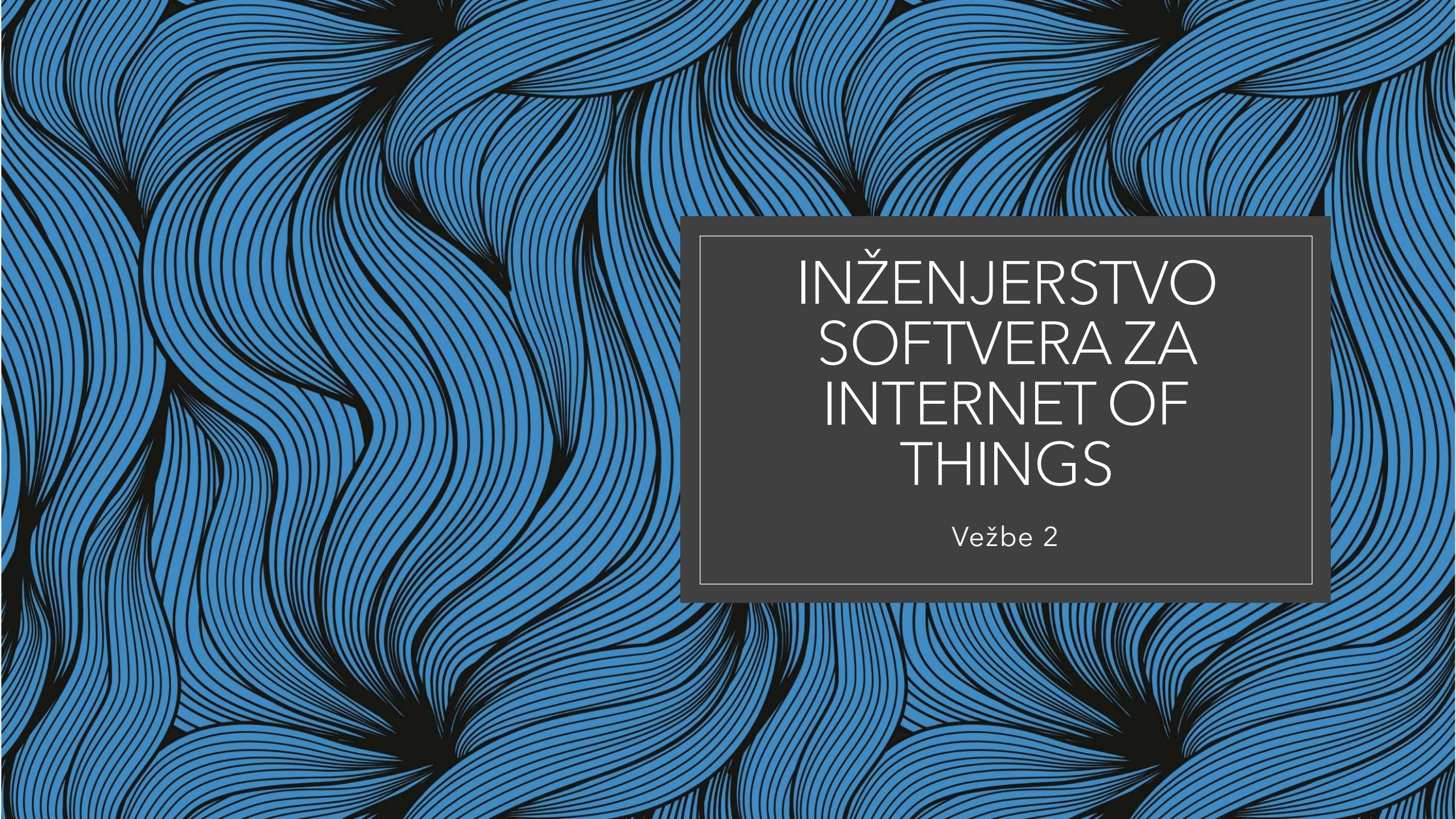
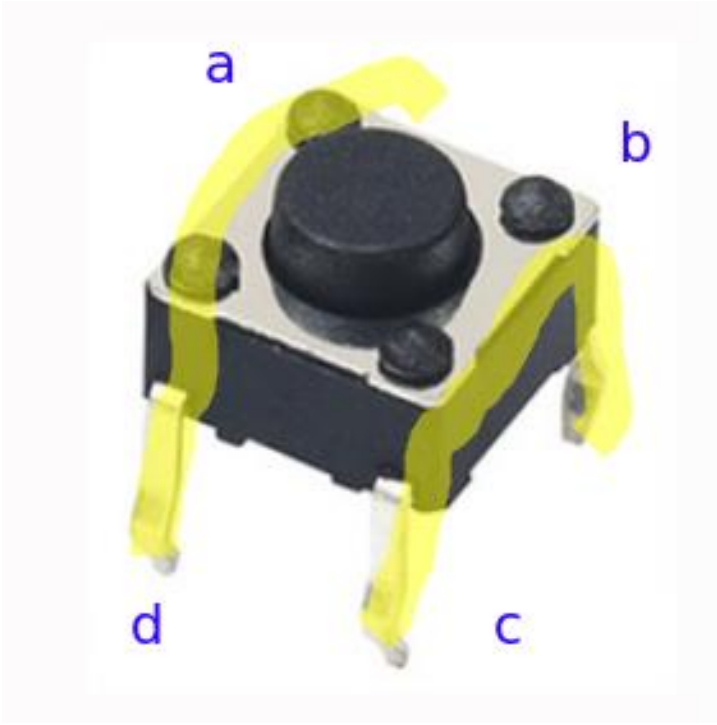


INŽENJERSTVO SOFTVERA ZA INTERNET OF THINGS

Vežbe 2



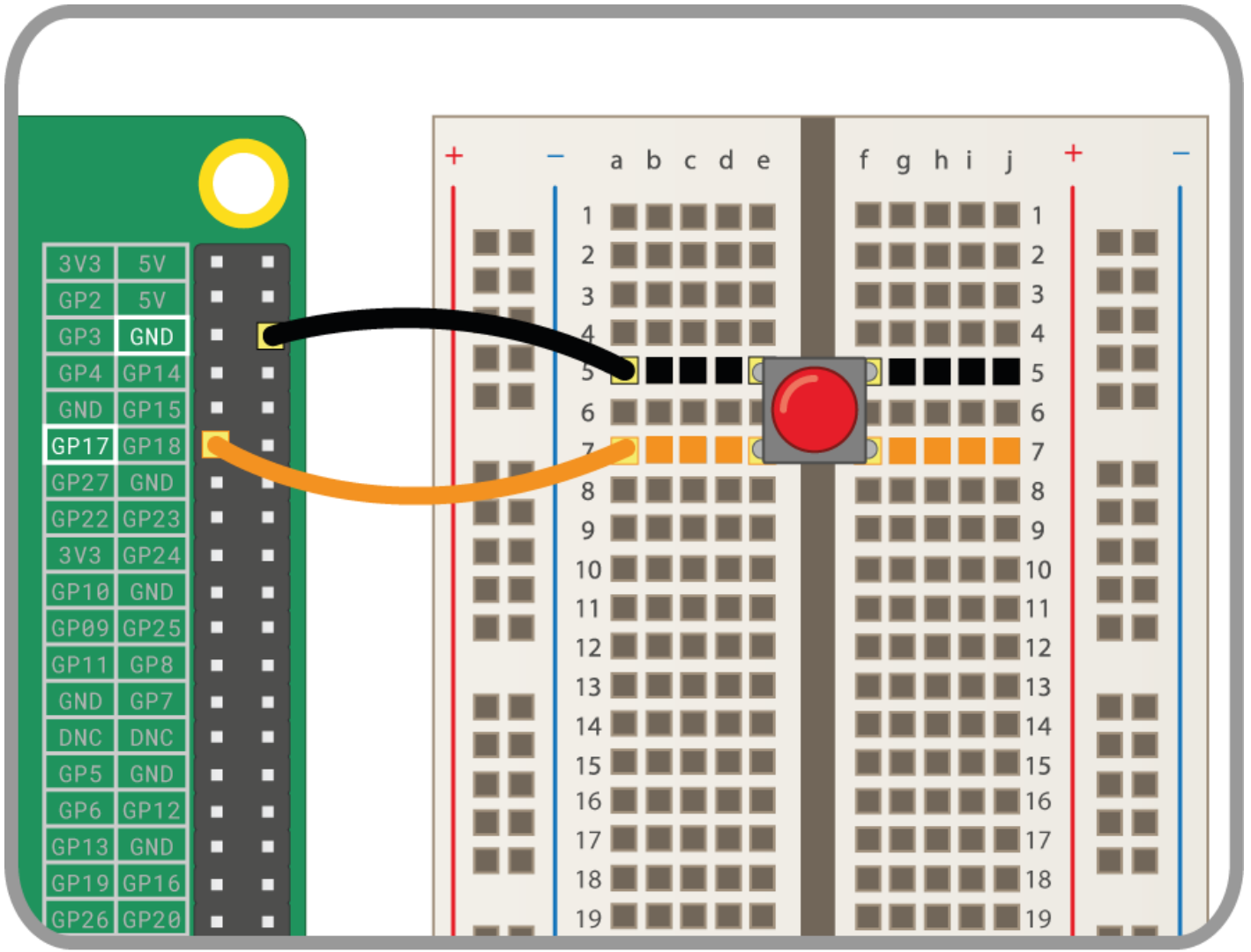
BUTTON (DUGMENCE)

PULL-UP VS PULL-DOWN OTPORNICI

Pull-Up otpornici	Pull-Down otpornici
Povežite između I/O pina i + napona, s otvorenim prekidačem povezanim između I/O pina i uzemljenja.	Povežite između I/O pina i uzemljenja, s otvorenim prekidačem povezanim između I/O pina i + napona.
Održava ulaz "Visokim"	Održava ulaz "Niskim"
Češće korišćeni	Ređe korišćeni

BUTTON (DUGMENCE)

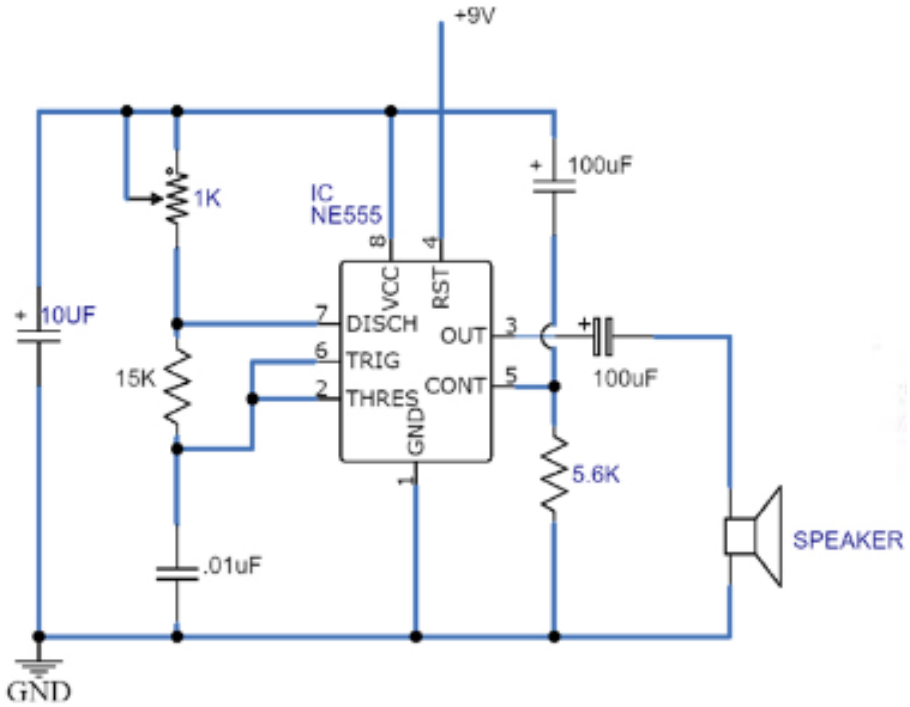
- Pull-Up wiring



Zadatak1 - code

```
import RPi.GPIO as GPIO


def button_pressed(event):
    print("BUTTON PRESS DETECTED")


PORT_BUTTON = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(PORT_BUTTON, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.add_event_detect(PORT_BUTTON, GPIO.RISING, callback =
button_pressed, bouncetime = 100)


input("Press any key to exit...")
```


Negative
(-)
Possitive
(+)

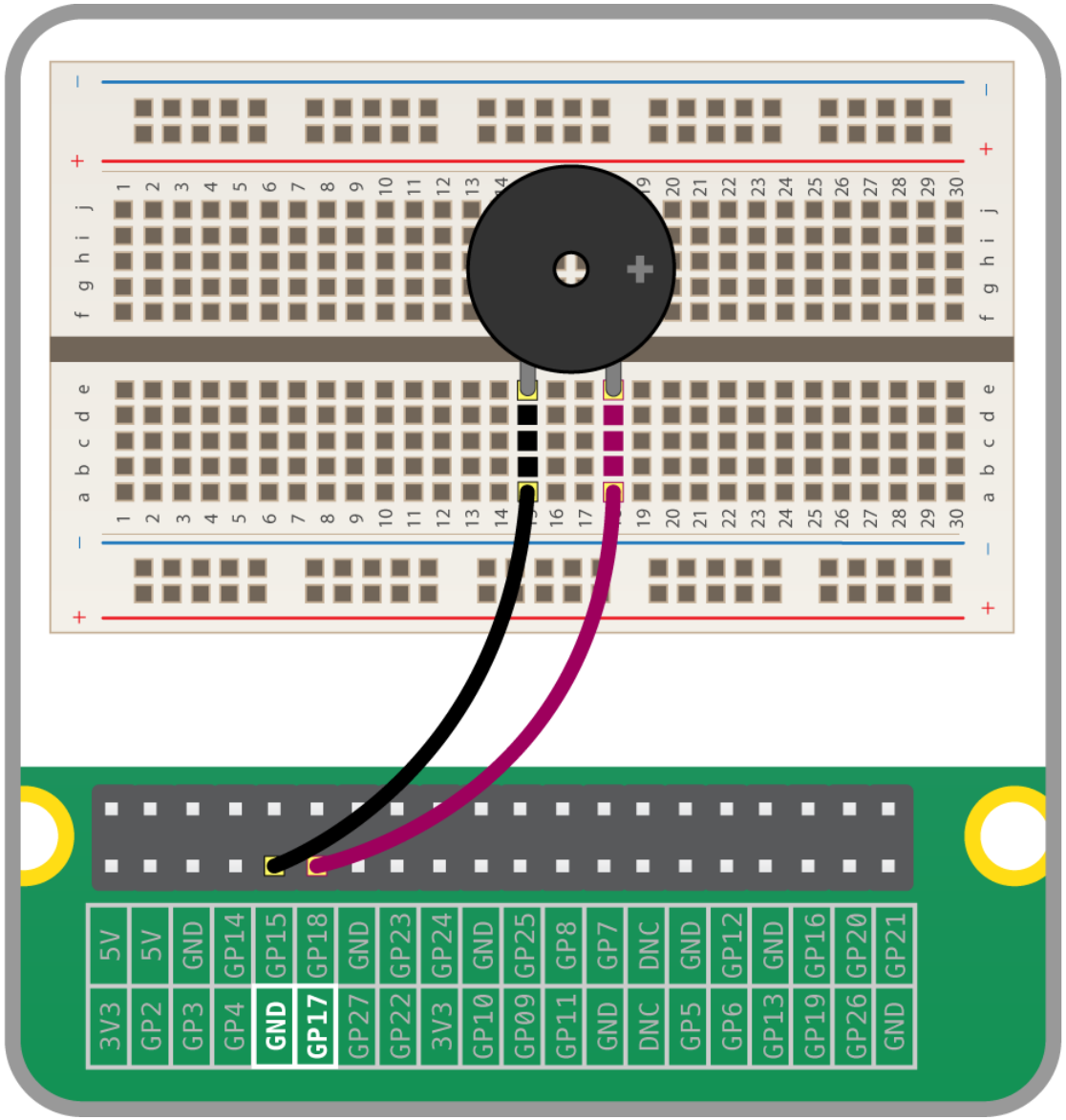

+9V
1K
IC
NE555
+ 100uF
+ 10UF
15K
VCC
RST
DISCH
OUT
TRIG
CONT
THRES
GND
100uF
5.6K
SPEAKER
.01uF
GND

BUZZER

Aktivni i pasivni Buzzer

- Aktivni buzzer ima ugrađen oscilator, tako da je dovoljno da mu dovedemo jednostmernu struju kako bi ispuštao zvuk.
- Uvek ispušta zvuk iste frekvencije

- Pasivni buzzer nema ugrađen oscilator, nego je građen nalik na zvučnik. Oscilovanjem signala upravljamo frenkvencijom zvuka koji ispušta.

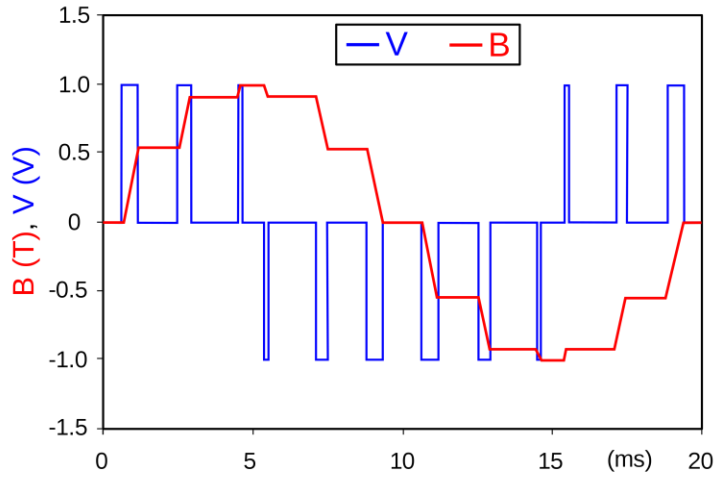


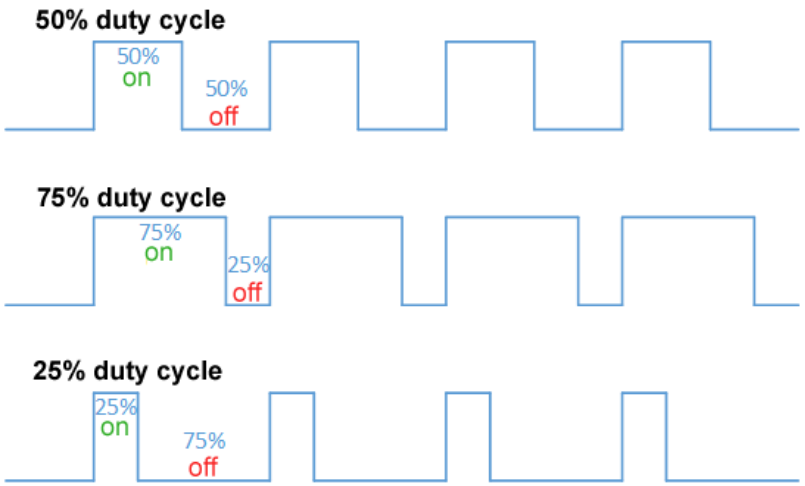
ZADATAK2 - WIRING

Zadatak2 – active buzzer code

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
buzzer_pin = 17
GPIO.setup(buzzer_pin, GPIO.OUT)

def buzz(pitch, duration):
    period = 1.0 / pitch
    delay = period / 2
    cycles = int(duration * pitch)
    for i in range(cycles):
        GPIO.output(buzzer_pin, True)
        time.sleep(delay)
        GPIO.output(buzzer_pin, False)
        time.sleep(delay)

try:
    while True:
        pitch = 440
        duration = 0.1
        buzz(pitch, duration)
        time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

- Možemo izmeniti frekvenciju percipiranog zvuka paljenjem/gašenjem buzzer-a odgovarajućom frekvenciju
- Rezultuje niskim kvalitetom zvuka





Impulsno širinska modulacija - *Pulse Width Modulation (PWM)*

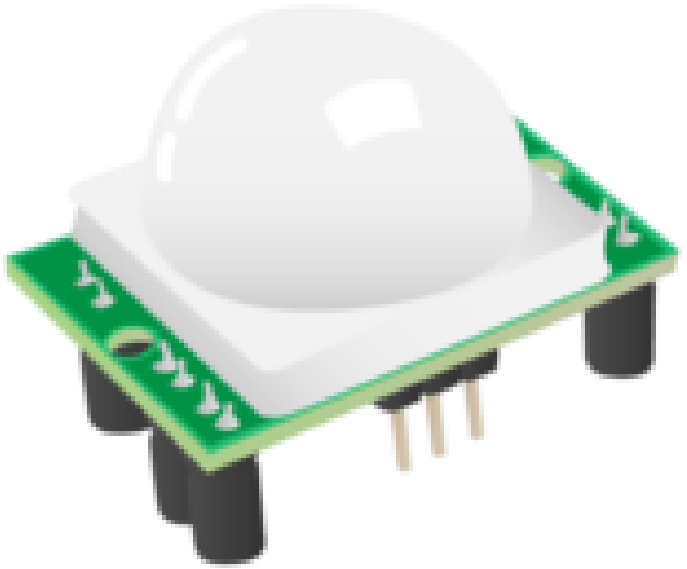
Služi da od digitalnog signala napravimo analogni, tako što se visokom frekvencijom na izlazu šalju 1 ili 0

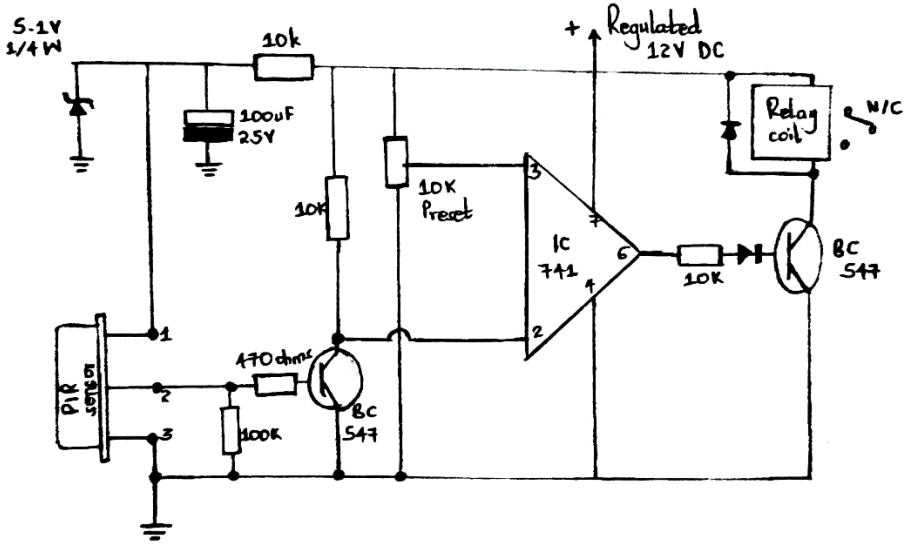
Frekvencija nam određuje koliko često menjamo signal

Radni ciklus *(Duty Cycle)* nam određuje odnos između trajanja 1 i 0

Passive buzzer code

```python
import RPi.GPIO as GPIO
import time


GPIO.setmode(GPIO.BCM)
BUZZER_PIN = 16

GPIO.setup(16, GPIO.OUT)
Buzz = GPIO.PWM(BUZZER_PIN, 440) # inital frequency

def buzz(pitch, duration):
    Buzz.ChangeFrequency(pitch) # frequency
    Buzz.start(50) # duty cycle
    time.sleep(duration)
    Buzz.stop()
```



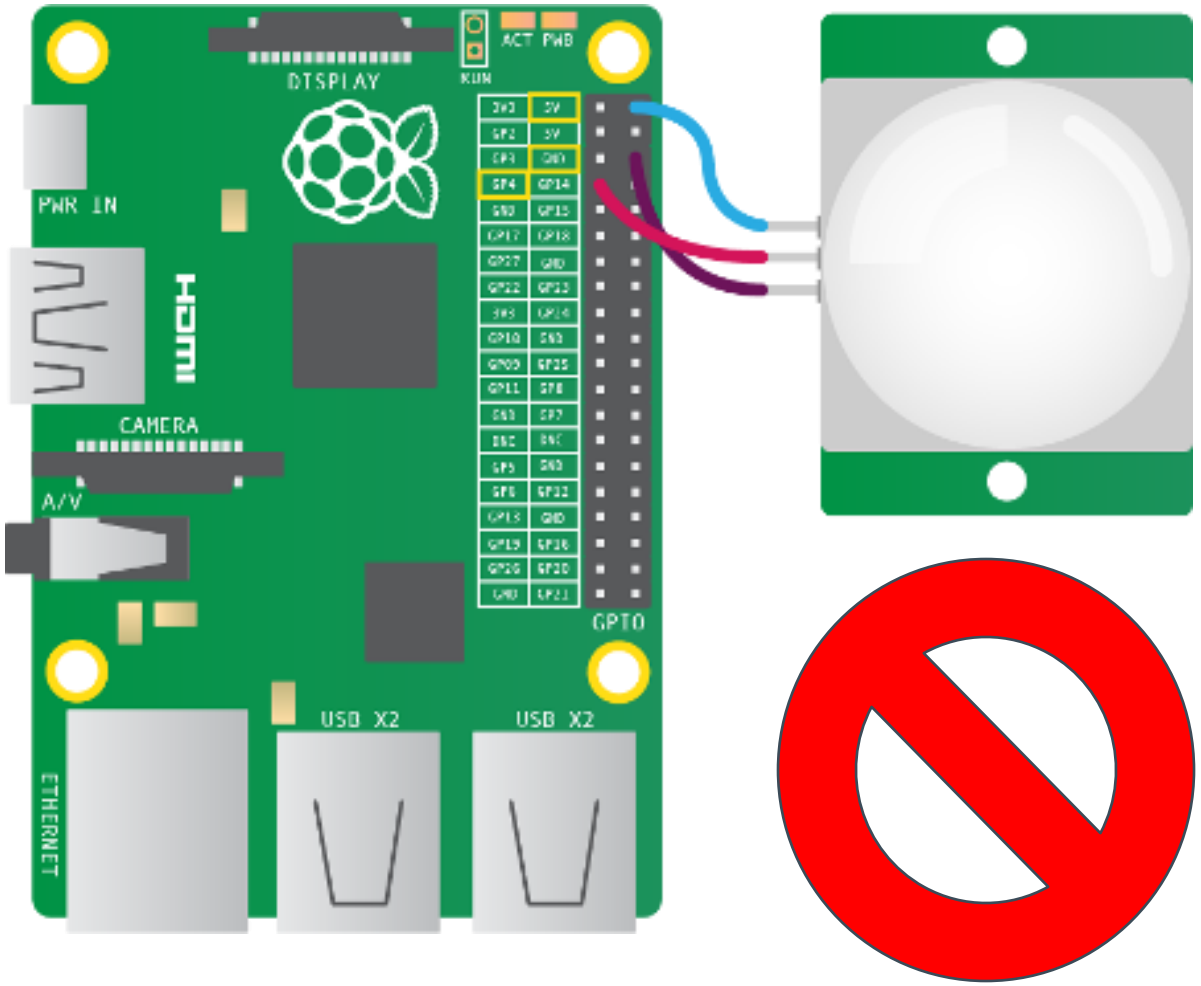


PIR (PASSIVE INFRARED) SENSOR

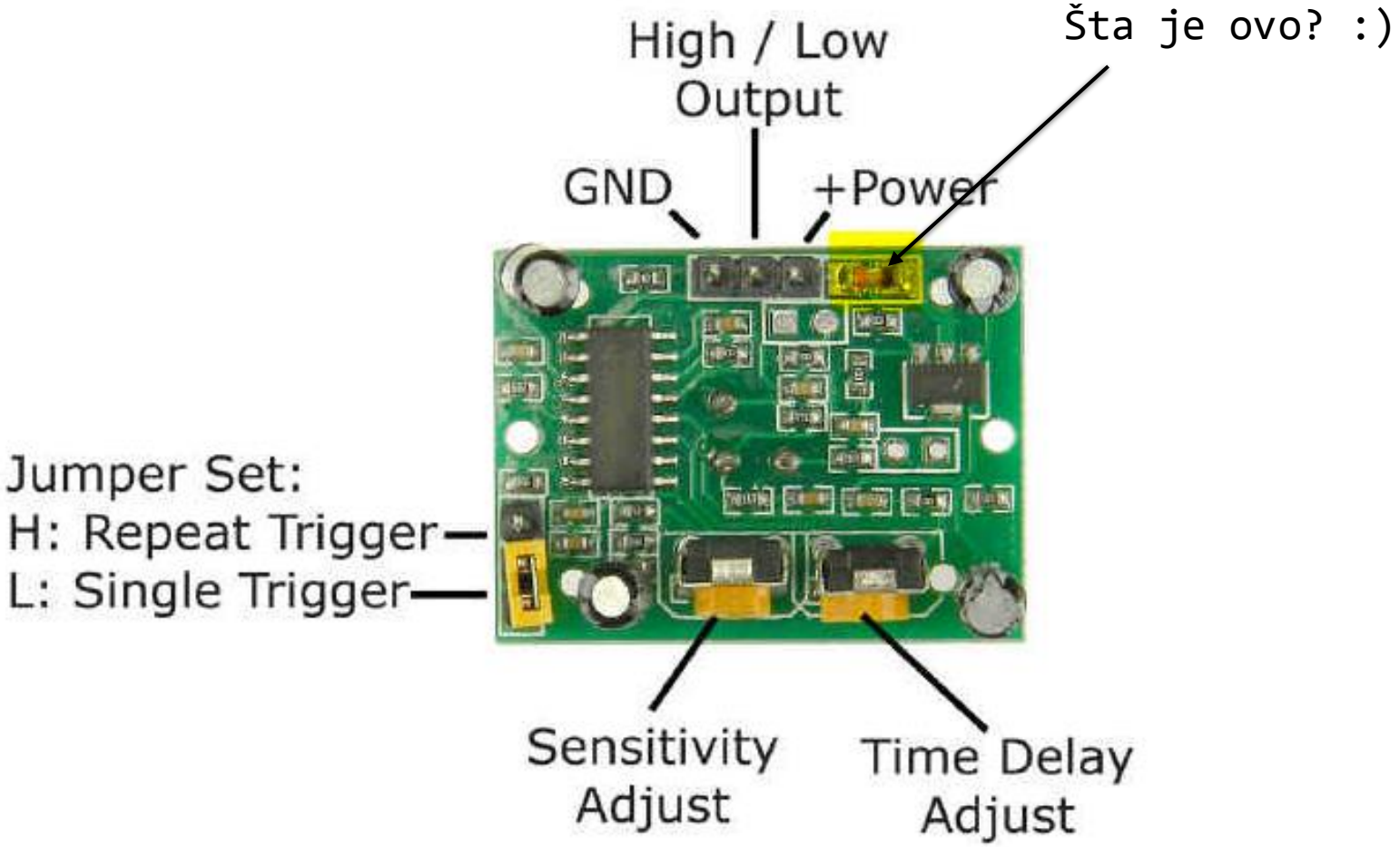
PIR Motion sensor

- U sebi sadrži dva IR senzora, koji registruju IR toplotno zračenje (npr. ljudskog tela)
- Pokret se detektuje kada se pojavi razlika između detektovanih vrednosti sa dva senzora
- U teoriji, neće detektovati kretanje objekata sobne temperature

- Povezuje se na napon od 5V i uzemljenje
- Izlaz je digitalni signal



ZADATAK3 – WIRING (ČESTA GREŠKA KOD NAŠEG MODELA)



ZADATAK3 – WIRING (ISPRAVAN ZA NAŠ MODEL)

Zadatak3 - code

```
import RPi.GPIO as GPIO

PIR_PIN = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_PIN, GPIO.IN)

def motion_detected(channel):
    print("You moved")

def no_motion(channel):
    print("You stopped moving")

GPIO.add_event_detect(PIR_PIN, GPIO.RISING, callback=motion_detected)
GPIO.add_event_detect(PIR_PIN, GPIO.FALLING, callback=no_motion)

input("Press any key to exit...")
```

Bonus zadatak

- Povezati sve 3 komponente da rade zajedno
- Kada se dugme pritisne buzzer treba da se oglasi
- Kada PIR senzor detektuje pokret buzzer takođe treba da se oglasi

- Bonus bonusa :) : napraviti da se pritiskom na dugme buzzer ne aktivira programski već da se direktno u mreži do istog dovede napajanje od 3.3V