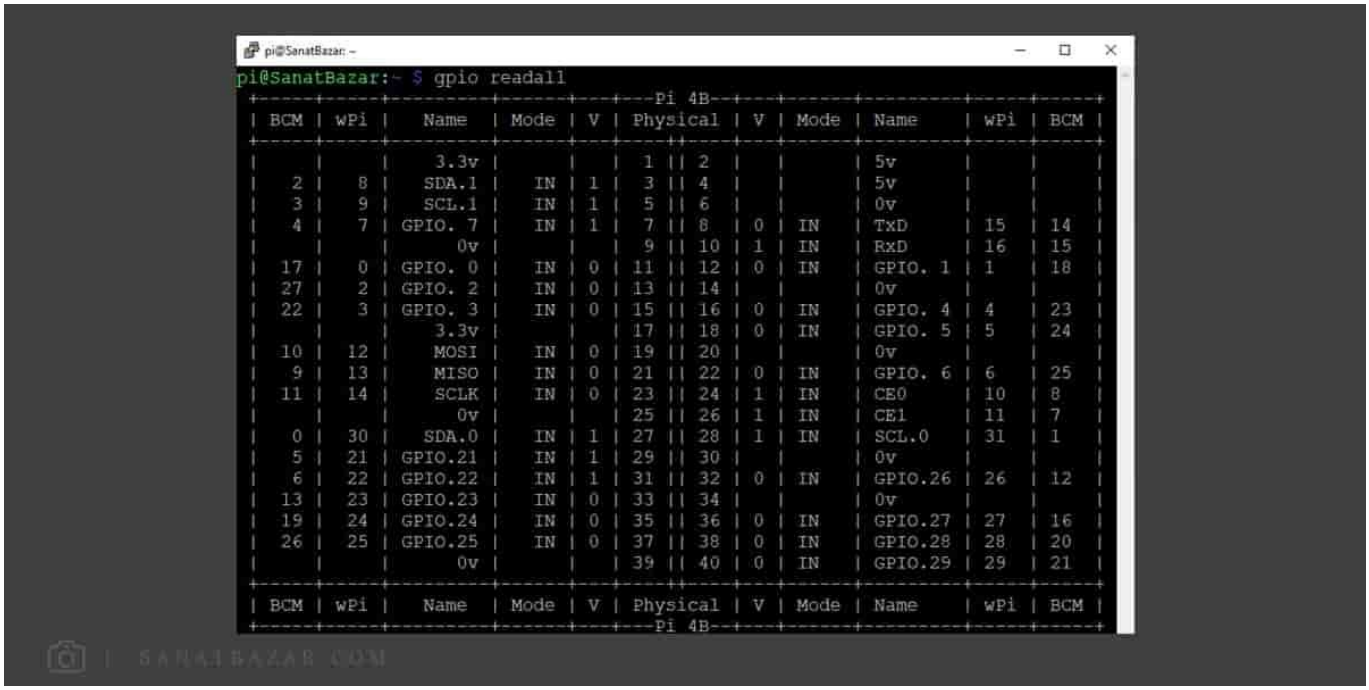
WiringPi کتابخانه‌ای برای دسترسی به GPIOهای برد، بر مبنای زبان‌های C و C++ است که به افراد آشنا به محیط کدنویسی آردوینو پیشنهاد می‌شود. چرا که تمام دستورات Arduino IDE در این کتابخانه برای رزبری‌پای قابل استفاده است. این کتابخانه به‌صورت رسمی برای رزبری‌پای با سیستم‌عال رزبین و به‌صورت غیر رسمی برای سایر پلتفرم‌ها منتشر شده است. WringPi به‌صورت پیش‌فرض روی نسخه‌های جدید رزبین نصب شده اما در صورت نیاز می‌توانید این کتابخانه را با دستور زیر نصب کنید:

```
$ sudo apt-get install wiringpi
```

قبل از انجام پروژه، با استفاده از دستور زیر می‌توانید با پین‌ها و GPIOها برد خود بیشتر آشنا شوید. در حالتی که به اینترنت دسترسی ندارید، این دستور خیلی می‌تواند به شما برای پیدا کردن اطلاعات هر پین کمک کند:

```
$ gpio readall
```

در صورت نصب و آپدیت بودن پکیج wiringpi، باید صفحه‌ی زیر را مشاهده کنید:

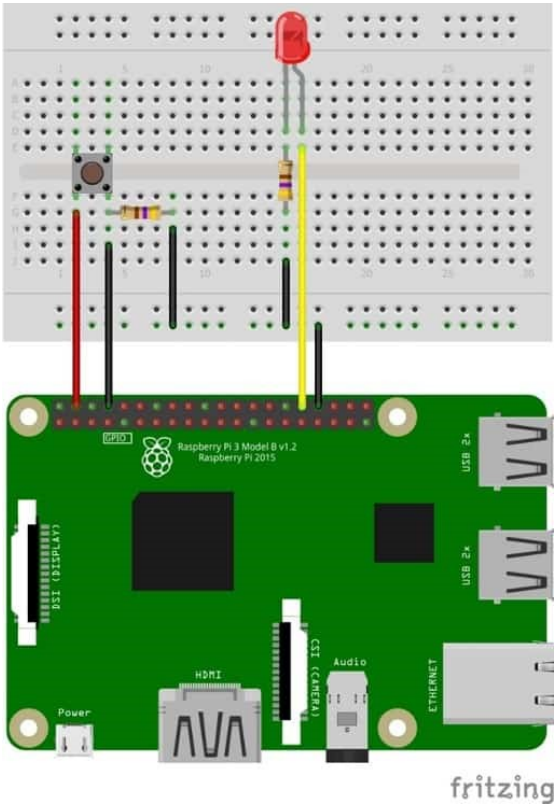


این صفحه تمام اطلاعات لازم در مورد شماره‌گذاری و قابلیت‌های هر پین را در اختیار شما قرار می‌دهد. همانطور که مشاهده می‌کنید، ما این دستور را روی رزبری‌پای ۴ اجرا کردیم ولی شما می‌توانید آن را روی سایر نسخه‌ها نیز اجرا کنید. ستون‌ها این جدول در ادامه توضیح داده شده است:

- BCM: شماره‌ی پایه‌ها در استاندارد BCM
- Wpi: شماره‌ی پایه‌ها برای کدنویسی با استفاده از کتابخانه‌ی WringPi
- NAME: عملکرد هر پایه. به‌عنوان مثال SPI، I2C و ...
- MODE: ورودی یا خروجی بودن هر پین را نشان می‌دهد.
- V: مقدار هر پایه (High یا Low) را نشان می‌دهد.
- Physical: شماره‌ی پایه‌ها در استاندارد BOARD

خب بعد از آشنایی با این دستور کاربردی می‌خواهم نحوه‌ی کار با WiringPi را با یک مثال ساده به شما آموزش دهم: دوباره روشن و خاموش کردن LED با کلید!

برای این کار اول مدار زیر را ببندید:



طبق معمول دوباره با استفاده از nano یک فایل متنی با نام دلخواه اما این بار با پسوند cpp ایجاد کنید. من نام این فایل را LED_button.cpp انتخاب کردم. سپس کد زیر را در آن وارد کنید:

```
# SanatBazar
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatbazar.com

#include <iostream>
#include <wiringPi.h>

#define PIN_LED 12
#define PIN_BUTTON 14
{
      wiringPiSetupGpio();
      pinMode(PIN_LED, OUTPUT);
      pinMode(PIN_BUTTON, INPUT);

      while (1)
{
            if (digitalRead(PIN_BUTTON) == HIGH)
              {
                  digitalWrite(PIN_LED, HIGH);
            else
```

```
            {
                  digitalWrite(PIN_LED, LOW);
            }
      delay(200)
}
```

همانطور که می‌بینید، کدها دقیقاً همان کدهای آردوینو است. در ابتدا کابخانه‌های مورد نیاز را معرفی و سپس با دستور define، پین‌های ورودی خروجی را نام‌گذاری می‌کنیم. در بخش بعدی از دستور WiringPiSetupGpio() برای تعریف استاندارد BCM استفاده کرده‌ایم. پس شماره پین‌ها در این برنامه BCM است. برای شماره گذاری BOARD از دستور WiringPiSetupSys() و برای استاندارد wpi) WringPi) از دستور WiringPiSetup() می‌توانید استفاده کنید. سپس در ادامه پین‌های مورد نظر را ورودی و خروجی تعریف و دستورات لازم برای روشن و خاموش کردن LED را می‌نویسیم. پس از پایان کد نویسی فایل را ذخیره کرده و با دستور زیر Compile کنید:

```
$ g++ -o LED_button LED_button.cpp –lwiringPi
```

حالا برای اجرا کافیست دستور زیر را اجرا کنید:

```
./ LED_button
```

در این برنامه با نگه داشتن کلید، LED روشن و با رها کردن آن، LED خاموش می‌شود.

نتیجه‌گیری

در این بخش آموزشی، هرآنچه که رای کار با GPIOهای رزبری‌پای نیاز داشتید، از مبتدی تا پیشرفته آموزش داده شد. همچنین با یاد گرفتن نحوه‌ی کار با کتابخانه‌ی WiringPi از این پس می‌توانید علاوه‌بر زبان پایتون، برنامه‌های خود را به زبان C++ و در قالب ک آردوینو به رزبری‌پای اعمال کنید. بنابراین از حالا به بعد آماده‌اید تا هر پروژه‌ای را متناسب با توانایی برنامه‌نویسی خود با رزبری‌پای انجام دهید. این بخش تنها جهت آشنایی با پایه‌های ورودی خروجی رزبری‌پای و نوه‌ی دستور دهی به آن‌ها بود. بنابراین سعی شد تا با مثال‌های ساده و روان، این مبحث به زبان ساده بیان شود. اما از این پس قصد داریم پروژه‌های حرفه‌ای و کاربردی‌تری به کمک رزبری‌پای به‌صورت DIY و گاهی در قالب IoT با هم انجام دهیم. در قسمت بعدی خواهیم دید که چگونه اطلاعات سنسورهای دما و رطوبت را توسط رزبری‌پای بخوانیم و از طریق اینترنت و ماژول esp8266 به آردوینو ارسال کنیم (و برعکس. یعنی با آردوینو بخوتنیم و به رزبری‌پای ارسال کنیم!) از این پروژه می‌توانید برای ساخت یک گلخانه‌ی هوشمند استفاده کنید! پس با من همراه باشید تا قدم به قدم نحوه‌ی انجام این پروژه‌ی جذاب را پیش ببریم.

نظرات شما باعث بهبود محتوای آموزشی ما می‌شود. اگر این آموزش را دوست داشتید، همین‌طور اگر سوالی در مورد آن دارید، از شنیدن نظراتتان خوشحال خواهیم شد.