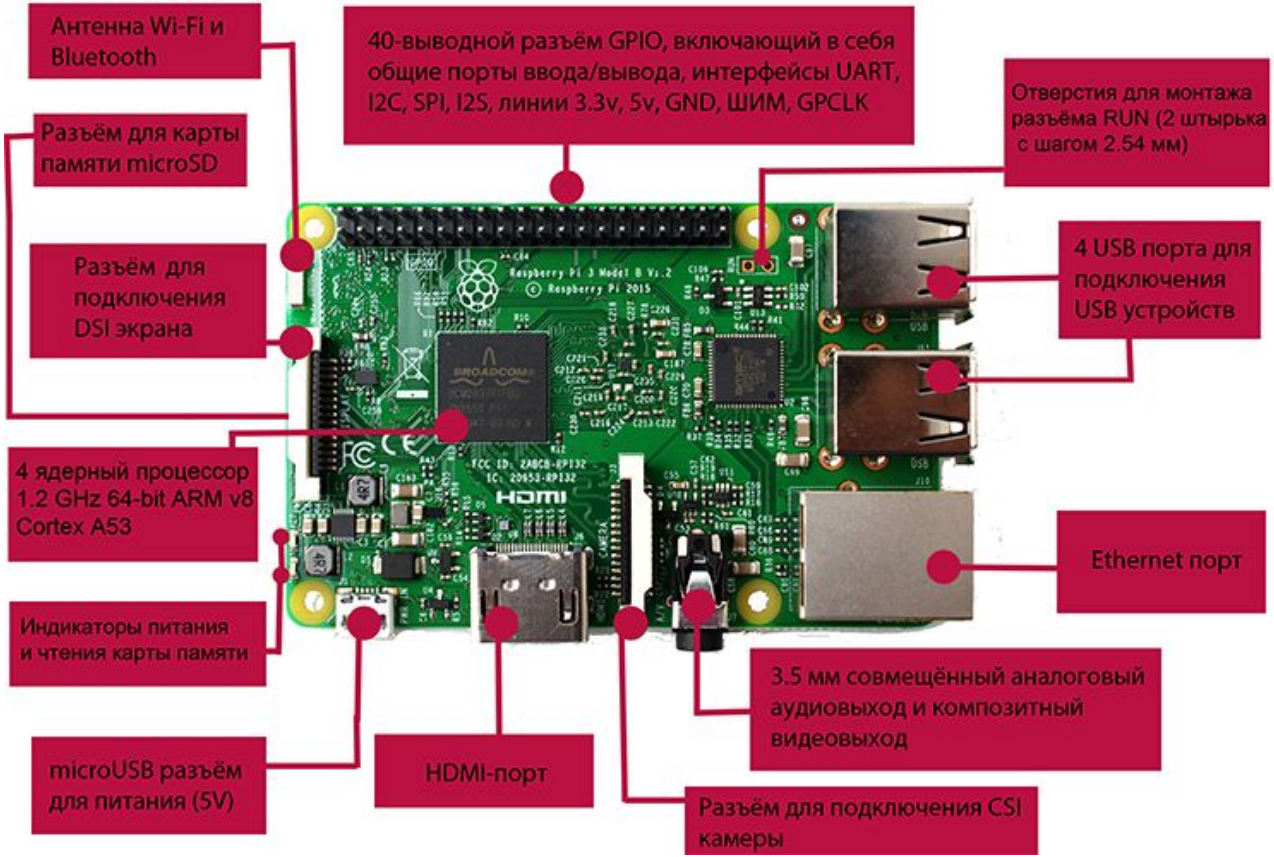
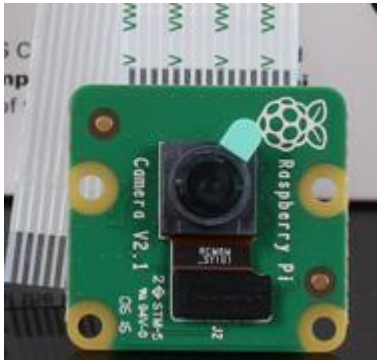
Raspberry PI. Описание возможностей GPIO

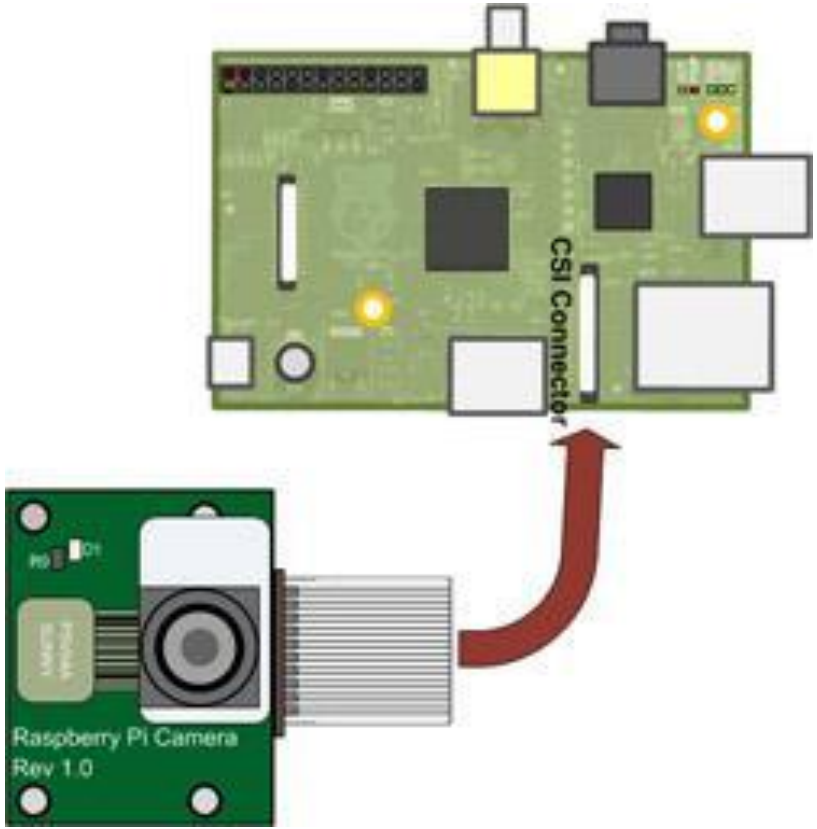
лекция 1

Raspberry PI 3



CSI камера

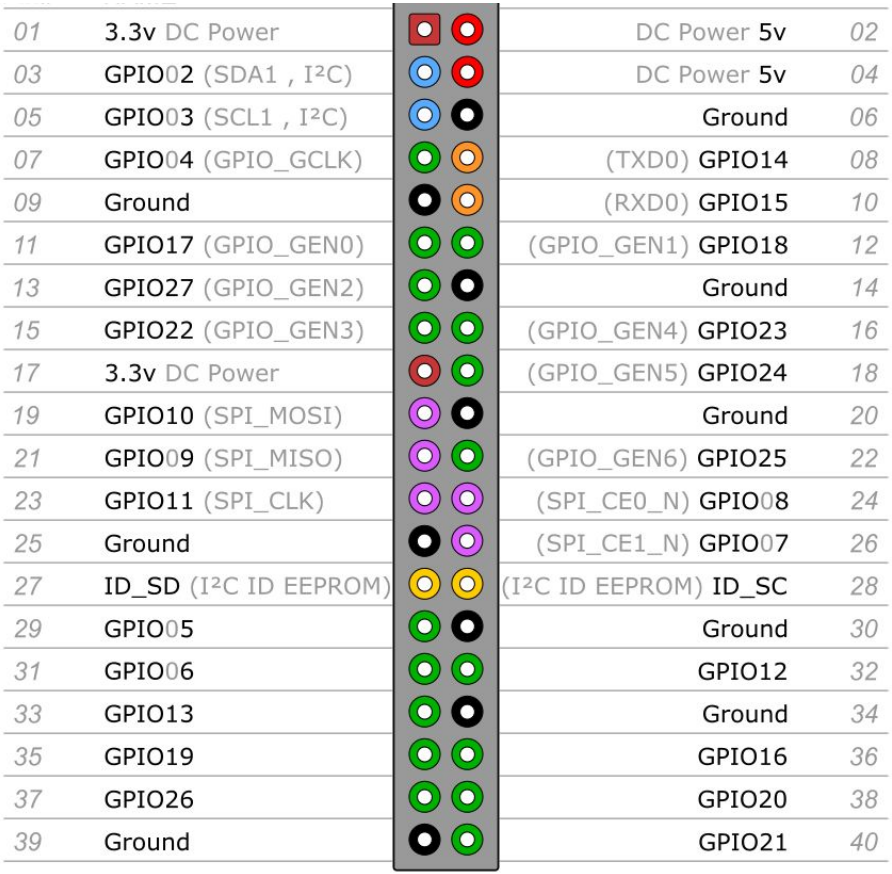




Данная камера использует аппаратные ресурсы видеопроцессора, в связи с чем не создаёт излишней нагрузки на центральный процессор в отличии от USB камер, которые также можно подключить к Raspberry Pi.

Подключается к CSI разъёму Raspberry Pi

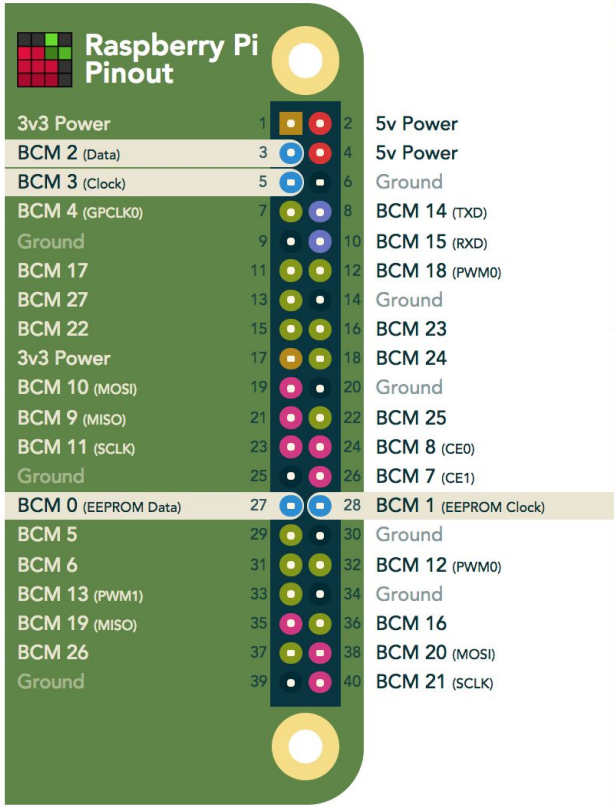
GPIO



01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

GPIO - группа портов ввода/вывода, на физическом уровне представлена разъёмом из двух рядов штырьковых выводов (пинов), расположенных с шагом 2.54мм

I2C

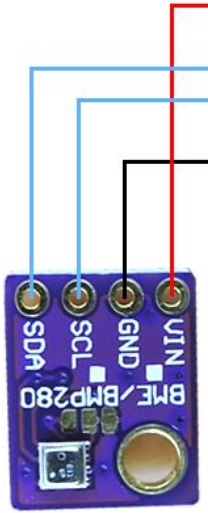


I2C - двунаправленная шина передачи данных, разработанная еще в 1980х годах компанией Philips для осуществления связи между разными схемами и устройствами. Передача данных осуществляется по двум проводам - SDA (Serial Data) и SCL (Serial Clock). На одной такой двухпроводной линии связи можно держать до 127 различных устройств и модулей которые умеют работать с шиной I²C.

10 кбит/с - 400 кбит/с

Применение
Микросхема DS1307 - часы реального времени;
Микросхема PCF8591 - аналогово<->цифровой преобразователь (4 аналоговых входа, 1 аналоговый выход);
ЖК-дисплеи

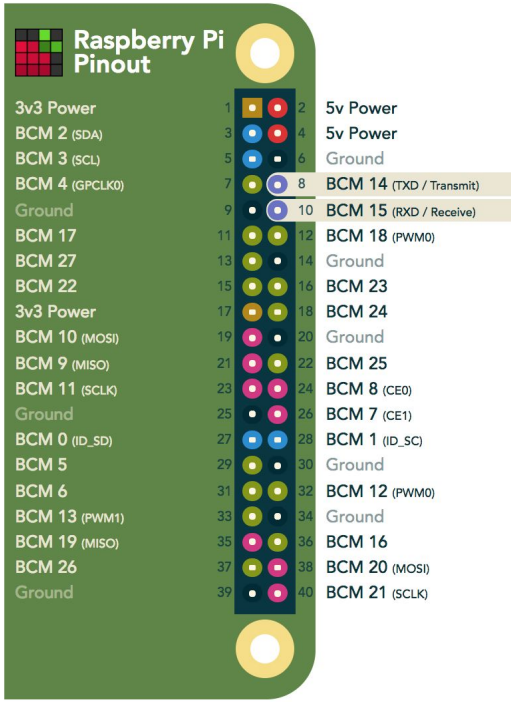
I2C. Датчика давления, температуры и влажности BME280 производства Bosch Sensortec



Raspberry Pi B+ J8 Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

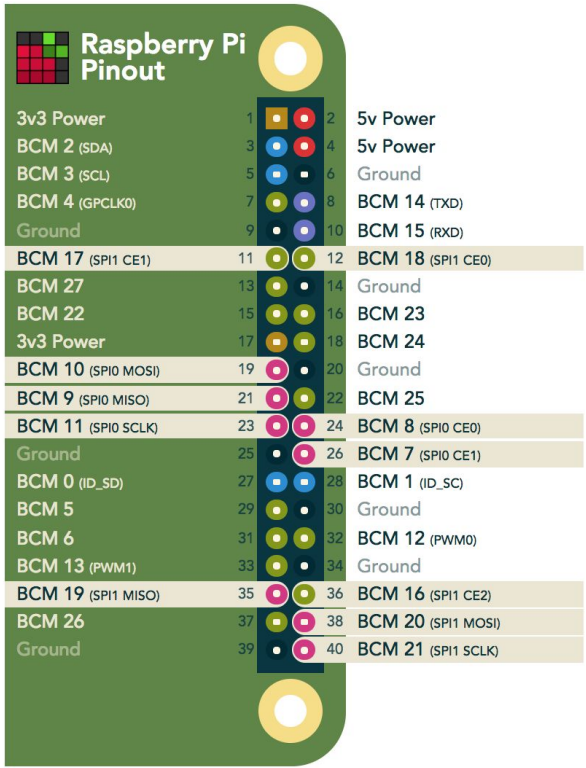
UART



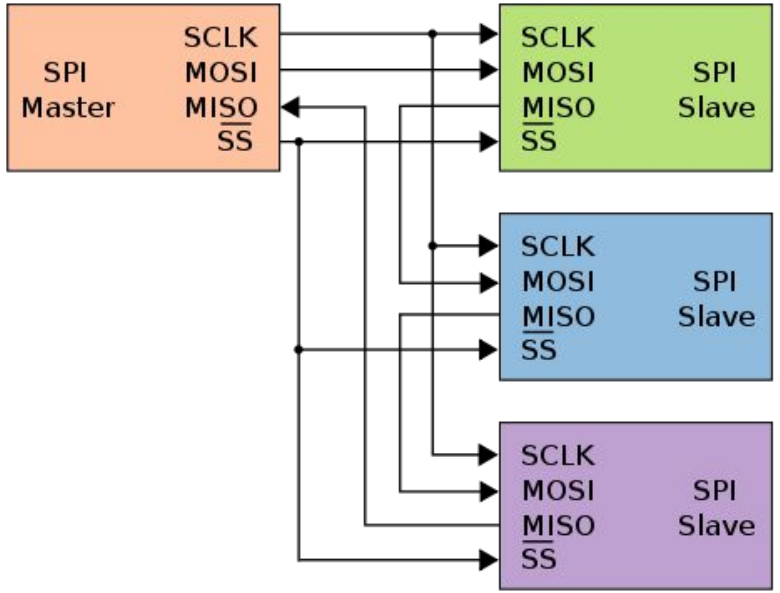
com-порт
RS-232

Примеры применения:
подключение модуля bluetooth
терминальная связь с компьютером

SPI

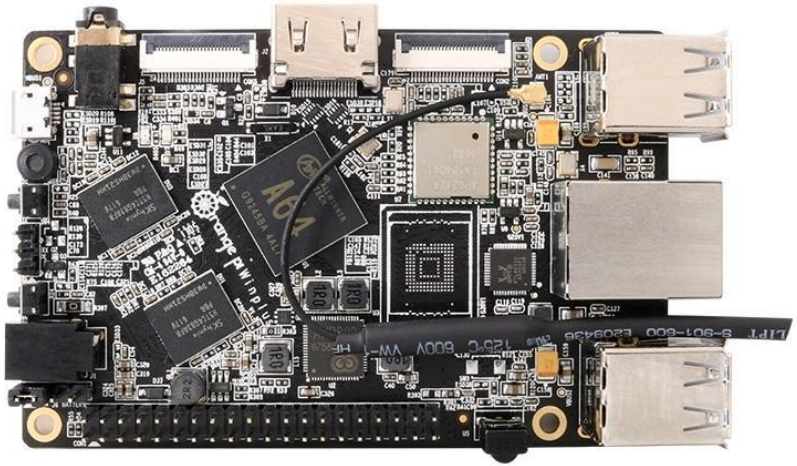


SPI (англ. Serial Peripheral Interface, SPI bus — последовательный периферийный интерфейс, шина SPI) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. SPI также иногда называют четырёхпроводным (англ. four-wire) интерфейсом.



Аналоги Raspberry

Orange Pi



Banana Pi BPI-M3

Raspbian

Raspbian is the Foundation's official supported operating system. You can install it with NOOBS or download the image below and follow our installation guide.

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more.

The Raspbian with Desktop image contained in the ZIP archive is over 4GB in size, which means that these archives use features which are not supported by older unzip tools on some platforms. If you find that the download appears to be corrupt or the file is not unzipping correctly, please try using 7Zip (Windows) or The Unarchiver (Macintosh). Both are free of charge and have been tested to unzip the image correctly.

RASPBIAN STRETCH WITH DESKTOP

Image with desktop based on Debian Stretch

Version:	November 2017
Release date:	2017-11-29
Kernel version:	4.9
Release notes:	Link

Download Torrent | Download ZIP

SHA-256: 64c4103316efe2a85fd2814f2af16313abac7d4ad68e3d95ae6709ec016e0c9f

RASPBIAN STRETCH LITE

Minimal image based on Debian Stretch

Version:	November 2017
Release date:	2017-11-29
Kernel version:	4.9
Release notes:	Link

Download Torrent | Download ZIP

SHA-256: e942b70072f2e83c446b9de6f202eb8f9692c06e7d92c343361340c

raspberry-gpio-python

Importing the module

```
try:
    import RPi.GPIO as GPIO
except RuntimeError:
    print("Error importing RPi.GPIO!  This is probably because you need superuser privileges.  You can
            achieve this by using 'sudo' to run your script")
```

Pin numbering

```
GPIO.setmode(GPIO.BOARD)
  # or
GPIO.setmode(GPIO.BCM)
```

```
mode = GPIO.getmode()
```

Channels

Warnings

```
GPIO.setwarnings(False)
```

Setup up a channel

```
GPIO.setup(channel, GPIO.IN)
GPIO.setup(channel, GPIO.OUT)
GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)

chan_list = [11,12]    # add as many channels as you want!
                       # you can tuples instead i.e.:
                       #    chan_list = (11,12)
GPIO.setup(chan_list, GPIO.OUT)
```

Input

To read the value of a GPIO pin:

```
GPIO.input(channel)
```

This will return either 0 / GPIO.LOW / False or 1 / GPIO.HIGH / True.

Output

To set the output state of a GPIO pin:

```
GPIO.output(channel, state)
```

State can be

0 / GPIO.LOW / False or 1 / GPIO.HIGH / True.

Output to several channels

```
chan_list = [11,12]                          # also works with tuples
GPIO.output(chan_list, GPIO.LOW)             # sets all to GPIO.LOW
GPIO.output(chan_list, (GPIO.HIGH, GPIO.LOW))   # sets first HIGH and second LOW
```

Cleanup

```
GPIO.cleanup(channel)
GPIO.cleanup( (channel1, channel2) )
GPIO.cleanup( [channel1, channel2] )
```

```
import atexit

def cleanup():
    GPIO.cleanup()
    print("Cleaning up!!")

atexit.register(cleanup)
```

atexit

atexit.**register**(*func*[, *args*[, *kwargs*]])

добавляет функцию в начало списка функции, которые должны вызываться перед выходом из программы

Inputs

Pull up / Pull down resistors

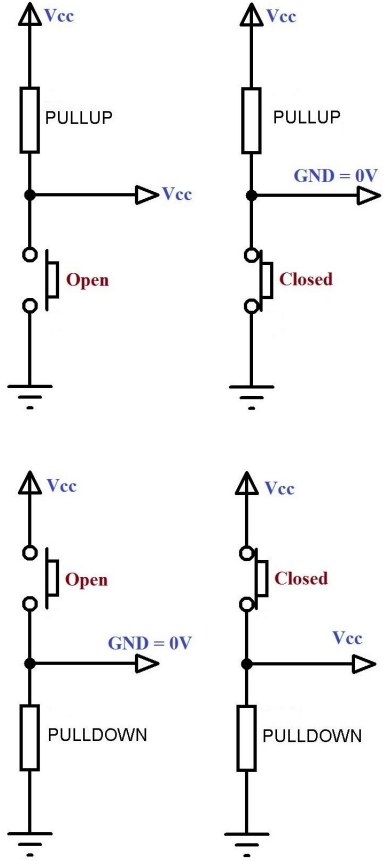
```
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)
  # or
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

Снимок в определенный момент

```
if GPIO.input(channel):
    print('Input was HIGH')
else:
    print('Input was LOW')
```

Или цикл:

```
(this assumes that pressing the button changes the input from LOW to
HIGH)
while GPIO.input(channel) == GPIO.LOW:
    time.sleep(0.01)  # wait 10 ms to give CPU chance to do other things
```



Inputs. Как опрашивать правильно?

Interrupts and Edge detection!

LOW -> HIGH = rising edge

HIGH -> LOW = falling edge

wait_for_edge() function

GPIO.wait_for_edge(channel, GPIO.RISING) Блокировка программы!!!

GPIO.RISING
GPIO.FALLING
GPIO.BOTH

```
# wait for up to 5 seconds for a rising edge (timeout is in milliseconds)
channel = GPIO.wait_for_edge(channel, GPIO_RISING, timeout=5000)
if channel is None:
    print('Timeout occurred')
else:
    print('Edge detected on channel', channel)
```

event_detected() function

```
GPIO.add_event_detect(channel, GPIO.RISING)  # add rising edge detection
on a channel
do_something()
if GPIO.event_detected(channel):
    print('Button pressed')
```

Remove event detection

```
GPIO.remove_event_detect(channel)
```

Threaded callbacks

```
def my_callback(channel):
    print('This is a edge event callback function!')
    print('Edge detected on channel %s'%channel)
    print('This is run in a different thread to your main program')

GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback)  # add rising edge detection on a channel
...the rest of your program...
```

```
def my_callback_one(channel):
    print('Callback one')

def my_callback_two(channel):
    print('Callback two')

GPIO.add_event_detect(channel, GPIO.RISING)
GPIO.add_event_callback(channel, my_callback_one)
GPIO.add_event_callback(channel, my_callback_two)
```

Switch debounce (шумоподавитель)

You may notice that the callbacks are called more than once for each button press. This is as a result of what is known as 'switch bounce'. There are two ways of dealing with switch bounce:

- add a 0.1uF capacitor across your switch.
- software debouncing
- a combination of both

To debounce using software, add the bouncetime= parameter to a function where you specify a callback function. Bouncetime should be specified in milliseconds. For example:

```
# add rising edge detection on a channel, ignoring further edges for 200ms for switch bounce handling
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback, bouncetime=200)
```

or

```
GPIO.add_event_callback(channel, my_callback, bouncetime=200)
```

Пример, на input

```
# coding: utf-8

import RPi.GPIO as GPIO
import time, datetime
import atexit
import lab_work_3_telegram

def cleanup():
    GPIO.cleanup()
    print("Cleaning up!!")

atexit.register(cleanup)

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(19,GPIO.IN)
GPIO.setup(20,GPIO.IN)

def mic_fc_04(channel): #power 3.3v
    if GPIO.event_detected(channel):
        print(str(datetime.datetime.now()),' Шум в помещении!')

def ir_sensor_hc_sr501(channel): #power 5v
    if GPIO.event_detected(channel):
        print(str(datetime.datetime.now()),' Зафиксировано
движение!')

GPIO.add_event_detect(19, GPIO.RISING, callback=mic_fc_04,
bouncetime=100)  # add rising edge detection on a channel
GPIO.add_event_detect(20, GPIO.RISING,
callback=ir_sensor_hc_sr501, bouncetime=100)


while True:
    time.sleep(10)
```

GPIO Outputs

1. First set up RPi.GPIO

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)
```

2. To set an output high:

```
GPIO.output(12, GPIO.HIGH)
 # or
GPIO.output(12, 1)
 # or
GPIO.output(12, True)
```

3. To set an output low:

```
GPIO.output(12, GPIO.LOW)
 # or
GPIO.output(12, 0)
 # or
GPIO.output(12, False)
```

4. To output to several channels at the same time:

```
chan_list = (11,12)
GPIO.output(chan_list, GPIO.LOW) # all LOW
GPIO.output(chan_list, (GPIO.HIGH,GPIO.LOW))  # first LOW, second HIGH
```

5. Clean up at the end of your program

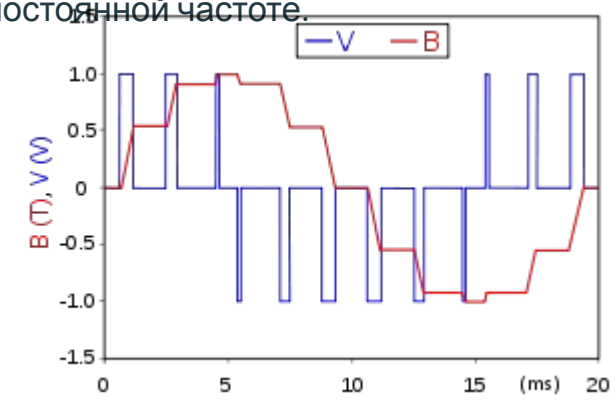
```
GPIO.cleanup()
```

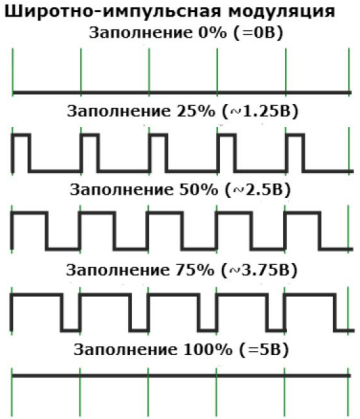
Note that you can read the current state of a channel set up as an output using the input() function. For example to toggle an output:

```
GPIO.output(12, not GPIO.input(12))
```

PWM

Широтно-импульсная модуляция (ШИМ, англ. pulse-width modulation (PWM)) — процесс управления мощностью, подводимой к нагрузке, путём изменения скважности импульсов, при постоянной частоте.





Основной причиной применения ШИМ является стремление к повышению КПД при построении вторичных источников питания электронной аппаратуры и в других узлах, например, ШИМ используется для регулировки яркости подсветки LCD-мониторов и дисплеев в телефонах, КПК и т.п..

Using PWM in RPi.GPIO

To create a PWM instance:

```
p = GPIO.PWM(channel, frequency)
```

To start PWM:

```
p.start(dc)   # where dc is the duty cycle (0.0 <= dc <= 100.0)
```

To change the frequency:

```
p.ChangeFrequency(freq)   # where freq is the new frequency in Hz
```

To change the duty cycle:

```
p.ChangeDutyCycle(dc)  # where 0.0 <= dc <= 100.0
```

To stop PWM:

```
p.stop()
```

An example to blink an LED once every two seconds:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 0.5)
p.start(1)
input('Press return to stop:')   # use raw_input for Python 2
p.stop()
GPIO.cleanup()
```

Using PWM in RPi.GPIO

An example to brighten/dim an LED:

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50)  # channel=12 frequency=50Hz
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

Checking function of GPIO channels

gpio_function(channel)

Shows the function of a GPIO channel.

For example:

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
func = GPIO.gpio_function(pin)
```

will return a value from:

GPIO.IN, GPIO.OUT, GPIO.SPI, GPIO.I2C, GPIO.HARD_PWM, GPIO.SERIAL, GPIO.UNKNOWN