Raspberry PI Zero W - Speech Synthesizer using MQTT protocol

Jun 12, 2017

The goal of this project is to make a low cost speech synthesizer using text to speech (TTS) for the Arduino or other Microcontroller to use. \*And one day voice Recognition\*

For the project you need just three things and some time.

1. Raspberry PI Zero W currently $10.00
2. 16 or larger Gigabyte SD card with Raspberian installed and configured $7 to $10
3. Fe-PI Audio sound card (I2S sound) for the Raspberry PI $14

Total Cost: about $34.

You will spent quite a bit of time setting this up and that maybe worth more than money.

I used the Zero W because in the end I needed it connected to the internet, you could use the Zero but you will most likely want to get it on the net.

A Raspberry PI (model A or B) 1st generation, B+, 2, and 3 should work as well.

I’m not going into how to setup the PI other than the software that needs to be installed.

* For a headless install follow this tutorial:
* <https://docs.google.com/document/d/1mSw7UgKg9hVrtHpAotEnCF3rDQkD_MX0PFc2BXLuK08/edit?usp=sharing>

One of the first things you’ll also want to do is setup the Fe-PI. See my notes here:

<https://docs.google.com/document/d/1Gyqjyr31InzIce2quLn-a6eTcNE7UU8Xl4Z8oM5uP5A/edit?usp=sharing>

* The Fe-PI is currently supported in the kernel and setup isn’t as bad as it seems.

After you have your SD card setup and the PI booting. And are able to connect to your network. You will most likely want to rename your Raspberry PI from the default name.

Again, I’m not going to go into this as there are hundreds of tutorials on the subject.

<https://www.howtogeek.com/167195/how-to-change-your-raspberry-pi-or-other-linux-devices-hostname/>

I renamed mine to rpi-speak

Now to connect to the PI with SSH or Web, or MQTT I can use “rpi-speak.local”

So:

ssh pi@rpi-speak.local will let me ssh into my system with out knowing the IP address.

* This works most of the time, and sometimes you do need the IP address, I’m not entirely sure why that is. Find the IP address tutorial (headless) <https://www.raspberrypi.org/documentation/remote-access/ip-address.md>

Next you’ll want to make sure you are up to date.

sudo apt-get update

sudo apt-get upgrade

* This might take a while, you’ll want to reboot when it done.

You’ll want to install flite - (festival-lite) speech synthesizer. <http://www.festvox.org/flite/>

sudo apt-get install flite

Once installed give it a test and make sure you hear the audio.

flite -t “Hello World”

* It should be noted my audio levels were/are pretty low, I forget how I fix/changed this
* It has to do with the ALSA mixer. Mine however keeps defaulting to a low volume.
* sudo apt-get install alsa-utils this has alsamixer and amixer
* <https://wiki.archlinux.org/index.php/Advanced_Linux_Sound_Architecture#ALSA_Utilities>

**\*\*\* OPTIONAL STEP \*\*\***

Install Apache2 and PHP, Why? At first I thought I would beable to just send a web form and have the PI speak. It seems that while I was able to execute the commands, everything was being piped to the webpage. And it just didn’t work.

I didn’t uninstall because there are some useful things that can still be done with a web site running on the PI. But at this point, it’s a optional step, and not needed.

Install instructions:

<https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>

* I did install some CGI and PHP scripts that I found on the net. For more Information see below. Again not required.
* <http://www.unix.com/shell-programming-and-scripting/250494-launch-shell-script-parameters-webpage-button.html>
* <http://raspberrywebserver.com/cgiscripting/writing-cgi-scripts-on-a-raspberry-pi.html>
* <http://www.yolinux.com/TUTORIALS/BashShellCgi.html> \*I installed most of these\*
* <https://www.cyberciti.biz/tips/executing-linuxunix-commands-from-web-page-part-i.html>
* <https://stackoverflow.com/questions/6235785/run-a-shell-script-with-an-html-button>
* <https://stackoverflow.com/questions/18058869/how-to-run-espeak-from-php-in-linux>
* All of the above is good information, but not needed to complete this project.

It was at this point that I still needed away for the Arduino to “talk” to the PI. There are many ways to do this, and here are just a few (I didn’t use).

* <https://stackoverflow.com/questions/31193686/communicating-between-raspberry-pi-and-arduino-over-lan>
* Arduino YUN Bridge - The YUN is a linux based controller with a “leonardo” on board as well, for the two processors to communicate, a Bridge is created. <https://www.arduino.cc/en/Reference/YunBridgeLibrary>
* The YUN Bridge was ported to the PI by Paul Stoffregen
* <https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=139599>
* <https://github.com/PaulStoffregen/Bridge>
* <https://github.com/PaulStoffregen/BridgeUSB>
* The Bridge seems like a good idea, but doing some reading may be a bit to setup.
* I settled on a MQTT protocol mainly because of easy of setup.

**What is MQTT?**

Almost every tutorial on the subject starts off asking this question. Some answers go into great detail, and others are very simple. I’ll stick with the simple.

MQTT is a machine to machine simple protocol, that uses a publish/subscribe. MQTT uses a BROKER which is really just a server that handles all the messages.

<http://mqtt.org/>

MQTT Brokers can either be public or private, and there are good uses for both.

A Public Broker generally is reachable from any where, and by any one on the internet. Most do provide some type of security. The draw back however is that anyone/everyone can subscribe to your information. It is unwise to publish sensitive data to a public Broker. Most of the FREE public brokers will not guarantee 100% update.

Here are some public Brokers: (There are many more)

<http://www.hivemq.com/try-out/>

<http://www.mqtt-dashboard.com/> (a HiveMQ Dashboard)

<https://test.mosquitto.org/>

<https://www.cloudmqtt.com/>

I used a Private MQTT Broker, and it’s running on the Raspberry PI.

Mosquitto is a open source, free broker that is easy to install and easy to use.

<https://mosquitto.org/>

Here are some tutorials I used to install this, but it comes down to a simple apt-get command on the PI.

sudo apt-get install mosquitto mosquitto-clients python-mosquitto

It’s really that simple, the mosquitto server starts as soon as it’s installed, the mosquitto-clients has the command line commands to publish and subscribe. And python-mosquitto is the library to use MQTT with python.

<https://arduinodiy.wordpress.com/2017/02/04/setting-up-a-local-mqtt-broker-on-the-raspberry-pi/>

<https://www.baldengineer.com/mqtt-tutorial.html>

If you are using linux, you can also install the mosquitto-clients package, and test your install out.

MQTT uses “Topics” - this is how messages are delivered. For my project I use two “topics” - Say and Said (I will probably end up changing at least one of these).

Topics are created by a subscribe or publish to the MQTT Broker.

Here is where it gets interesting thou.

On the Raspberry PI I have a python script running that subscribes to “Say”, and publishes to “Said”. Using the desktop MQTT client I can run one CLI to publish to “Say” and one CLI to subscribe to “Said”. The effect, the two computers can talk to each other.

So with the MQTT (mosquitto broker) running on the PI,

ssh into your pi, and issue the command:

mosquitto\_sub -h localhost -t “Say” -v

And issue this command on your desktop.

mosquitto\_pub -h rpi-speak.local -t “Say” -m ‘-voice slt -t “Hello I am Sally”’

You should see:

-voice slt -t “Hello I am Sally”

In your SSH window.

So the **-h** is the host flag, it is the server running the MQTT broker, and can be public or private (such as in this case)

**-t** is the topic we want to subscribe or publish to “Say”.

It’s really that simple to send a message to a machine running a MQTT subscription.

Once you verify that is working, you’ll need to install a couple of more things, and a python script to get it all working.

Installing paho-mqtt for python.

<https://eclipse.org/paho/clients/python/>

Type: pip install paho-mqtt on the Raspberry Pi.

I published my python script to github here:

<https://gist.github.com/kd8bxp/6520a4dcda1a43834a6cda402fa441d2>

Most of what I learned, I learned from here:

<http://lelylan.github.io/lab-projects/raspberry-pi-light/>

And while he is turning GPIO on and off, it contained enough information to make my “Speech” script.

You will want to copy and paste the script to your Raspberry PI. I just put it in my home directory. “/home/pi/say.py”

You can test the script at this point by typing:

python say.py

And on your desktop, publish a message to the “Say” topic.

mosquitto\_pub -h rpi-speak.local -t “Say” -m ‘-t “Hello World!”’

You should hear, the PI say “Hello World”.

We are almost done with the Raspberry PI, we will want to set the script to start when the PI is booted or rebooted.

The easiest way is to setup a cron job. You probably don’t have any cron jobs, yet - so the PI will create a new cron file.

Type: crontab -e to edit the file, and add @reboot python /home/pi/say.py & to the end of the file, save and exit nano, and reboot. If all goes well when the system is ready - it will say “Ready”

* Don’t forget to the & at the end of the reboot command.

That is everything you need for the Raspberry PI, at the start of this project I wanted a cheap

speech synthesizer that an Arduino, or other microcontroller could use. So we need to add a MQTT library to the Arduino IDE. There are a number of MQTT libraries, but this tutorial

<https://www.baldengineer.com/mqtt-tutorial.html> recommends the PubSubClient library.

The easiest way to install is with the “library manager” on 1.6.x and above IDE.

The github for the library is here: <https://github.com/knolleary/pubsubclient> and the full documentation for it is here: <http://pubsubclient.knolleary.net/>

This is a pretty easy to use library, I modified the mqtt\_esp8266 example to publish to the “Say” topic, and subscribe to the “Said” topic.

The only problem I had was the library was unable to connect with the host name “rpi-speak”

Or “rpi-speak.local” and needed an IP address.

This was the only hickup - But being able to SSH into the PI, or using the Webserver and CGI script for the environment (env.sh) it was pretty easy to get the IP address.

My modified sketch can be found here:

<https://gist.github.com/kd8bxp/59b09e72d4f5ff0e4375fcb24df527d9>

Jun 19, 2017

Added a PHP script that starts at reboot, checks the IP address and if it’s changed will send a SMS message with the new IP address.

For this to work, I added a sleep delay of about 3 minutes to the start of the PHP script, and a crontab job to run the script at reboot.

The script checks for a “ip.txt” file, compares the contents to the current IP address - if it is different, it will trigger a IFTTT SMS message. It writes the IP address back into the file for next reboot. The delay (sleep) at the begin of the script, allows enough time for the network to come online and get an IP address, too short of a delay and you’ll not get a good IP, too long of delay and you’ll wait for a long time.

This tutorial on how to send a SMS using a ESP8266 is for the most part what I used create the SMS trigger.

<https://circuitdigest.com/microcontroller-projects/sending-sms-using-esp8266>

I did add the “value1” for the IP address and changed the message that was sent but otherwise.

My PHP script can be found here:

<https://gist.github.com/kd8bxp/280b2f0cc301a49c3b2bb7520806a191>

I added a line to the “SAY.PY” python script that checks for “@@shutdown” and will shutdown the PI safely.