Добавь голосовое управление до электроники с Raspberry Pi и Arduino

По **maxwelhelp** - 09.05.2018

Добавь голосовое управление до электроники с Raspberry Pi и Arduino

Компьютеры не очень умные, потому что они делают только то, что мы заставляем их делать. Теперь вы можете сказать своему RPi сделать что-то разумное, например, управлять подключенными устройствами в своем доме, используя только голос.

Это не трудно — вы можете легко это сделать с помощью некоторого открытого исходного кода и RPi. Затем добавьте Arduino с інфрачервонним (IR) светодиодом и также сможете сказать своему роботу Roomba, что ему делать. Ранее мы уже рассматривали распознавание голоса с помощью Raspberry Pi, а сейчас попробуем расширить наши знания и навыки.

Давайте скажем Game?

Этот проект стал возможным благодаря годам исследований десятков ученых, инженеров и лингвистов во всем мире, которые работают для распознавания речи в режиме реального времени, которое смогло бы работать на скромном оборудовании — это те достижения, которые принесли нам Siri на устройствах Apple и возможности распознавания речи, встроенные в Android от Google.

В частности, мы увидим, как использовать открытый код инструментария распознавания речи от специалистов из университета Карнеги-Меллона с названием PocketSphinx, предназначенного для использования во встраиваемых системах. Эта удивительная библиотека позволяет делегировать сложное преобразование звука в текст, так что мы сможем сосредоточиться на реализации логики более высокого уровня, чтобы преобразовать текст в значимые действия типа управления системой освещения в вашем доме и даже вашим роботом-пылесосом Roomba. Это также позволяет нам, как исследователям, залезть под капот и экспериментировать с аспектами распознавания речи, которые, как правило, зарезервированы для тех, кто реализует инструментарии или изучает с научными целями.

Подготовка Raspberry Pi

Прежде чем сможете использовать RPi для распознавания речи, вы должны убедиться, что сможете записывать звук с USB-микрофона. Дистрибутив Raspbian настроен на использование для звука Advanced Linux Sound Architecture (ALSA). Несмотря на свое название, эта система вполне зрелая и будет работать со многими пакетами программного

обеспечения и звуковыми картами без изменений или проблем конфигурации. Для конкретного случая RPi есть несколько настроек, которые мы должны сделать, чтобы убедиться, что USB-карте отдано преимущество перед встроенным аудио.

Сначала подключите USB-гарнитуру и подайте питание на RPi. После того, как RPi закончит загрузку, можете проверить, обнаружено и готово к использованию USB-аудио, выполнив команду aplay -L. Должна отобразиться название вашей карты, например, в нашем примере: Logitech H570e Stereo, USB Audio. Если ваша звуковая USB-карта появилась в этом списке, то можете перейти дальше, чтобы сделает ее использование для системы по умолчанию. Чтобы сделать это, используйте свой любимый текстовый редактор для редактирования файла alsa-base.conf, например,так:

sudo nano /etc/modprobe.d/alsa-base.conf

Вы должны изменить строку options snd-usb-audio index=-2 на options snd-usb-audio index=0 и добавить строку options snd_bcm2835 index=1. После того, как закончите, сохраните файл и перезагрузите RPi с sudo reboot, чтобы использовать новую конфигурацию.

Чтобы проверить изменения, воспользуйтесь командой arecord -vv —duration=7 -fdat ~/test.wav, чтобы записать короткий 7-секундный отрезок аудио с микрофона. Попробуйте его воспроизвести с aplay ~/test.wav, и должны услышать через USB-наушники то, что записали ранее. Если нет, попробуйте воспроизвести заранее записанный звук, как aplay /usr/share/sounds/alsa/Front_Center.wav, чтобы обнаружить, проблема в микрофоне или колонках. (Интернет будет хорошей помощью при устранении этих проблем.)

Если вы услышали свою запись, то замечательно! Вы готовы перейти к настройке программного обеспечения.

Компиляция программного обеспечения необходимых компонентов

Так, как мы стоим на плечах программных гигантов, то есть несколько пакетов для установки и несколько частей программного обеспечения для компиляции, прежде чем ваш RPi будет готов принять программу голосового управления.

Во-первых, перейдите к получению пакетов, необходимых для выполнения SphinxBase, выполнив:

```
sudo apt-get install libasound2-dev libtool autoconf bison \
swig python-dev python-pyaudio
```

Вы также должны установить некоторые библиотеки Python для использования с нашим демо-приложением. Чтобы сделать это, нужно будет установить и использовать команду pip с помощью:

```
curl -O bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo pip install gevent grequests
```

Совет: Если подключение к RPi сбоит и склонно к отключениям, то можете немного сэкономить душевной боли, выполнив следующие команды в screen session (сессии экрана). Чтобы сделать это, выполните следующее, прежде чем продолжить:

```
sudo apt-get install screen
screen -DR sphinx
```

Если на любом этапе получите отключение от RPi (и он не перезапустився), то можете запустить screen -DR sphinx, чтобы снова соединиться и продолжить оттуда, где вы остановились.

Получение инструментов Sphinx

Теперь можете перейти к получению пакета SphinxBase, который используется PocketSphinx, а также другим программным обеспечением семейства CMU Sphinx.

Чтобы получить SphinxBase, выполните следующие команды:

```
git clone git://github.com/cmusphinx/sphinxbase.git
cd sphinxbase
git checkout 3b34d87
./autogen.sh
make
```

(На этом этапе можете сходить выпить кофе …)

```
sudo make install
cd ..
```

Вы готовы перейти к PocketSphinx. Чтобы получить PocketSphinx, выполните следующие команды:

```
git clone git://github.com/cmusphinx/pocketsphinx.git
cd pocketsphinx
git checkout 4e4e607
./autogen.sh
make
```

(Время для второй чашки кофе …)

```
sudo make install
cd ..
```

Чтобы обновить систему с новыми библиотеками, запустите sudo ldconfig.

Тестирование распознавания речи

Теперь у вас на месте строительные блоки для распознавания речи, но желательно проверить, что это на самом деле работает, прежде чем продолжить.
Можете запустить тест PocketSphinx, использовав pocketsphinx_continuous -inmic yes.

Вы должны увидеть нечто подобное, которое показывает, что система готова начать говорить:

```
Listening…
Input overrun, read calls are too rare (non-fatal)
```

Можете проигнорировать предупреждение. Идите вперед и говорите!

Когда закончите, то должны увидеть техническую информацию вместе с лучшим предложением от PocketSphinx о том, что сказали, а потом еще строка READY, который дает вам знать, что система готова к дальнейшему ввода:

```
INFO: ngram_search.c(874): bestpath 0.10 CPU 0.071 xRT
INFO: ngram_search.c(877): bestpath 0.11 wall 0.078 xRT
What
READY….
```

В данный момент распознавания речи запущено и работает. Вы готовы перейти к реальному удовольствие от создания пользовательского приложения голосового управления!

Управление всеми вещами

Для демо-приложении, автор запрограммировал систему, которая имеет возможность контролировать три отдельные системы: Philips Hue, осветительную систему Insteon и iRobot Roomba – робота-пылесоса. С первыми двумя вы будете общаться с помощью моста или концентратора, подключенного к сети. Для третьего, будете общаться с Arduino через последовательный порт USB, а затем Arduino будет переводить ваши команды в инфракрасные (IR) сигналы, которые имитируют пульт дистанционного управления Roomba.

Если вы просто хотите окунуться в демо-приложения и попробовать их, то можете использовать следующие команды, чтобы получить исходный код Python и запустить его на RPi:

```
git clone github.com/bynds/makevoicedemo
cd makevoicedemo
python main.py
```

На этом этапе вы должны получить на экране сообщение, что RPi готов к вводу. Попробуйте сказать одну из команд – “Turn on the kitchen light” (Включить свет на кухне) или “Turn off the bedroom light” (Выключить свет в спальне) — и посмотрите слова, которые появляются на экране. Поскольку мы еще не создали файл configuration.json, то свет на кухне должен быть еще выключен.

Использование PocketSphinx

Есть несколько режимов, которые можно настроить для PocketSphinx. Например, можно предложить прослушивать для определенного *keyword* — ключевое слово (он будет пытаться игнорировать все, что слышит, кроме ключевого слова), или ему можно запропоновати использовать *grammar* — грамматику, которую укажете (он будет пытаться отвечать на все, что слышит, ограничиваясь грамматикой). В нашем примере, мы используем режим грамматики, которая была разработана, чтобы позволить нам захватить все команды, которые будем использовать. Файл грамматики указан в формате JSGF или JSpeech Grammar Format, который имеет мощный, но простой синтаксис для указания языка, которую он ожидает услышать, в терминах простых правил.

В дополнение к файлу грамматики, вам нужно собрать еще три файла для того, чтобы использовать PocketSphinx в нашем приложении: файл словаря (*dictionary*), который будет определять слова в терминах того, как они звучат, файл модели языка (*language model*), который содержит статистические данные о слова и их порядок, и акустическую модель (*acoustic model*), которая используется для определения того, как аудио коррелируется со звуками в словах. Файл грамматики, словарь, и модель языка — все будут генерироваться специально для нашего проекта, в то время как акустическая модель будет универсальной моделью для американского английского языка (*U. S. English*).

Создание словаря

Для того, чтобы сгенерировать наш словарь, используем lmtool, инструмент на веб-основе, созданный CMU специально для быстрой генерации подобных файлов. Входом для lmtool есть файл corpus, который содержит все или большинство из предложений, которые вы хотели бы

иметь в состоянии распознать. В нашем простом случае мы имеем следующие предложения в corpus:

turn on the light kitchen
turn off the light kitchen
turn on the bedroom light
turn off the bedroom light
turn on the roomba
turn off the roomba
roomba clean
roomba go home

Вы можете ввести их в текстовом редакторе и сохранить файл как corpus.txt или можете скачать готовую версию с репозитория GitHub.

Теперь, когда у вас есть свой corpus-файл, перейдите к использованию lmtool. Чтобы скачать corpus-файл, нажмите кнопку Browse (Обзор), чтобы з'илося диалоговое окно, в котором можно выбрать corpus-файл, который только что создали.

Затем нажмите кнопку Compile Knowledge Base (Компиляция базы знаний). Вы попадете на страницу со ссылками на скачивание результата. Можете скачать сжатый файл .tgz, который содержит все созданные файлы, или просто скачать файл .dic, обозначенный как Pronunciation Dictionary. Скопируйте этот файл в ту же директорию makevoicedemo, которая была создан на RPi ранее. Вы можете переименовать файл, используя команду mv *.dic dictionary.dic, чтобы было удобнее с ним работать.

После того, как сделаете это, загрузите предварительно созданную акустическую модель с Sphinx Sourceforge. После того, как переместите ее в каталог makevoicedemo, извлеките ее из архива:

tar -xvf cmusphinx-en-us-ptm-5.2.tar.gz

Создание Grammar File

Как упоминалось ранее, все, что слышит PocketSphinx, он будет пытаться записывать в словах грамматики. Посмотрите, как описан формат JSGF в примечании W3C. Он начинается с декларации формата с последующей декларацией названия грамматики. Мы просто назовем его «commands».

Будем использовать три основных правила: *action* (действие), *object* (объект) и *command* (команда). Для каждого правила определите "*tokens*" (маркеры), которые вы ожидаете, будут сказанные пользователем.Например, двумя маркерами для нашего правила action является TURN ON (включение) и TURN OFF (выключение). Поэтому мы представляем правило как:

<action> = TURN ON |
TURN OFF ;

Точно так же правило _object_ мы определяем как:

<object> = KITCHEN LIGHT|
BEDROOM LIGHT|
ROOMBA ;

Наконец, чтобы продемонстрировать, что мы можем разместить правила или создать их с явными маркерами, определяем команду как:

```
public <command> = <action> THE <object> |
ROOMBA CLEAN |
ROOMBA GO HOME ;
```

Обратите внимание на ключевое слово public перед <command>. Это позволяет нам использовать правило <command>, а в будущем импортируя его в другие файлы грамматики.

Инициализация декодера

Мы используем Python как нашу язык программирования, потому что ее код легко читать, он мощный и, благодаря предусмотрительности разработчиков PocketSphinx, его также очень легко использовать с PocketSphinx.

Основной рабочей лошадкой при распознавании речи с PocketSphinx есть *decoder* (декодер). Для того, чтобы использовать декодер, нужно сначала установить *config* (конфигурация), которую будет использовать декодер.

```
from pocketsphinx import *
hmm = 'cmusphinx-5prealpha-en-us-ptm-2.0/'
dic = 'dictionary.dic'
grammar = 'grammar.jsgf'
config = Decoder.default_config()
config.set_string('-hmm', hmm)
config.set_string('-dict', dic)
config.set_string('-jsgf', grammar)
```

Как только это будет сделано, инициализация декодера также будет простым с помощью decoder = Decoder(config).

Для примера программы, мы используем библиотеку pyAudio, чтобы получить язык пользователя с микрофона для преобразования с PocketSphinx. Специфика этой библиотеки менее важная для наших целей (исследования распознавания речи) и поэтому мы просто примем как должное, что pyAudio работает так, как рекламируется.

Специфика получения декодером текста высказывания немного сложная, однако основной процесс может быть сведен к следующим шагам.

```
\# Запуск 'высказывания'
decoder.start_utt()
\# Обработка soundbite
decoder.process_raw(soundBite, False, False)
\# Закончить высказывание, когда пользователь перестал говорить
decoder.end_utt()
\# Получить гипотезу (для того, что было сказано)
hypothesis = decoder.hyp()
\# Получить текст гипотезы
bestGuess = hypothesis.hypstr
\# Вывести то, что было сказано
print 'I just heard you say:»{}»'.format(bestGuess)
```

Те, кто заинтересован в получении дополнительной информации об отдельных деталях этого процесса, должны обратить свое внимание на код pocketSphinxListener.py в примере проекта.

Есть много различных параметров настройки, с которыми вы можете экспериментировать, и, как упоминалось ранее, другие режимы распознавания, чтобы попробовать. Например, исследовать вариант конфигурации allphone_ci PocketSphinx и его влияние на точность декодирования. Или попробуйте ключевое слово spotting для активации света. Или

попробуйте статистическую модель языка, подобную той, которая была сгенерирована, которую мы ранее использовали с lmtool, вместо файла грамматики. Как практик, вы можете экспериментировать почти бесконечно исследуя полосу, что возможно. Одна вещь, которую вы быстро заметите, что PocketSphinx является исследовательской системой, которая активно развивается, а это иногда означает, что вы должны переписывать приложения в соответствии с новыми API и именами функций.

Теперь, когда мы рассмотрели то, что нужно для преобразования речи в текст, давайте сделаем что-то интересное с ним! В следующем разделе рассмотрим некоторые рудиментарные связи с сетевыми огнями Insteon и Philips Hue.

Пусть будет GET /Lights HTTP/1.1

В течение многих лет были спроектированы, построены и развернуты бесчисленные системы, чтобы включить и выключить скромную лампочку. Обе системы Insteon и Philips Hue имеют такие возможности, но обе имеют гораздо больше. Они общаются с системой через беспроводные протоколы, а Insteon имеет дополнительное преимущество, потому что также может связываться через линии электропитания дома.Общение напрямую с лампами в обеих этих системах было бы для некоторых эпичным хаки, однако на данный момент мы устанавливаем наши сигналы чуть ниже и получим место для общения через посредников.

Обе системы оснащены сетевым концентратором (“hub”) или мостом (“bridge”), которые выполняют необходимую работу, чтобы такие устройства в сети, как смартфоны с установленными соответствующими приложениями, могли передавать команды до светильников.

Оказывается, обе эти системы имеют API, которые базируются на HTTP и которые можем использовать в нашем примере голосового управлением системами. Обе компании имеют программы для разработчиков, к которым можно присоединиться, чтобы в полной мере воспользоваться API:

» Philips Hue Developer Program

» Insteon Developer Program

Для тех, кто хотел пополнить свои знания немного больше «партизанского стиля», есть много ресурсов в Интернете, которые объясняют основы общения с концентраторами обоих этих производителей. Много мастеров опубликовали веб статьи на эту тему, раскрывая их секреты через вековые навыки тщательного обследования, анализа сети и обратного инжиниринга. Эти навыки имеют неоценимое значение ибо и мастера, и эти системы предлагают большой опыт для тех, кто готов работать с ними.

Пример проекта имеет минимальный набор команд Python, которые могут быть использованы для общения как со старым Insteon 2242-222 Hub, так и текущим мостом Philips Hue, чтобы вы могли стартовать.

Я приказываю тебе, робот

Roomba, подготовься, чтобы выполнить мои желания!

Чтобы расширить наш зверинец красочных устройств, мы также захватили Arduino Leonardo, к которому подключен ик-светодиод и запрограммировали для отправки команд в нашем iRobot Roomba – роботу-пылесосу.

Arduino подключается к RPi посредством USB-кабеля, который также подает питание и позволяет последовательную связь. Мы используем библиотеку IRremote, чтобы сделать тяжелую работу мигание IR LED с соответствующим точным временем.

Самодельная плата подключения инфракрасного светодиода к Arduino. Конечно, она может быть меньше.

Принципиальная схема для простого круга IR LED.

Для того, чтобы в библиотеке что-то иметь для общения, мы должны подключить некоторую схему передачи ИК к соответствующим контактам нашего Arduino. Аппаратно все можно свести к транзистора, ИК-подсветкой и резистора или двух, которые могут быть размещены на плате и подключены к Arduino. В данном примере тестировали набор SparkFun Max Power IR LED Kit и установку минималистского ИК-передатчика с резистором 330 Ом, транзистором 2N3904 NPN и инфракрасным светодиодом, подключенным к Arduino через три штыри выводов.

Используем библиотеку IRremote, чтобы позволить создавать и отправлять ИК сигналы к Roomba, которые выполняют эмуляцию пульта дистанционного управления Roomba. Эти сигналы являются серией кодированных импульсов, длительность которых соответствует двоичным сигналам. Когда они закодированы для передачи с помощью библиотеки, то они выглядят подобно следующего скетча нашего примера:

```
// Очистить
const unsigned int clean[15] =
{3000,1000,1000,3000,1000,3000,1000,3000,3000,1000,1000,3000,1000,3000,1000};
// кнопка питания
const unsigned int power[15] =
{3000,1000,1000,3000,1000,3000,1000,3000,3000,1000,1000,3000,3000,1000,1000};
// Док
const unsigned int dock[15] =
{3000,1000,1000,3000,1000,3000,1000,3000,3000,1000,3000,1000,3000,1000,3000};
```

Для того, чтобы Roomba получил и понял эти сигналы, автор нашел, что лучше отправить их четыре раза, что сделаем в следующей функции sendRoombaCommand:

```
void sendRoombaCommand(unsigned int* command){
for (int i = 0; i < 4; i++){
irsend.sendRaw(command, 15, 38);
delay(50);
}
}
```

Добавив библиотеку IRremote кода Arduino, вы сможете отправлять инфракрасные (ИК) сигналы управления для Roomba.

После того, как ИК-оборудование было подключено к соответствующим контактам и скомпилированный и загруженный скетч через Arduino IDE, вы сможете отправлять команды через последовательный порт, который будет переведавати то, что Roomba может понять. Бинарность действительно становится универсальным языком!

Теперь все вместе

Так что у вас есть полная система управления огнями Insteon, светом Philips Hue и iRobot Roomba лишь звуками своего голоса!

Raspberry Pi (в черном корпусе) + Arduino + инфракрасный светодиод = полный устройство для голосового управления вашим освещением и работами.

Все это было бы невозможно без щедрого вклада многих проектов с открытым исходным кодом, которые мы использовали. Данный пример проекта также является открытым источником, поэтому вы можете обратиться к нему при осуществлении собственных проектов голосового управления на RPi и повторить его части в своих следующих проектах.

maxwelhelp