

---

---

# 18. Роботика

— 21 декември 2023 —

---

---

# Въпрос

Каква е разликата между конкурентност и паралелизъм?

**Конкурентност:** Когато две изчисления нямат ясно дефинирана последователност на изпълнение.

**Паралелизъм:** Две изчисления, които реално се изпълняват едновременно.

# Въпрос+

Каква е разликата между `threading.Semaphore` и `threading.Lock`?

`Lock` позволява само една нишка, а `Semaphore` позволява няколко.

# Въпрос++

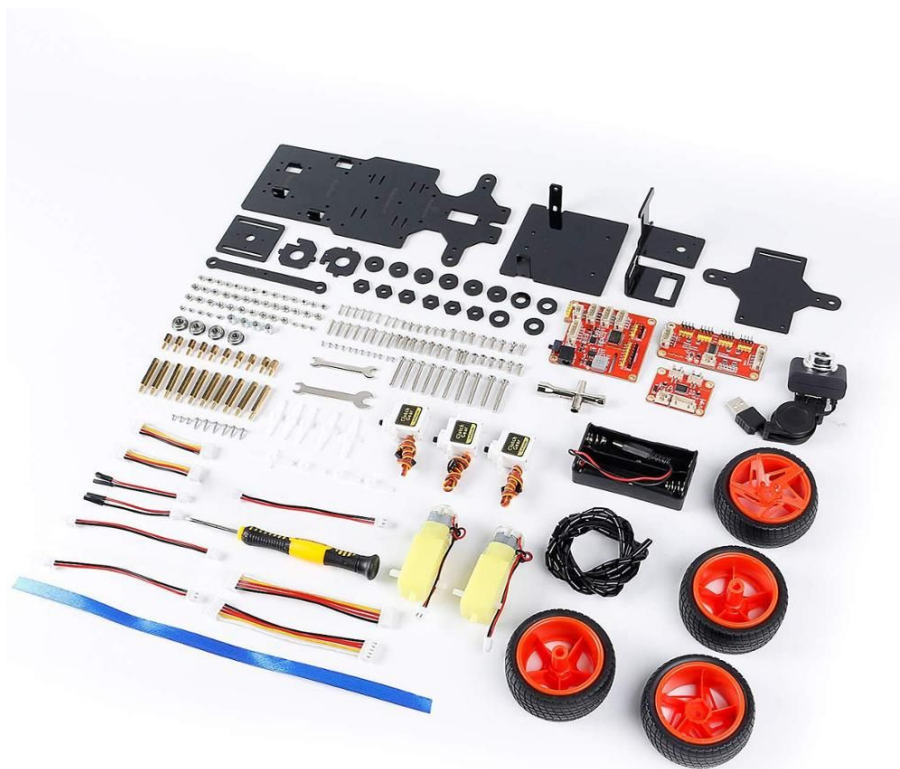
За какво се използва `os.fork`?

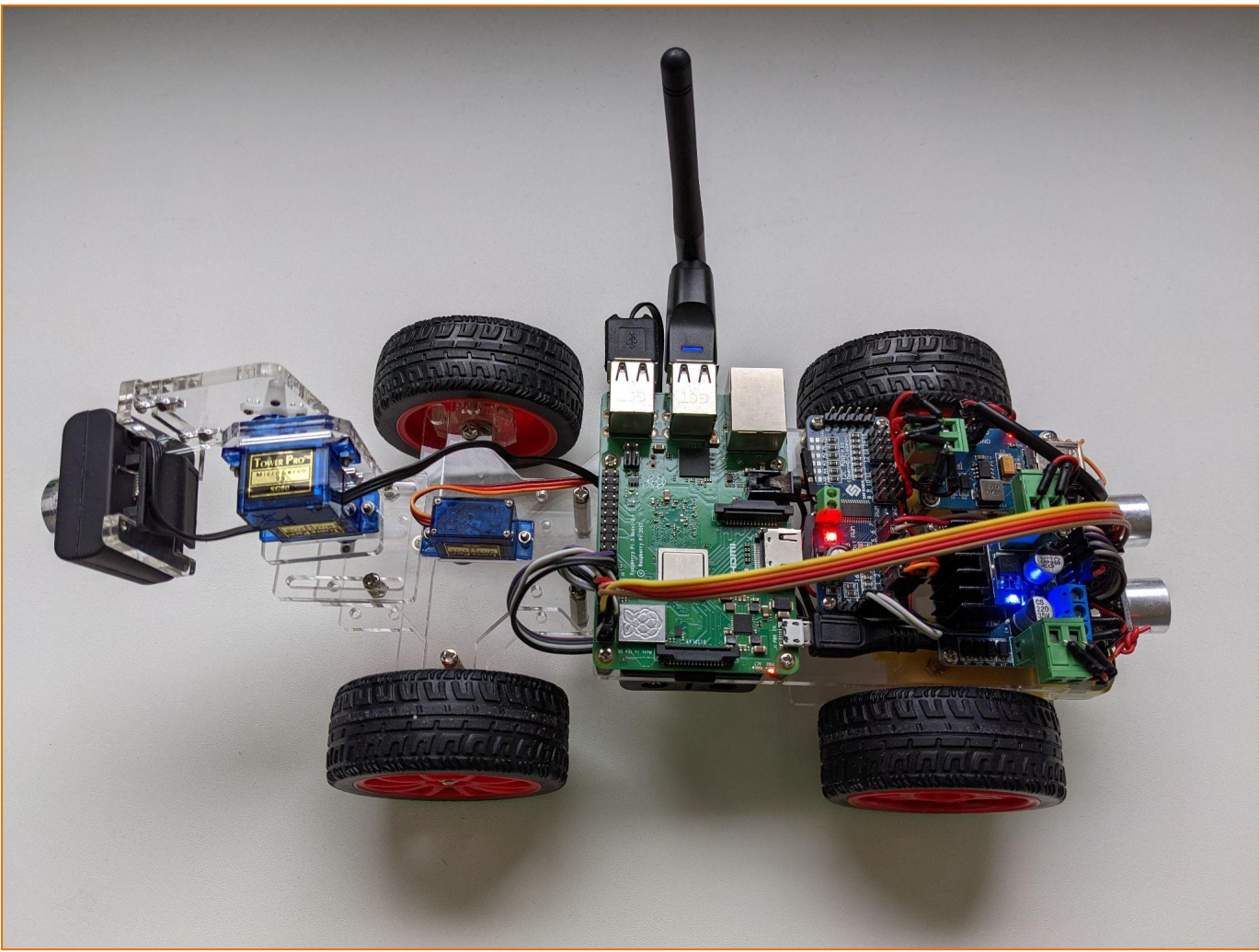
За разделяне на програмата в два отделни процеса.

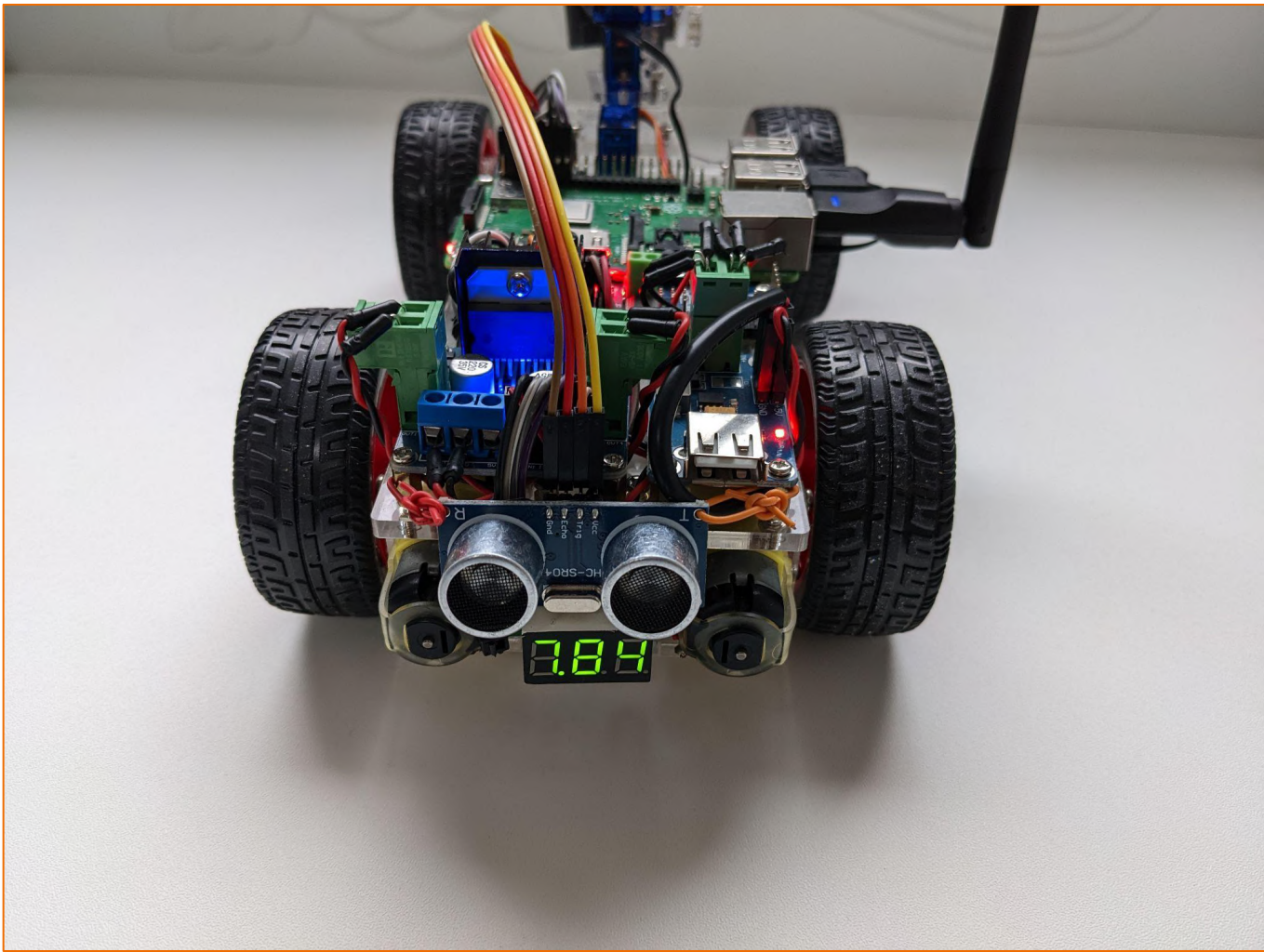
# Какво ще правим днес?



# Комплектът преди да запретнете ръкави

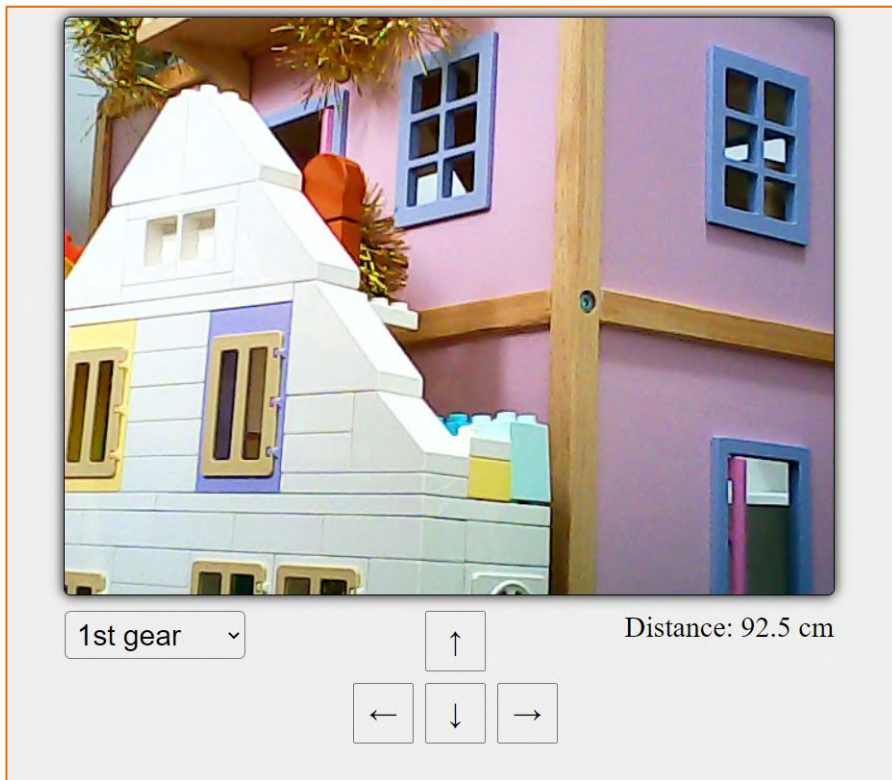








# Интерфейсът изглежда така

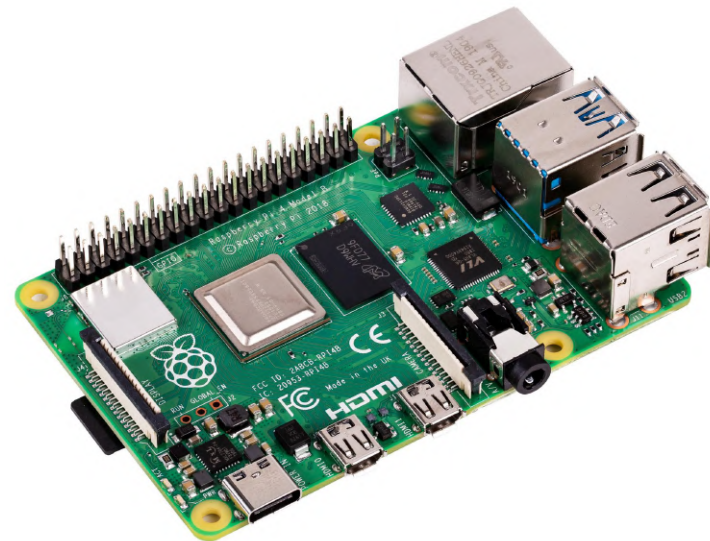


# Да разгледаме компонентите

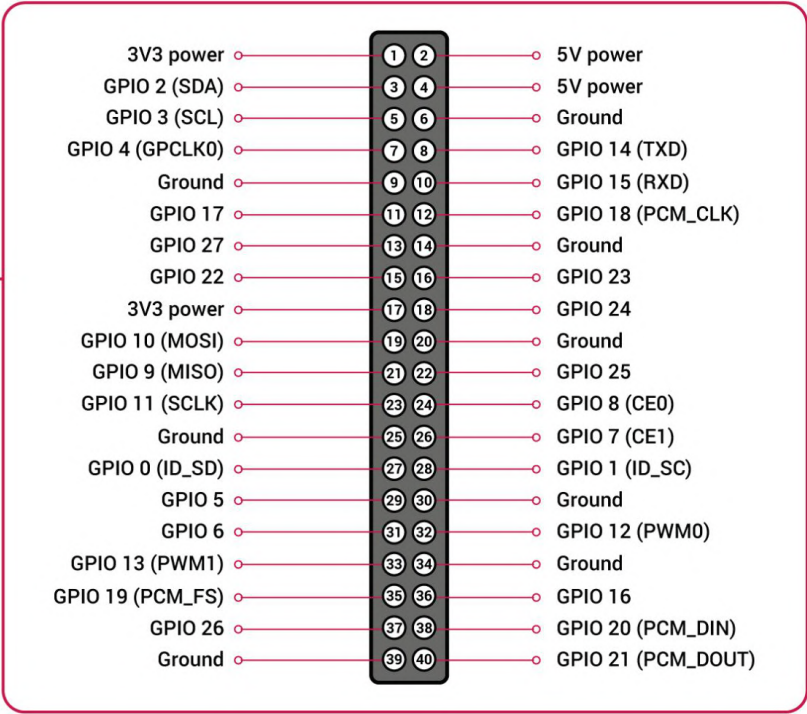
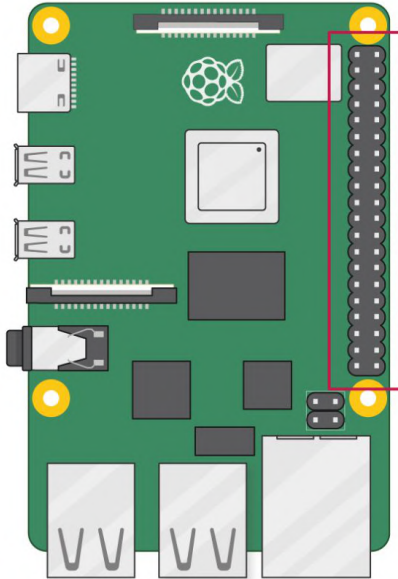
- Raspberry Pi 4
- Sunfounders DC-DC converter module
- HC-SR04 Ultrasonic Distance Sensor
- L298N Motor Driver
- PCA9685 16 Channel Servo I2C Module
- 3x SG90 180° Servo
- 2x DC Gear Motor
- USB Video Camera
- 7 segment voltmeter
- USB Wifi adapter

# Raspberry Pi

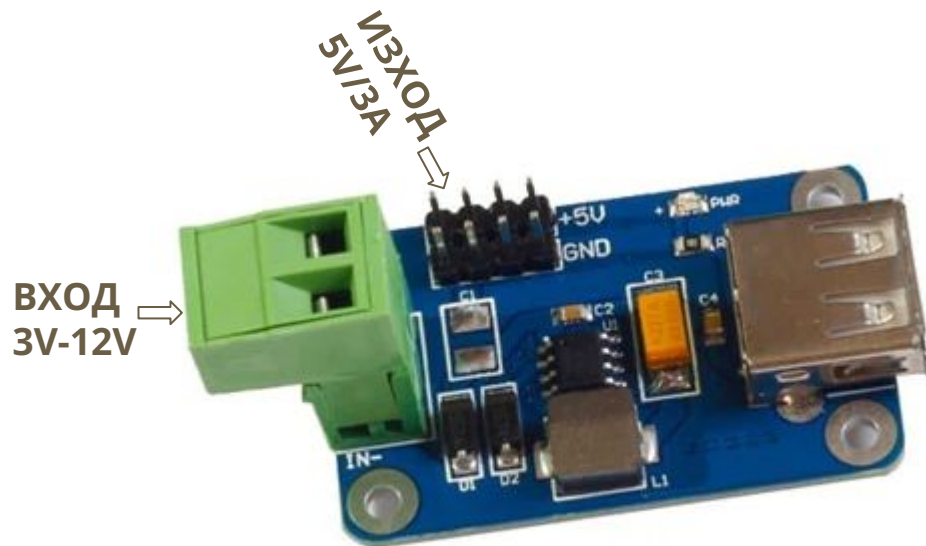
- Напълно функционален едноплатков компютър
- CPU, GPU, RAM, Networking, USB, HDMI, Audio
- Но най-важното - GPIO:
  - General Purpose Input/Output
- Моята версия е 4:
  - 1.8 GHz 64-bit quad core ARM processor
  - 4GB RAM
- Последният модел е 5:
  - 2.4 GHz 64-bit quad core ARM processor
  - 8GB RAM
- Има специално Линукс дистрибуция - Raspberry Pi OS
  - Debian базирано



# Pinout



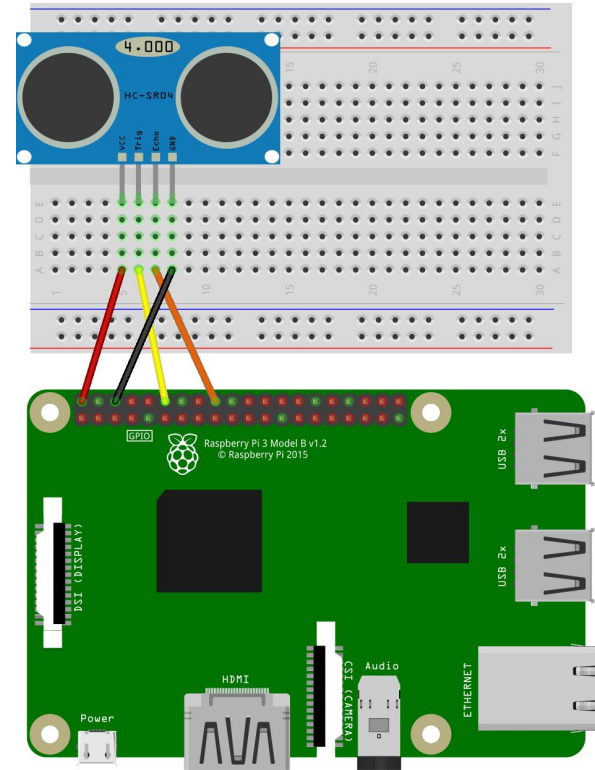
# Захранването



# RPi.GPIO

- pip install RPi.GPIO
- `import RPi.GPIO as GPIO`
- `GPIO.setmode(GPIO.BOARD/GPIO.BCM)` # BCM => Broadcom
- `GPIO.getmode()`
- `GPIO.setwarnings(False)`
- `GPIO.setup(channel, GPIO.IN/OUT)`
- `GPIO.setup(channel, GPIO.OUT, initial=GPIO.LOW/HIGH)`
- `GPIO.input(channel)`
- `GPIO.output(channel, GPIO.LOW/HIGH)`
- `GPIO.cleanup()`
- `GPIO.RPI_INFO`
- `GPIO.VERSION`

# HC-SR04 Ultrasonic Distance Sensor

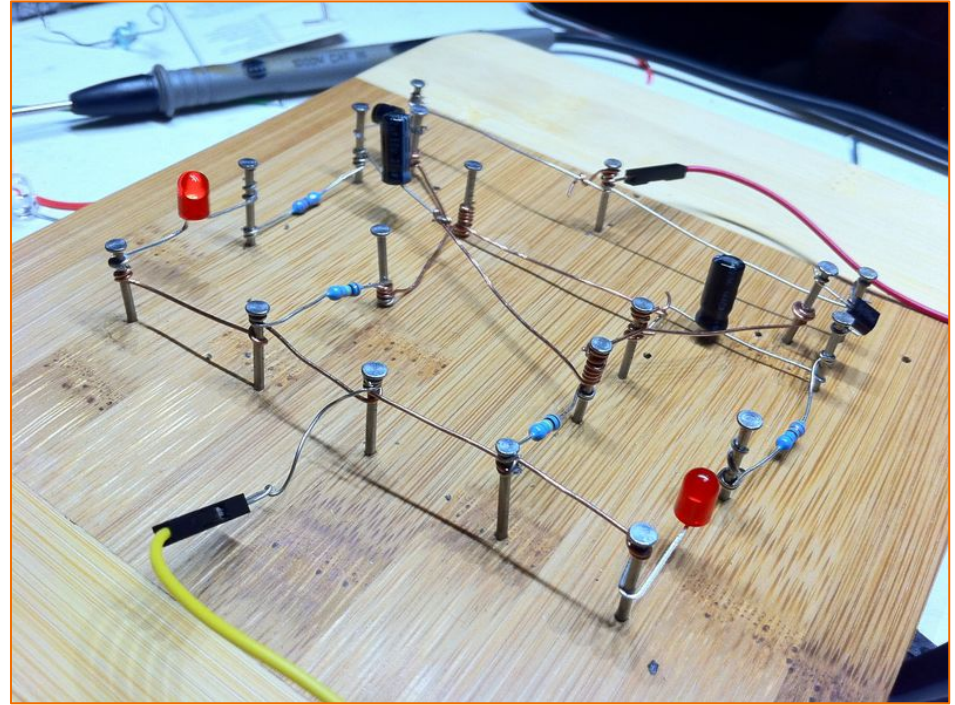
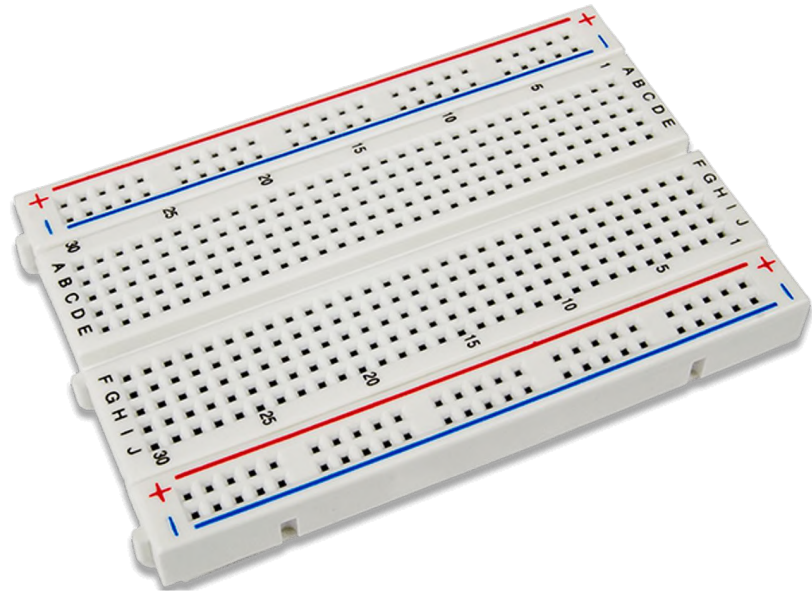


# Breadboard?

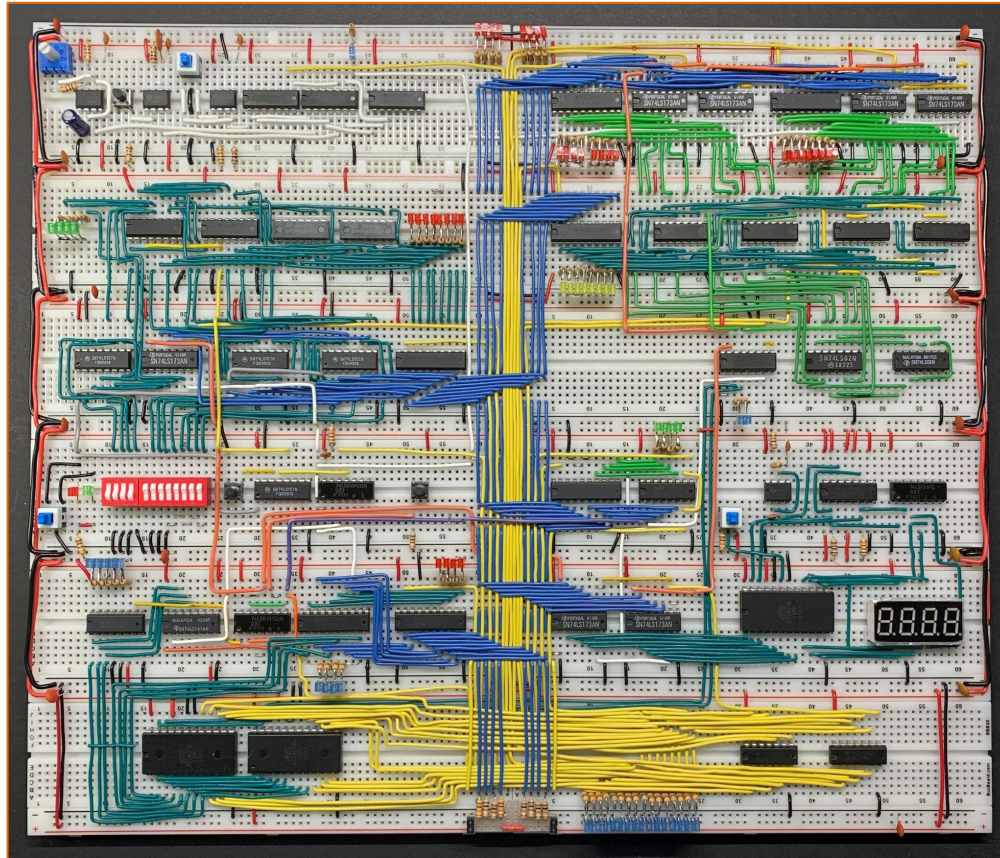




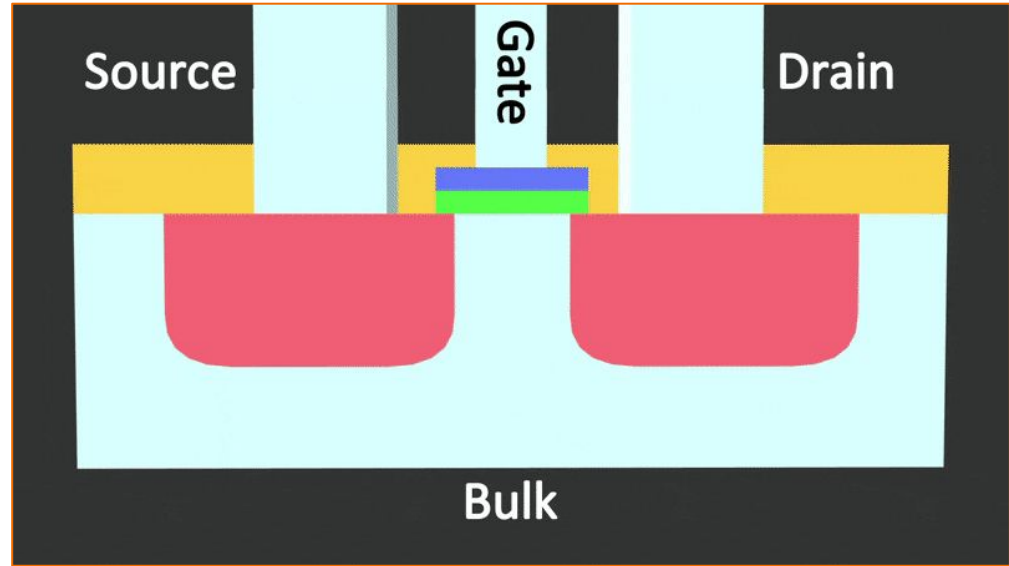
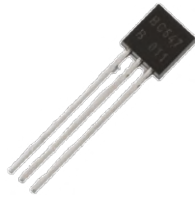
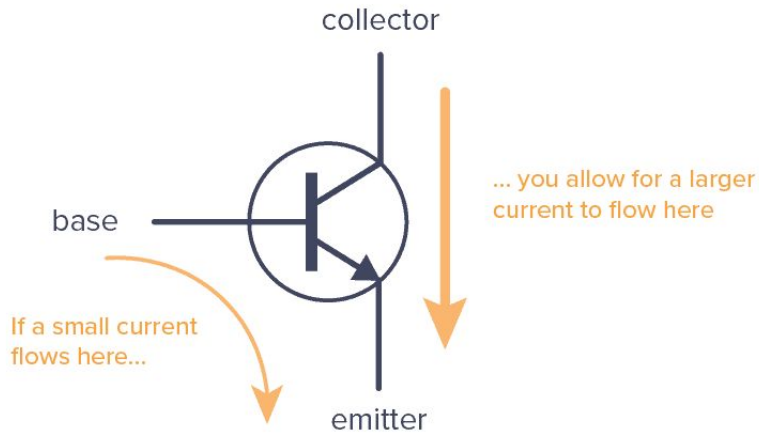
# Breadboard



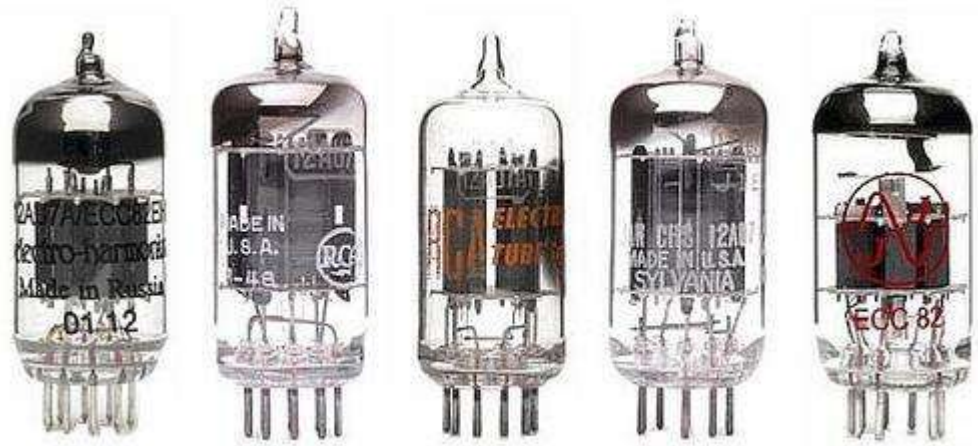
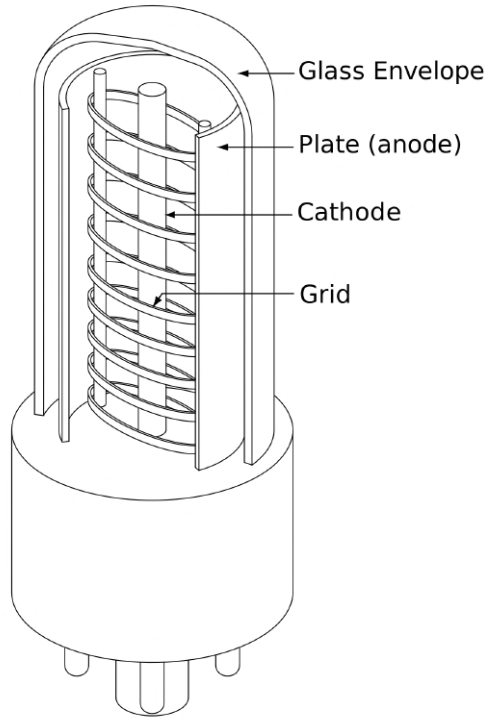
# Breadboard 8-bit computer by Ben Eater



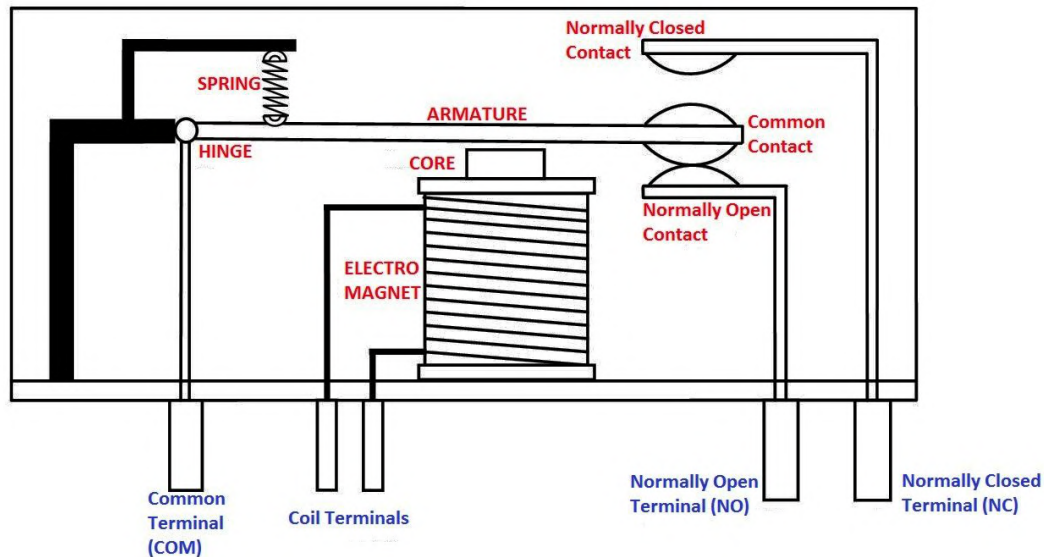
# Основа - транзистор



# Преди транзисторите - ламби



# Преди ламбите - релета



# Логиката - Gates



AND



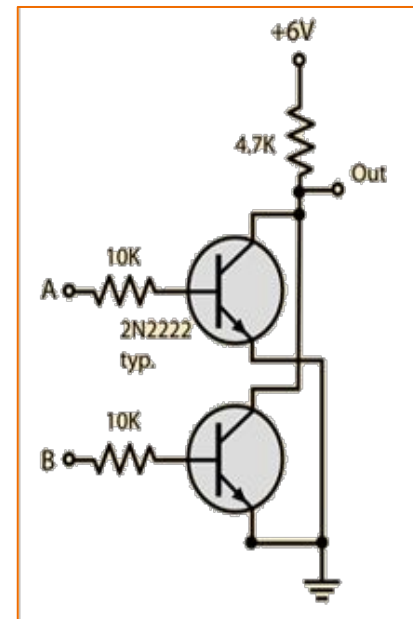
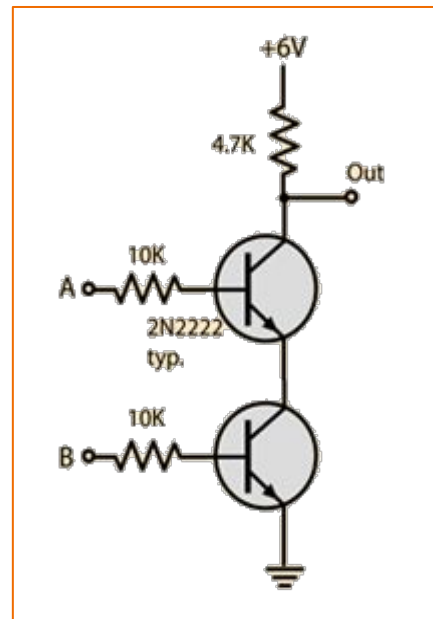
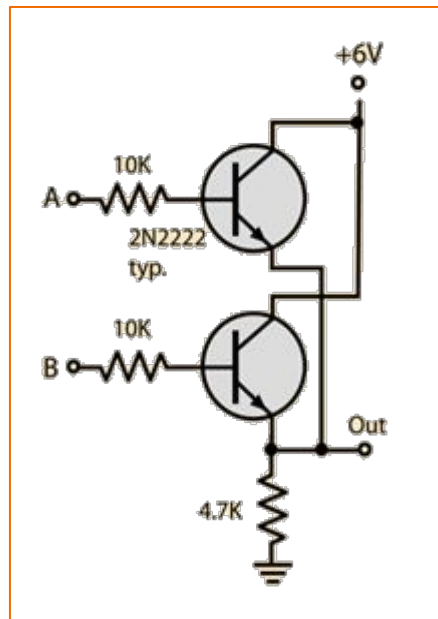
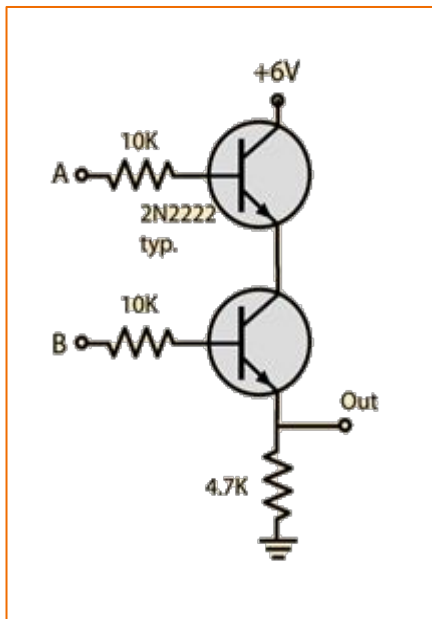
OR



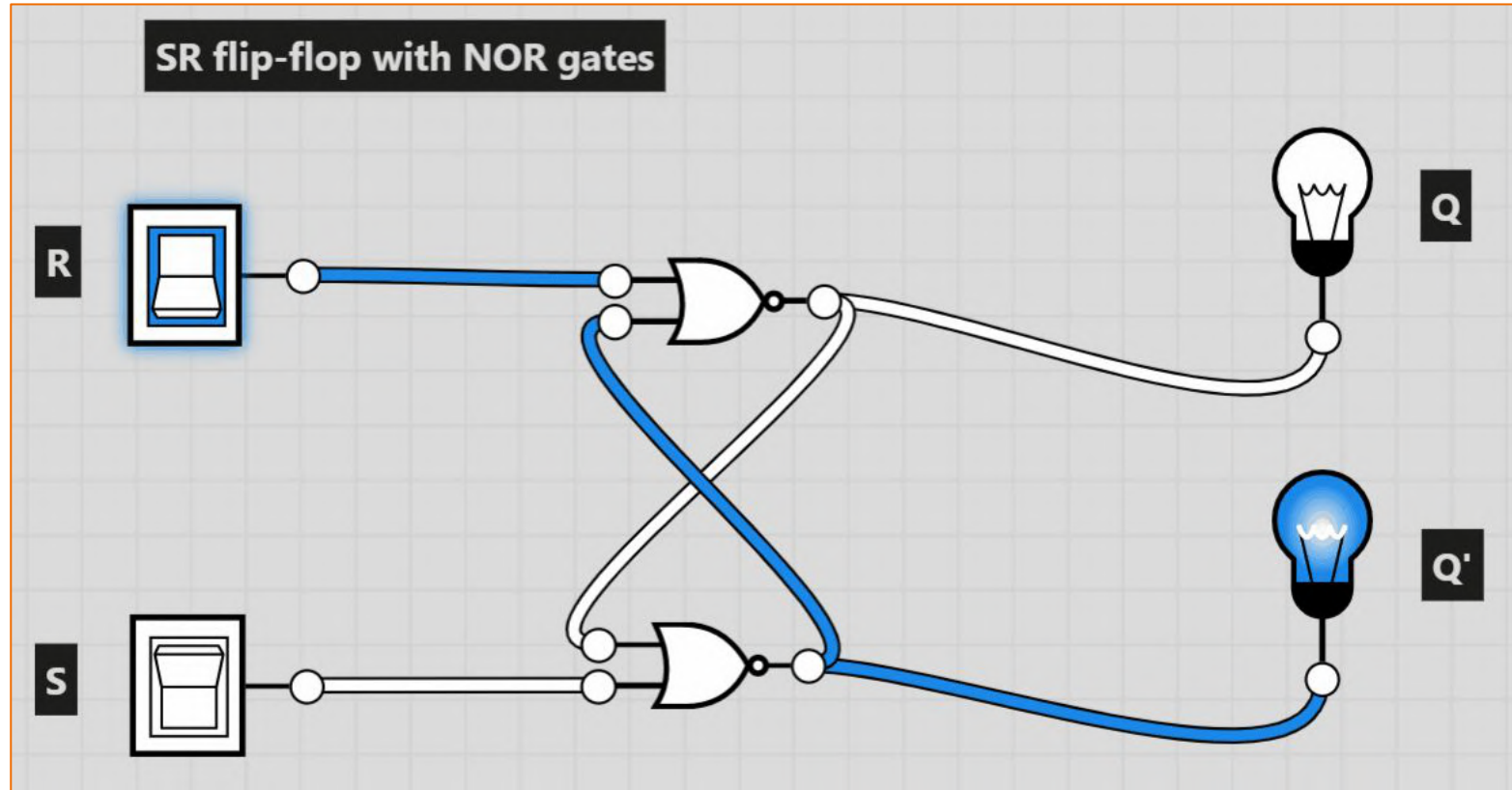
NAND



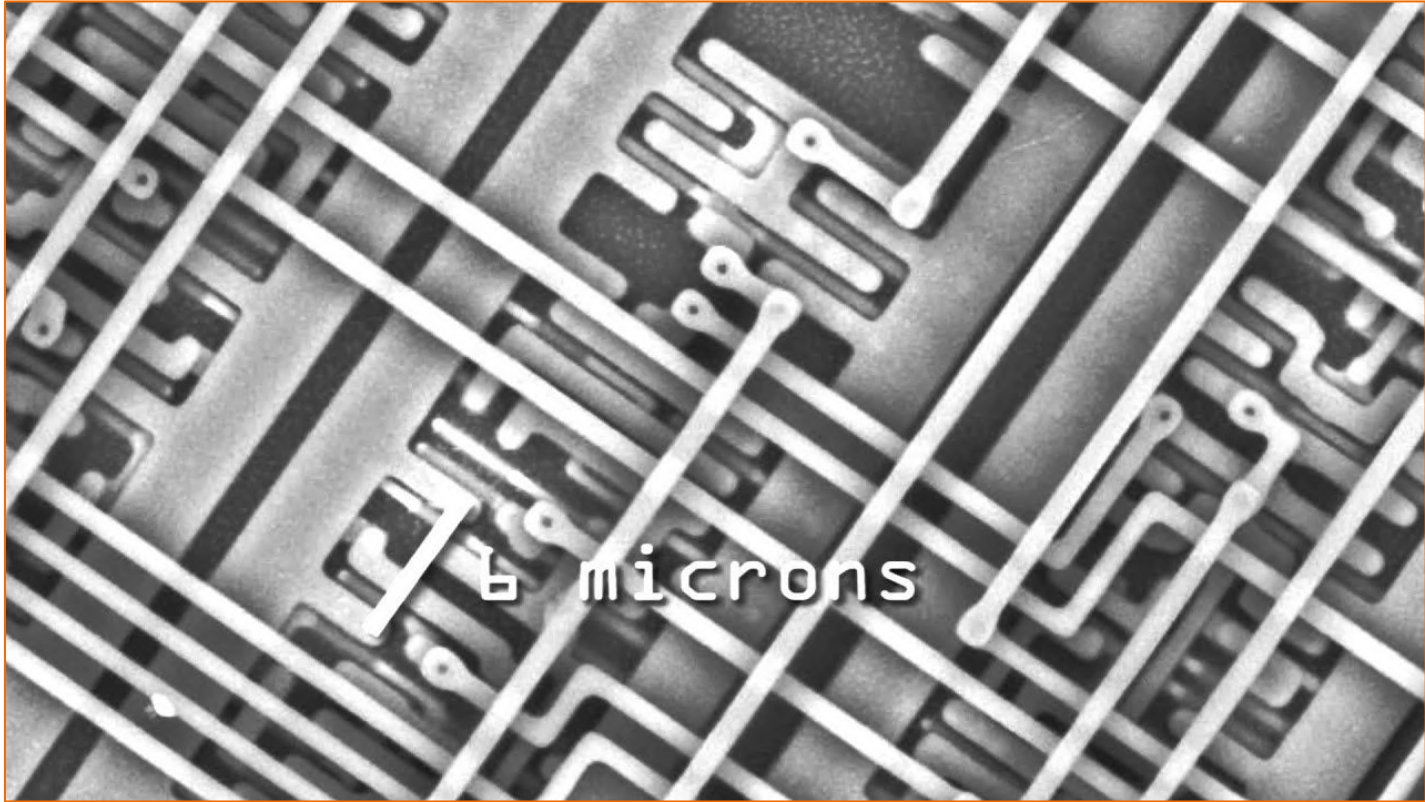
NOR



# Flip-flop

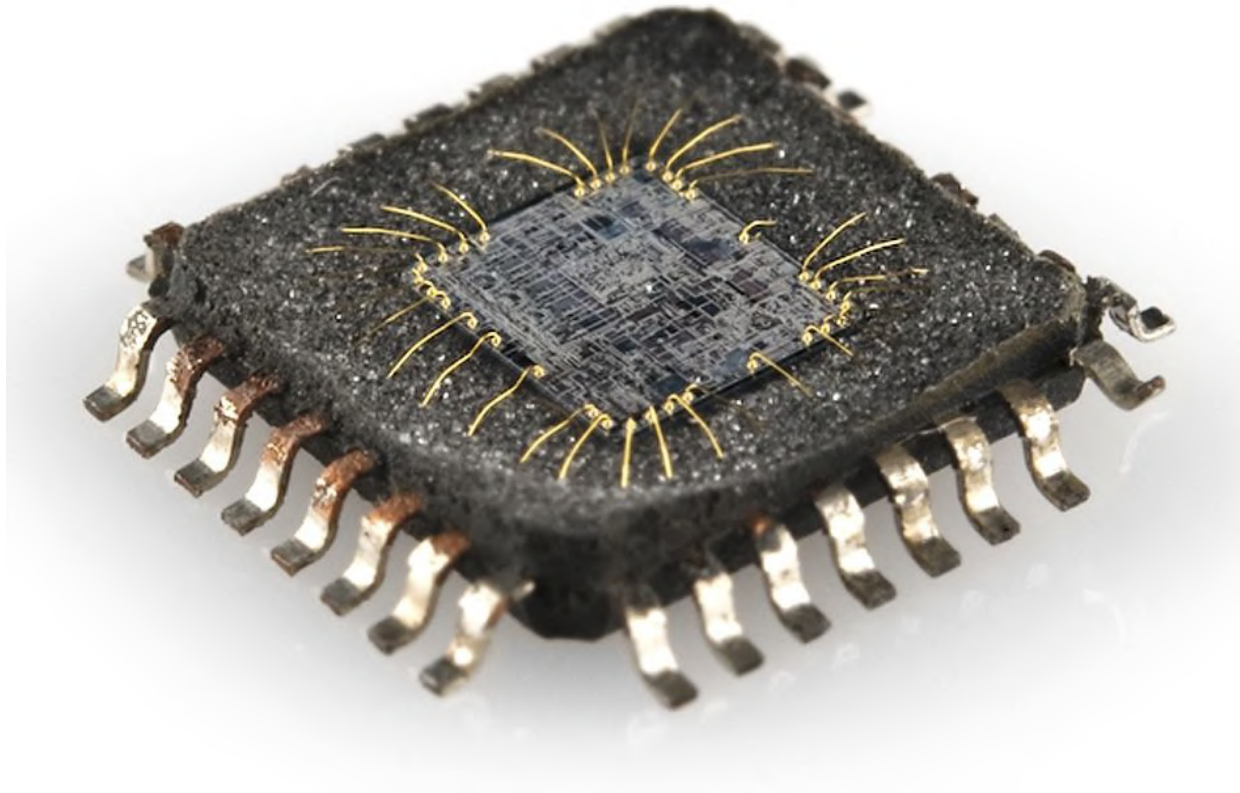


# “Половин” век по-късно





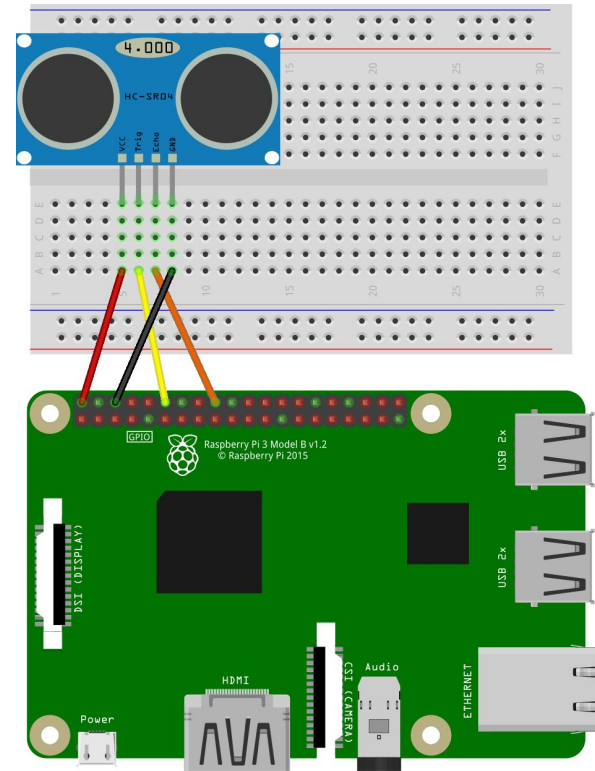
# Integrated circuit



# Да се върнем на сензора



●	VCC
●	GND
●	TRIG
●	ECHO



# Кодът 1/2

```
import RPi.GPIO as GPIO
import time

class DistanceSensor:

    def __init__(self, trig, echo):
        self._trig = trig
        self._echo = echo
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self._trig, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(self._echo, GPIO.IN)

    def __del__(self):
        GPIO.cleanup()
```

## Кодът 2/2

```
def read(self):
    """Calculate distance in cm and return it."""
    GPIO.output(self._trig, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(self._trig, GPIO.LOW)
    while not GPIO.input(self._echo):
        pulse_start = time.time()
    while GPIO.input(self._echo):
        pulse_end = time.time()
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17165 # Half the speed of sound in cm/s
    return round(distance, 1)
```

# Video Camera



# Кодът 1/2

```
import cv2

class Camera:

    def __init__(self):
        self._video = cv2.VideoCapture(0)

    def __del__(self):
        self._video.release()
```

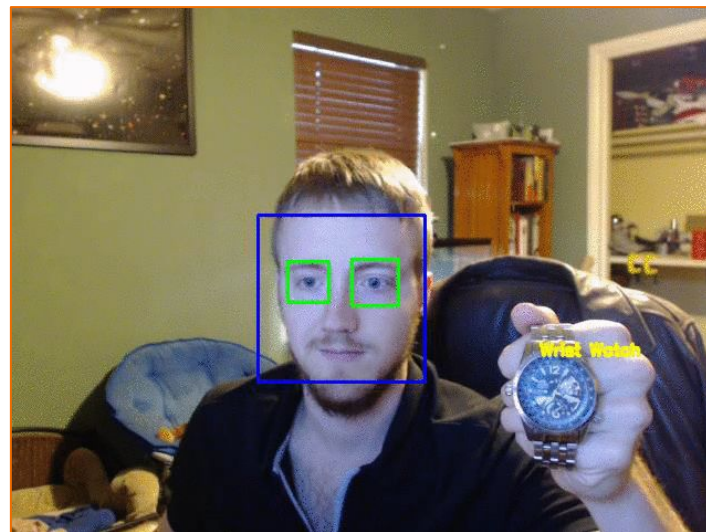
## Кодът 2/2

```
def _get_frame(self):
    success, image = self._video.read()
    if not success:
        print('Cannot read from video camera.')
        return False
    _, jpeg = cv2.imencode('.jpg', image)
    return jpeg.tostring()

def get_stream(self):
    while True:
        frame = self._get_frame()
        if frame:
            yield(b'--frame\n'
                 b'Content-Type: image/jpeg\n\n' + frame + b'\n\n')
```

# Open CV

- pip install opencv-python
- CV - Computer Vision
- Реализирано е на C++
- Всевъзможна функционалност за обработка на изображения
- Включително ML неща като:
  - face recognition
  - Object detection
  - Motion tracking

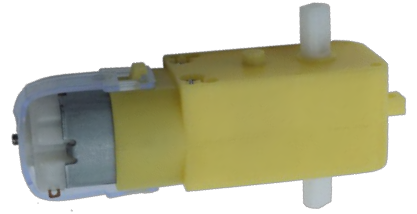
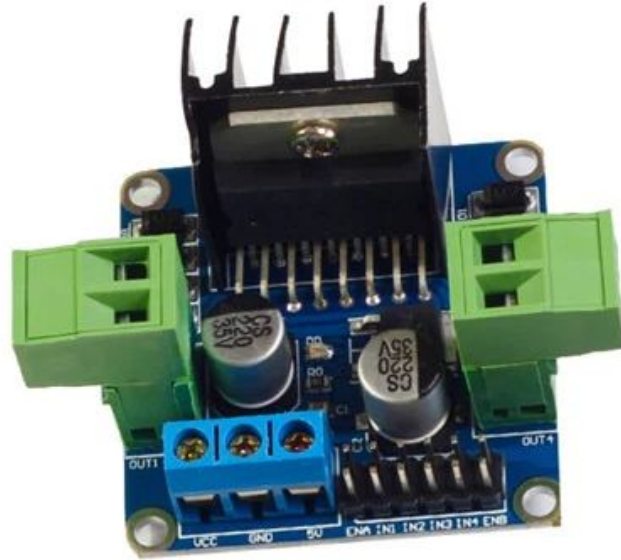
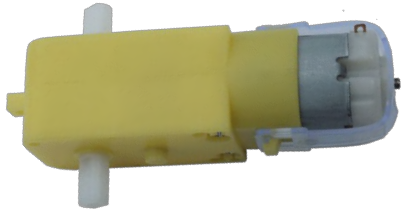




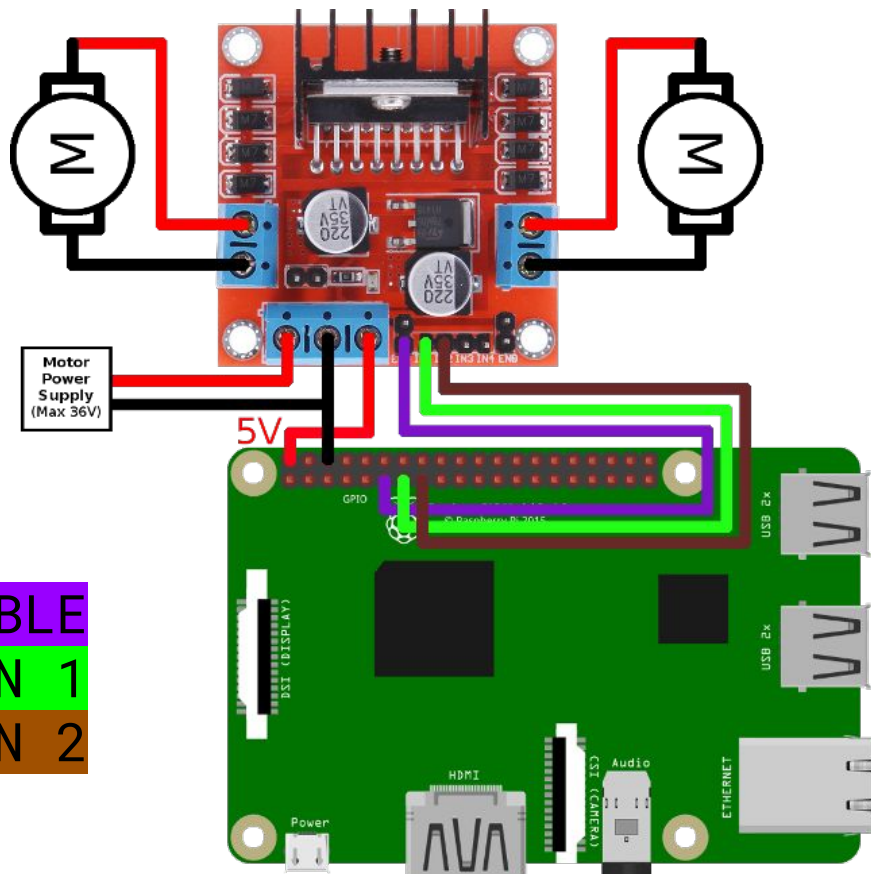
# Да добавим face recognition и за нашия клас

```
# Machine Learning магия, която си взимаш от нета:  
face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  
  
def _add_faces(self, image):  
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
    faces = face.detectMultiScale(gray, 1.1, 4)  
    for (x, y, w, h) in faces:  
        cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)
```

# Motor Driver + Motors



# Как работи?



- ENABLE
- DIRECTION 1
- DIRECTION 2

# Кодът 1/2

```
import RPi.GPIO as GPIO
```

```
class Motor:
```

```
    def __init__(self, fwd, bwd):  
        self._fwd = fwd  
        self._bwd = bwd  
        self._set_board()
```

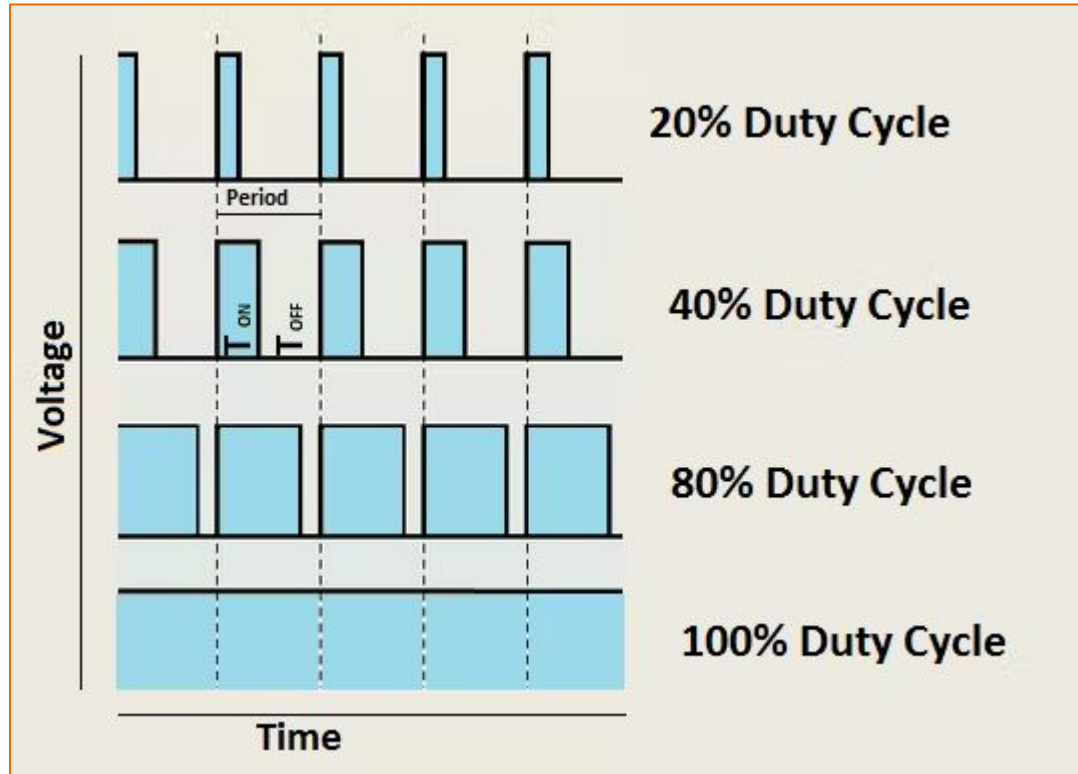
```
    def _set_board(self):  
        GPIO.setmode(GPIO.BCM)  
        GPIO.setup(self._fwd, GPIO.OUT, initial=GPIO.LOW)  
        GPIO.setup(self._bwd, GPIO.OUT, initial=GPIO.LOW)
```

```
    def __del__(self):  
        GPIO.cleanup()
```

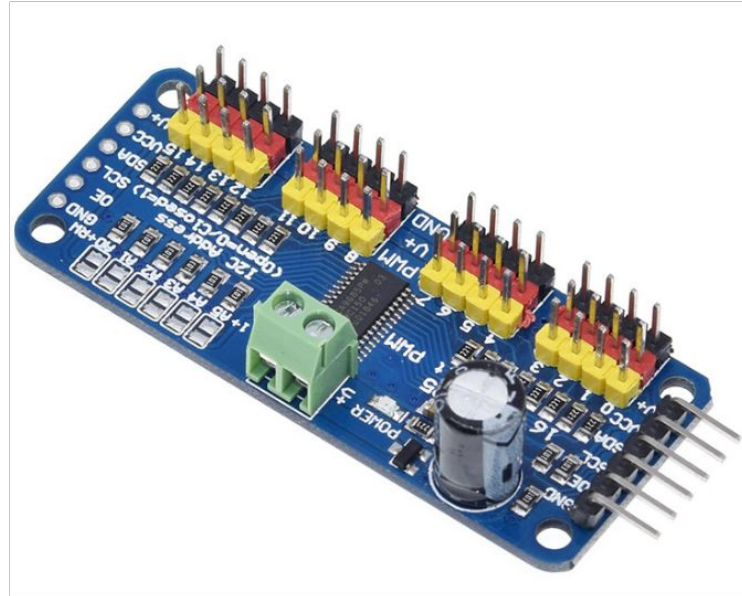
## Кодът 2/2

```
def fwd(self):  
    GPIO.output(self._fwd, GPIO.HIGH)  
    GPIO.output(self._bwd, GPIO.LOW)  
  
def bwd(self):  
    GPIO.output(self._fwd, GPIO.LOW)  
    GPIO.output(self._bwd, GPIO.HIGH)  
  
def stop(self):  
    GPIO.output(self._fwd, GPIO.LOW)  
    GPIO.output(self._bwd, GPIO.LOW)
```

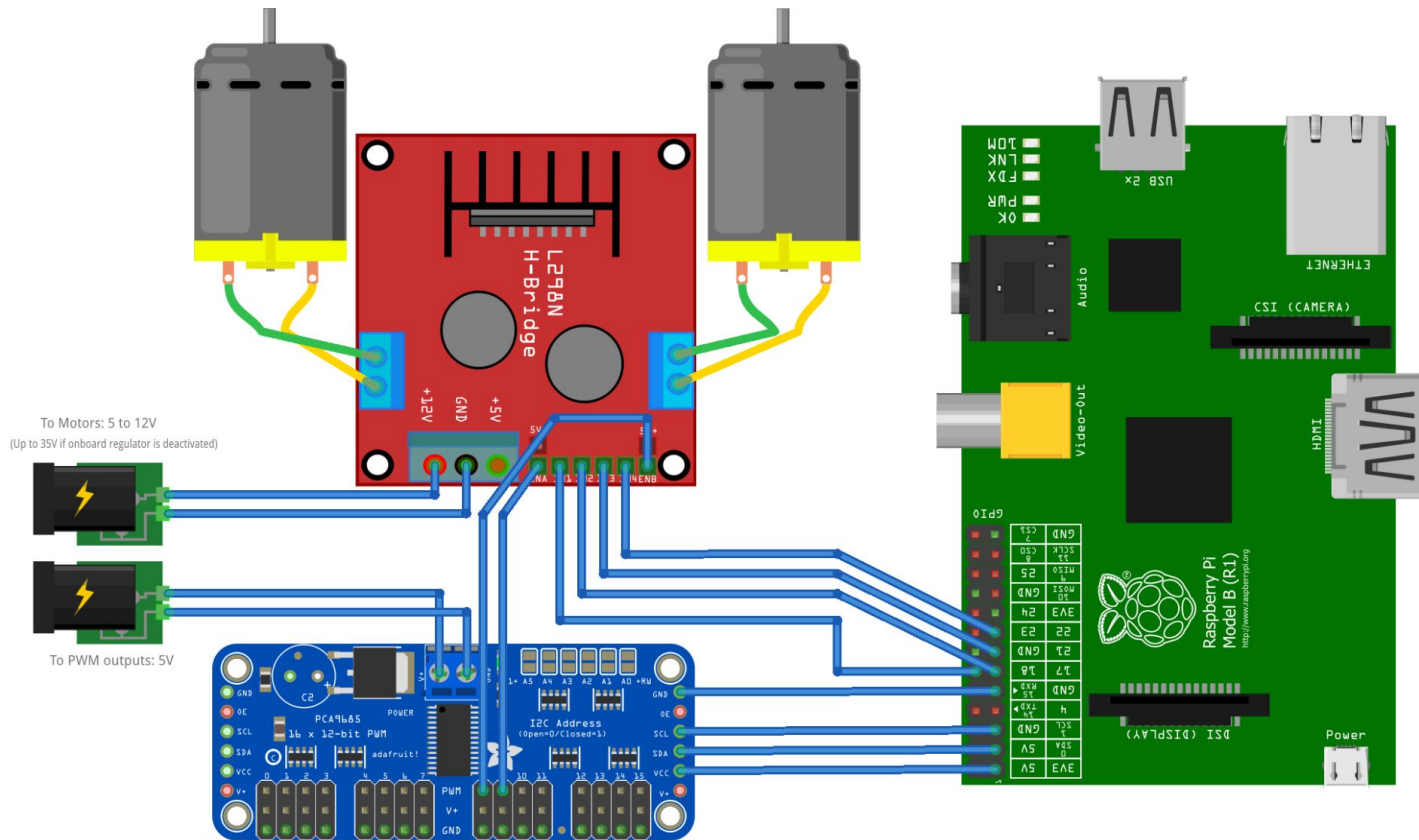
# PWM (Pulse Width Modulation)



# PCA9685 16 Channel Servo I2C Module



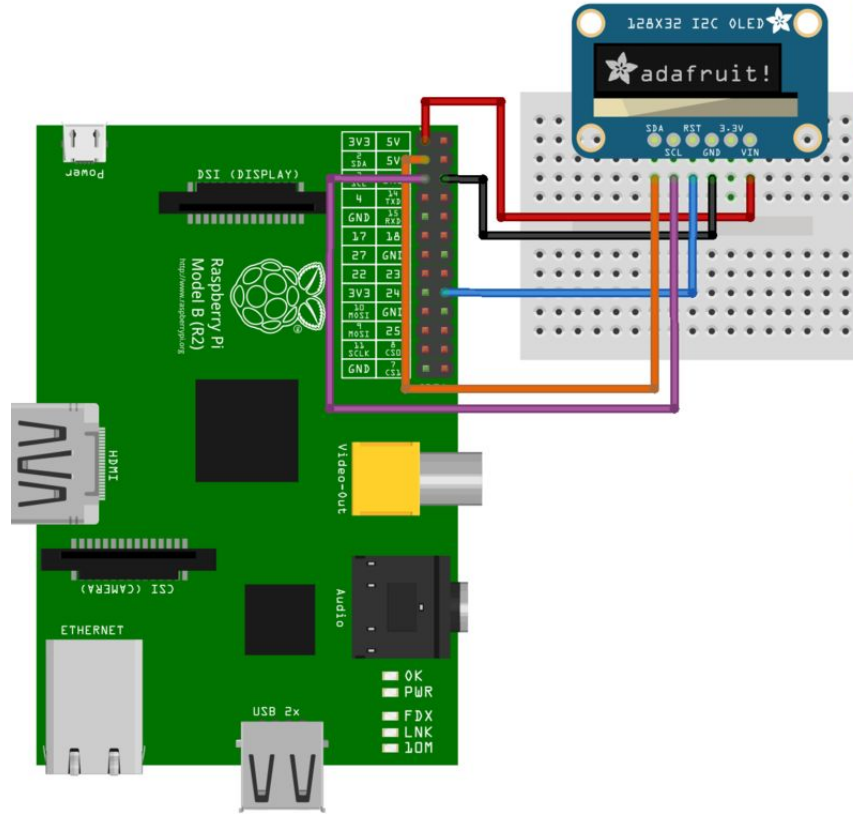
# Как работи?



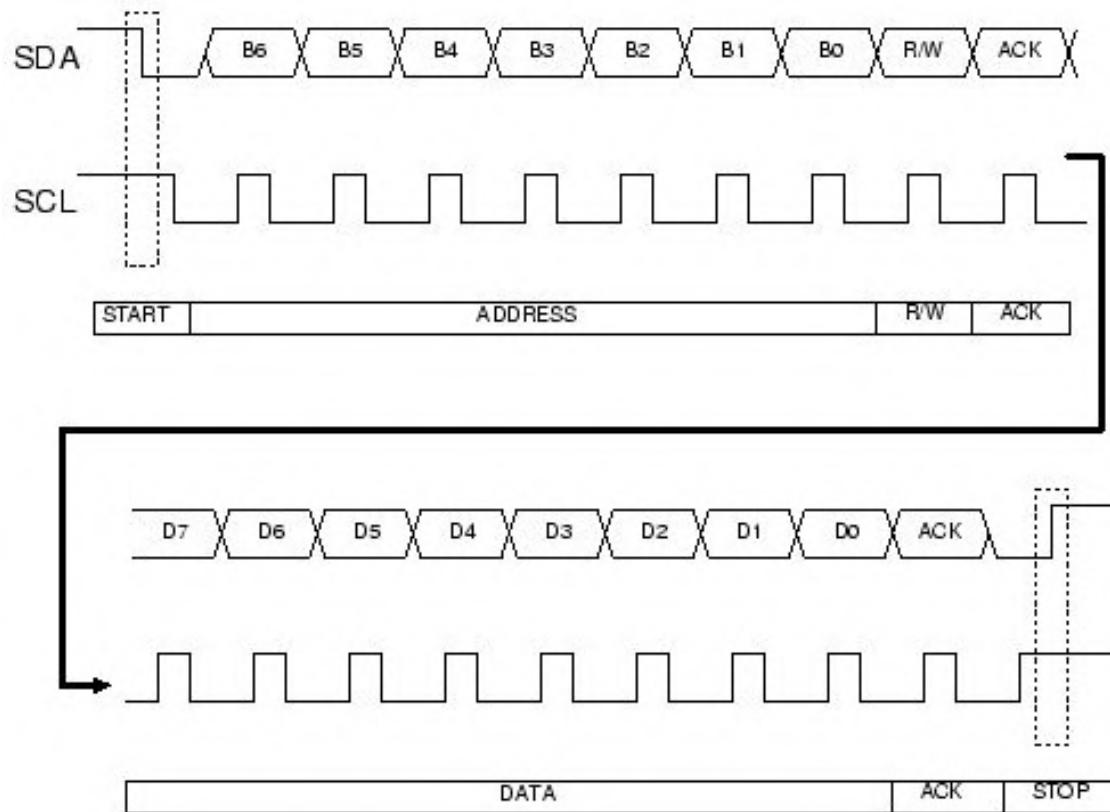


# I<sup>2</sup>C (Inter-Integrated Circuit) Interface

- VCC
- GND
- RST
- SCL
- SDA



# I<sup>2</sup>C (Inter-Integrated Circuit) Interface



# WTF?

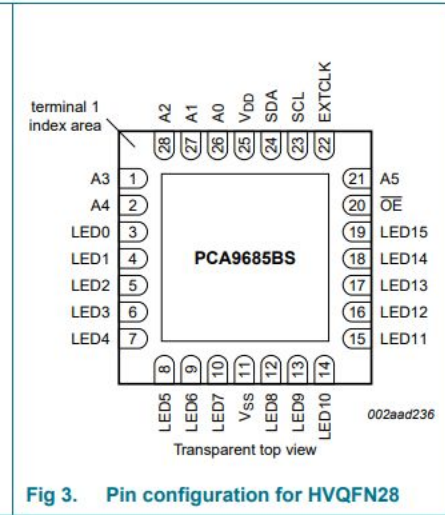
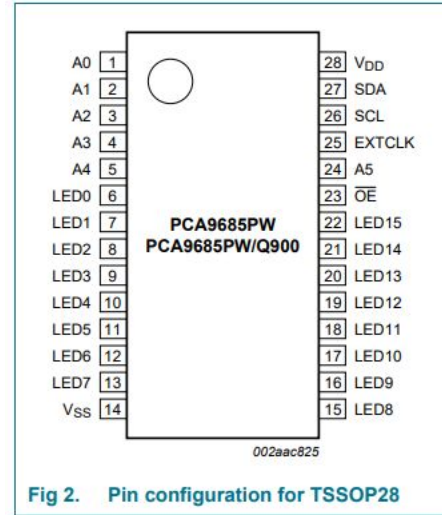
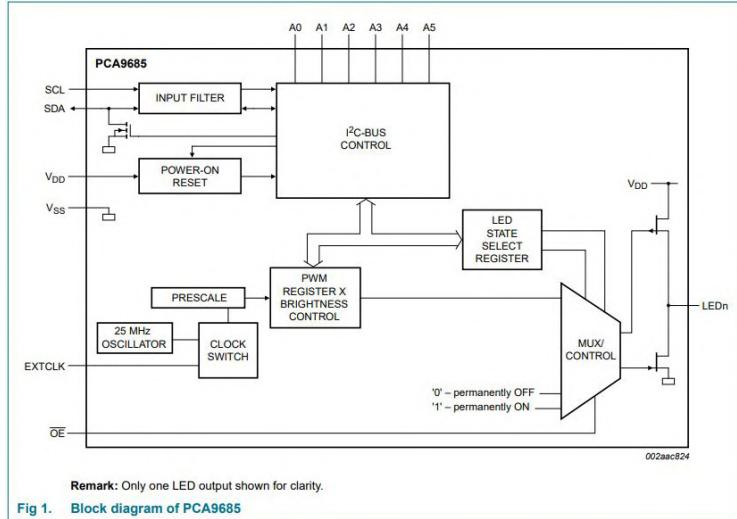


## PCA9685

16-channel, 12-bit PWM Fm+ I<sup>2</sup>C-bus LED controller

Rev. 4 — 16 April 2015

Product data sheet



# WTF?

Table 4. Register summary

Register# (decimal)	Register# (hex)	D7	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
0	00	0	0	0	0	0	0	0	0	MODE1	read/write	Mode register 1
1	01	0	0	0	0	0	0	0	1	MODE2	read/write	Mode register 2
2	02	0	0	0	0	0	0	1	0	SUBADR1	read/write	I <sup>2</sup> C-bus subaddress 1
3	03	0	0	0	0	0	0	1	1	SUBADR2	read/write	I <sup>2</sup> C-bus subaddress 2
4	04	0	0	0	0	0	1	0	0	SUBADR3	read/write	I <sup>2</sup> C-bus subaddress 3
5	05	0	0	0	0	0	1	0	1	ALLCALLADR	read/write	LED All Call I <sup>2</sup> C-bus address
6	06	0	0	0	0	0	1	1	0	LED0_ON_L	read/write	LED0 output and brightness control byte 0
7	07	0	0	0	0	0	1	1	1	LED0_ON_H	read/write	LED0 output and brightness control byte 1
8	08	0	0	0	0	1	0	0	0	LED0_OFF_L	read/write	LED0 output and brightness control byte 2
9	09	0	0	0	0	1	0	0	1	LED0_OFF_H	read/write	LED0 output and brightness control byte 3
10	0A	0	0	0	0	1	0	1	0	LED1_ON_L	read/write	LED1 output and brightness control byte 0

# Кодът

 [gvkunchev / fmi-robotics](#) Public

# Имплементацията

```
from .hardware.pwm import Pwm
```

```
class Engine:
```

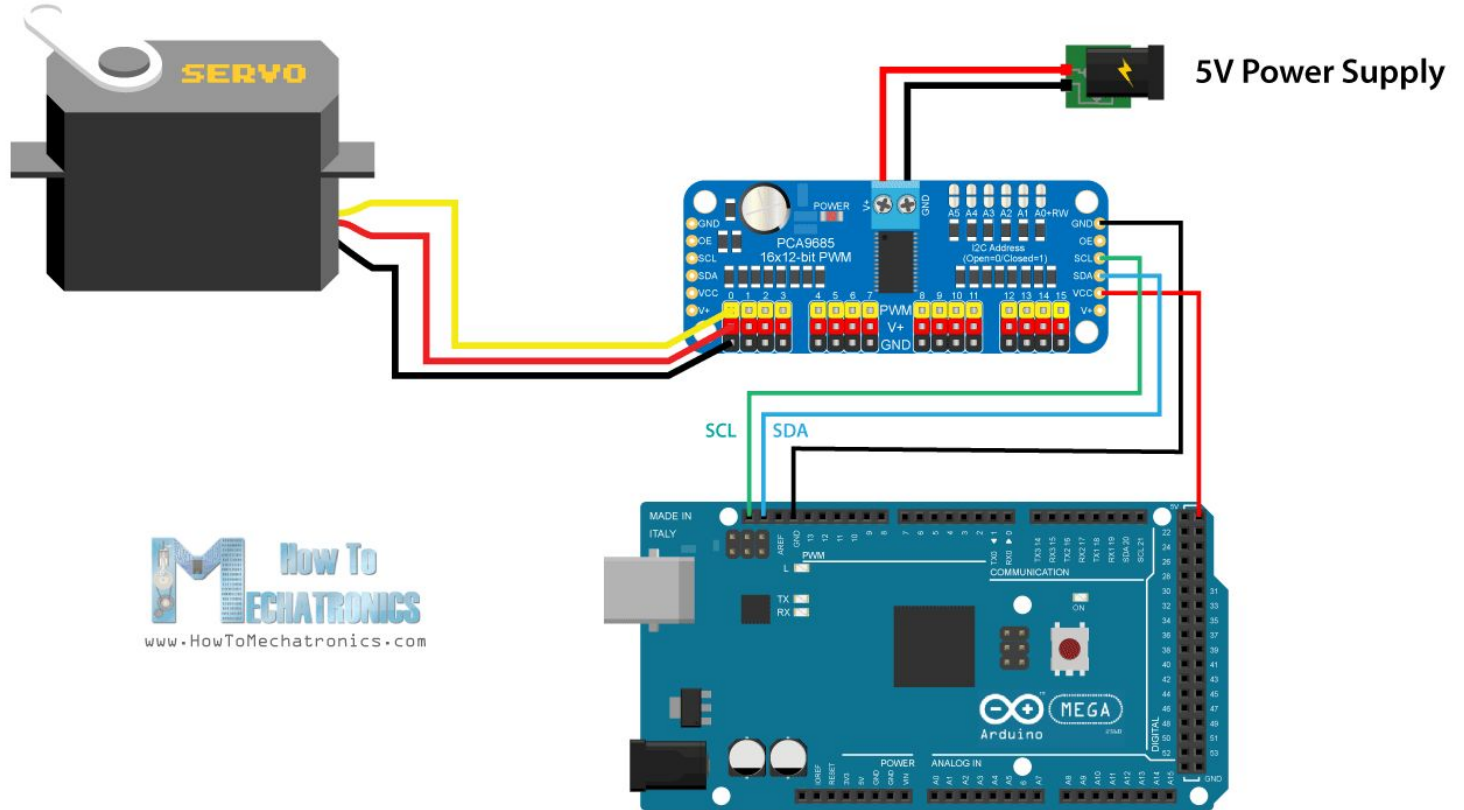
```
    def __init__(self, left, right):  
        self._pwm = Pwm()  
        self._motor_left = left  
        self._motor_right = right
```

```
    def set_speed(self, speed):  
        self._pwm.set_value(self._motor_left, 0, speed)  
        self._pwm.set_value(self._motor_right, 0, speed)
```

# SG90 180° Servo

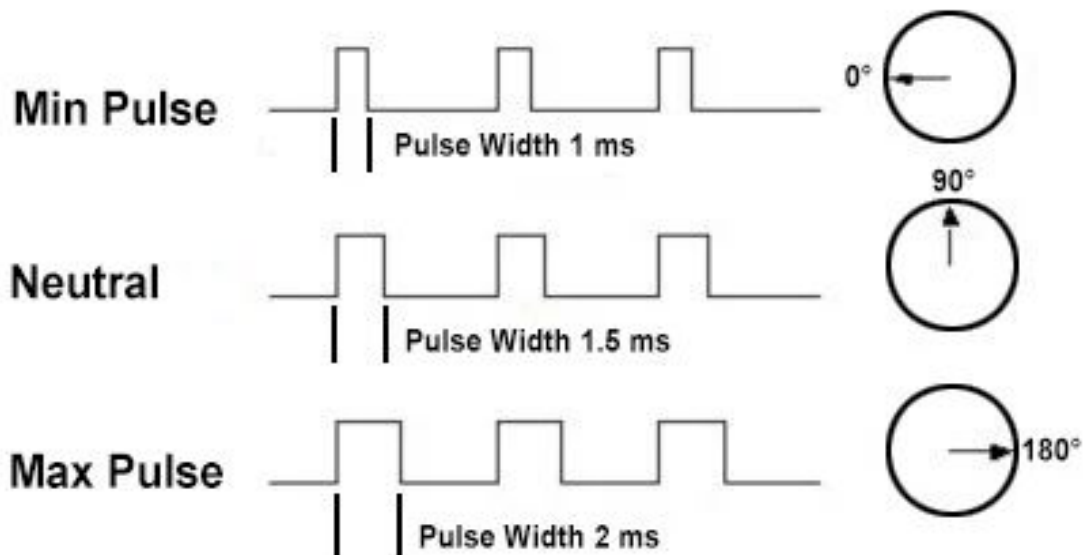


# Как се свързва?





# Как работи?



# Кодът

```
from adafruit_servokit import ServoKit

class Servo:

    def __init__(self, port, min, max):
        self._port = port
        self._iface = ServoKit(channels=8).servo[port]
        self._min = min
        self._max = max

    def set(self, value):
        value = max(value, self._min)
        value = min(value, self._max)
        self._iface.angle = value
```

# Имплементация

```
class Wheel:

    def __init__(self, port, min, max):
        self._min = min
        self._max = max
        self._servo = Servo(port, min, max)

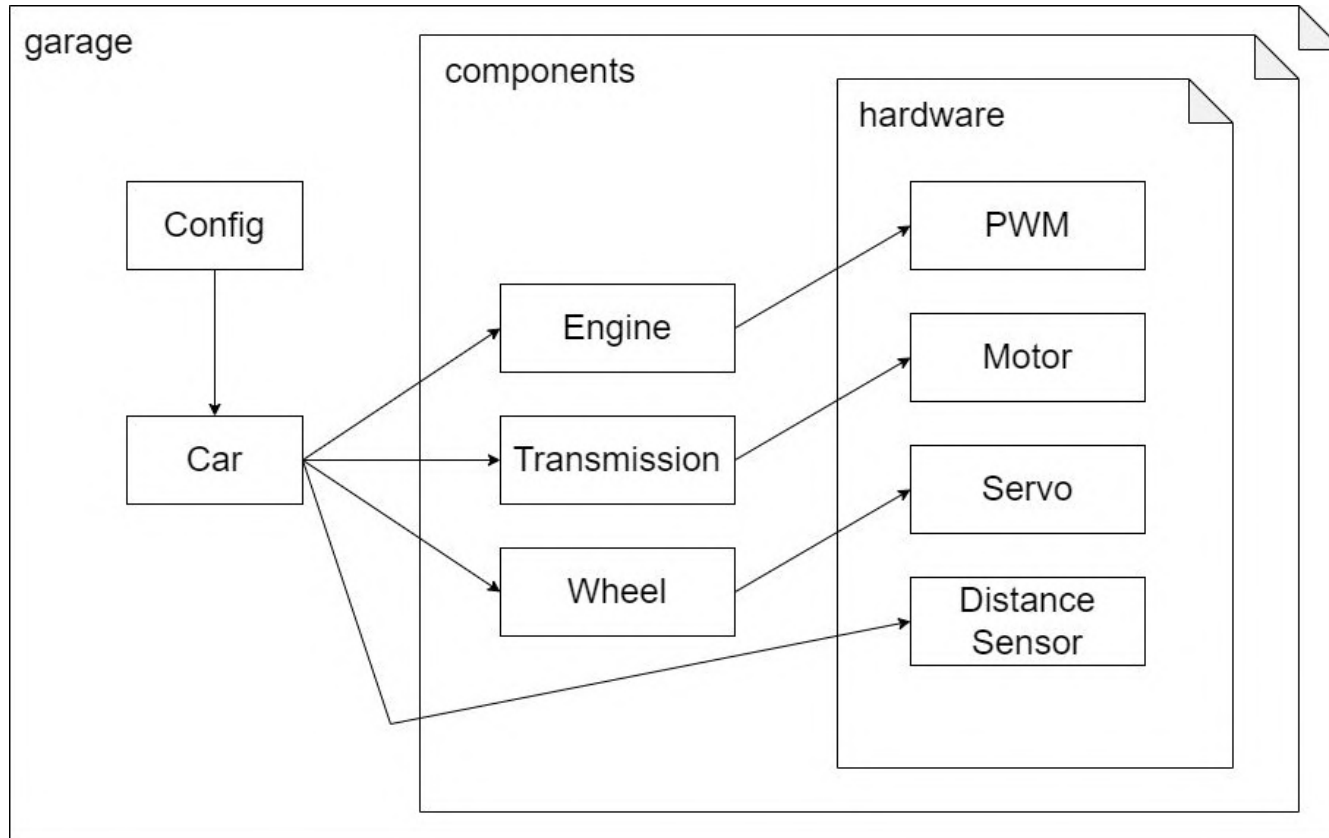
    def left(self):
        self._servo.set(self._min)

    def right(self):
        self._servo.set(self._max)

    def center(self):
        self._servo.set(self._min + (self._max - self._min) // 2)
```

# Дизайн

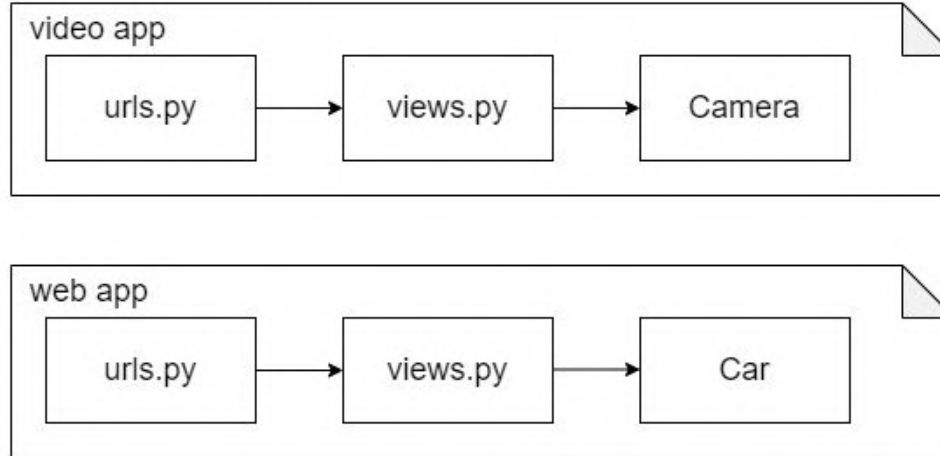
<https://github.com/gvkunchev/fmi-robotics/tree/main/car/web/garage>



# Interface

<https://github.com/gvkunchev/fmi-robotics/tree/main/car>

Django



# Бонус - COT ала Raspberry Pi + Python (Сървър)



# Бонус - СОТ ала Raspberry Pi + Python (Клиент)



**Въпроси?**