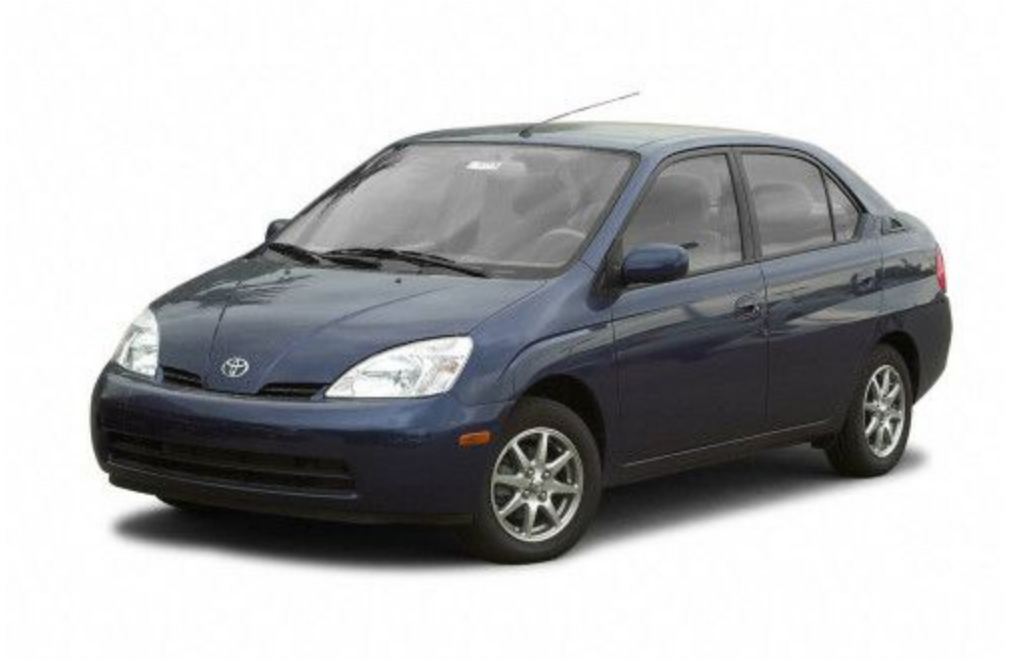
The Statistic

Raspberry Pi Media Player for Older/Outdated Cars



Saraf Ray CS 121 5/04/18

Final Report

Table of Contents

Introduction

For this project I made a Raspberry Pi car music player. It runs on the official Raspberry Pi touch screen and is coded using Python and Kivy, an open source Python library.

I wanted to create this project because of its functionality. It is mounted in my roommate's car (as I don't drive), which I spend a lot of time in. Often times, when outside of Burlington, we lose phone signal, so music services like Spotify or Soundcloud won't work. Other newer cars have music players built in, so adding a music player to a 2002 Prius is both a functional and aesthetic upgrade.

Adding music to the Pi is very simple. The code directs the program to look at a certain directory (/home/pi/Desktop/RPi Music) where the music files are located. To add music, one can simply use a program such as FileZilla to move downloaded files from a laptop or desktop to the correct location on the Pi.

Cost Analysis

Projected Costs

Tontec Touch Screen - $64
OR
Kuman Touch Screen - $54

Power Supply - $10

Cassette Adaptor - $20

USB Stick - $5

microSD Card - $15

Total - $114 with Tontec Touch Screen or $104 with Kuman Touch Screen

Overall Cost Estimate (including miscellaneous fees and miscalculations): **$150**

Actual Costs

Raspberry Pi Touch Screen - $75

Raspberry Pi Touch Screen Case - $15

USB Microphone - $7

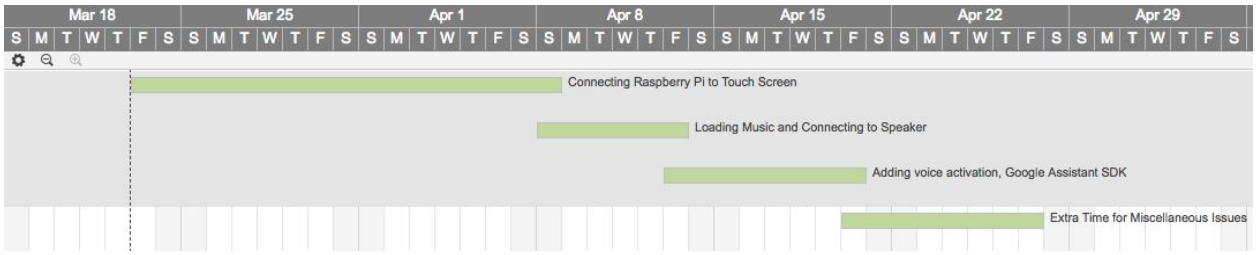
Cassette Adaptor - $20

Mounts - $13

Total Cost: $120

Time Analysis

Initial Gantt Chart:



My time projections were quite off, but that is partly due to the fact that I took a different approach to this project than I initially thought I would. When I decided to code the interface using Python and Kivy, as opposed to using a distribution such as XBian, I realized that this would lengthen this stage of the process. It took me a few days just to get a grasp of Kivy, before I could make actual progress on the project. This didn't give me time to add voice activation as I would have liked. However, using Python and Kivy was still a rewarding, if sometimes frustrating, experience.

Reflections

Doing this project was definitely a learning experience. One of the first steps, purchasing the touch screen, presented an important choice. There are many different touch screens compatible with the Raspberry Pi, all with their own advantages and disadvantages. I chose the Official Raspberry Pi touch screen due to its generally good reviews and the fact that it is the official screen made by the company. However, the screen is designed for use with Raspbian and is not compatible with all operating systems. I was unsure if my original plan to use XBian would still be feasible.

When I found the Kivy library, I knew immediately that I had made an important discovery. It seemed like an ideal way to create a music player interface using Python that would run on the touch screen I had purchased. Unfortunately, installing Kivy on my laptop and the Pi took way longer than it should have. This is because Kivy runs with Python 2.7, which was making the installation difficult. After a few hours of figuring that out, I was ready to go.

I spent the next few days watching Kivy tutorials and playing around with different layouts and interfaces. Once I felt I had a grasp of it, I asked Sam (my roommate and owner of car) what he wanted the screen to look like. One important thing he mentioned was the date, as it isn't shown anywhere in the car. I designed my own interfaces, which functioned fine, but I found a more aesthetically pleasing one on github. This interface, though, was not fully functional. I had to make changes, adding and removing some things, to make it what I wanted. Getting the play song function to resume from the paused point instead of starting the song over was especially annoying. In the end, I did not have time to add functional voice activation.

Mounting the touch screen in the car was a lot easier than I thought it would be. I purchased a case with built in holes for wall mount, and purchased metal mounts from Michael's. The Pi stands firmly, even when going over bumpy roads. It looks good in the car, if I am being honest. In addition, the adaptor that the Pi plugs into is both an adaptor and a power bank. It continues to power the Pi, even if unplugged from the car.

Final Code

MusicPlayerforPi.py

```
# If on Windows to avoid fullscreen, use the following two lines of code
from kivy.config import Config

Config.set('graphics', 'fullscreen', '0')

from kivy.app import App
from kivy.lang import Builder
from kivy.uix.popup import Popup
from kivy.uix.button import Button
from kivy.uix.widget import Widget
from kivy.core.audio import SoundLoader
from kivy.properties import ObjectProperty
from kivy.uix.gridlayout import GridLayout
from kivy.uix.floatlayout import FloatLayout

from os import listdir, path

Builder.load_string('''
#: kivy 1.10.0
#: import datetime datetime
<MusicPlayer>:

  canvas.before:
    Color:
      rgba: 0, 0, .1, 1
    Rectangle:
      pos: self.pos
      size: self.size

  Label:
    id: date
    text: datetime.datetime.now().strftime("%A %d %B %Y")
    size: 200,35
    background_color: 0,.5,1,1
    pos: root.width-200, root.top-35
```

```
ScrollView:
    size_hint: None, None
    size: root.width, root.height-135
    pos: 0, 100
    GridLayout:
        id: scroll
        cols: 1
        spacing: 10
        size_hint_y: None
        row_force_default: True
        row_default_height: 40

GridLayout:
    rows: 1
    pos: 0, 50
    size: root.width, 50
    Button:
        id: pause
        text: '||'
        background_color: 0,.5,1,1
        on_press: root.pauseSong()
    Button:
        id: play
        text: 'Play'
        background_color: 0,.5,1,1
        on_press: root.playSong(root.spot)
Button:
    id: nowplay
    text: 'Now Playing'
    pos: 0,0
    size: root.width, 50
    background_color: 0,.5,1,1

Label:
    id: status
    text: ''
    center: root.center
```

```
''')


class MusicPlayer(Widget):
    directory = "/home/pi/Desktop/RPi Music"  # location of songs folder
    nowPlaying = ''  # Song that is currently playing

    spot = None
    songs = []

    def getpath(self):
        f = open("sav.dat", "r")
        f.close()
        self.getSongs()

    def savepath(self, path):
        f = open("sav.dat", "w")
        f.write(path)
        f.close()

    def select(self, path):
        self.directory = path
        self.ids.direct.text = self.directory
        self.savepath(self.directory)
        self.getSongs()

    def pauseSong(self):
        if self.nowPlaying.state == 'play':
            spot = self.nowPlaying.get_pos()
            self.nowPlaying.stop()
            self.spot = spot

    def playSong(self, p):
        if self.nowPlaying.state == 'stop':
            self.nowPlaying.play()
            self.nowPlaying.seek(p)
```

```
def getSongs(self):

    songs = []  # List to hold songs from music directory
    self.directory = "/home/pi/Desktop/RPi Music"

    if self.directory == '':
        self.fileSelect()

    # To make sure that the directory ends with a '/'
    if not self.directory.endswith('/'):
        self.directory += '/'

    # Check if directory exists
    if not path.exists(self.directory):
        self.ids.status.text = 'Folder Not Found'
        self.ids.status.color = (1, 0, 0, 1)

    else:

        self.ids.status.text = ''

        self.ids.scroll.bind(minimum_height=self.ids.scroll.setter('height'))

        # get mp3 files from directory
        for fil in listdir(self.directory):
            if fil.endswith('.mp3'):
                songs.append(fil)

        # If there are no mp3 files in the chosen directory
        if songs == [] and self.directory != '':
            self.ids.status.text = 'No Music Found'
            self.ids.status.color = (1, 0, 0, 1)

        songs.sort()

        for song in songs:

            def playSong(bt):
                try:
```

```
                self.nowPlaying.stop()
            except:
                pass
            finally:
                self.nowPlaying = SoundLoader.load(self.directory + bt.text + '.mp3')
                self.nowPlaying.play()
                self.ids.nowplay.text = bt.text

        btn = Button(text=song[:-4], on_press=playSong)

        # Color Buttons Alternatively
        if songs.index(song) % 2 == 0:
            btn.background_color = (0, 0, 1, 1)
        else:
            btn.background_color = (0, 0, 2, 1)

        self.ids.scroll.add_widget(btn)  # Add btn to layout

    self.songs = songs



class MusicApp(App):

  def build(self):
    music = MusicPlayer()
    music.getpath()
    return music



if __name__ == "__main__":
  MusicApp().run()
```

Citations

Kivy library

https://kivy.org/#home

Base code for Music Player

https://github.com/JasonHinds13/KVMusicPlayer

YouTube Kivy Tutorials

https://www.youtube.com/playlist?list=PLGLfVvz_LVvTAZ-OcNIXe05srJRXaJRd9

Thank You!

A video of my project can be found here: https://www.youtube.com/watch?v=ZqPi7cyNUVQ