Raspberry pi + PocketSphinx: Оффлайн распознавание речи и управление голосом.

Опубликовано 10.03.2017 Запись добавил Alex

Хотите создать свою маленькую Siri, которая может работать без телефона и без интернета? А может онлайн и кроссплатформенно!

Очень кратко, что нас ждет и как это будет работать:

- Установим легковесный и быстрый движок PocketSphinx.
- Будем использовать акустическую языковую модель (hmm) и статическую языковую модель (lm). У нас будет свой словарь употребляемых слов (dict)
- Создадим строгий каркас (разрешенную последовательность слов) в произносимых фразах (jsgf).

Обо всем этом более подробно можно почитать тут: CMUSphinx.

Принцип работы таков: говорите в микрофон необходимые, заранее описанные в словаре слова, Raspberry распознает предложение, после чего происходит интерпретация в соответствующую команду. Как вариант, включается свет/розетка. Данный процесс не требует подключения к интернету и весьма быстр (на небольшом словаре — менее секунды на распознание). Дополнительно (в данной статье описано не будет, но сразу забегаем вперед) можно осуществлять распознавание из записанного аудио файла, например, передавая его малине из Telegram. Телеграм, в свою очередь, является кроссплатформенным пультом управления от всего, не требующим ВПНов, белой IP-адресации и работающий где угодно.

Для осуществления нашей затеи понадобится любой одноплатник (в нашем случае Raspberry pi3), USB-микрофон и, желательно, какой-нибудь управляемый ключ / диод или реле. В общем, что-то управляемое, чтобы в конечном итоге мы визуально оценили плоды работы. Микрофон можно использовать от веб камеры, если нет простого.

Подготовим систему для установки PocketSphinx

Здесь и далее будем работать из под root.

```
$ sudo -i
# apt-get update
# apt-get upgrade
```

Вставляем микрофон, загружаемся и проверяем, что он присутствует в системе:

```
# cat /proc/asound/cards
```

Вы должны увидеть, что появился еще один девайс помимо ALSA:

```
cat /proc/asound/cards                                                    Shell
root@raspberrypi_lobby:~# cat /proc/asound/cards
 0 [ALSA           ]: bcm2835 - bcm2835 ALSA
                      bcm2835 ALSA
 1 [Device         ]: USB-Audio - USB PnP Sound Device
                      C-Media Electronics Inc. USB PnP Sound Device at usb-3f980000.usb-1.4, fu
```

Девайс с индефикатором «1» и есть наш микрофон.

Что делать, если микрофон **не представлен** в системе?

Показать решение

Отлично, идем дальше.

Ставим необходимые пакеты:

```
$ sudo -i
# apt-get install bison
# apt-get install alsa-utils libasound2-dev
# apt-get install swig
# apt-get install python-dev
```

Сразу уберем из системы pulseaudio, чтобы не получить ошибки о недоступности микрофона в дальнейшем:

```
# apt-get remove pulseaudio -y
# aptitude purge pulseaudio -y
# sudo mv /usr/include/pulse/pulseaudio.h /usr/include/pulse/pulseaudio.h.old
```

Как проверить микрофон в вашем Raspberry?

Текущие параметры микрофона посмотреть можно следующим образом:

amixer -c X sget 'Mic',0 Где X Номер девайса из *cat /proc/asound/cards*.

В нашем случае команда выглядит так:

```
# amixer -c 1 sget 'Mic',0
```

Установка чувствительности микрофона.

Подберите нужную по необходимости. Конечно, если девайс поддерживает данную настройку:

```
# amixer -c 1 sset 'Mic' 100
# amixer -c 1 sset 'Mic' 70
...
и тд
```

Блокировка или toggle

```
# amixer -c 1 sset 'Mic' toggle
```

Разблокировка происходит точно так же, как и блокировка:

```
# amixer -c 1 sset 'Mic' toggle
```

Запись.

Теперь давайте запишем тестовый фрагмент аудио.

Номер после двоеточия в команде ниже — это номер вашего девайса. У нас 1:

```
# arecord -D plughw:1,0 -f cd ./test_record.wav
```

Для окончания записи нажмите **Ctrl+C**.

Воспроизведение.

Убедитесь что в вашем Raspberry для начала включена поддержка аудио.

В конфиге загрузки */boot/config.txt* должен присутствовать параметр *dtparam=audio=on*.

Плюс потребуются наушники или колонки. Воспроизведем тестовую запись:

```
# aplay ./test_record.wav
```

С микрофоном определились, теперь давайте ставить **PocketSphinx**

Установка Sphinxbase

для удобства копипасты используйте значок Shell

```
$ sudo -i
# cd ~/
# wget http://sourceforge.net/projects/cmusphinx/files/sphinxbase/5prealpha/sphinxbase-5prealph
# tar -zxvf ./sphinxbase-5prealpha.tar.gz
# cd ./sphinxbase-5prealpha
# ./configure --enable-fixed
# make
# sudo make install
```

Установка PocketSphinx

для удобства копипасты используйте значок Shell

```
# cd ~/
# wget http://sourceforge.net/projects/cmusphinx/files/pocketsphinx/5prealpha/pocketsphinx-5pre
# tar -zxvf pocketsphinx-5prealpha.tar.gz
# cd ./pocketsphinx-5prealpha
# ./configure
# make
# sudo make install
# export LD_LIBRARY_PATH=/usr/local/lib
# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

Теперь давайте разберемся, что делать дальше. Что мы хотим? Мы хотим управлять розеткой/лампой голосом, говоря в микрофон. При этом требуется произносить структурированную последовательность слов, которые нужно интерпретировать в команды. Правильно сформулированная задача — это наполовину решенная задача.

Для ее решения нам потребуется утилита pocketsphinx_continuous, акустическая языковая модель, статическая языковая модель и словарь. Английская акустическая языковая модель уже присутствует после установки и используется по умолчанию, русскую модель и словарь мы разберем чуть позже.

Пример распознавания речи на английском

Как создать словарь?

Создаем на компьютере текстовый файлик txt со всеми вашими словами, которые вы хотите распознавать.

Вот пример минимального словаря для управления розеткой (каждое слово с новой строки):

file.txt Shell

```
PLUG
ACTIVE
DISABLE
```

Загружаем созданный файл на сайт CMU в утилиту lmtool.

Нажимаем **compile knoweledge base**. Далее правым щелчком в появившейся странице нажимите на ваш персональный архив, копируйте ссылку на него:

Sphinx knowledge base generator [lmtool.3a]

Your Sphinx knowledge base compilation has been successfully processed!

The base name for this set is **6797**. TAR6797.tgz is the compressed version.
Note that this set of files is internally consistent and is best used together.

IMPORTANT: Please download these files as soon as possible; they will be deleted in approximately a half hour.

```
SESSION 1488625377_12302
[_INFO_] Found corpus: 6 sentences, 6 unique words
[_INFO_] Found 0 words in extras  (0)
[_INFO_] Language model completed  (0)
[_INFO_] Pronounce completed  (0)
[_STAT_] Elapsed time: 0.028 sec
```

Please include these messages in bug reports.

Name	Size	Description
6797.dic	108	*Pronunciation Dictionary*
6797.lm	1.2K	*Language Model*
6797.log_pronounce	79	*Log File*
6797.sent	84	*Corpus (processed)*
6797.vocab	30	*Word List*
TAR6797.tgz	908	**COMPRESSED TARBALL**

потом вставляете ссылку с командой скачивания:

```
# cd ~/
# wget http://www.speech.cs.cmu.edu/tools/product/1488625377_12302/TAR6797.tgz
```

Распаковываем архив. Введите имя **вашего** архива (оно уникально):

```
# tar zxvf TAR6797.tgz
```

Проверим, что файлы разархивировались и лежат в домашней директории текущего пользователя (6797 — замените в команде на номер своего архива):

```
ls -an | grep 6797
```

```
ls -an | grep 6797                                                    Shell
-rw-r--r--   1   33   33      108 мар  4  2017 6797.dic
-rw-r--r--   1   33   33     1200 мар  4  2017 6797.lm
-rw-r--r--   1   33   33       79 мар  4  2017 6797.log_pronounce
-rw-r--r--   1   33   33       84 мар  4  2017 6797.sent
-rw-r--r--   1   33   33       30 мар  4  2017 6797.vocab
-rw-r--r--   1    0    0      908 мар  4  2017 TAR6797.tgz
```

Теперь можем проверить работу распознания из микрофона в связке с загруженным словарем и вашей статической языковой моделью:

```
pocketsphinx_continuous -adcdev plughw:1,0 -dict /root/6797.dic -lm /root/6797.lm -inmic yes
```

В консоли сначала отобразится текущий конфиг, потом вы можете начинать говорить слова из вашего словаря.

Что делать, если вы получили ошибку следующего содержания:

```
Error                                                                  Shell
Error opening audio device plughw:1,0 for capture: Connection refused
FATAL: "continuous.c", line 245: Failed to open audio device
```

Есть два варианта, почему это происходит:

Первый — вы выбрали не тот девайс, что маловероятно (например, plughw:0,0).

Второй — у вас нестыковочка с пакетами и вам придется кое-что переставить.

Общие рекомендации можно прочитать тут.

Как исправить такую ошибку:

Способ первый

Способ второй

Второй способ более сложный и нужен, если все-таки хотим оставить pulse.

Теперь сделаем процесс распознавания более структурированным, а именно: опишем как должны строиться предложения (каков должен быть порядок слов в предложении). Это необходимо для того, чтобы увеличить процент удачных распознанных фраз, облегчить работу программе и не ждать ненужных слов на входе. Чтобы распознавшие происходило в строгой определенной заранее последовательности, а все, что будет произноситься не по заданному порядку, было бы откинуто.
Для малого количества слов и заранее продуманной структуры предложений можно использовать JavaScript Grammar File вместо статической языковой модели.
Прочитать о JavaScript Grammar File можно тут.
Создаем файл с грамматикой pi.gram:

```
vim pi.gram
```

pi.gram JavaScript

```
#JSGF V1.0;
grammar PI;
public <cmd> = ( PLUG ) ( ACTIVE | DISABLE );
```

Круглые скобки говорят о том, что объект и действие обязательны в произносимой фразе, а знак вертикальной черты — возможность выбора между элементами в скобках.

Файл с грамматикой предложений создан. Теперь запустим pocketsphinx_continuous с опцией, указывающей на грамматическую конструкцию в **pi.gram**

для удобства копипасты используйте значок Shell

```
pocketsphinx_continuous -adcdev plughw:1,0 -jsgf /root/pi.gram -dict /root/6797.dic -inmic yes
```

Видим, что распознавание стало происходить только в определенном порядке. Остальные варианты, не описанные в pi.gram, отбрасываются, поэтому говорить теперь стоит несколько четче.

Распознавание речи в Pocketsphinx на русском

Теперь давайте займемся «русификацией» нашего детища. Для этого нам потребуется скачать русскую фонетическую модель, создать новый словарь и прописать грамматику произносимых фраз на русском. Поехали.
Качаем фонетическую модель от sourceforge вот с этого ресурса.
Архив **zero_ru_cont_8k_v3.tar.gz** переносим к себе в домашнюю директорию пользователя pi через winscp (если у вас Windows) или через scp (если мак/линукс).
Дальше переместим архив в домашнюю директорию root (конечно, это не обязательное условие, вы можете держать все файлы где хотите, просто для наглядности мы помещаем все в одно место, чтобы не надо было далеко ходить).

```
# mv /home/pi/zero_ru_cont_8k_v3.tar.gz /root/zero_ru_cont_8k_v3.tar.gz
```

Разархивируем, скопированный архив:

```
# cd ~/
# tar -zxvf ./zero_ru_cont_8k_v3.tar.gz
```

Скачиваем небольшую программу которая будет генерировать словари для русской фонетической модели. Это аналог того, что вы делали через браузер. Для этого склонируем репозитрий из GitHub. git должен быть установлен в малине. если нет — *apt-get install git*:

```
# cd ~/
# git clone https://github.com/zamiron/ru4sphinx/ ru4sphinx
```

Создаем в Raspberry файл со всеми необходимыми русскими словами (каждое слово с новой строки):

```
# vim raw_rus_dict
```

raw_rus_dict — Python

```
Зинаида Александровна
Зиночка
включи
погаси
показывай
розетку
лампу
отчет
```

Теперь генерируем словарь с помощью скачанного скрипта, указывая в параметрах путь до файла с нашими словами и путь для записи результирующего файла:

```
# /root/ru4sphinx/text2dict/dict2transcript.pl raw_rus_dict rus_pi_dict
```

После отработки программы проверим сгенерированный словарь:

```
# cat rus_pi_dict
```

cat rus_pi_dict — Shell

```
Александровна a ll i k s ay n d r ay v n ay
Зинаида z y n a ii d ay
Зиночка z y n a ch k aa
включи f k ll uj ch ii
лампу l aa m p u
отчет a ch jo t
погаси p ay g a ss ii
показывай p a k aa z y v ay j
розетку r a zz je t k u
```

Добавим немного последовательности для произносимых фраз:

```
# vim rus_pi.gram
```

rus_pi.gram — Python

```
#JSGF V1.0;
grammar PI;
public <cmd> = <name> <obj> <action>;
<name> = [ Зинаида Александровна | Зиночка];
<obj> = ( лампу | розетку | отчет);
<action> = ( погаси | включи | показывай);
```

В квадратных скобках мы указываем необязательное условие, в круглых — обязательное.

Готово. Запускаем pocketsphinx_continuous аналогично примеру на английском. Дополнительно необходимо указать русскую акустическую модель:

для удобства копипасты используйте значок — Shell

```
# pocketsphinx_continuous -adcdev plughw:1,0 -hmm /root/zero_ru_cont_8k_v3/zero_ru.cd_semi_4000
```

Также в этой команде мы убрали подробное логирование, чтобы видеть только итоговый результат распознавания. Результат команды будет примерно такой:

результат распознавания — Shell

```
лампу включи
Зиночка лампу включи
Зиночка лампу погаси
Зинаида Александровна отчет показывай
```

Вы можете увидеть самостоятельно, что «Зиночка» можно произносить, а можно и опустить, так как данная опция стала произвольной. Но для итоговой конфигурации я рекомендую выбрать какое-то одно имя и сделать его принудительным в ожидаемой фразе (указать круглые скобки в грамматике JSGF), дабы уменьшить число возможных ложных срабатываний.

Дополнительно в файле JSGF можно использовать более структурированные конструкции следующего вида:

file.gram Python
```
public <cmd> = <a> <b> <c>..;
<a> = <d> <e>;
<d> = [x1|x2|..];
<e> = (y1|y2|..);
...
```

Такие грамматические обороты могут в конечном итоге помочь отсеять лишние не ожидаемые варианты распознания.

Подключим управляемое реле 220v к Raspberry.

Как было сказано выше, хотелось бы увидеть результат на чем-то осязаемом. Для индикации результата я выбрал обычное коммутируемое реле 220v с 3-5-тивольтовой логикой управления. Как подключить такое реле, вы можете узнать, прочитав подробную статью.
Совсем необязательно подключать на него все 220 вольт, можно использовать цепи с меньшим вольтажом. Для экспериментов вы можете подключить фонарик/диод или все, что дает какую-то индикацию на такое реле, разорвав через него цепь питания. Но статья не об этом. Ниже я напишу два питоновскох скрипта, которые будут включать и выключать такое реле. Сигнальный провод от реле подключен на 17Pin(BCM).

Включение:

```
# vim /home/pi/relay_on.py
```

relay_on.py Python
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#project: home-smart-home.ru
import RPi.GPIO as GPIO

pin_number = 17

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(pin_number, GPIO.OUT)   #устанавливаем пин на выходной сигнал
GPIO.output(pin_number, GPIO.HIGH) #ставим логическую еденицу на выходе
print "включила реле"
```

Делаем сценарий исполняемым:

```
# chmod +x /home/pi/relay_on.py
```

Выключение:

```
# vim /home/pi/relay_off.py
```

relay_off.py Python
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#project: home-smart-home.ru
import RPi.GPIO as GPIO

pin_number = 17

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(pin_number, GPIO.OUT)   #устанавливаем пин на выходной сигнал
```

```
GPIO.output(pin_number, GPIO.LOW)
print "отключила реле"
```

И не забываем про бит исполнения:

```
# chmod +x /home/pi/relay_off.py
```

Преобразование результата распознания PocketSphinx в команды

Теперь самое интересное. Как полученный результат распознавания переводить в системные команды?

На гитхабе есть различные готовые варианты программ различных конструкций. Но не все варианты могут работать так как вы хотите 🙂

Покажу простейший вариант на питоне, который можно будет наращивать самостоятельно с помощью новых блоков elif:

```
# vim recognizer.py
```

recognizer.py Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#project: home-smart-home.ru
import subprocess
import time

exe = '''pocketsphinx_continuous -adcdev plughw:1,0 -hmm /root/zero_ru_cont_8k_v3/zero_ru.cd_se
p = subprocess.Popen(["%s" % exe], shell=True, stdout=subprocess.PIPE)

while True:
        retcode = p.returncode
        line = p.stdout.readline()
        if line == "отчет показывай\n":
                print line
                subprocess.call('uname -a', shell=True)
                subprocess.call('uptime', shell=True)

        elif line == "Зиночка лампу включи\n":
                print line
                subprocess.call('/home/pi/relay_on.py', shell=True)

        elif line == "Зиночка лампу погаси\n":
                print line
                subprocess.call('/home/pi/relay_off.py', shell=True)
        # можете продолжать дальше в том же духе с новыми блоками elif

        else:
                print line
                time.sleep(0.15)
        if(retcode is not None):
                break
```

Разрешаем исполнение сценария:

```
# chmod +x /root/recognizer.py
```

Можем запускать и радоваться проделанной работе. Распознавание наглядно отрабатывает. Родилась интернет-вещь:

```
# /root/recognizer.py
```

Для удобства работы, в дальнейшем можете добавить сценарий в автозагрузку. Для этого нужно дописать в файл /etc/rc.local строчку **/root/recognizer.py &** выше строки **exit 0**

Оптимизация или работа напильником

Если вас не очень устраивает качество распознавания «из коробки», выход есть. Правда, он может быть для вас не очень простым. Но, так или иначе, процентов на десять точно можно улучшить коробочные результаты. Какие для этого есть варианты?

1. Самый простой вариант. Запустите pocketsphix_continous без параметров и вы увидите длинный список возможных входных аргументов:

```
root@raspberrypi:~# pocketsphinx_continuous
```

```
Arguments list definition:
[NAME]          [DEFLT]     [DESCR]
-adcdev                 Name of audio device to use for input.
-agc            none        Automatic gain control for c0 ('max', 'emax', 'noise', or 'none')
-agcthresh      2.0         Initial threshold for automatic gain control
```

Показать полный вывод команды

Определенно стоит поиграть с параметрами пауз, определением шума, чувствительностью и т.д.

1. Можно попытаться произвести адаптацию звуковой модели на основе своих надиктованных данных. Частично можно прочитать об этом тут, а также google вам в помощь.

Заключение

На мой взгляд, описанная связка гораздо удобнее и круче всяких выключений света по хлопку и прочих подобных решений. Это вообще другой уровень инженерной мысли. Вы можете составлять произвольные словари, в которых будет все необходимое. Мой опыт показывает, что для небольших домашних проектов хватает словаря с 20-30 элементами, перечисления всех операций над ними, а также необходимых числительных. Такой объем информации pocketsphinx «на ура» отрабатывает практически на любом одноплатнике. Как быть с USB микрофонами в реальных проектах? Из опыта: можно удлинить стандартный полутораметровый юсб шнур с помощью усилителей или несколько нарастить через преобразования usb-ethernet. Примерно 10 метров кабеля между микрофоном и малиной обычно хватает для удобного размещения и первого и второго девайсов.

Всем упорства, упорства и еще раз терпения. Подписывайтесь на блог в правой верхней части экрана. Полезной информации для дальнейших публикации еще очень много 🙂

Похожее

Проходной выключатель с обратной связью + Raspberry pi.
07.02.2017
В "Алгоритмы, сценарии, автоматизация."

Raspberry pi + реле, или как управлять 220v.
05.02.2017
В "Алгоритмы, сценарии, автоматизация."

Raspberry pi. Настройка сети.
29.01.2017
В "Базовое администрирование"

Опубликовано в разделе: Алгоритмы, сценарии, автоматизация. Теги: pocketsphinx, raspberry pi, голосовое управление, распознание речи

Telegram bot на Raspberry pi: сигнализация,
контроль и управление.

Комментарии к “Raspberry pi + PocketSphinx: Оффлайн распознавание речи и управление голосом.”: 17

ell:
27.07.2017 в 22:16

error while loading shared libraries: libpocketsphinx.so.3: cannot open shared object file: No such file or directory
вылезает ошибка, не могу разобрать в чем дело

Ответить

Александр:
28.07.2017 в 17:02

переменные окружения проверьте.

Ответить

ell:
28.07.2017 в 23:29

Да, спасибо с этим разобрался. Появилась новая проблема, если ничего не говорить, он выводит случайные команды из словаря. как будто бы я их говорю. и вообще распознавание происходит медленно.
(Orange Pi+2e, ubuntu server 16.04.2; карта transcend 32gb)

Ответить

Александр:
29.07.2017 в 00:15

вам необходимо или снизить чувствительность микрофона (если подлерживает) или попытаться задать пороги шума у покетсфинкса (смотрите документацию) или начать использовать коловое слово(можно через покетконтиноус, можно самому написать алгаритм, чтобы отрабатывать фразы только после прризнесения кодовой)

данный вывод произвольных команд связан с тем , что из микрофона у вас идет не тишина а постоянный поток данных.

как вариант можете, когда не нужно ставить микрофон на тоггл програмно. насчет медленно не уверен 1.5-2 секунды — у меня средний результат отработки команды с распознанием в комнатных помещениях.

Ответить

ell:

29.07.2017 в 00:44

спасибо за ответ, как допилю — отпишусь. Снижение чувствительности действительно работает, ложных распознаваний стало меньше.

Александр:

29.07.2017 в 05:41

использование нормальной (не ходовой в разговоре) ключевой фразы перед реальной командой уберет еще 90-95% ложных срабатываний.

Ответить

Алексей:

22.01.2018 в 20:36

Александр! Как это сделать? Вот, например, хочу чтобы ключевой фразой была «Лиза». Но как с ней, так и без нее на выходе получаем строчку «Лиза включи свет в гостевой»… у вас как-нибудь получилось использовать кодовое слово?

Ответить

ell:

02.08.2017 в 00:58

Алекс, спасибо огромно за статью! Наконец допилил включение по голосу, хотя пару мелочей пришлось делать не так как тут. но все равно, 5+ тебе!

Ответить

Евгений:

13.08.2017 в 17:11

Я задал несколько грамматических правил и все они хорошо распознаются, но проблема в том, что если я скажу левую фразу PocketSphinx выдает как-нибудь вариант из всех правил. Т.е. в обычной речи он улавливает правила. Как от этого избавиться? Можно ли активировать его по ключевой фразе -kws, а потом послушать фразу из грамматических правил (как известно -jsgf конфликтует с -kws)

Ответить

Станислав:

17.10.2017 в 21:11

Здравствуйте Александр.
Вопрос по аппаратной части. Будет ли (потянет, соберется) эта связка работать на Raspberry Pi Zero?

Ответить

Алексей:

21.01.2018 в 01:53

Огромное спасибо за статью!!! Собрал пакеты на Orange Pi Zero, микрофон аналоговый подключен к соответствующим входам — распознавание хорошее. По ключевым фразам управляю внешними контроллерами с помощью get-запросов. На питоне «пишу» первый раз, подключил urllib2 и отправляю команды так:
if line == «Лиза включи лампу гостевой\n»:
print line
response = urllib2.urlopen(‘http://192.168.12.21/sec/?pt=27&cmd=27:1’)
print «Ответ контроллера: «, response.code
elif line == «Лиза выключи лампу гостевой\n»:
print line
response = urllib2.urlopen(‘http://192.168.12.21/sec/?pt=27&cmd=27:0’)
print «Ответ контроллера: «, response.code
В принципе все работает! Однако при прерывании процесса вылазит следующее:
Traceback (most recent call last):
File «/root/recognizer.py», line 13, in

line = p.stdout.readline()
KeyboardInterrupt

С чем это может быть связано???

Ответить

Артём:
06.06.2018 в 07:34

KeyboardInterrupt означает, что вы просто напросто нажали CTRL+C. Это что-то типа обработки сигнала в Python. Попробуйте ввести ключевое слово для завершения и по нему завершать процесс.

Ответить

Артём:
08.06.2018 в 09:01

А можно не пользоваться консольной утилитой, а дёргать прямо из питона обёртки для неё.
Правда у меня она не совсем стабильно работает. и не получилось Sphinxbase для 3 питона завести

Ответить

Альберт:
11.08.2018 в 17:22

добрый день.
благодаря мануалу собрал робота, прицепил к нему один из лучших девайс-микрофонов — respeaker v.2 (вышла в 2018). там есть и шумоподавление, и определение с какой стороны звук пришел и т.п.
все работает, но полно ложных срабатываний, несмотря на то, что тестировал в полной тишине в комнате.
если робот едет, то из-за шума моторов (траки танка) ложных срабатываний еще больше. грусть (

Ответить

ell:
15.08.2018 в 01:25

настройте срабатывание по ключевой фразе

Ответить

Альберт:

15.08.2018 в 08:48

разумеется я использую ключевую фразу — https://youtu.be/xfJjybJs8GE
может быть есть еще какие-то шаманства ?

Ответить

ell:

21.09.2018 в 18:03

не знаю даже. есть невероятная версия о наводках) попробуйте экранировать микрофон, типа мелкой металлической сектой

Ответить

Добавить комментарий

Комментарий

Имя *

E-mail *

Сайт

А не бот ли вы? *

+ 5 = 9

ОТПРАВИТЬ КОММЕНТАРИЙ

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.

Подписаться на блог

Укажите свой адрес электронной почты, чтобы получать уведомления о новых записях в этом блоге.

E-mail адрес

ПОДПИСАТЬСЯ

Поиск

Поиск…

Теги

220v apple backup bot homebridge homekit HTU21 motion pocketsphinx python3.5 raspberry pi raspbian jessie lite SHT21 si7021 siri telegram wifi

влажность

голосовое управление

датчик воды движение

импульсное реле

контроль протечек

кроссплатформеное управление

опторазвязка

проходной выключатель

распознание речи реле

сетевые настройки

сигнализация температура

управление нагрузкой

установка OS

О нас

Мы не пытаемся заработать на перепродаже готовых систем. Мы создаём максимально простые и бюджетные решения или описываем, как сделать это самостоятельно.

Вербальная связь

+79268234783

Социальные сети