<http://www.avislab.com/blog/raspberry-pi-pwm/>

https://www.youtube.com/watch?v=LwEBB6v559I

문제발생:

서보모터 PWM제어를 하는데 일반적인 방법으로 서보모터가 바들바들 떠는 현상 발생

이에따른 방법으로 다른모듈 사용

**apt-get install python-setuptools**

**easy\_install -U RPIO**

|  |
| --- |
| import time  from RPIO import PWM  servo = PWM.Servo()  # Set servo on GPIO17 to 900.s (0.9ms)  servo.set\_servo(4, 900)  # Set servo on GPIO17 to 2000.s (2.0ms)  #servo.set\_servo(17, 2000)  try:  while True:  servo.set\_servo(4, 750)  print "Left"  time.sleep(5)  servo.set\_servo(4, 1500)  print "Center"  time.sleep(5)  print "Right"  servo.set\_servo(4, 2500)  time.sleep(5)  except KeyboardInterrupt:  # Clear servo on GPIO17  servo.stop\_servo(4) |

* [Home](http://www.avislab.com/blog/)
* [About](http://www.avislab.com/blog/about/)
* [Video](http://www.youtube.com/channel/UCRu6XayQfW-augguSYhStyg)

[**Avislab**](http://www.avislab.com/blog/)

**сайт для палких паяльників**

양식의 맨 위



양식의 맨 아래

[Raspberry Pi – ШИМ и Сервопривод](http://www.avislab.com/blog/raspberry-pi-pwm_ru/) »

« [Raspberry Pi – FT232](http://www.avislab.com/blog/raspberry-pi-ft232_ru/)

# Raspberry Pi – PWM і Сервопривод

Опубліковано 10.10.2014

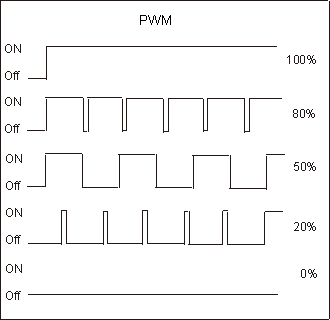
* [Facebook](http://www.facebook.com/share.php?u=http%3A%2F%2Fwww.avislab.com%2Fblog%2Fraspberry-pi-pwm%2F&t=Raspberry+Pi+%26%238211%3B+PWM+%D1%96+%D0%A1%D0%B5%D1%80%D0%B2%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4)
* [Twitter](http://twitter.com/home?status=Raspberry+Pi+%26%238211%3B+PWM+%D1%96+%D0%A1%D0%B5%D1%80%D0%B2%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4%20-%20http%3A%2F%2Fwww.avislab.com%2Fblog%2Fraspberry-pi-pwm%2F%20)
* [Google Plus](https://plus.google.com/share?url=http%3A%2F%2Fwww.avislab.com%2Fblog%2Fraspberry-pi-pwm%2F)
* [Add to favorites](http://www.avislab.com/blog/raspberry-pi-pwm/)
* [Email](mailto:?to=&subject=Raspberry+Pi+%26%238211%3B+PWM+%D1%96+%D0%A1%D0%B5%D1%80%D0%B2%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4&body=%D0%AD%D1%82%D0%B0+%D1%81%D1%82%D0%B0%D1%82%D1%8C%D1%8F+%D0%BD%D0%B0+%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%BE%D0%BC+%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%0D%0ARaspberry+Pi+%D0%BC%D0%B0%D1%94+%D0%B4%D0%B5%D0%BA%D1%96%D0%BB%D1%8C%D0%BA%D0%B0+%D1%88%D0%BB%D1%8F%D1%85%D1%96%D0%B2+%D0%B4%D0%BB%D1%8F+%D1%80%D0%B5%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97+PWM+%28%D0%A8%D0%B8%D1%80%D0%BE%D1%82%D0%BD%D0%BE-%D1%96%D0%BC%D0%BF%D1%83%D0%BB%D1%8C%D1%81%D0%BD%D0%BE%D1%97+%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%97%29.+%D0%9C%D0%B8+%D1%80%D0%BE%D0%B7%D0%B3%D0%BB%D1%8F%D0%BD%D0%B5%D0%BC%D0%BE+%D1%8F%D0%BA+%D1%80%D0%B5%D0%B0%D0%BB%D1%96%D0%B7%D1%83%D0%B2%D0%B0%D1%82%D0%B8%2C+PWM%20-%20http://www.avislab.com/blog/raspberry-pi-pwm/)
* [Print](http://www.printfriendly.com/print?url=http%3A%2F%2Fwww.avislab.com%2Fblog%2Fraspberry-pi-pwm%2F)
* [RSS](http://www.avislab.com/blog/feed/rss/)

[Эта статья на русском языке](http://www.avislab.com/blog/raspberry-pi-pwm_ru/)

Raspberry Pi має декілька шляхів для реалізації**PWM** (Широтно-імпульсної модуляції). Ми розглянемо як реалізувати, **PWM** програмно, та задіємо для генерації **PWM** апаратні ресурси Raspberry Pi. Спочатку будемо змінювати яскравість світлодіода, а потім навчимося керувати **сервоприводом**.

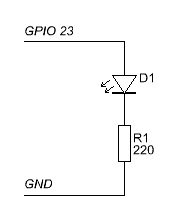
## Що таке PWM (ШІМ) ?

Широтно-імпульсна модуляція (ШІМ) – це імпульсний сигнал постійної частоти і змінної шпаруватості, тобто відношення тривалості імпульсу до періоду його проходження. За допомогою завдання шпаруватості (тривалості імпульсів) можна міняти середнє значення напруги на виході ШІМ.

[](http://www.avislab.com/blog/wp-content/uploads/2014/10/pwm.png)

## Програмна реалізація PWM

Підключимо світлодіода до **GPIO23** як вказано на схемі:

[](http://www.avislab.com/blog/wp-content/uploads/2014/10/pwm1.gif)

Напишемо скрипт **pwm\_soft.py**:

|  |  |
| --- | --- |
| 1 | nano ./pwm\_soft.py |

Текст скрипта:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | import time  import RPi.GPIO as GPIO  GPIO.setmode(GPIO.BCM)  GPIO.setup(23, GPIO.OUT)  p = GPIO.PWM(23, 50)  # channel=23 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() |

Запустимо його:

|  |  |
| --- | --- |
| 1 | python ./pwm\_soft.py |

Світлодіод буде плавно загоратися і плавно гаснути.

Програмна реалізація PWM дозволяє сформувати PWM сигнал на будь-кому виводі. У цьому прикладі ми використовуємо **RPi.GPIO** для програмної генерації PWM сигналу. А це означає, що витрачаються обчислювальні ресурси мікрокомп’ютера. Якщо мікрокомп’ютер буде відволікатися на інші задачі, PWM сигнал буде викривлятися і не буде стабільним. Це не принципово, якщо PWM застосовується для керування яскравістю світлодіода. Але може стати неприйнятним, коли PWM застосовується для формування керуючого сигналу. Наприклад, при керуванні сервоприводами програмна реалізація PWM не може стабільно утримувати сервоприводи у заданому положенні.

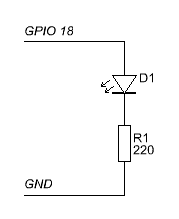
Raspberry Pi має технічну можливість використовувати апаратний ресурс для генерації PWM.

## Генерація PWM сигналу завдяки апаратним ресурсам Raspberry Pi

Проект **WiringPi** – це бібліотека, яка містить утиліти для простого доступу до GPIO. Вона дозволяє налаштувати апаратні модулі для спеціальних виходів PWM. Встановлюємо wiringPi:

|  |  |
| --- | --- |
| 1  2  3  4  5 | sudo apt-get install git-core  git clone git://git.drogon.net/wiringPi  cd wiringPi  ./build  cd .. |

Підключимо світлодіод до **GPIO18** як вказано на схемі:

[](http://www.avislab.com/blog/wp-content/uploads/2014/10/pwm2.gif)

Перший вихід PWM заведений на **GPIO18**, інші канали PWM задіяні на аудіо-виході. Виконаємо наступні команди для формування на **GPIO18** PWM сигналу. Налаштовуємо перший канал PWM (GPIO18):

|  |  |
| --- | --- |
| 1 | gpio mode 1 pwm |

Задаємо шпаруватість від 0 до 1024:

|  |  |
| --- | --- |
| 1 | gpio pwm 1 500 |

Світлодіод має світитися напівсили. Проекспериментуйте з PWM. Спробуйте задати наступні значення:

|  |  |
| --- | --- |
| 1  2 | gpio pwm 1 10  gpio pwm 1 1023 |

Відключаємо PWM:

|  |  |
| --- | --- |
| 1 | gpio unexport 1 |

або

|  |  |
| --- | --- |
| 1 | gpio unexportall |

## Генерація апаратного PWM сигналу на Python

Щоб використовувати **PWM** у **Python** треба встановили **WiringPi-Python**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | sudo apt-get install python-dev python-setuptools  git clone https://github.com/WiringPi/WiringPi-Python  cd WiringPi-Python  git submodule update --init  python setup.py install  cd .. |

Створимо срипт **pwm.py**:

|  |  |
| --- | --- |
| 1 | nano pwm.py |

Текст скрипта:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | import time  import wiringpi  # GPIO pin 12 = BCM pin 18 = wiringpi pin 1  led\_pin  = 1  wiringpi.wiringPiSetup()  wiringpi.pinMode(led\_pin, 2)  wiringpi.pwmWrite(led\_pin, 0)  def led(led\_value):      wiringpi.pwmWrite(led\_pin, led\_value)  led(0)  while 1:          for dc in range(0, 1023, 5):                  led(dc)                  time.sleep(0.01)          for dc in range(1024, 0, -5):                  led(dc)                  time.sleep(0.01) |

Запустимо його:

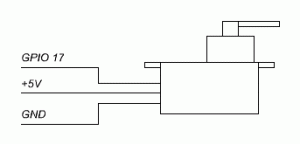
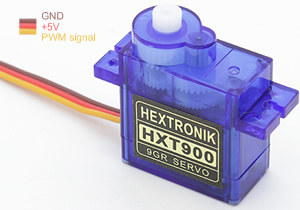
|  |  |
| --- | --- |
| 1 | python ./pwm.py |

Світлодіод буде плавно загоратися і плавно гаснути. Апаратна реалізація **PWM** забезпечує більш стабільний результат. Нажаль апаратний вихід у **Raspberry Pi** тільки один. Але існує ще пара методів генерування **PWM**. Через **DMA**, та використання зовнішнього **PWM контролера**. Ці методи розглянемо нижче для керування сервоприводами, оскільки на світлодіодах різниця не буде помітна.

## Керування Сервоприводом

Про [**сервоприводи**](http://www.avislab.com/blog/serva/)та характеристики сигналу управління я писав раніше у статті [Управление сервоприводом (сервомашинкой) с помощью микроконтроллера ATMega](http://www.avislab.com/blog/serva/).

Зазвичай сервоприводи використовують живлення 5В. Малопотужний сервопривід можна живити від**Raspberry Pi**. Але якщо привід споживає досить великий струм, або Вам потрібно підключити декілька сервомашинок, краще не навантажувати **Raspberry Pi** і використовувати окреме джерело живлення. Схема під’єднання сервопривода:

[](http://www.avislab.com/blog/wp-content/uploads/2014/10/pwm3.gif) [](http://www.avislab.com/blog/wp-content/uploads/2014/10/textronikHXT900.jpg)

## Керування сервоприводом за допомогою програмно сформованого PWM

Спочатку спробуємо формувати **PWM** для керування сервоприводом програмно. Cтворимо скрипт **servo.py**:

|  |  |
| --- | --- |
| 1 | nano servo.py |

Текст скрипта:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | import RPi.GPIO as GPIO  import time  GPIO.setmode(GPIO.BCM)  GPIO.setup(17,GPIO.OUT)  p=GPIO.PWM(17,50)  p.start(7.5)  try:          while True:                  p.ChangeDutyCycle(7.5)                  print "Left"                  time.sleep(1)                  p.ChangeDutyCycle(12.5)                  print "Center"                  time.sleep(1)                  p.ChangeDutyCycle(2.5)                  print "Right"                  time.sleep(1)  except KeyboardInterrupt:          p.stop()          GPIO.cleanup() |

Запустимо **servo.py**:

|  |  |
| --- | --- |
| 1 | python ./servo.py |

Я навмисно вставив функції **print** у код. Наявність цих функцій виявляє описану раніше проблему нестабільності програмно сформованого **PWM**. Сервомашинка не фіксується у заданому положенні і смикається. Якщо видалити інструкції **print**, проблема зменшується або взагалі зникає.

## Керування сервоприводом за допомогою PWM, сформованого через DMA

Встановлюємо **RPIO**:

|  |  |
| --- | --- |
| **1**  **2** | **apt-get install python-setuptools**  **easy\_install -U RPIO** |

Створимо срипт **servo\_dma.py**:

|  |  |
| --- | --- |
| 1 | nano ./servo\_dma.py |

Текст скрипта:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | import time  from RPIO import PWM  servo = PWM.Servo()  # Set servo on GPIO17 to 900.s (0.9ms)  servo.set\_servo(17, 900)  # Set servo on GPIO17 to 2000.s (2.0ms)  #servo.set\_servo(17, 2000)  try:          while True:                  servo.set\_servo(17, 750)                  print "Left"                  time.sleep(1)                  servo.set\_servo(17, 1500)                  print "Center"                  time.sleep(1)                  print "Right"                  servo.set\_servo(17, 2500)                  time.sleep(1)  except KeyboardInterrupt:          # Clear servo on GPIO17          servo.stop\_servo(17) |

Запускаємо його:

|  |  |
| --- | --- |
| 1 | python ./servo\_dma.py |

Тепер сервопривід працює стабільно. Тобто, для керування сервоприводами про програмний **PWM** бажано взагалі забути.

Подробиці використання RPIO читайте тут: <http://pythonhosted.org/RPIO/pwm_py.html#examples>

На цьому відео видно різницю між різними способами генерації PWM сигналу:

## Servoblaster

Існує проект [**Servoblaster**](https://github.com/richardghirst/PiBits/tree/master/ServoBlaster), який теж працює через **DMA**. За допомогою нього можна керувати до 8 сервоприводами.**Servoblaster** встановлюється як демон і дозволяє керувати сервоприводами через файли пристроїв. Тобто, керувати сервами можна через файлову систему. Це дозволяє керувати сервами використовуючи будь-яку мову програмування або з командного рядка і не потребує встановлення додаткових модулів таких як **RPIO**, котрий ми встановлювали раніше для **Python**.

Встановлюємо **Servoblaster**:

|  |  |
| --- | --- |
| 1  2  3  4 | git clone http://github.com/richardghirst/PiBits.git  cd PiBits/ServoBlaster/user  make  sudo make install |

Перевіряємо чи Servoblaster встановлено:

|  |  |
| --- | --- |
| 1 | ls /dev | grep servoblaster |

Маємо побачити:

|  |  |
| --- | --- |
| 1  2 | servoblaster  servoblaster-cfg |

Можна переглянути конфіг Servoblaster-а:

|  |  |
| --- | --- |
| 1 | cat /dev/servoblaster-cfg |

Виводи на які можна підключати сервоприводи наведені у наступній таблиці:

|  |  |  |
| --- | --- | --- |
| **Servo number** | **GPIO number** | **Pin in P1 header** |
| 0 | 4 | P1-7 |
| 1 | 17 | P1-11 |
| 2 | 18 | P1-12 |
| 3 | 21/27 \* | P1-13 |
| 4 | 22 | P1-15 |
| 5 | 23 | P1-16 |
| 6 | 24 | P1-18 |
| 7 | 25 | P1-22 |

\*– **RaspberryPi B** і **RaspberryPi B Revision 2** мають різницю у розпіновках. Детальніше тут:<http://www.avislab.com/blog/raspberry-pi-install/>

Допустимі значення положення залежать від вашого сервоприводу. У більшості вони лежать у діапазоні від 80 до 249. Напишемо наступний срипт для керування сервою:

|  |  |
| --- | --- |
| 1 | nano ./servo\_blaster.py |

Текст скрипта:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import time  # Servo Channel 1 =&gt; GPIO 17  servoChannel = 1  def setServo(servoChannel, position):      servoStr ="%u=%u\n" % (servoChannel, position)      with open("/dev/servoblaster", "wb") as f:          f.write(servoStr)  if \_\_name\_\_ == '\_\_main\_\_':      val = 50      direction = 1      while True:          #print val          setServo(servoChannel, val)          time.sleep(.01)          if val == 249:              direction -= 1          elif val == 50:              direction = 1          val += direction |

Запустимо його:

|  |  |
| --- | --- |
| 1 | python ./servo\_blaster.py |

Є одна особливість у **Servoblaster**. Поки він запущений, він займає 8 вказаних в таблиці виходів і під інші цілі ви їх вже не зможете задіяти. Спробуйте запустити раніше написані скрипти:

|  |  |
| --- | --- |
| 1  2 | python ./servo.py  python ./servo\_dma.py |

Сервомашинка не працює як слід.

Спробуємо зупинити демон **Servoblaster** і повторити спробу. Зупиняємо **Servoblaster** за допомогою команди:

|  |  |
| --- | --- |
| 1 | sudo killall servod |

Перевіряємо чи нема тепер **servod** у запущених процесах:

|  |  |
| --- | --- |
| 1 | ps ax | grep servod |

Повторюємо запуск скриптів:

|  |  |
| --- | --- |
| 1  2 | python ./servo.py  python ./servo\_dma.py |

Все працює як слід. Запускаємо **servod** командою:

|  |  |
| --- | --- |
| 1 | /usr/local/sbin/servod --idle-timeout=2000 |

Виникає питання: а що робити, якщо треба задіяти лише декілька каналів, а не всі 8? **Servoblaster** можна конфігурувати завдяки наступним опціям:

–pcm tells servod to use PCM rather than PWM hardware to implement delays  
–idle-timeout=Nms tells servod to stop sending servo pulses for a given output N milliseconds after the last update  
–cycle-time=Nus Control pulse cycle time in microseconds, default 20000us  
–step-size=Nus Pulse width increment step size in microseconds, default 10us  
–min={N|Nus|N%} specifies the minimum allowed pulse width, default 50 steps or 500us  
–max={N|Nus|N%} specifies the maximum allowed pulse width, default 250 steps or 2500us  
–invert Inverts outputs  
–dma-chan=N tells servod which dma channel to use, default 14  
–p1pins=<list> tells servod which pins on the P1 header to use  
–p5pins=<list> tells servod which pins on the P5 header to use

Зараз нас цікавить опція **–p1pins**. Зупинимо **Servoblaster**:

|  |  |
| --- | --- |
| 1 | sudo killall servod |

Та запустимо з новими опціями:

|  |  |
| --- | --- |
| 1 | /usr/local/sbin/servod --idle-timeout=2000 --p1pins=11 |

Тепер **Servoblaster** займатиме одну ногу **P1-11** (**GPIO17**). Перевірити це можна переглянувши конфіг:

|  |  |
| --- | --- |
| 1 | cat /dev/servoblaster-cfg |

Якщо перезавантажити **Raspbery Pi**, **Servoblaster** знову буде працювати з початковими опціями. Для того, щоб при старті системи **Servoblaster** запускався з Вашими налаштуваннями, вкажіть їх у файлі **/etc/init.d/servoblaster**

Ви можете прочитати детальніше про **Servoblaster** тут: <https://github.com/richardghirst/PiBits/tree/master/ServoBlaster>

Деінсталяція Servoblaster:

|  |  |
| --- | --- |
| 1  2  3 | sudo killall servod  cd PiBits/ServoBlaster/user  make uninstall |

## Adafruit 16-channel servo driver

Якщо Вам потрібно керувати надзвичайною кількістю сервоприводів, доречно використовувати багатоканальний зовнішній **PWM контролер**. Наприклад **Adafruit 16-channel servo driver**, або його аналоги. Ця плата підключається до **Raspberry Pi** по інтерфейсу **I2C** і може одночасно керувати 16-ма сервоприводами. Документацію можна скачати тут: <https://learn.adafruit.com/downloads/pdf/adafruit-16-channel-servo-driver-with-raspberry-pi.pdf>

Про інтерфейс **I2C** мова піде у наступних статтях.

Успіхів!

Дивись також:

* [Raspberry Pi — що це таке?](http://www.avislab.com/blog/raspberry-pi-install/)
* [Raspberry Pi — GPIO](http://www.avislab.com/blog/raspberry-pi-gpio/)
* [Raspberry Pi — UART](http://www.avislab.com/blog/raspberry-pi-uart/)
* [Raspberry Pi — FT232](http://www.avislab.com/blog/raspberry-pi-ft232/)
* [Raspberry Pi — PWM і Сервопривод](http://www.avislab.com/blog/raspberry-pi-pwm/)
* [Raspberry Pi — DHT11](http://www.avislab.com/blog/raspberry-pi-dht11/)
* [Raspberry Pi - FM Transmitter](http://www.avislab.com/blog/pifm/)
* [Прошивка AVR мікроконтролерів за допомогою Raspberry Pi](http://www.avislab.com/blog/raspberry-pi-avrdude/)
* [Raspberry Pi — LCD display 1602](http://www.avislab.com/blog/raspberry-pi-wh1602/)
* [Raspberry Pi — WiFi](http://www.avislab.com/blog/raspberry-pi-wifi/)
* [Raspberry-Pi — I2C (TWI)](http://www.avislab.com/blog/raspberry-pi-i2c/)
* [Raspberry Pi - DS18B20](http://www.avislab.com/blog/raspberry-pi-ds18b20/)
* [Raspberry Pi Camera](http://www.avislab.com/blog/raspberry-pi-camera/)
* [nRF24L01+ ATMEGA + Raspberry Pi](http://www.avislab.com/blog/nrf24l01/)
* [BMP180 + Raspberry Pi + WH1602](http://www.avislab.com/blog/raspberry-pi-bmp180/)
* [Wi-Fi Метео станція](http://www.avislab.com/blog/wi-fi-meteo/)
* [Raspbian. Apache + PHP + Python + MySQL](http://www.avislab.com/blog/apache-php-python-mysql/)
* [Встановлюємо Raspbian на Raspberry Pi без клавіатури та монітору](http://www.avislab.com/blog/raspbian_install/)
* [IR-дистанційне керування. Використання LIRC у Python](http://www.avislab.com/blog/raspberry-ir/)
* [Raspberry Pi. Raspbian. Відправлення пошти через аккаунт Gmail](http://www.avislab.com/blog/raspberry-pi-gmail/)

Рубрика: [Raspberry Pi](http://www.avislab.com/blog/category/raspberry-pi/)

#### 5 комментариев: Raspberry Pi – PWM і Сервопривод

* barteroff говорить:

[13.10.2014 в 21:02](http://www.avislab.com/blog/raspberry-pi-pwm/#comment-106827)

Було б непогано, якщо у статті було відео роботи.

* + andre говорить:

[13.10.2014 в 21:11](http://www.avislab.com/blog/raspberry-pi-pwm/#comment-106828)

Дякую за зауваження. Відео обов’язково додам.

* + andre говорить:

[15.10.2014 в 10:03](http://www.avislab.com/blog/raspberry-pi-pwm/#comment-106829)

Добавив відео у статтю.

* + - barteroff говорить:

[15.10.2014 в 17:37](http://www.avislab.com/blog/raspberry-pi-pwm/#comment-106830)

Доречі, підкажіть, будь-ласка, чим обумовлено вибір мови Python для реалізації керування ШІМ?  
Цікавлюсь, оскільки стою перед вибором на чому писати програми під CubieBoard (аналог Raspberry, але не таке велике комюніті).

* + - * andre говорить:

[16.10.2014 в 07:46](http://www.avislab.com/blog/raspberry-pi-pwm/#comment-106831)

Python дуже потужний і популярний. На ньому можна писати скрипти з графічним інтерфейсом. До тогож у Respbian він вже встановлений. Якщо не потрібна супершвидкість Python – саме те. Для іншого краще C. Доречі, треба буде добавити приклади на С.

**Translate**



**[[https://www.gstatic.com/images/branding/googlelogo/1x/googlelogo_color_42x16dp.png](https://translate.google.com/)번역](https://translate.google.com/" \t "_blank)**에서 제공

**Категорії**

* [Brushless Motors](http://www.avislab.com/blog/category/brushlessmotors/" \o ) (12)
* [Raspberry Pi](http://www.avislab.com/blog/category/raspberry-pi/" \o ) (44)
* [Довідники](http://www.avislab.com/blog/category/guides/" \o ) (5)
* [Книги про електроніку](http://www.avislab.com/blog/category/books/" \o ) (20)
* [Корисно знати](http://www.avislab.com/blog/category/info/" \o ) (69)
* [Новини](http://www.avislab.com/blog/category/news/" \o ) (9)
* [Початківцям](http://www.avislab.com/blog/category/begin/" \o ) (25)
* [Схеми і прошивки](http://www.avislab.com/blog/category/circuit_diagrams/" \o ) (64)

**Недавні записи**

* [MPU-6050 – гіроскоп – акселерометр](http://www.avislab.com/blog/mpu-6050/)
* [Neoway M590 – GSM/GPRS modem](http://www.avislab.com/blog/neoway-m590/)
* [MAX1674, NCP1400, Max1724](http://www.avislab.com/blog/max1674-ncp1400-max1724/)
* [GSM Alarm Monitor](http://www.avislab.com/blog/gsm-alarm-monitor/)
* [Raspberry Pi. Raspbian. Відправлення пошти через аккаунт Gmail](http://www.avislab.com/blog/raspberry-pi-gmail/)

**Останні коментарі**

* Олег до [ИК-дистанционное управление. Использование LIRC в Python](http://www.avislab.com/blog/raspberry-ir_ru/" \l "comment-107983)
* Борис до [MAX1674, NCP1400, Max1724](http://www.avislab.com/blog/max1674-ncp1400-max1724/" \l "comment-107962)
* Борис до [MAX1674, NCP1400, Max1724](http://www.avislab.com/blog/max1674-ncp1400-max1724/" \l "comment-107960)
* Борис до [MAX1674, NCP1400, Max1724](http://www.avislab.com/blog/max1674-ncp1400-max1724/" \l "comment-107959)
* Борис до [MAX1674, NCP1400, Max1724](http://www.avislab.com/blog/max1674-ncp1400-max1724/" \l "comment-107958)

**Архіви**

* [Листопад 2015](http://www.avislab.com/blog/2015/11/)
* [Жовтень 2015](http://www.avislab.com/blog/2015/10/)
* [Вересень 2015](http://www.avislab.com/blog/2015/09/)
* [Серпень 2015](http://www.avislab.com/blog/2015/08/)
* [Липень 2015](http://www.avislab.com/blog/2015/07/)
* [Червень 2015](http://www.avislab.com/blog/2015/06/)
* [Травень 2015](http://www.avislab.com/blog/2015/05/)
* [Квітень 2015](http://www.avislab.com/blog/2015/04/)
* [Березень 2015](http://www.avislab.com/blog/2015/03/)
* [Лютий 2015](http://www.avislab.com/blog/2015/02/)
* [Січень 2015](http://www.avislab.com/blog/2015/01/)
* [Грудень 2014](http://www.avislab.com/blog/2014/12/)
* [Листопад 2014](http://www.avislab.com/blog/2014/11/)
* [Жовтень 2014](http://www.avislab.com/blog/2014/10/)
* [Вересень 2014](http://www.avislab.com/blog/2014/09/)
* [Серпень 2014](http://www.avislab.com/blog/2014/08/)
* [Червень 2014](http://www.avislab.com/blog/2014/06/)
* [Травень 2014](http://www.avislab.com/blog/2014/05/)
* [Квітень 2014](http://www.avislab.com/blog/2014/04/)
* [Березень 2014](http://www.avislab.com/blog/2014/03/)
* [Лютий 2014](http://www.avislab.com/blog/2014/02/)
* [Грудень 2013](http://www.avislab.com/blog/2013/12/)
* [Листопад 2013](http://www.avislab.com/blog/2013/11/)
* [Жовтень 2013](http://www.avislab.com/blog/2013/10/)
* [Вересень 2013](http://www.avislab.com/blog/2013/09/)
* [Серпень 2013](http://www.avislab.com/blog/2013/08/)
* [Липень 2013](http://www.avislab.com/blog/2013/07/)
* [Травень 2013](http://www.avislab.com/blog/2013/05/)
* [Квітень 2013](http://www.avislab.com/blog/2013/04/)
* [Березень 2013](http://www.avislab.com/blog/2013/03/)
* [Лютий 2013](http://www.avislab.com/blog/2013/02/)
* [Січень 2013](http://www.avislab.com/blog/2013/01/)
* [Грудень 2012](http://www.avislab.com/blog/2012/12/)
* [Листопад 2012](http://www.avislab.com/blog/2012/11/)
* [Жовтень 2012](http://www.avislab.com/blog/2012/10/)
* [Вересень 2012](http://www.avislab.com/blog/2012/09/)
* [Серпень 2012](http://www.avislab.com/blog/2012/08/)
* [Липень 2012](http://www.avislab.com/blog/2012/07/)
* [Червень 2012](http://www.avislab.com/blog/2012/06/)
* [Квітень 2012](http://www.avislab.com/blog/2012/04/)
* [Березень 2012](http://www.avislab.com/blog/2012/03/)
* [Січень 2012](http://www.avislab.com/blog/2012/01/)
* [Листопад 2011](http://www.avislab.com/blog/2011/11/)
* [Вересень 2011](http://www.avislab.com/blog/2011/09/)
* [Серпень 2011](http://www.avislab.com/blog/2011/08/)
* [Липень 2011](http://www.avislab.com/blog/2011/07/)
* [Червень 2011](http://www.avislab.com/blog/2011/06/)
* [Травень 2011](http://www.avislab.com/blog/2011/05/)
* [Квітень 2011](http://www.avislab.com/blog/2011/04/)
* [Березень 2011](http://www.avislab.com/blog/2011/03/)
* [Лютий 2011](http://www.avislab.com/blog/2011/02/)

**Tags**

[3D printer](http://www.avislab.com/blog/tag/3d-printer/" \o "2 теми) [ATMega](http://www.avislab.com/blog/tag/atmega/" \o "27 тем)[AVR](http://www.avislab.com/blog/tag/avr/" \o "17 тем) [battery](http://www.avislab.com/blog/tag/battery/" \o "8 тем) [BLDC](http://www.avislab.com/blog/tag/bldc/" \o "16 тем)[Bluetooth](http://www.avislab.com/blog/tag/bluetooth/" \o "1 тема ) [bmp](http://www.avislab.com/blog/tag/bmp/" \o "2 теми) [BMP085](http://www.avislab.com/blog/tag/bmp085/" \o "8 тем)[books](http://www.avislab.com/blog/tag/books-2/" \o "29 тем) [Brushless](http://www.avislab.com/blog/tag/brushless/" \o "12 тем)[DC-DC](http://www.avislab.com/blog/tag/dc-dc/" \o "4 теми) [DHT11](http://www.avislab.com/blog/tag/dht11/" \o "3 теми) [displays](http://www.avislab.com/blog/tag/displays/" \o "5 тем)[EB-500](http://www.avislab.com/blog/tag/eb-500/" \o "4 теми) [Ethernet](http://www.avislab.com/blog/tag/ethernet/" \o "1 тема ) [GPS](http://www.avislab.com/blog/tag/gps/" \o "4 теми) [HIH-4000](http://www.avislab.com/blog/tag/hih-4000/" \o "2 теми)[I2C](http://www.avislab.com/blog/tag/i2c/" \o "2 теми) [LCD](http://www.avislab.com/blog/tag/lcd/" \o "5 тем) [LED](http://www.avislab.com/blog/tag/led/" \o "4 теми) [MAX1674](http://www.avislab.com/blog/tag/max1674/" \o "1 тема )[meteo](http://www.avislab.com/blog/tag/meteo/" \o "9 тем) [motor](http://www.avislab.com/blog/tag/motor/" \o "12 тем) [MPU-6050](http://www.avislab.com/blog/tag/mpu-6050/" \o "2 теми)[MPX4115A](http://www.avislab.com/blog/tag/mpx4115a/" \o "2 теми) [news](http://www.avislab.com/blog/tag/news-2/" \o "3 теми) [Piezo](http://www.avislab.com/blog/tag/piezo/" \o "2 теми)[programmator](http://www.avislab.com/blog/tag/programmator/" \o "6 тем) [purchase](http://www.avislab.com/blog/tag/purchase/" \o "3 теми)[Python](http://www.avislab.com/blog/tag/python/" \o "2 теми) [Raspberry](http://www.avislab.com/blog/tag/raspberry/" \o "46 тем)[RFID](http://www.avislab.com/blog/tag/rfid/" \o "4 теми) [RS-232](http://www.avislab.com/blog/tag/rs-232/" \o "8 тем)[sensors](http://www.avislab.com/blog/tag/sensors/" \o "21 тема ) [servo](http://www.avislab.com/blog/tag/servo/" \o "3 теми) [SMD](http://www.avislab.com/blog/tag/smd/" \o "8 тем)[SMS](http://www.avislab.com/blog/tag/sms/" \o "2 теми) [solar](http://www.avislab.com/blog/tag/solar/" \o "9 тем) [soldering](http://www.avislab.com/blog/tag/soldering/" \o "2 теми) [UART](http://www.avislab.com/blog/tag/uart/" \o "2 теми)[USB](http://www.avislab.com/blog/tag/usb/" \o "9 тем) [vario](http://www.avislab.com/blog/tag/vario/" \o "8 тем) [WiFi](http://www.avislab.com/blog/tag/wifi/" \o "4 теми)

© 2011-2015 Андрій Корягін, Кременчук, Україна

Powered by [WordPress](http://wordpress.org/" \t "_blank) and [WordPress Theme](http://www.artisteer.com/?p=wordpress_themes" \t "_blank) created with Artisteer.