Raspberry Pi Radio Scanner Control Web Application

A Senior Project
presented to
the Faculty of the Computer Science Department
California Polytechnic State University, San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science

by
Jackson Marshall Strand
June, 2016

Table of Contents

Abstract

In this senior project I design and develop software for the Raspberry Pi which allows the user to listen to and control a scanner radio anywhere in their home. The solution involves utilizing a Raspberry Pi 2 interfaced with the radio, a software package to communicate directly with the radio, and a web app hosting a web client providing the interface for user control. Audio streaming is achieved through the use of FFmpeg. Listening on the client is achieved by using an external software such as Video Lan (VLC) to open the audio stream. We provide an operable proof-of-concept with less than 1-second of latency.

List of Figures

List of Tables

Introduction

Radio scanning is a popular hobby interesting to those in many fields. Popular services to listen to include public safety, railroads, aviation, amateur radio, and many more. Traditionally one uses a scanner, a radio receiver with a large memory of programmable frequencies which it 'scans' through searching for activity. Scanners are available in both small handheld and larger base station type devices. Handheld scanners with a small attached antenna can perform well with nearby signals in certain frequency bands, but varying patterns of interference can occur within buildings. Also, more distant signals and those of longer wavelengths are not received well on handheld scanners with attached antennas. For optimal performance an external antenna mounted at a higher elevation, such as a roof or pole is used. It is then connected to either a base station or handheld scanner. This setup, while ideal for reception, physically ties the listening experience to where the cable is connected.

Still, portability of the listening experience is desirable. Being able to listen to the scanner radio from any location within the home can increase listening time while performing other tasks. Many scanners include an audio out port, thus various audio transmitting devices could be used to provide portability of audio. However, when listening to radio scanners, one often needs to adjust the settings, pause on a frequency, or otherwise control the scanner. Therefore, the ideal solution for a scanning enthusiast includes both audio streaming, and remote control of the scanner radio.

Uniden Corp, in its newest scanner model (BCD536HP) has included a WIFI module and applications for PC, Apple, and Android that providing audio streaming and remote control. [1] But, this scanner has a market value of approximately 550 USD [2].

Uniden includes audio line out and USB API control for all of their scanners of recent years, including scanners in the 90-350$ range. A few desktop software packages utilize the Uniden API to provide programming and remote control over direct USB connection. However, none are available that provide remote control and audio distribution over a network.

This project utilizes a Raspberry Pi, a scanner and several software packages to provide an operable proof of concept. The software components are broken down to two major categories: audio streaming, and remote control.

Hardware

The hardware used for this project includes: a radio scanner, Raspberry Pi 2 and a USB sound adapter. As shown in figure 1, the Raspberry Pi is connected to the local area network via the onboard Ethernet and to the scanner via an USB cable. The USB sound card is connected to the Raspberry Pi to allow for capture of audio. The external speaker port of the scanner is connected to the microphone on the sound card via 3.5mm cable.

Table 1: Hardware Components

Component	Model	Purpose	Approximate Cost USD	Citation
Scanner	Uniden BCD996P2	Radio Receiver	400	[3]
Raspberry Pi	Raspberry Pi 2	Connect and control scanner, Host Web Control App, Stream Audio	35 + accessories (case, power supply, etc.)	[4]
Sound Card	Sabrent AU-MMSA	Capture Audio for Pi	6	[5]

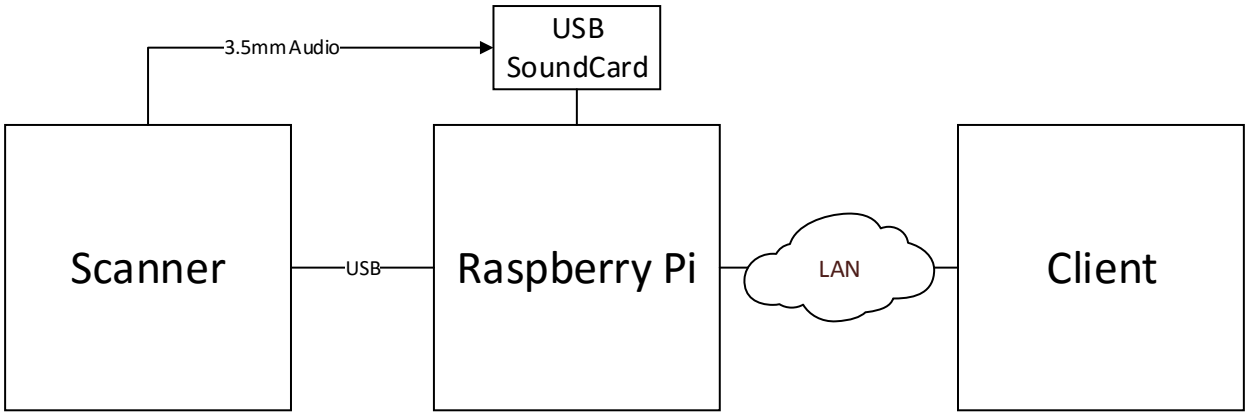

Figure 1: High Level Hardware Overview

Software

Operating System Selection

We considered various operating systems for the Raspberry Pi including: Raspberian, Ubuntu, Ubuntu Snappy, and Windows IoT. We choose a community supported image of Ubuntu 14.04 [6] for compatibility with existing software packages and user familiarity. You can read more about available operating systems in Appendix C.

Audio Streaming

Audio streaming is an important facet of this project as latency of more than 1 second would make control of the scanner difficult. We explored using both HTTP Live Streaming (HLS) [7] and FFmpeg [8] for Real Time Protocol (RTP) streaming of audio.

For HLS streaming we used two software packages. The first package captures audio and sends it to the second known as a 'server'. The server can be located on another device, however in our implementation it is also on the Raspberry Pi. The server manages and serves audio to attached clients. This is a robust system and has the advantage of working in most browsers. Unfortunately, latency is too high ranging from 5-15 seconds. In our testing we used Darkice [9] as the source software and Icecast2 [10] as the server.

The second option is the software FFmpeg, self-described as a 'Cross-platform solution to record, convert and stream audio and video.' [8] After testing various configurations, RTP streaming using the mp2 codec produced acceptable latency and low cpu usage; see Table 2 and below.

The command is:

ffmpeg -re -f alsa -i plughw:1,0 -acodec mp2 -f rtp rtp://dest-ip:openport

This results in cpu usage of around 15% (measured using the top command). While latency is difficult to measure in this case, the times in Table 2 (next page) are measured by muting the scanner and using a stopwatch to measure the time delay. All times are taken with the Raspberry Pi connected via Ethernet.

Table 2: Audio Stream Latency

Client	Connection	Network Caching Level	Latency
Desktop (Windows) VLC Media Player	Ethernet	100ms	< 300ms
Laptop (Windows) VLC Media Player	Wi-Fi	100ms	< 300ms
Apple IPad Air 2 VLC for Mobile	Wi-Fi	"Lowest Latency"	~ 700ms

In order to distribute the stream there are two options. The first option is to stream directly to the client; this requires a separate instance of FFmpeg for each client. However, due to access control of the USB soundcard, only one instance of FFmpeg can run at a given time.

Alternatively we can stream to a multicast address, e.g. 224.0.0.0-239.x.x.x. [11] In our testing, while streaming to a multicast address, Ethernet connected clients work fine; unfortunately, Wi-Fi connected clients are unable to receive the stream clearly. Therefore, for this project we use unicast streaming. This is acceptable as the most likely use case is a single client.

The web app of this project manages the creation and closing of audio streams. The client software gives the user options to manage the destination and status of the stream. See section: Client Side Software.

To receive the stream, the user uses an external software such as VideoLan (VLC) [12] , which is an open source cross-platform multimedia player. It is available for Windows, Linux, OSX, IOS and Android. The stream is opened by "opening a network stream" at rtp://@:port. '@' acts as a localhost pointer in VLC.

For this project, FFmpeg was built from source, following standard procedure to ensure performance and to allow for additional libraries (none are needed to run this project). It is dependent on FFmpeg being compiled with ALSA (Advanced Linux Sound Architecture) support, which it utilizes to capture the raw audio from the soundcard. FFmpeg can also be obtained from the standard repositories with ALSA support.

Remote Control

In this project we design software to provide remote control of the scanner and allow for configuration of an audio stream. As shown in figure 2, the solution is broken into several parts: a 'scanner controller', the web application, and the client web page and JavaScript.

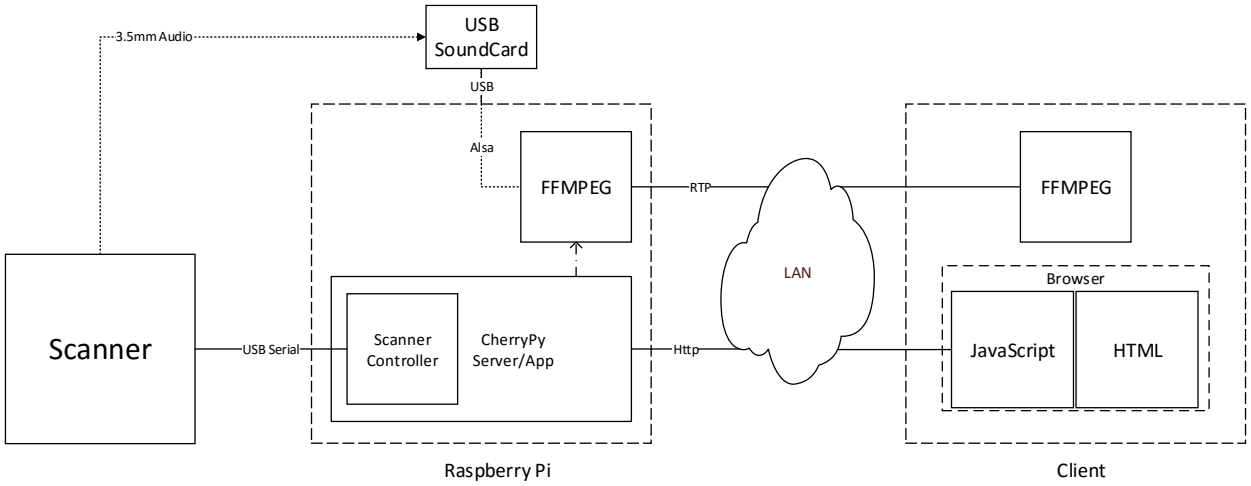

Figure 2: Software Overview

Scanner Controller

The primary purpose of the controller is to send commands, forwarded from the web app, to the scanner. It is Python based for compatibility with Pyserial [13] and the web application. The decision to separate the controller from the web application, allows for modularity and future compatibility of addition scanner models.

The controller, uses the serial module provided by Pyserial [13] to communicate with the scanner over USB. The controller, contains the configuration for initializing the serial connection. The proper settings for the BCD996P2 are provided in the remote control API document [14]. This connection is opened in the 'init()' method. If the scanner is not connected or turned off, it will fail to open the serial connection. See future improvements, for additional notes on this.

Currently, 'get_status' is the only command supported by the controller that is not in the scanner's API. To handle this command, the controller sends 'STS' to the scanner; this command gets the current status, including the display. Next, the controller formats the data for the clients display. Uniden uses a custom character mapping so it strips non-utf-8 characters and replaces those it can.

To handle asynchronous web inputs, the built-in Threading.lock [15] module is used to control access to the USB medium.

Web Application Sever

After comparing various minimalist web frameworks, we choose CherryPy [16] for its simplicity, robustness and excellent documentation. The framework, being Python based, allows for integration with the controller.

The CherryPy web app provides a basic web API for the client to interact with. As shown in table 3, it contains several URL handlers: index, command, setup_stream, stream_status, and disconnect_stream. It also contains an instance of the controller object, to enable communication with the scanner. The web app is started by the command 'python scannerapp.py'. The client can navigate to the address of the Raspberry Pi at port 8080, e.g. http://ip:8080 to open the control page.

Table 3: Web API

URL	Parameters	Description
/		Returns index.html; the client control page.
/command	cmd	Forwards the parameter cmd to the controller, returns the result.
/stream_status		Returns whether a stream is active and if so the destination.
/setup_stream	host	Closes any active stream and starts a new stream to the 'host' destination at port 1234
/disconnect_stream		Closes any active stream.

Client Side Software

The client side software includes HTML and JavaScript. Since these provide the remote interface for controlling the scanner, each supported scanner will require unique versions of these. The index.html client page provides software buttons for all physical buttons, a display area, and buttons for stream management.

The JavaScript file 'streamManagement.js' utilizes JQuery [17] and provides button handlers for the stream control interface. The button handlers perform the associated posts to the web app.

The JavaScript file 'uniden996p2controls.js' also utilizes JQuery and provides button handlers for all scanner control buttons, as well as functions to manage updating the display. In addition, it manages Volume and Squelch levels. To update the display, it has a recursive function that polls for the display status. More discussion on this function follows in Future Improvements. For comparison of the physical interface of the scanner with the client software interface see figures 3 and 4.

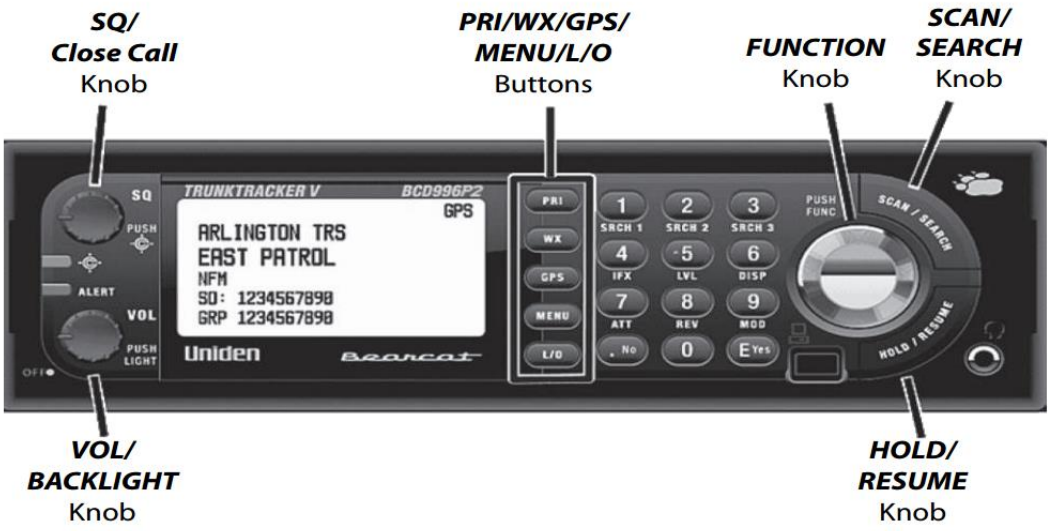

Figure 3: Scanner Hardware Interface (from Uniden Official Manual) [18]

BCD996P2

Rx: 1
Cal Trans
DOT Pfeiffer
856.7625
S1:1*3456----
GRP****5*--**

Disable Display Refresh
Volume 8 - +
Squelch 3 - +

PRI | Scan/Search
WX | 1 | 2 | 3 | Hold/Resume
GPS | 4 | 5 | 6 | short_press | VFO Right
MENU | 7 | 8 | 9 | Close Call | Function
L/O | . No | 0 | E Yes | light | VFO Left

No Stream Active | Refresh Stream Status
Stream to This Device | | Stream to input | Disconnect Stream

Figure 4: Software User Interface

Future Improvements

Many Improvements could be made to this project. Currently, it contains several shortcomings that should be addressed. The most pressing of which is the current method of refreshing the display in the client. Currently, this is achieved through a recursive JavaScript function, as discussed in the client side software section. This is an issue because if a second client connects, it will double the amount of commands the controller sends to the scanner. As more clients connect, this can cause congestion of the USB medium.

One method to fix this is to have the controller poll for the status. It should store the formatted display status data, and return the stored data to 'get_status' requests from clients. This fixes the issue by making the number of polling processes constant at 1. Another improvement is to have the web app push the updates to the clients instead of having clients request an update. This could reduce web traffic by one half.

Another issue is Uniden's custom character mapping that supports special characters. Currently the project converts some of the non-standard characters to standard UTF. While this provides good coverage of normal usage, special modes requiring the special characters cannot be displayed. For reference the key mapping can be found in the BCD996P2_remote_commands spec sheet [14]. Creating a solution to display these characters would involve either a custom font or creating a custom display element.

Another functional issue is currently when the web app is started it attempts to instantiate a controller object. This requires the scanner to be powered on and connected via USB; if not, it will exit. In addition, if the scanner is disconnected or powered off while the web app is up, the web app will have to be restarted to reinstate control. Instead, the web app should provide an option to connect to the scanner at any time, minimizing the need for the web app to be restarted.

Feature wise, the controller could be expanded to handle memory management or provide non-API specified control options. This would improve upon the native usability of the scanner and allow remote programming of the scanner's memory. In addition, the application could be setup to detect which scanner model is attached and tailor the controller and web page to the specific model. This requires providing either separate or dynamic[1] controllers and web pages compatible with the specific models.

[1] Here, dynamic refers to a controller which can handle several scanners that have similar APIs.

Conclusion

This project enables scanner listeners to improve their listening experience by removing the physical limitation usually associated scanning. It provides remote scanner control and audio streaming with low latency. It is designed to allow improvement and expandability and is usable from desktop and mobile clients.

We hope that others may find this project useful and expand its functionality; source code is available on BitBucket at https://bitbucket.org/jackscrj/piscannerwebcontrol and in Appendix D.

References

[1] Uniden America Corporation, "BCD536HP HomePatrol Series Scanner with Wi-Fi," [Online]. Available: https://www.uniden.com/scanner/id-BCD536HP/BCD536HP_HomePatrol_Series_Scanner_with_Wi-Fi. [Accessed 3 6 2016].

[2] Scanner Master, "Uniden Bearcat BCD536HP Police Scanner | ScannerMaster.com," [Online]. Available: http://www.scannermaster.com/Uniden_Bearcat_BCD536HP_Police_Scanner_p/10-501854.htm. [Accessed 3 6 2016].

[3] Scanner Master, "Uniden Bearcat BCD996P2 Police Scanner | Scanner Master," [Online]. Available: http://www.scannermaster.com/Uniden_Bearcat_BCD996P2_Police_Scanner_Radio_p/10-501891.htm. [Accessed 3 6 2016].

[4] Allied Electronics, "Raspberry Pi - RASPBERRY PI 2, MODEL B - 85.6 x 56 x HDMI 27 GPIO USB 1GB RAM 900Mhz Quad-Core Open Frame Computer - Allied Electronics," [Online]. Available: http://www.alliedelec.com/raspberry-pi-raspberry-pi-2-model-b/70465426/. [Accessed 3 6 2016].

[5] Amazon, "Amazon.com: Sabrent USB External Stereo Sound Adapter for Windows and Mac. Plug and play No drivers Needed. (AU-MMSA): Computers & Accessories," [Online]. Available: http://www.amazon.com/Sabrent-External-Adapter-Windows-AU-MMSA/dp/B00IRVQ0F8. [Accessed 3 6 2016].

[6] R. Finnie, "ARM/RaspberryPi - Ubuntu Wiki," [Online]. Available: https://wiki.ubuntu.com/ARM/RaspberryPi. [Accessed 3 6 2016].

[7] Apple Inc., "HTTP Live Streaming (HLS) - Apple Developer," [Online]. Available: https://developer.apple.com/streaming/. [Accessed 3 6 2016].

[8] "FFmpeg," [Online]. Available: https://www.ffmpeg.org/. [Accessed 3 6 2016].

[9] R. Diniz, "DarkIce live audio streamer," [Online]. Available: http://www.darkice.org/. [Accessed 3 6 2016].

[10] "Icecast," [Online]. Available: http://icecast.org/. [Accessed 3 6 2016].

[11] S. Venaas, "IPv4 Multicast Address Space Registry," 27 5 2016. [Online]. Available: http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml. [Accessed 3 6 2016].

[12] "VideoLAN - VLC: Official site - Free multimedia solutions for all OS!," [Online]. Available: http://www.videolan.org/index.html. [Accessed 3 6 2016].

[13] C. Liechti, "pyserial 3.1 : Python Package Index," [Online]. Available: https://pypi.python.org/pypi/pyserial. [Accessed 3 6 2016].

[14] Uniden America Corporation, "BCD996P2_Remote_Commands.pdf," [Online]. Available: http://info.uniden.com/twiki/pub/UnidenMan4/BCD996P2/BCD996P2_Remote_Commands.pdf. [Accessed 5 6 2016].

[15] Python Software Foundation, "17.1. threading — Thread-based parallelism - Python 3.5.1 documentation," 2016 5 6. [Online]. Available: https://docs.python.org/3/library/threading.html#lock-objects. [Accessed 5 6 2016].

[16] The CherryPy team, "CherryPy — A Minimalist Python Web Framework," [Online]. Available: http://www.cherrypy.org/. [Accessed 5 6 2016].

[17] The JQuery Foundation, "JQuery," [Online]. Available: https://jquery.com/. [Accessed 5 6 2016].

[18] Uniden America Corporation, "BCD996P2om.pdf," 2015. [Online]. Available: https://www.uniden.com/File%20Library/FooterNav/Product%20Information/Owners%20Manuals/BCD996P2om.pdf. [Accessed 5 6 2016].

[19] Uniden America Corporation, "BCD996P2 Digital Mobile TrunkTracker V Scanner," [Online]. Available: https://www.uniden.com/scanner/id-BCD996P2/BCD996P2_Digital_Mobile_TrunkTracker_V_Scanner. [Accessed 3 6 2016].

Appendix A: Senior Project Requirements and Timeline

A senior project is required to an independent, self-owned project requiring research and creativity. With respect to independence and self-ownership, this project was researched and completed by myself. It also required research into the hardware, Uniden's API, various audio streaming techniques, and web development. Creatively, this project involved using multiple hardware components and creating custom software to solve a unique problem.

Timeline

Senior projects are expected to take two quarters to complete. This project was started winter quarter of 2016 (January 4, 2016), and was submitted on June, 7 2016 at the conclusion of the 2016 spring quarter. Initial research of software and acquisition of hardware took place during January. February, consisted of researching audio streaming options. During March, development of the controller and serial interfacing took place. April through middle of May was devoted to research and development of the web app and interface as well as combining the individual components into a cohesive system. The remainder of May and few days of June were devoted to this write up.

Appendix B: Setup and Use Guide

Hardware:

Currently, the project only supports the Uniden BCD996P2 [19] scanner. If using a Raspberry Pi, you will need an audio capture device, such as the Sabrent U-MMSA [5]. It is recommended you use a Raspberry Pi 2 or 3.

Setup:

First, install the operating system of your choice on the Pi; Ubuntu [6] is recommended. You will need to connect your scanner to the Raspberry Pi with the USB programming cable. Also you will need to connect the audio port from the scanner to the capture port on your audio capture device.

Next, you will need to install the dependencies of the project. You will need FFmpeg with alsa support. This can be obtained from the standard repository or compiled from source.

sudo apt-get install ffmpeg

You will also need a Python 3 environment with CherryPy and Pyserial installed. These can both be installed from Python Pip. You can use a Virtual Environment to manage this:

sudo apt-get install python3

sudo apt-get install virtualenv

virtualenv -p python3 envname

source envname/bin/activate to exit the env *deactivate*

Your terminal should display '(envname) user@machine$' now

Install CherryPy and Pyserial:

pip install CherryPy

pip install pyserial

Clone the source:

Install git if necessary *sudo apt-get install git*

git clone https://bitbucket.org/jackscrj/piscannerwebcontrol

Usage:

Ensure your scanner is turned on and connected to the Pi. Ensure the VirtualEnv is activated.

Enter the source directory:

cd path-to-source/pirscannerwebcontrol

Start the web app:

python scannerapp.py

You may now navigate to your Raspberry Pi's address at port 8080 from another client.

Ex. http://192.168.1.39:8080

You will be greeted by the web interface.

To start an audio stream to your current device click "stream to this device". Or to stream to another client, type in the address and click "stream to input".

To receive the stream, you will need to utilize VLC Player [12], it is available for Windows, OSX, Android and IOS. To open the stream, click open a network stream and type in rtp://@:1234. This needs to be done on the client which the stream is pointed at.
If you are using a desktop version, click show advanced options and set network-caching to 100ms.

You should now have full control and audio.

Shutdown and Notes:

To shut down the app send control-c to the web app. It will properly shutdown.

If the scanner is disconnected or shutoff the web app will need to be restarted to re-enable control.

Appendix C: Operating Systems Comparison

Raspberian is a Debian based OS, and is the standard operating system for Raspberry Pi. It is good on resource consumption and compatibility with the hardware is excellent. While good for educational environments, it is a bit lacking in package availability and compatibility compared to Ubuntu.

Ubuntu Snappy is 'a new transactionally-updated Ubuntu for IoT devices, clouds and more.' It is light on resource usage. It also includes a new package management system. It packages applications as a single bundle with their dependencies, so this could prove to be a good solution for distribution and use by less technical users.

While this system would seem ideal for Raspberry Pi, for this project time was not available to learn a new development package system. In addition, more users have familiarity with traditional Ubuntu packages which would allow more users to use and extend the end product.

Ubuntu 14.04 this is a community supported image made for the Raspberry Pi. This provides a nice clean installation of a headless Ubuntu desktop machine. This is ideal for this purpose as it is intended to be accessed remotely, and keeps the resource usage and filesystem size down. But this still allows us to access preexisting software intended for Ubuntu/Linux desktop. In addition, developers who are familiar with Linux development will have no difficulty in further extending the application. Ubuntu 16.04 has since been released with images for Raspberry Pi 2 and 3 available. This project has not been extensively tested on 16.04; however, there are no anticipated compatibility issues.

Windows IoT Microsoft has provided a nice system in Windows Iot Core, which is free to run on Raspberry Pi. However, the decision was made to stick with Linux for open platform, developmental familiarity as well as preexisting software packages which would prove useful.

Appendix D: Source Code

Note: all source code is available at https://bitbucket.org/jackscrj/piscannerwebcontrol

Unidencontroller.py

```
import time
import serial
import threading

class unidencontroller(object):
        lock = threading.Lock()
        def __init__(self):
                # configure the serial connections
                self.ser = serial.Serial(
                        port='/dev/ttyACM0',
                        baudrate=115200,
                        parity=serial.PARITY_NONE,
                        stopbits=serial.STOPBITS_ONE,
                        bytesize=serial.EIGHTBITS
                )
                if(self.ser.isOpen):
                        print("BCD996P2CONTROLLER Initialized")
                else:
                        print("bcd966p2controller failed to initialize")
                        exit()

        #accept a command and send to the scanner
        def command(self, cmd):
                if cmd == 'get_status':
                        return get_status()
                else:
                        return self.send_command(cmd)

        #send command to scanner, returns response
        def send_command(self, cmd):
                self.lock.acquire()
                cmd += '\r\n'
                self.ser.write(cmd.encode())
                response = self.get_response()
                self.lock.release()
                if len(response) == 0:
                        return b'Error: Scanner did not respond'
                return response

        #get serial response
        def get_response(self):
                out = bytes()
                time.sleep(.1)
                while self.ser.inWaiting() > 0:
                        out += self.ser.read(1)
                return out
```

```
#get STS from scanner
def get_status(self):
        sts = self.send_command('STS\r\n')
        if sts == b'Error: Scanner did not respond':
                return sts

        sts = self.clean_text(sts)
        #refer to API to understand formating of return information
        stslist = sts.split(',')
        numrows = len(stslist[1])
        dsplist = list()

        #format display text
        j = 2
        for i in range(numrows):
                cur_row = ''
                if stslist[j+1] == '________________':
                        cur_row = '<u>' + stslist[j] + '</u><br>'
                elif stslist[j+1] == '****************':
                        cur_row = '<span class="highlight">' +
                                        stslist[j] + '</span><br>'
                else:
                        cur_row = stslist[j] + '<br>'
                dsplist.append(cur_row)
                j += 2
        dispstring = ''.join(dsplist)
        return dispstring

#remove special character mappings and replace mappable ones
def clean_text(self, txt):
        txt = txt.replace(b'\x8D\x8E\x8F\x90',b'HOLD')

        #RSI levels
        txt = txt.replace(b'\xA6', b'Rx: 1')
        txt = txt.replace(b'\xA7', b'Rx: 2')
        txt = txt.replace(b'\xA8\xA9', b'Rx: 3')
        txt = txt.replace(b'\xAA\xAB', b'Rx: 4')
        txt = txt.replace(b'\xAC\xAD', b'Rx: 5')

        #PRI
        txt = txt.replace(b'\xA1\xA2', b'PRI')

        #ATT
        txt = txt.replace(b'\xa3\xa4\xa5', b'ATT')

        #L/O
        txt = txt.replace(b'\x95\x96\x97', b'L/O')
        txt = txt.replace(b'\xDD\xDE\xDF', b'L/O')

        #NAC
        txt = txt.replace(b'\xD4\xd5\xd6', b'NAC:')

        #MAX
        txt = txt.replace(b'\xd0\xd1\xd2\xd3', b'MAX')

        #REP
        txt = txt.replace(b'\xcd\xce\xcf', b'REP')
```

```
#scr
txt = txt.replace(b'\xc8\xc9\xca', b'SCR')

#IFX
txt = txt.replace(b'\xc5\xc6\xc7', b'IFX')

#SRCH
txt = txt.replace(b'\xc1\xc2\xc3\xc4', b'SRCH')

#DSKP
txt = txt.replace(b'\x91\x92\x93\x94', b'DSKP')

#AM FM NFM WFM FMB
txt = txt.replace(b'\x98\x99\x9a', b' AM')
txt = txt.replace(b'\x9B\x9C\x9a', b' FM')
txt = txt.replace(b'\x9D\x9E\x9C\x9a', b'NFM')
txt = txt.replace(b'\x9f\xa0\x9c\x9a', b'WFM')
txt = txt.replace(b'\x9b\x9c\xb9\xba', b'FMB')

txt = txt.decode('utf-8', 'ignore')
return txt
```

Scannerapp.py

```
import cherrypy
import os
import subprocess
import threading
from unidencontroller import unidencontroller


class ScannerWebApp(object):
        def __init__(self,ctrl):
                self.controller = ctrl
                self.stream = None
                self.stream_url = None
                self.stream_port = None
                #'plughw:1,0' for raspberry pi
                self.alsaplug = 'plughw:1,0'

        @cherrypy.expose
        def index(self):
                return open('index.html')

        @cherrypy.expose
        def command(self, cmd):
                return self.controller.command(cmd)

        @cherrypy.expose
        def setup_stream(self, host):
                rhost = cherrypy.request.remote.ip
                if host=='this_client':
                        host = rhost

                if self.stream != None:
                        self.close_stream()

                port = '1234'
                self.open_stream(host, port)

                rtn = 'Active at other client: ' if host != rhost else 'Active
at your Client: '
                return rtn + self.stream_url + ":" + port;

        @cherrypy.expose
        def stream_status(self):
                rhost = cherrypy.request.remote.ip
                if self.stream_url == None:
                        return 'No Stream Active'
                rtn = 'Active at other client: ' if self.stream_url != rhost
else 'Active at your Client: '
                return rtn + self.stream_url + ":" + self.stream_port
```

```
        @cherrypy.expose
        def disconnect_stream(self):
                self.close_stream()
                return 'OK'

        def close_stream(self):
                self.stream.terminate()
                self.stream.wait(100)
                self.stream = None
                self.stream_url = None
                self.stream_port = None

        def open_stream(self, host, port):
                self.stream = subprocess.Popen(("ffmpeg -re -f alsa -i " +
self.alsaplug + " -codec libmp3lame -f rtp rtp://"
                                + host + ":" + port).split())
                self.stream_url = host
                self.stream_port = port


if __name__ == '__main__':
        conf = {
                '/': {
             'tools.staticdir.root': os.path.abspath(os.getcwd())
          },
          '/static': {
              'tools.staticdir.on': True,
              'tools.staticdir.dir': './public'
          }
      }
        ctrl = unidencontroller()

        cherrypy.config.update({'server.socket_host': '0.0.0.0'})
        cherrypy.quickstart(ScannerWebApp(ctrl), '/', conf)
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
        <link href="/static/css/style.css" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-2.2.3.min.js"></script>
        <script src="static/js/uniden996p2controls.js"
type="text/javascript"></script>
        <script src="static/js/streamManagement.js"
type="text/javascript"></script>
</head>
        <body>
                <div id="header">BCD996P2<br>
                        <p id="display"></p>
                        <button id="button_display_refresh">Disable Display
Refresh</button>
            <div class="arrange-horizontally">
                <span id="display_volume">Volumne: 0</span>
                <button id="button_vol_down">-</button>
                <button id="button_vol_up">+</button>
            </div>
            <div class="arrange-horizontally">
                <span id="display_squelch">Squelch: 0</span>
                <button id="button_squelch_down">-</button>
                <button id="button_squelch_up">+</button>
            </div>
               </div>

        <div  class="arrange-horizontally">
                <div class="arrange-vertically">
                        <button id="key_PRI">PRI</button>
                        <button id="key_WX">WX</button>
                        <button id="key_GPS">GPS</button>
                        <button id="key_MENU">MENU</button>
                        <button id="key_LO">L/O</button>
                </div>
                <div class="arrange-horizontally">
                        <div class="arrange-vertically">
                                <button id="key_1">1</button>
                                <button id="key_4">4</button>
                                <button id="key_7">7</button>
                                <button id="key_dot">. No</button>
                        </div>
                        <div class="arrange-vertically">
                                <button id="key_2">2</button>
                                <button id="key_5">5</button>
                                <button id="key_8">8</button>
                                <button id="key_0">0</button>
                        </div>
                        <div class="arrange-vertically">
                                <button id="key_3">3</button>
                                <button id="key_6">6</button>
                                <button id="key_9">9</button>
                                <button id="key_E">E Yes</button>
                        </div>
```

```
                        <div class="arrange-vertically">
                                <button id="key_SCAN">Scan/Search</button>
                                <button id="key_HOLD">Hold/Resume</button>
                                <button id="key_longpress">short_press</button>
                                <button id="key_SQ_press">Close Call</button>
                                <button id="key_vol_press">light</button>
                        </div>
            <div class="arrange-vertically">
                <button id="key_VFO_RIGHT">VFO Right</button>
                <button id="key_function">Function</button>
                <button id="key_VFO_LEFT">VFO Left</button>
            </div>
               </div>
        </div>
<br>    <div id="stream_config" visible="false">
                        <span id="active_stream_url">Stream Not Active.</span>
                        <button id="button_refresh_stream_status">Refresh
Stream Status</button>
                        <br>
                        <button id="button_stream_here">Stream to This
Device</button>
                        <input type="text" id="stream_url_address_input">
                        <button id="button_stream_to_input">Stream to
input</button>
                        <button id="button_disconnect_stream">Disconnect
Stream</button>
        </div>

</body>
</html>
```

streamManagement.js

```
//Handles buttons and setup for getting
//and displaying stream address
$(document).ready(function() {


    get_init_stream_state();

    function get_init_stream_state () {
        $.post("/stream_status")
        .done(handle_stream_url_callback);
    }

    function handle_stream_url_callback (rtn) {
        $("#active_stream_url").html(rtn);
    }

    $("#button_stream_here").click(function (e) {
        $.post("/setup_stream", {"host" : "this_client"})
        .done(handle_stream_url_callback);
    });

    $("#button_stream_to_input").click(function (e) {
        $.post("/setup_stream", {"host" :
$("#stream_url_address_input").val()})
        .done(handle_stream_url_callback);
    });

    $("#button_refresh_stream_status").click(function (e) {
        $.post("/stream_status")
        .done(handle_stream_url_callback)
    });

    $("#button_disconnect_stream").click(function (e) {
        $.post("/disconnect_stream")
        .done(function (rtn) {
            if (rtn == "OK") {
                $("#active_stream_url").html("No Stream Active");
            } else {
                window.alert("Error disconnecting stream");
            }
        });
    });
});
```

Uniden996p2controls.js

```
$(document).ready(function() {
    var display = document.getElementById("display");
    var long_press = false;
    var volume_level = 0;
    var sql_level = 0;
        var status_poll_bool = true;

    initialize_display();

        update_status();

        function update_status() {
                $.post("/command", {"cmd" : "get_status"})
                .done(function(rtn) {
                        $("#display").html(rtn);
                        if(status_poll_bool) {
                                setTimeout(update_status, 300);
                        }
                });
    }

    function initialize_display() {
        $.post("/command", {"cmd" : "VOL"})
        .done(handle_vol_return_val);

        $.post("/command", {"cmd" : "SQL"})
        .done(handle_sql_return_val);
    }



        //used to handle return from VOL when expecting a value eg
    // "VOL,12"
    function handle_vol_return_val(rtn) {
        if (rtn.slice(0,4) === "VOL,") {
            volume_level = parseInt(rtn.slice(4));
            $("#display_volume").html('Volume ' + volume_level);
        } else {
            window.alert("Expected VOL,... from scanner got:" + rtn);
        }
    }

    function handle_vol_return_ok(rtn) {
        if (rtn == "VOL,OK\r") {
            $("#display_volume").html('Volume ' + volume_level);
                        beep();
        } else {
            $.post("/command", {"cmd" : "VOL"})
            .done(handle_vol_return_val);
        }
    }
```

```
function handle_sql_return_val(rtn) {
    if (rtn.slice(0,4) === "SQL,") {
        sql_level = parseInt(rtn.slice(4));
        $("#display_squelch").html('Squelch ' + sql_level);
    } else {
        window.alert("Expected SQL,... from scanner got:" + rtn);
    }
}

function handle_sql_return_ok(rtn) {
    if (rtn === "SQL,OK\r") {
        $("#display_squelch").html('Squelch ' + sql_level);
                  beep();
    } else {
        $.post("/command", {"cmd" : "SQL"})
        .done(handle_sql_return_val);
    }
}

function handle_button_return(rtn) {
    if (rtn == "KEY,OK\r") {
       beep();
    } else {
        window.alert("Button error: " + rtn);
    }
}

$("#button_vol_up").click(function(e) {
    if (volume_level === 29) {
       window.alert("already max vol");
    } else {
        volume_level += 1;
        cmd = "VOL," + volume_level.toString();
        $.post("/command", {"cmd" : cmd})
        .done(handle_vol_return_ok);
    }
});

$("#button_vol_down").click(function(e) {
    if (volume_level === 0) {
        window.alert("already min vol");
    } else {
        volume_level -=1;
        cmd = "VOL," + volume_level.toString();
        $.post("/command", {"cmd" : cmd})
        .done(handle_vol_return_ok);
    }
});
```

```
$("#button_squelch_up").click(function(e) {
        if (sql_level === 19) {
           window.alert("already max radio closed");
        } else {
            sql_level += 1;
            cmd = "SQL," + sql_level.toString();
            $.post("/command", {"cmd" : cmd})
            .done(handle_sql_return_ok);
        }
    });

    $("#button_squelch_down").click(function(e) {
        if (sql_level === 0) {
            window.alert("already min sql");
        } else {
            sql_level -=1;
            cmd = "SQL," + sql_level.toString();
            $.post("/command", {"cmd" : cmd})
            .done(handle_sql_return_ok);
        }
    });

    $("#button_display_refresh").click(function(e) {
               if (status_poll_bool) {
                       status_poll_bool = false;
                       $("#button_display_refresh").html("Enable Display
Refresh");
               } else {
                       status_poll_bool = true;
                        $("#button_display_refresh").html("Disable Display
Refresh");
                        update_status();
               }
        });

        $("#key_longpress").click(function(e) {
               if (long_press) {
                       long_press = false;
                       $("#key_longpress").html("Short Press");
               } else {
                       long_press = true;
                       $("#key_longpress").html("Long Press");
               }
        });

        $("#key_PRI").click(function(e) {
               $.post("/command", {"cmd" : "KEY,P,P"})
               .done(handle_button_return);
        });
```

```
$("#key_WX").click(function(e) {
        if (long_press) {
                cmd = "KEY,W,L";
        } else {
                cmd = "KEY,W,P";
        }
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return);
});

$("#key_GPS").click(function(e) {
        $.post("/command", {"cmd" : "KEY,G,P"})
        .done(handle_button_return);
});

$("#key_MENU").click(function(e) {
        $.post("/command", {"cmd" : "KEY,M,P"})
        .done(handle_button_return);
});

$("#key_LO").click(function(e) {
        if (long_press) {
                cmd = "KEY,L,L";
        } else {
                cmd = "KEY,L,P";
        }
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return);
});

$("#key_1").click(function(e) {
        $.post("/command", {"cmd" : "KEY,1,P"})
        .done(handle_button_return);
});

$("#key_2").click(function(e) {
        $.post("/command", {"cmd" : "KEY,2,P"})
        .done(handle_button_return);
});

$("#key_3").click(function(e) {
        $.post("/command", {"cmd" : "KEY,3,P"})
        .done(handle_button_return);
});

$("#key_4").click(function(e) {
        $.post("/command", {"cmd" : "KEY,4,P"})
        .done(handle_button_return);
});

$("#key_5").click(function(e) {
        $.post("/command", {"cmd" : "KEY,5,P"})
        .done(handle_button_return);
});
```

```
$("#key_6").click(function(e) {
        $.post("/command", {"cmd" : "KEY,6,P"})
        .done(handle_button_return);
});

$("#key_7").click(function(e) {
        $.post("/command", {"cmd" : "KEY,7,P"})
        .done(handle_button_return);
});

$("#key_8").click(function(e) {
        $.post("/command", {"cmd" : "KEY,8,P"})
        .done(handle_button_return);
});

$("#key_9").click(function(e) {
        $.post("/command", {"cmd" : "KEY,9,P"})
        .done(handle_button_return);
});

$("#key_0").click(function(e) {
        $.post("/command", {"cmd" : "KEY,0,P"})
        .done(handle_button_return);
});

$("#key_dot").click(function(e) {
        $.post("/command", {"cmd" : "KEY,.,P"})
        .done(handle_button_return);
});

$("#key_E").click(function(e) {
        $.post("/command", {"cmd" : "KEY,E,P"})
        .done(handle_button_return);
})

$("#key_SQ_press").click(function(e) {
        if (long_press) {
                cmd = "KEY,Q,L";
        } else {
                cmd = "KEY,Q,P";
        }
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return)
});

$("#key_vol_press").click(function(e) {
        $.post("/command", {"cmd" : "KEY,V,P"})
        .done(handle_button_return);
})

$("#key_SCAN").click(function(e) {
        $.post("/command", {"cmd" : "KEY,S,P"})
        .done(handle_button_return);
})
```

```
   $("#key_HOLD").click(function(e) {
               if (long_press) {
                       cmd = "KEY,H,L";
               } else {
                       cmd = "KEY,H,P";
               }
               $.post("/command", {"cmd" : cmd})
               .done(handle_button_return)
       });

    $("#key_function").click(function(e) {
        if (long_press) {
            cmd = "KEY,F,L";
        } else {
            cmd = "KEY,F,P";
        }
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return)
    });

    $("#key_VFO_RIGHT").click(function(e) {
        cmd = "KEY,>,P";
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return)
    });

    $("#key_VFO_LEFT").click(function(e) {
        cmd = "KEY,<,P";
        $.post("/command", {"cmd" : cmd})
        .done(handle_button_return)
    });


        //http://stackoverflow.com/a/23395136
    function beep() {
   var snd = new
Audio("data:audio/wav;base64,//uQRAAAAWMSLwUIYAAsYkXgoQwAEaYLWfkWgAI0wWs/ItAA
AGDgYtAgAyN+QWaAAihwMWm4G8QQRDiMcCBcH3Cc+CDv/7xA4Tvh9Rz/y8QADBwMWgQAZG/ILNAAR
Q4GLTcDeIIIhxGOBAuD7hOfBB3/94gcJ3w+o5/5eIAIAAAVwWgQAVQ2ORaIQwEMAJiDg95G4nQL7m
QVWI6GwRcfsZAcsKkJvxgxEjzFUgfHoSQ9Qq7KNwqHwuB13MA4a1q/DmBrHgPcmjiGoh//EwC5nGP
EmS4RcfkVKOhJf+WOgoxJclFz3kgn//dBA+ya1GhurNn8zb//9NNutNuhz31f////9vt///z+IdAE
AAAK4LQIAKobHItEIYCGAExBwe8jcToF9zIKrEdDYIuP2MgOWFSE34wYiR5iqQPj0JIeoVdlG4VD4
XA67mAcNa1fhzA1jwHuTRxDUQ//iYBczjHiTJcIuPyKlHQkv/LHQUYkuSi57yQT//uggfZNajQ3Vm
z+Zt//+mm3Wm3Q576v////+32///5/EOgAAADVghQAAAAA//uQZAUAB1WI0PZugAAAAAoQwAAAEk3
nRd2qAAAAACiDgAAAAAAABCqEEQRLCgwpBGMlJkIz8jKhGvj4k6jzRnqasNKIeoh5gI7BJaC1A1Ao
NBjJgbyApVS4IDlZgDU5WUAxEKDNmmALHzZp0Fkz1FMTmGFl1FMEyodIavcCAUHDWrKAIA4aa2oCg
ILEBupZgHvAhEBcZ6joQBxS76AgccrFlczBvKLC0QI2cBoCFvfTDAo7eoOQInqDPBtvrDEZBNYN5x
wNwxQRfw8ZQ5wQVLvO8OYU+mHvFLlDh05Mdg7BT6YrRPpCBznMB2r//xKJjyyOh+cImr2/4doscwD
6neZjuZR4AgAABYAAAABy1xcdQtxYBYYZdifkUDgzzXaXn98Z0oi9ILU5mBjFANmRwlVJ3/6jYDAm
xaiDG3/6xjQQCCKkRb/6kg/wW+kSJ5//rLobkLSiKmqP/0ikJuDaSaSf/6JiLYLEYnW/+kXg1WRVJ
L/9EmQ1YZIsv/6Qzwy5qk7/+tEU0nkls3/zIUMPKNX/6yZLf+kFgAfgGyLFAUwY//uQZAUABcd5Ui
NPVXAAAApAAAAAE0VZQKw9ISAAACgAAAAAVQIygIElVrFkBS+Jhi+EAuu+lKAkYUEIsmEAEoMeDmC
ETMvfSHTGkF5RWH7kz/ESHWPAq/kcCRhqBtMdokPdM7vil7RG98A2sc7zO6ZvTdM7pmOUAZTnJW+N
Xxqmd41dqJ6mLTXxrPpnV8avaIf5SvL7pndPvPpndJR9Kuu8fePvuiuhorgWjp7Mf/PRjxcFCPDkW
31srioCExivv9lcwKEaHsf/7ow2Fl1T/9RkXgEhYElAoCLFtMArxwivDJJ+bR1HTKJdlEoTELCIqg
EwVGSQ+hIm0NbK8WXcTEI0UPoa2NbG4y2K00JEWbZavJXkYaqo9CRHS55FcZTjKEk3NKoCYUnSQ0r
WxrZbFKbKIhOKPZe1cJKzZSaQrIyULHDZmV5K4xySsDRKWOruanGtjLJXFEmwaIbDLX0hIPBUQPVF
```

```
VkQkDoUNfSoDgQGKPekoxeGzA4DUvnn4bxzcZrtJyipKfPNy5w+9lnXwgqsiyHNeSVpemw4bWb9ps
Yeq//uQZBoABQt4yMVxYAIAAAkQoAAAHvYpL5m6AAgAACXDAAAAD59jblTirQe9upFsmZbpMudy7L
z1X1DYsxOOSWpfPqNX2WqktK0DMvuGwlbNj44TleLPQ+Gsfb+GOWOKJoIrWb3cIMeeON6lz2umTqM
XV8Mj30yWPpjoSa9ujK8SyeJP5y5mOW1D6hvLepeveEAEDo0mgCRClOEgANv3B9a6fikgUSu/DmAM
ATrGx7nng5p5iimPNZsfQLYB2sDLIkzRKZOHGAaUyDcpFBSLG9MCQALgAIgQs2YunOszLSAyQYPVC
2YdGGeHD2dTdJk1pAHGAWDjnkcLKFymS3RQZTInzySoBwMG0QueC3gMsCEYxUqlrcxK6k1LQQcsmy
YeQPdC2YfuGPASCBkcVMQQqpVJshui1tkXQJQV0OXGAZMXSOEEBRirXbVRQW7ugq7IM7rPWSZyDlM
3IuNEkxzCOJ0ny2ThNkyRai1b6ev//3dzNGzNb//4uAvHT5sURcZCFcuKLhOFs8mLAAEAt4UWAAIA
BAAAAAB4qbHo0tIjVkUU//uQZAwABfSFz3ZqQAAAAAngwAAAE1HjMp2qAAAAACZDgAAAD5UkTE1Ug
ZEUExqYynN1qZvqIOREEFmBcJQkwdxiFtw0qEOkGYfRDifBui9MQg4QAHAqWtAWHoCxu1Yf4VfWLP
IM2mHDFsbQEVGwyqQoQcwnfHeIkNt9YnkiaS1oizycqJrx4KOQjahZxWbcZgztj2c49nKmkId44S7
1j0c8eV9yDK6uPRzx5X18eDvjvQ6yKo9ZSS6l//8elePK/Lf//IInrOF/FvDoADYAGBMGb7FtErm5
MXMlmPAJQVgWta7Zx2go+8xJ0UiCb8LHHdftWyLJE0QIAIsI+UbXu67dZMjmgDGCGl1H+vpF4NSDc
kSIkk7Vd+sxEhBQMRU8j/12UIRhzSaUdQ+rQU5kGeFxm+hb1oh6pWWmv3uvmReDl0UnvtapVaIzo1
jZbf/pD6ElLqSX+rUmOQNpJFa/r+sa4e/pBlAABoAAAAA3CUgShLdGIxsY7AUABPRrgCABdDuQ5GC
7DqPQCgbbJUAoRSUj+NIEig0YfyWUho1VBBBA//uQZB4ABZx5zfMakeAAAAmwAAAAF5F3P0w9GtAA
ACfAAAAAwLhMDmAYWMgVEG1U0FIGCBgXBXAtfMH10000EEEEEECUBYln03TTTdNBDZopopYvrTTdN
a325mImNg3TTPV9q3pmY0xoO6bv3r00y+IDGid/9aaaZTGMuj9mpu9Mpio1dXrr5HERTZSmqU36A3
CumzN/9Robv/Xx4v9ijkSRSNLQhAWumap82WRSBUqXStV/YcS+XVLnSS+WLDroqArFkMEsAS+eWmr
UzrO0oEmE40RlMZ5+ODIkAyKAGUwZ3mVKmcamcJnMW26MRPgUw6j+LkhyHGVGYjSUUKNpuJUQoOIA
yDvEyG8S5yfK6dhZc0Tx1KI/gviKL6qvvFs1+bWtaz58uUNnryq6kt5RzOCkPWlVqVX2a/EEBUdU1
KrXLf40GoiiFXK///qpoiDXrOgqDR38JB0bw7SoL+ZB9o1RCkQjQ2CBYZKd/+VJxZRRZlqSkKiws0
WFxUyCwsKiMy7hUVFhIaCrNQsKkTIsLivwKKigsj8XYlwt/WKi2N4d//uQRCSAAjURNIHpMZBGYia
QPSYyAAABLAAAAAAAACWAAAAApUF/Mg+0aohSIRobBAsMlO//Kk4soosy1JSFRYWaLC4qZBYWFRGZ
dwqKiwkNBVmoWFSJkWFxX4FFRQWR+LsS4W/rFRb//////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////VEFHAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU291bmRib3kuZGUAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAwNGh0dHA6Ly93d3cuc291bmRib3kuZGUA
AAAAAAAACU=");
            snd.play();
    }
});
```