Raspberry Pi - Support #407

Building Jasper From Scratch on a Raspberry Pi

06/18/2014 12:36 PM - Daniel Curtis

Status:	Suspended	**Start date:**	06/18/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	0%
Category:		**Estimated time:**	8.00 hours
Target version:		**Spent time:**	2.00 hours

Description

Follow these instructions only if you wish to compile your Jasper software from scratch.

Burn Raspbian image onto SD card

We'll first clear the SD card using GParted on Ubuntu, but you can use an equivalent utility or operating system. In GParted: right-click on each partition of the SD card, then select Unmount and Delete. Apply the changes with Edit -> Apply All Operations.

Download Raspbian Wheezy from http://downloads.raspberrypi.org/raspbian_latest. While we've tested Jasper on the 2014-01-07 release, newer releases may also work.

- We'll use dd to burn the image to the disk. Obtain the address of the SD card with:

```
sudo fdisk -l
```

- Our device address was '/dev/mmcblk0', so the following command burns the image to the disk:

```
sudo dd if=2013-12-20-wheezy-raspbian.img of=/dev/mmcblk0 bs=2M
```

When it's done, remove your SD card, insert it into your Raspberry Pi and connect it to your computer via ethernet.

Configure Raspbian

We're now going to do some basic housekeeping and install some of the required libraries. You should SSH into your Pi with a command similar to the following. The IP address usually falls in the 192.168.2.3-192.168.2.10 range.

- SSH into the Pi

```
ssh pi@192.168.2.3
```

password (default): raspberry

- Run the following, select to 'Expand Filesystem' and restart your Pi:

```
sudo raspi-config
```

- Run the following commands to update Pi and some install some useful tools.

```
sudo apt-get update
sudo apt-get upgrade --yes
sudo apt-get install vim git-core espeak python-dev python-pip bison libasound2-dev libportaud
io-dev python-pyaudio --yes
```

- Plug in your USB microphone. Let's open up an ALSA configuration file in vim:

  ```
  sudo vim /etc/modprobe.d/alsa-base.conf
  ```

  Change the following line:

  options snd-usb-audio index=-2

To this:

  options snd-usb-audio index=0

- Back in the shell, run:

  ```
  sudo alsa force-reload
  ```

- Next, test that recording works (you may need to restart your Pi) by recording some audio with the following command:

  ```
  arecord temp.wav
  ```

- Make sure you have speakers or headphones connected to the audio jack of your Pi. You can play back the recorded file:

  ```
  aplay -D hw:1,0 temp.wav
  ```

  NOTE: If nothing is played, try turning up the microphone volume using alsamixer.

- Add the following line to the end of ~/.bash_profile (you may need to run touch ~/.bash_profile if the file doesn't exist already):

  ```
  vi ~/.bash_profile
  ```

  export LD_LIBRARY_PATH="/usr/local/lib"
  source .bashrc

- And this to your ~/.bashrc or ~/.bash_profile:

  ```
  vi ~/.bash_profile
  ```

  LD_LIBRARY_PATH="/usr/local/lib"
  export LD_LIBRARY_PATH
  PATH=$PATH:/usr/local/lib/
  export PATH

With that, we're ready to install the core software that powers Jasper.

Install Pocketsphinx

- Jasper uses Pocketsphinx for voice recognition. Let's download and unzip the sphinxbase and pocketsphinx packages:

  ```
  wget http://downloads.sourceforge.net/project/cmusphinx/sphinxbase/0.8/sphinxbase-0.8.tar.gz
  wget http://downloads.sourceforge.net/project/cmusphinx/pocketsphinx/0.8/pocketsphinx-0.8.tar.
  ```

```
gz
tar -zxvf sphinxbase-0.8.tar.gz
tar -zxvf pocketsphinx-0.8.tar.gz
```

- Now we build and install sphinxbase:

```
cd ~/sphinxbase-0.8/
./configure --enable-fixed
make
sudo make install
```

- And pocketsphinx:

```
cd ~/pocketsphinx-0.8/
./configure
make
sudo make install
```

Once the installations are complete, restart your Pi:

- [redacted]

Install CMUCLMTK, OpenFST, MIT Language Modeling Toolkit, m2m-aligner, Phonetisaurus

Note that some of these installation steps take more time to complete than previous steps.

- Begin by installing some dependencies:

```
sudo apt-get install subversion autoconf libtool automake gfortran g++ --yes
```

- Next, move into your home (or Jasper) directory to check out and install CMUCLMTK:

```
svn co https://svn.code.sf.net/p/cmusphinx/code/trunk/cmuclmtk/
cd cmuclmtk/
sudo ./autogen.sh && sudo make && sudo make install
cd ..
```

- Then, when you've left the CMUCLTK directory, download the following libraries:

```
wget http://distfiles.macports.org/openfst/openfst-1.3.3.tar.gz
wget https://mitlm.googlecode.com/files/mitlm-0.4.1.tar.gz
wget https://m2m-aligner.googlecode.com/files/m2m-aligner-1.2.tar.gz
wget https://phonetisaurus.googlecode.com/files/phonetisaurus-0.7.8.tgz
wget http://phonetisaurus.googlecode.com/files/g014b2b.tgz
```

- Untar the downloads:

```
tar -xvf m2m-aligner-1.2.tar.gz
tar -xvf openfst-1.3.3.tar.gz
tar -xvf phonetisaurus-0.7.8.tgz
tar -xvf mitlm-0.4.1.tar.gz
tar -xvf g014b2b.tgz
```

- Build OpenFST:

```
cd openfst-1.3.3/
sudo ./configure --enable-compact-fsts --enable-const-fsts --enable-far --enable-lookahead-fst
s --enable-pdt
sudo make install # come back after a really long time
```

- Build M2M:

```
cd m2m-aligner-1.2/
sudo make
```

- Build MITLMT:

```
cd mitlm-0.4.1/
sudo ./configure
sudo make install
```

- Build Phonetisaurus:

```
cd phonetisaurus-0.7.8/
cd src
sudo make
```

- Move some of the compiled files:

```
sudo cp ~/m2m-aligner-1.2/m2m-aligner /usr/local/bin/m2m-aligner
sudo cp ~/phonetisaurus-0.7.8/phonetisaurus-g2p /usr/local/bin/phonetisaurus-g2p
```

- Build Phonetisaurus model:

```
cd g014b2b/
./compile-fst.sh
```

- Finally, rename the following folder for convenience:

```
mv ~/g014b2b ~/phonetisaurus
```

At this point, we've installed Jasper and all the necessary software to run it. Before we start playing around, though, we need to configure Jasper and provide it with some basic information.

Configuring the Jasper Client

Install Jasper Client

- In the home directory of your Pi, clone the Jasper source code:

```
git clone https://github.com/jasperproject/jasper-client.git jasper
```

- Jasper requires various Python libraries that we can install in one line with:

```
sudo pip install --upgrade setuptools
sudo pip install -r jasper/client/requirements.txt
```

- Run crontab -e, then add the following line, if it's not there already:

  @reboot /home/pi/jasper/boot/boot.sh;

- Set permissions inside the home directory:

```
sudo chmod 777 -R *
```

Restart your Raspberry Pi. Doing so will run boot.py, which generates the languagemodel.lm file in the client/ folder.

Generating a user profile

In order for Jasper to accurately report local weather conditions, send you text messages, and more, you first need to generate a user profile.

- To facilitate the process, run the profile population module that comes packaged with Jasper

```
cd ~/jasper/client
python populate.py
```

The process is fairly self-explanatory: fill in the requested information, or hit 'Enter' to defer at any step. The resulting profile will be stored as a YML file at profile.yml.

Important: populate.py will request your Gmail password. Of course, this is purely optional and will never leave the device. This password allows Jasper to report on incoming emails and send you text or email notifications, but be aware that the password will be stored as plaintext in profile.yml. The Gmail address can be entered alone (without a password) and will be used to send you notifications if you configure a Mailgun account, as described below.

Mailgun as a Gmail alternative

If you'd prefer not to enter your Gmail password, you can setup a free Mailgun account that Jasper will use to send you notifications. It's incredibly painless and Jasper is already setup for instant Mailgun integration. Note that if you don't enter your Gmail address, however, Jasper will only be able to send you notifications by text message (as he won't know your email address).

In slightly more detail:

1. Register for a free Mailgun account.
2. Navigate to the "Domains" tab and click on the sandbox server that should be provided initially.
3. Click "Default Password" and choose a password.
4. Take note of the "Default SMTP Login" email address.
5. Edit your profile.yml to read:

   ...
   mailgun:
   username: postmaster@sandbox95948.mailgun.org
   password: your_password

1. Enjoy your notifications.

Facebook tokens

To enable Facebook integration, Jasper requires an API key. Unfortunately, this is a multi-step process that requires manual editing of profile.yml. Fortunately, it's not particularly difficult:

1. Go to https://developers.facebook.com and select ‘Apps’, then ‘Create a new app’.
2. Give your app a name, category, etc. The choices here are arbitrary.
3. Go to the Facebook Graph API Explorer and select your App from the drop down list in the top right (the default choice is ‘Graph API Explorer’).
4. Hit ‘Get Access Token’ with the default permissions.
5. Take the resulting API key and add it to profile.yml in the following format:

   ...
   prefers_email: false
   timezone: US/Eastern
   keys:
   FB_TOKEN: abcdefghijklmnopqrstuvwxyz

Note that similar keys could be added when developing other modules. For example, a Twitter key might be required to create a Twitter module. The goal of the profile is to be straightforward and extensible.

Spotify integration

Jasper has the ability to play playlists from your Spotify library. This feature is optional and requires a Spotify Premium account. To configure Spotify on Jasper, just perform the following steps.

- Install Mopidy with:

```
wget -q -O - http://apt.mopidy.com/mopidy.gpg | sudo apt-key add -
sudo wget -q -O /etc/apt/sources.list.d/mopidy.list http://apt.mopidy.com/mopidy.list
sudo apt-get update
sudo apt-get install mopidy mopidy-spotify --yes
```

- We need to enable IPv6:

```
sudo modprobe ipv6
echo ipv6 | sudo tee -a /etc/modules
```

Now run sudo vim /root/.asoundrc, and insert the following contents:

```
pcm.!default {
        type hw
        card 1
}
ctl.!default {
        type hw
        card 1
}
```

- We need to create the following new file and delete the default startup script:

```
sudo mkdir /root/.config
sudo mkdir /root/.config/mopidy
sudo rm /etc/init.d/mopidy
```

- Now let’s run sudo vim /root/.config/mopidy/mopidy.conf and insert the following

  [spotify]
  username = YOUR_SPOTIFY_USERNAME
  password = YOUR_SPOTIFY_PASSWORD

  [mpd]
  hostname = ::

```
[local]
media_dir = ~/music

[scrobbler]
enabled = false

[audio]
output = alsasink
```

Finally, let's configure crontab to run Mopidy by running sudo crontab -e and inserting the following entry:

```
@reboot mopidy;
```

Upon restarting your Jasper, you should be able to issue a "Spotify" command that will enter Spotify mode. For more information on how to use Spotify with your voice, check out the Usage guide.

Software Architecture

Having installed the required libraries, it is worth taking a moment to understand how they interact and how the client code is architected.

Jasper utilizes a number of open source libraries to function. Pocketsphinx performs speech recognition via Python bindings to the CMUSphinx engine. Jasper's voice is owed to the popular TTS program, eSpeak. Phonetisaurus and CMUCLMTK enable Jasper to generate dictionaries and language models on-the-fly based on the custom module vocabularies. Mopidy enables streaming from Spotify, for those users who wish to use the module.

The client architecture is organized into a number of different components:

Jasper Client Architecture

main.py is the program that orchestrates all of Jasper. It creates mic, profile, and conversation instances. Next, the conversation instance is fed the mic and profile as inputs, from which it creates a notifier and a brain.

The brain then receives the mic and profile originally descended from main and loads all the interactive components into memory. The brain is essentially the interface between developer-written modules and the core framework. Each module must implement isValid() and handle() functions, as well as define a WORDS = [...] list.

To learn more about how Jasper interactive modules work and how to write your own, check out the API guide

Resources

- http://jasperproject.github.io/documentation/software/#manual-installation

History

#1 - 07/12/2014 05:19 PM - Daniel Curtis

- Description updated

#2 - 06/04/2017 08:11 PM - Daniel Curtis

- Status changed from New to Suspended