VINTAGE TELEVISION OPERATING SYSTEM

Contact: Andrew D. Marques

Website: www.AndrewDMarques.com

Email: _____
_____

Address: _____
_____

Phone: _____

Version: 8
Updated: 2/5/2024

INTRODUCTION

This manual serves as a comprehensive guide for using and maintaining the television set. While it covers the hardware build process, the primary focus is on the accompanying software, Vintage Television Operating System (VINTOS). VINTOS is an operating system designed to "reanimate" vintage tube television sets for use in modern applications.

For television sets that are beyond economical repair, VINTOS provides a viable solution to give them new life. It populates the shell of the vintage set to emulate any preferred television or film period. VINTOS aims to retain the original functionality of the unit for power, channel, volume, and brightness controls.

Each channel that the owner selects can be dedicated to specific types of content. For example, channel 1 may be dedicated to 1950's period sitcoms, channel 2 to the owner's private 8mm home film collection, channel 3 to 1940s movies, and so on. Alternatively, each channel could be set to specific content for a museum exhibit.

Operation Modes:

1. **Front Mode**: Control the television using original knobs located in the front of the cabinet.
2. **Internal Mode**: Control the television using the internal knobs while disabling the front knobs.
3. **Default Mode**: Disable all front and internal knobs. Only play content from the first channel by default with no knob controls to change channels, volume, or brightness.

For use in museums, a "curator override" system can be added to disable front knobs and provide control through an internal set of knobs. A "default" mode is also available to disable all knob inputs and play the content of channel 1 on repeat in case of issues with the front or internal knobs.

The cabinet used in this project is an RCA 6-T-76 with a 16" black and white TV and a maple cabinet built between 1949-1951. The updated set includes a Samsung S24D360HL monitor, JBL Platinum Series SP08A11 speakers, an Arduino UNO REV3 microcontroller board, and a Lenovo ThinkCentre M93p i5-4570T 2.90GHz 8GB RAM 250GB SSD main computer.

A video describing the finished product can be found at this link:
https://photos.app.goo.gl/2rV23QRDTUTvKgD26.

Together, this modified cabinet and VINTOS software provide an immersive viewing experience that simulates interactive vintage television.

TABLE OF CONTENTS

QUICKSTART GUIDE

Operating Steps:

1. Plug in the television.
2. Wait approximately 1 minute for the system to boot up.
3. Control of the TV depends on which mode is running:
    a. **Front mode** (front controls activated)
        i. Power the monitor, speaker, and indicator light with the front left-most knob.
        ii. Change screen brightness with the front's middle left knob.
        iii. Change volume with the front's middle right knob.
        iv. Change channels with the front's right-most knob.
    b. **Internal mode** (internal controls activated, except power)
        i. Power the monitor, speaker, and indicator light with the front left-most knob.
        ii. Change screen brightness with the internal top knob.
        iii. Change volume with the internal middle knob.
        iv. Change channels with the internal bottom knob.
    c. **Default mode** (limit control of knobs)
        i. Power the monitor, speaker, and indicator light with the front left-most knob.
        ii. Screen brightness will be set to max by default, no knob controls this.
        iii. Volume will be set to max by default, no knob controls this.
        iv. Channel will be set to play channel 1 content by default, no knob controls this.
4. To disable the front left-most knob from powering on the monitor, speaker, and indicator light, the included surge protector could be used. All devices could be connected to this, bypassing the GFCI outlets. When connected to the surge protector, please disconnect the brown vintage cable. This is recommended for units to be displayed in museums.

Change Modes:

- To activate **Front mode**:
    - Connect Arduino plug to Arduino socket.
    - Turn "internal control" switch to **OFF**.
    - Reboot the system by disconnecting and reconnecting the main power and power cycle the Lenovo computer.
- To activate **Internal mode**:
    - Connect Arduino plug to Arduino socket.
    - Turn "internal control" switch to **ON**.
    - Reboot the system by disconnecting and reconnecting the main power and power cycle the Lenovo computer.
- To activate **Default mode**:
    - Disconnect Arduino plug from Arduino socket.
    - Reboot the system by disconnecting and reconnecting the main power and power cycle the Lenovo computer.

Adding New Programs:

1. Adding new programs is easy, begin by powering on the system.
2. Use the included Keyboard/Trackpad.
3. Press the "Windows" key and search for "Files".
4. Insert a USB drive to the main computer that contains the video file (mp4 is preferred but most common video types are accepted).
5. Select the files from the flashdrive you would like too add.
6. Go to "/home/andrewdmarques/Desktop/TV/Channels" on the main computer and copy the selected files to the designated channel you would like to add them to.
7. Disconnect the flashdrive.
8. Generate a new set of program schedules.
    a. Go to "/home/andrewdmarques/Desktop/TV/Bin/Program-Schedule/" and delete ALL the contents of this folder. All files should have been CSV files.
    b. Disconnect and reconnect the main power supply to the cabinet.
    c. This will automatically generate a new list of schedules that now include the added program.

Moving the Cabinet:

1. Moving the cabinet should be done carefully -- first disconnect the main power.
2. Unplug and remove the following devices, it is recommended to leave the wires in the cabinet.
    a. Monitor
    b. Main computer
    c. Speakers
3. The Arduino board should remain in the cabinet. It is recommended to secure the board using tape to not strain the jumper wires.
4. Transport the cabinet in an upright position. You may want to secure the cabinet doors closed.
5. After transporting the TV, plug in the devices and place the components back in their locations. See the images in this document for where the devices belong.
    a. TIP: Make it so the bottom of the monitor is pressed as closely to the tan bakelite mount as possible.

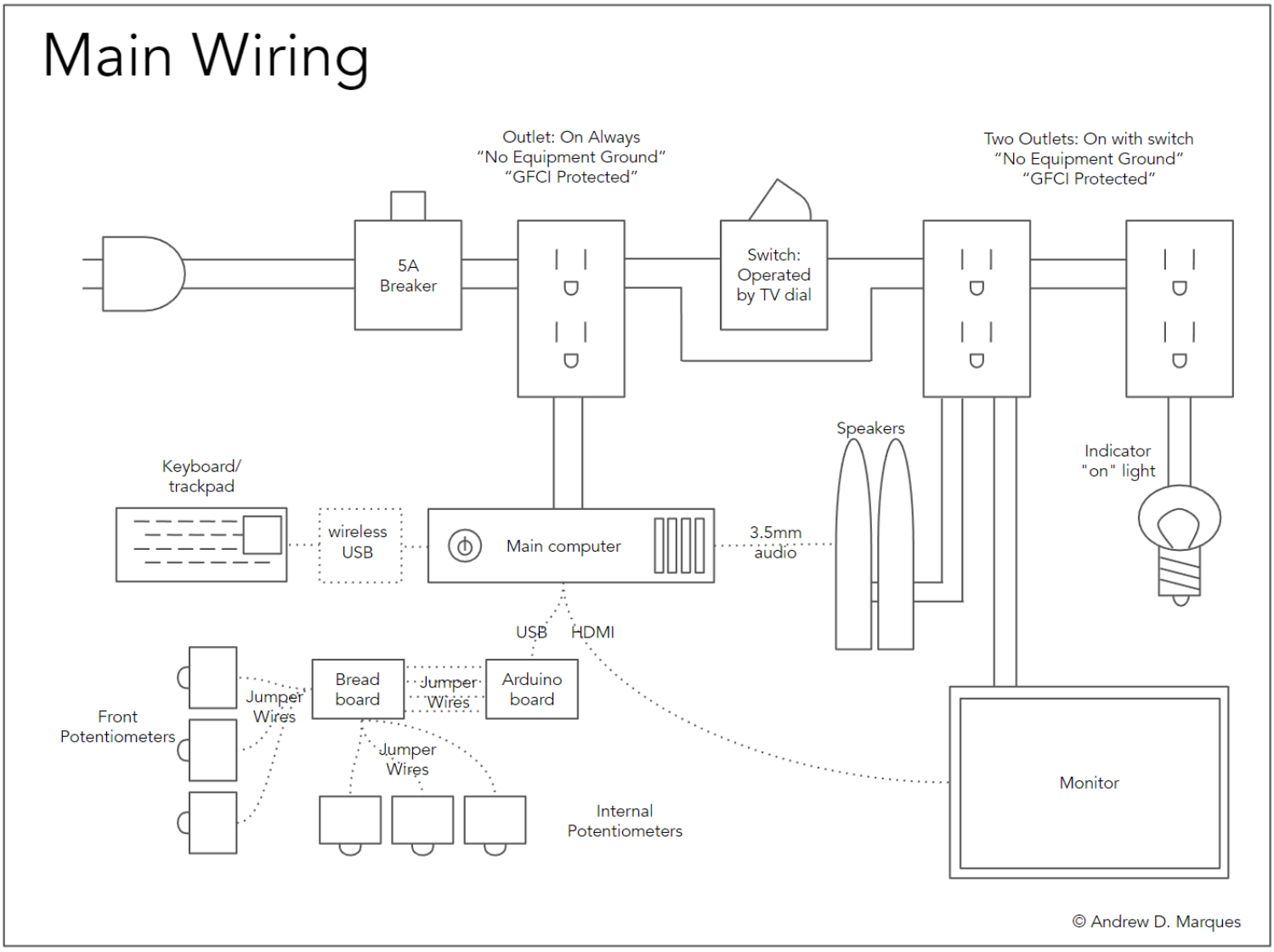
Long-Term Storage:

1. The cabinet should be stored in dry cool spaces without rodents. This cabinet should be thought of like a computer, so make sure that it is stored appropriately.
2. Check the integrity of the wiring connections.
3. The internal clock for the main computer may be reset if stored for more than a few weeks without power. You may have to manually input the time in the settings.
    a. Open the Activities overview (press "Windows" button and start typing Settings.
    b. Click on Settings.
    c. Click Date & Time in the sidebar to open the panel.
    d. If you have the Automatic Date & Time switch set to on, your date and time should update automatically if you have an internet connection. To update your date and time manually, set this to off.
    e. Click Date & Time, then adjust the time and date.
    f. You can change how the hour is displayed by selecting 24-hour or AM/PM for Time Format.

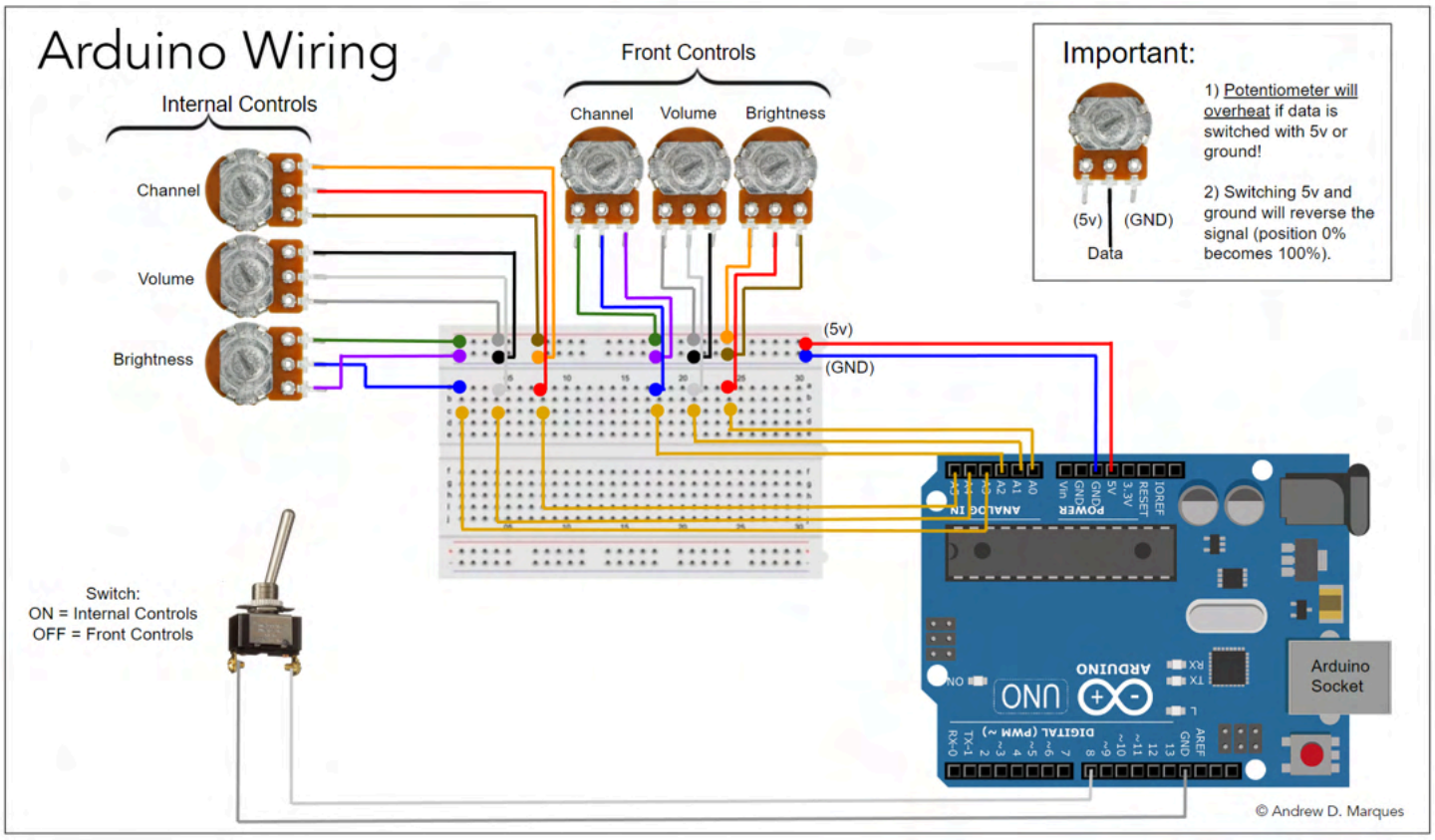
Fast Facts:

- **System reboot:** For system health, it is recommended to power on and off the main power supply (not just the front knob) at least 1x a day. A fail safe is built into the software which will reboot the computer once a day. This will occur several minutes before the 24 hour mark before the last time it was turned on.
- **System stopping/starting**: If you wish to stop the operating system and keep the computer running, while the system is rebooting press the "Windows" key and navigate to Desktop/TV/Bin/Scripts. At this location, delete "tv-on.txt". This will indicate to the system that it should stop immediately. If a video is already playing while this file is deleted, the video will continue to play. You can close the video's tab.
- **Main computer power:** The main computer power will remain on even after the front power knob is turned off. This is because it stays running in the background for a seemingly instantaneous "on" experience when the front left-most knob is turned.
- **On always outlets**: Two outlets will always have power sent to them, regardless of the state of the power knob.
- **On with switch outlets**: 4 outlets will have power only when the front left-most knob is set to power.
- **5A max**: Power consumption is limited to 5A maximum. This is controlled by a 5A circuit breaker.
- **Plugging in other devices to GFCI outlets**: Do not use the empty outlets for powering non-TV related devices. This may overdraw current and trip the breaker.
- **GFCI outlets:** GFCI outlets are used for non-grounded 3-pronged receptacles. This follows local and federal code. Electrical code 406.4(D)(2) Non-Grounding-Type Receptacles states that when replacing a non-grounding type receptacle with a GFCI type, the receptacle or the cover plate must be marked “No equipment ground”. It is recommended to test outlets 1x per month.
- **Changing file locations:** It is not recommended to change file locations from their current directories. If this must be done, then the operating system code will need to be modified to have the correct locations.
- **Channel directories:** The channel directories should be named with a "00_name" format, where "00" is a two-digit number and "name" can be anything without spaces or special characters. Non-video files should not be placed in these channel directories. All channels
- **Wifi:** The computer is configured to connect to the internet, but it requires a wifi dongle (USB connector). For internet security reasons, and to prevent issues with updating popups, this should be left disconnected. If a wifi dongle is not included in your package, you can purchase one. It is recommended to use the following dongle: TP-Link TL-WN725N N150 150Mbps Wireless Nano USB WiFi Network Adapter Dongle.

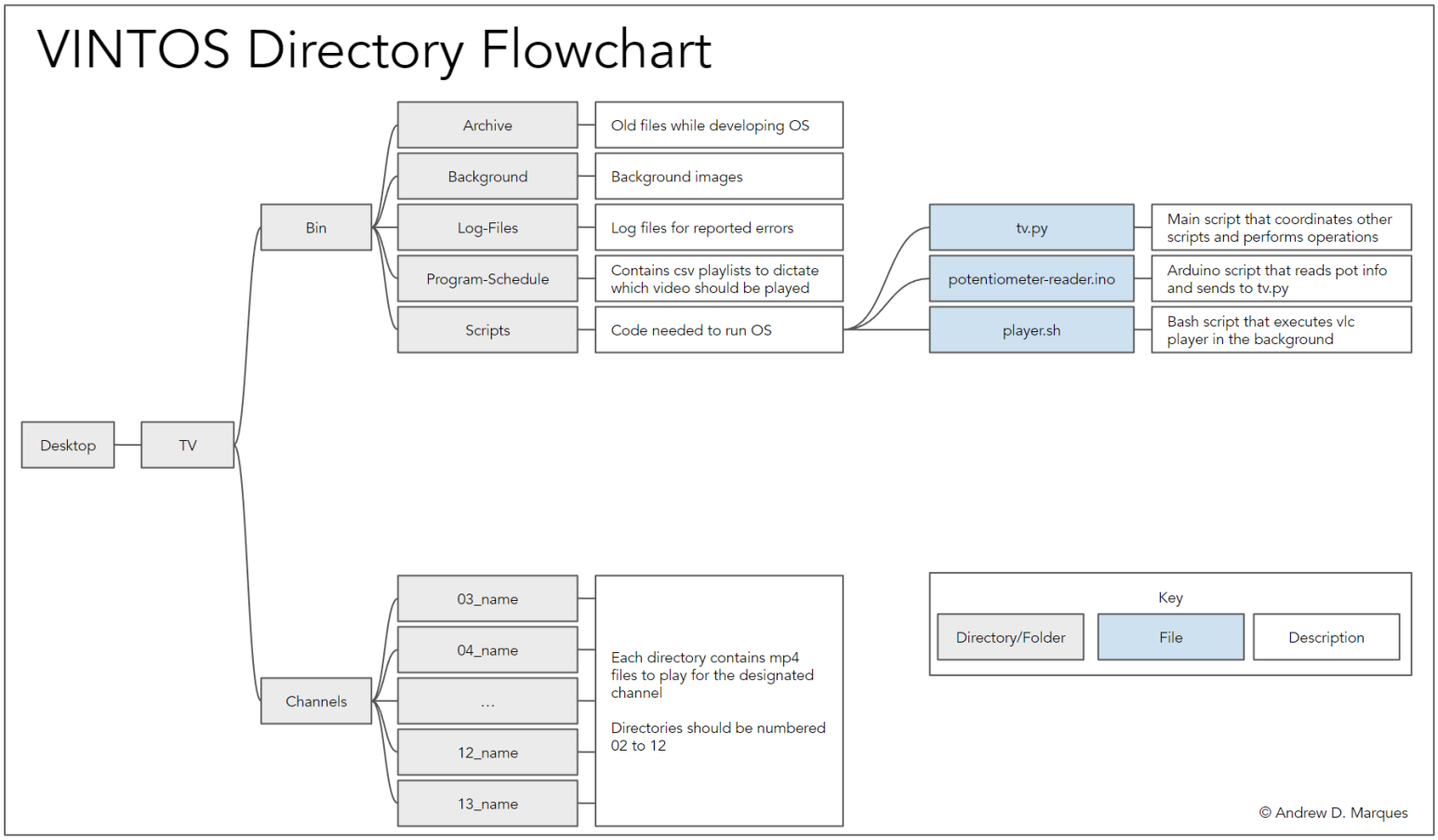
DIAGRAMS



The main electrical wiring diagram shown above can be used as a reference when troubleshooting if connections are not made to the appropriate devices.



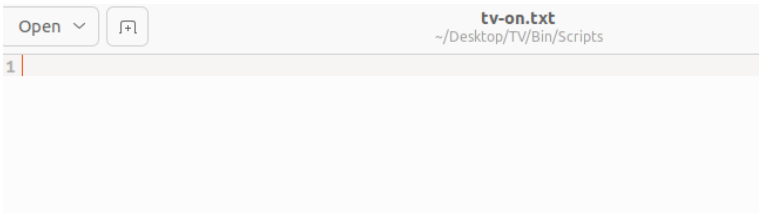
The Arduino wiring diagram indicates how the jumper wires should be connected to the breadboard and controls. Remember to disconnect power before touching any cables. Failure to follow the diagram may result in dysfunctional connections or destruction of potentiometers by overheating.



The VINTOS file structure is shown above, and all relevant files are located on the Desktop. To run VINTOS, the potentiometer-reader.ino file must be uploaded to the Arduino, but a reference copy is kept in Desktop/TV/Bin/Scripts.

For proper organization, all mp4 files should be placed in their respective channel directories. The channel directories should be named with a "00_name" format, where "00" is a two-digit number and "name" can be anything without spaces or special characters. Non-video files should not be placed in these channel directories.

File Descriptions:

- **tv-on.txt**: File that allows the tv script to run. Delete this file to prevent the script from playing. Make a file with this name to get it to play. The content of this file is not important, what matters is that there is a file with this name. File location: /home/andrewdmarques/Desktop/TV/Bin/Scripts/tv-on.txt
  - 

- **player-on.txt**: File that allows the player.sh script to run. This file is automatically made by the tv.py script to indicate to player.sh that there is a new video to play i.e. the channel has changed or the computer is booting up for the first time. The contents of this script are written as the time that it was written. It will be deleted by the player.sh script once the video has begun playing. File location: /home/andrewdmarques/Desktop/TV/Bin/Scripts/player-on.txt
- **tv.py**: The main script that controls volume, brightness, and indicates what video should be played. This coordinates the operation. When channel changes are prompted, it will write the temp.py script with the command to commence the vlc player, and generate the player-on.txt file to initiate the player.sh script to execute the command located in temp.py. File location: /home/andrewdmarques/Desktop/TV/Bin/Scripts/tv.py
  - 2023-01-25_tv_v1.31_get-running-on-thinkpad.py

```
#!/usr/bin/python3
# import the libraries.
from gpiozero import MCP3008   # To interpret the potentiometers when u
import serial                  # To interpret the potentiometers when u
import time                    # To allow for gaps in time before going
from datetime import datetime, timedelta  # To get time and make the tv
import os                      # To get the directories that are presen
import subprocess              # To determine the lengths of each of th
import pandas as pd            # To save the program schedules as csv f
import random                  # To randomize the programs when making
from pandas import *           # To read in the program schedule.
import math                    # To help with determining the channel f

#################################################################
#User input variables.
#####################v###########################################
dir_channels = '/home/andrewdmarques/Desktop/TV/Channels' # This is the
```

- **temp.py**: Script that contains the command to begin running the vlc player. It is created by the tv.py script and executed by the player.sh script. File location: /home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.py
  - temp.py

```
#!/usr/bin/python3
import os
os.system('DISPLAY=:0 vlc /home/andrewdmarques/Desktop/TV/Channels/06_C
```

- **player.sh**: An auxiliary script that waits for the cue to begin playing a new video. The queue to play a new video will be if player-on.txt is present. Once player.sh is activated, it will delete player-on.txt and run the command written in temp.py. File location: /home/andrewdmarques/Desktop/TV/Bin/Scripts/player.sh
  - Open | player.sh ~/Desktop/TV/Bin/Scripts | Save

```
#!/bin/bash
sleep 30
while :
do
        sleep 0.1
        FILE=/home/andrewdmarques/Desktop/TV/Bin/Scripts/player-on.txt
        if [ -f "$FILE" ]; then
                rm -r /home/andrewdmarques/Desktop/TV/Bin/Scripts/player-on.txt
                FILE=/home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.py
```

- **schedule_#.csv**: CSV file that is the program schedule. There are n number of program schedules and they are numbers schedule_0.csv, schedule_1.csv, etc. It includes information such as channel, the program on the channel, the file location of the program on that channel, the program length in seconds, the Location: /home/andrewdmarques/Desktop/TV/Bin/Program-Schedule/schedule_0.csv
  -

	A	B	C	D	E	F	G
1	prog_channel	prog	prog_dir	prog_len	prog_end	prog_after_start	prog_time
2	02_Tucker-Movie	Tucker.The.Man	/home/andrewdm	6630.666	6630.666	0	2023-01-14 09:34:51.930620
3	02_Tucker-Movie	Tucker.The.Man	/home/andrewdm	6630.666	13261.332	13261.332	2023-01-14 13:15:53.262620
4	02_Tucker-Movie	Tucker.The.Man	/home/andrewdm	6630.666	19891.998	19891.998	2023-01-14 18:47:25.260620
5	02_Tucker-Movie	Tucker.The.Man	/home/andrewdm	6630.666	26522.664	26522.664	2023-01-15 02:09:27.924620
6	02_Tucker-Movie	Tucker.The.Man	/home/andrewdm	6630.666	33153.33	33153.33	2023-01-15 11:22:01.254620

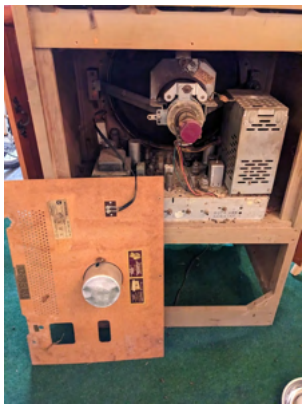
BUILD DOCUMENTATION

1/4/2023


The cabinet of the original television set showed signs of age and water damage. It appears the television cabinet sat in about 1.5 feet of water at some point of its life. There was also extensive damage to the coating on the top and front cabinets.


More images of the front of the console with the cabinet door open shows the surface damage present on the unit.





The first two images show the cabinet before and after being cleaned and re-varnished. This illustrates the condition of damage. The third image shows the condition of the original electronics. There was some minor evidence of water damage, but there was clear damage from rodents. Evidence of rust from urine,

and the presence of feces was found throughout the unenclosed locations. Rust did not appear on surfaces that could not be accessed by rodents.

1/10/2023



Above are the original electronics being removed. Some of these components will be cleaned and reused and others will be either disposed of responsibly, or made to be reused in other antique electronics.

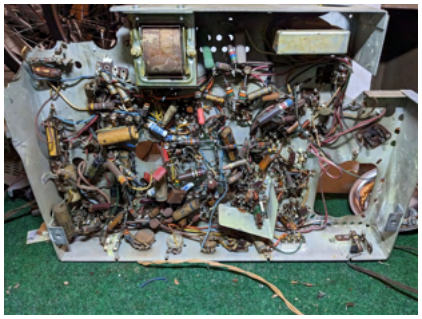
1/14/2023

The cathode-ray tube was physically damaged and did not appear to be salvageable. It was placed at the Philadelphia waste department site for televisions.

1/15/2023



The tan bakelite mount that the cathode-ray tube mounts to had to be cut to fit the monitor. While being cut, the material appeared to melt while being cut, rather than simply being cut.

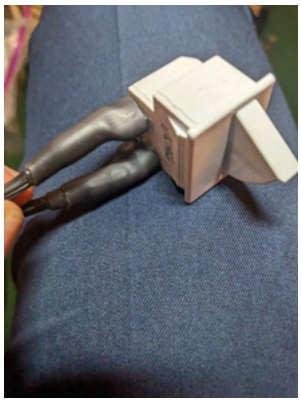



The board of this RCA television is attached to the potentiometer knobs. These must be cut from the main electronics board to maintain the position and sturdiness of the dial connections to the cabinet.


The cuts show the portion of the main board that is kept.

1/17/2023

A USB-powered LED light strip will be used as a light indicator. This will be the backlight for the channel dial. Several of the diodes would illuminate parts of the cabinet that are not behind the channel selector diode. These are simply covered with black electrical tape to limit light leaks.

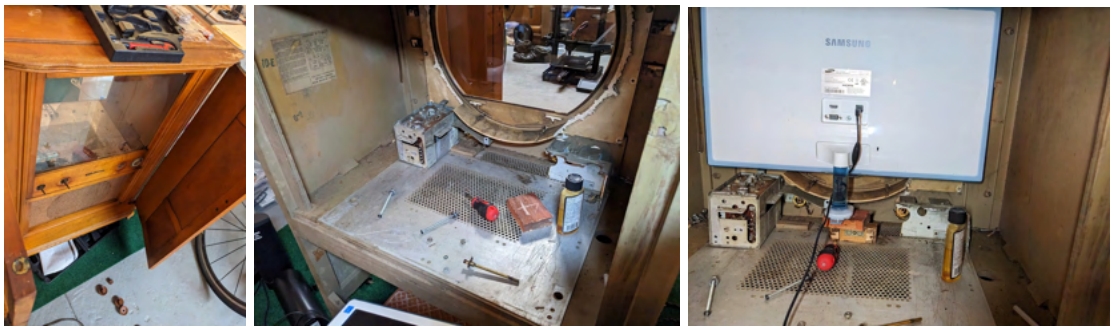


A fridge-switch is used in conjunction with the front left-most dial to cut power to the monitor and speakers. A 5A fuse breaker will act as a safety breaker to prevent passing current through the switch beyond what it is rated.

1/19/2023

The original dials are disassembled and prepared to be used with the modern potentiometers.

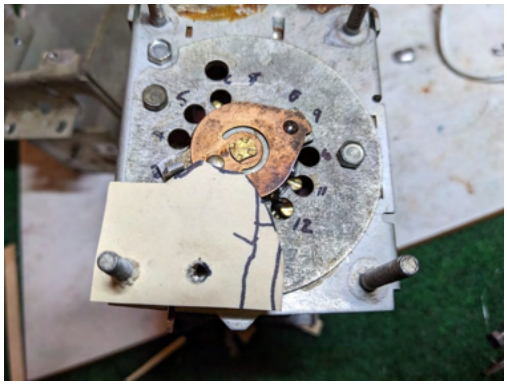
1/21/2023



After removing the cathode-ray tube, the bakelite mount, and the glass front, the components are cleaned and sized to mount the new monitor. New mounts and further cuts are made for the unit to fit together.

For the first time, the computer is turned on and booted up in the cabinet.

1/28/2023


A new potentiometer is placed in the control unit. A piece of scrap metal is used as a limiter to prevent the dial from spinning beyond the potentiometer's ability to interpret. This was first measured using the manilla paper for sizing before making metal cuts.

1/29/2023



The bakelite tv mount is measured for fit with the monitor and the case for the electrical equipment.

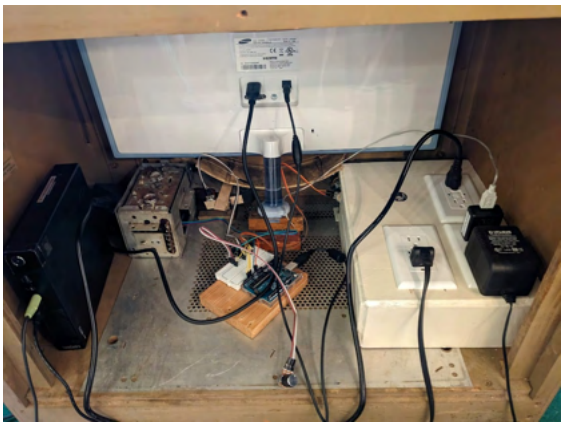
1/30/2023


The damaged wood is sanded and refinished. The results are better, but the damage is still clearly noticeable.

2/4/2023







A custom case is built from wood and painted to house the electrical equipment. For the first time, the electronics are connected and tested together.

2/8/2023

Improvements are made to the knob mounts. Additionally, the jumper wires are soldered to the potentiometers.

2/15/2023

Electrical wires are shown once connected to the outlets.

2/17/2023

Cable management is performed and the automatic fire extinguisher is installed.

2/24/2023

The internal knobs are connected to the main computer. The components and cables are labeled.
Completed back-view, with and without the cardboard backing.

Completed cabinet, front view before delivery (left image). VINTOS 2023 TV after installation in the AACA Hershey Museum as part of the Tucker Exhibit (right image). In the photo is Rob Kain (Director of Museum Advancement) and Andrew Marques (creator of VINTOS 2023 TV). Photo credit for the right image goes to Stanley Spiko (Museum Curator).

The television is on display in the Tucker Sales Center section of the AACA Hershey Museum 161 Museum Dr, Hershey, PA 17033.

Installation guide: For first time installing VINTOS

1. Using the command terminal, install the dependencies and prepare the files with the following commands:

```
sudo apt update && sudo apt upgrade
sudo apt install htop
sudo snap install mediainfo # For getting the duration of videos
sudo snap install vlc
sudo visudo
sudo apt install thonny
chmod +x player.sh
chmod +x tv.py
```

   a. username ALL=(ALL) NOPASSWD: /sbin/reboot # Add this line to the end of visudo

2. Install python dependencies.

```
sudo apt install python3-pip
pip3 install pyserial # For potentiometer reading on Arduino
pip3 install gpiozero # For potentiometer reading on raspberry pi
pip3 install pandas # For main script
```

3. Install arduino dependencies.
   a. https://linuxconfig.org/how-to-install-Arduino-ide-on-ubuntu-20-04-focal-fossa

```
sudo snap install arduino
```

4. Prepare scripts to be run at boot.

```
crontab -e
```

   a. At the end of the crontab file, add:

```
@reboot /home/andrewdmarques/Desktop/TV/Bin/Scripts/tv.py 1>
/home/andrewdmarques/Desktop/TV/Bin/Log-Files/log-root.txt 2>&1
@reboot /home/andrewdmarques/Desktop/TV/Bin/Scripts/player.sh
```

   b. Save by CTRL+X, "y", ENTER

5. Allow volume to be changed by the root user.
https://unix.stackexchange.com/questions/473769/sound-doesnt-work-properly-in-root-but-does-in-normal-user

```
sudo apt install pulseaudio
sudo su
mkdir /root/.config/pulse
cp -r /home/andrewdmarques/.config/pulse/* /root/.config/pulse/
mkdir /root/.config/autostart
cp /home/andrewdmarques/Desktop/TV/Bin/Archive/pulseaudio.desktop
/root/.config/autostart/
```

6. Prepare VLC player
   a. In Tools>Preferences>Interface
      1) Uncheck "Resize interface to video size"
      2) Uncheck "Show controls in full screen mode"
      3) Check "Start in minimal view mode"
      4) Change "Show media change popup" to never
      5) Check "Use only one instance when started from file manager"
   b. In Tools>Preferences>Subtitles/OSD then uncheck "Enable On Screen Display" and "Show media title on video start"
   c. Save the preferences before exiting
7. Enable screen brightness control
   a. when booting up ubuntu go to the log in page and select the user but before putting the password, click the gear settings icon on the bottom right then change this to xorg -- this will allow the xrandr operation to
8. Remove the automatic login prompt:
   a. https://linuxconfig.org/wp-content/uploads/2020/01/04-how-to-enable-automatic-login-on-ubuntu-20-04-focal-fossa.png
   b. https://linuxconfig.org/how-to-disable-keyring-popup-on-ubuntu#:~:text=The%20first%20option%20is%20you,master%20password%20for%20your%20keyring
   c. Set the screen to never sleep (settings>power>screen blank set to never)
9. Prepare the Arduino by uploading the potentiometer reader script. Follow popup window instructions from the Arduino software.

BUILD COSTS

Total: $433.35

- $100 RCA Victor TV cabinet
- $16.21 vintage electrical cable https://www.etsy.com/listing/181921564/custom-length-cord-w-plug-attached-26?click_key=cce03646b3c2b5d39c24897df7ff9ba14c7cc9e3%3A181921564&click_sum=40c915ac&ref=shop_home_feat_1&clickFromShopCard=1&frs=1&sts=1
- $23.32 keyboard and trackpad (Logitech K400R) https://www.ebay.com/itm/185724276957
- $23.06 Processor for interpreting potentiometers (Arduino Uno Rev3) https://www.ebay.com/itm/354493058859
- $11.21 Extension cables (rated for 13A at 125V) https://www.amazon.com/gp/product/B0B57V1992/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
- $8.47 Power switch (rated for 5A at 125V) https://www.amazon.com/gp/product/B08QMKX7M8/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1
- $50 Monitor (SAMSUNG SD360 Series Monitor 23.6")
- $74.19 Computer (Lenovo ThinkCentre M93p Tiny Desktop i5-4570T 2.90GHz 8GB RAM 250GB SSD) https://www.ebay.com/itm/225052512428
- $6 Potentiometers (6x items at $1 each) https://www.amazon.com/dp/B07DHKQVG5/ref=redir_mobile_desktop?_encoding=UTF8&ref_=yo_ii_img&th=1
- $2.80 Jumper cables (28x items at $0.10 each, $11.99 for 120) https://www.amazon.com/dp/B07GD1TH2K/ref=redir_mobile_desktop?_encoding=UTF8&ref_=ya_aw_od_pi&th=1
- $35.45 Speakers (JBL Platinum Series SP08A11) https://www.ebay.com/itm/155353679122
- $6.83 Display port to HDMI cable https://www.amazon.com/dp/B015OW3GJK?psc=1&ref=ppx_yo2ov_dt_b_product_details
- $8.43 One-gang device box for GFCI outlets (3x items at $3.04 each) https://www.amazon.com/gp/product/B00H8NUVQK/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1
- $33.49 GFCI outlets rated for 15A at 125V(3x items at $11.13 each) https://www.amazon.com/gp/product/B0B7FHMNK7/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&th=1
- $5.29 5A circuit breaker (selected because the power switch is rated for 5A at 125V) https://www.amazon.com/gp/product/B08RBLTT14/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&th=1
- $13.77 Black & Decker Timer with 7 day setting https://www.amazon.com/dp/B094PFJ5JW?psc=1&ref=ppx_yo2ov_dt_b_product_details
- $14.83 GE 6-outlet surge protector https://www.amazon.com/dp/B00DOMYL24?psc=1&ref=ppx_yo2ov_dt_b_product_details

OPERATING SYSTEM CODE

Python code: tv.py is the main program that coordinates other scripts.

```
#!/usr/bin/python3
# import the libraries.
from gpiozero import MCP3008   # To interpret the potentiometers when using a
raspberry pi
import serial                  # To interpret the potentiometers when using an arduino
import time                    # To allow for gaps in time before going through the
while loop again.
from datetime import datetime, timedelta  # To get time and make the tv program list
import os                      # To get the directories that are present
import subprocess              # To determine the lengths of each of the programs
import pandas as pd            # To save the program schedules as csv files. sudo
apt-get install python3-pandas
import random                  # To randomize the programs when making a list.
from pandas import *           # To read in the program schedule.
import math                    # To help with determining the channel for the
potentiometer.

############################################################
#User input variables.
######################v#######################################
dir_channels = '/home/andrewdmarques/Desktop/TV/Channels' # This is the location that
the data is stored.
dir_os = '/home/andrewdmarques/Desktop/TV/Bin' # Location of the bin, scripts, and
files needed to run the operating system.
dir_default = '/home/andrewdmarques/Desktop/TV/Channels/03_Tucker-Documentary-1/' #
This is the directory that, if the arduino is not detected, will then be played by
default.
time_delay_sensor = 0.05    # Time (seconds) to pause before checking the
potentiometer inputs.
time_delay_channel_input=2  # Time (seconds) to pause before committing to changing
the channel.
time_delay_channel = 2      # Time (seconds) to allow for transition before playing
the program after a channel change or a program ends.
time_bri_steps = 4          # Time (seconds) to incrementally brighten the screen
before a channel is displayed.
vol_max = 0.9               # Volume (proportion of max) to limit the volume to. If
potentiomter reads 100 and vol_max is 0.6, then it will be scaled to 60%.
bri_min = 0.0               # The minimum brightness (0-1).
board = 'ard'               # Indicate where the hardware set up has the potentiometer
being read by 'pi' for raspberry pi or 'ard' for arduino.
ard_path = '/dev/ttyACM0'   # The port that the arduino will communicate with. If a
reaspberry pi is used, then this will take any value.
tv_on_file = '/home/andrewdmarques/Desktop/TV/Bin/Scripts/tv-on.txt' # The file that,
if deleted, will allow the script to be stopped.
sched_count = 10            # Number of new schedules to generate that will be
randomly pulled from.
prog_min = 36               # Time (hours) that the minimum playlist length should be
made.
time_reboot = 23.9          # Time (hours) that the program will run before
automatically forcing a reboot. ***This should be at least several hours shorter than
prog_min
make_new_schedule = not bool(os.listdir(dir_os+'/Program-Schedule/'))  # Make new
schedules set as True or False depending on if the direcotry is empty.

time.sleep(20)
os.system('pulseaudio -D')

############################################################
# Define Functions
############################################################

# function for playing a video through player.sh.
def play_video(dir_os,pdc,pts,tt):
    # Begin playing the program.
    with open(dir_os + "/Scripts/temp.py", "w") as job_file:
        job_file.write("#!/usr/bin/python3\n")
        job_file.write("import os\n")
        #job_file.write("media = '"+prog_dir_curr[0]+"'\n")
        # Set the start time to play the video a little early.
        pts = max(pts-5,0) # sets the program start time to either 0 or 5 seconds
early
        job_file.write("os.system('DISPLAY=:0 vlc " + pdc + " --video-wallpaper
--qt-start-minimized --start-time="+str(pts)+"')")
    # Indicate to the player script that it should begin playing the program.
    with open(dir_os + "/Scripts/player-on.txt", "w") as job_file: # xhost + local: #
to give the permissions
        job_file.write(str(tt))
#play_video(dir_os,prog_dir_curr[0],prog_time_start[0],datetime.now())

# Function for getting potentiomter readings if using a Raspberry Pi.
if board == 'pi':
    def get_pot():
        numbers = [str(MCP3008(channel = 0)),str(MCP3008(channel =
1)),str(MCP3008(channel = 2))]
        pot0, pot1, pot2 = map(int,numbers)
        numbers2 = [pot0*100, pot1*100, pot2*100]
    # The arduino must be connected for this to continue.
    if os.path.exists(ard_path):
        # set up the serial connection to the Arduino
        ser = serial.Serial(ard_path, 9600)
        time.sleep(2) # wait for the Arduino to reset
        def get_pot():
            # send a command to the Arduino to read the potentiometer values
            ser.write(b'r')

            # read the response from the Arduino
            response = ser.readline().decode().strip()
            values = response.split(',')

            # check if we received the expected number of values
            if len(values) != 3:
                raise ValueError("Expected 3 values, but received
{}".format(len(values)))

            # normalize the values and return them as a list
            pots = [round(int(val) / 1023 * 100) for val in values]
            return pots
    else:
        print('arduino not connected')

# Function to determine the duration of the mp4 files.
def get_length(filename):
    time_out = os.popen("mediainfo --Inform='Video;%Duration/String3%' " +
filename).read()
    time_out1 = time_out.split('.', 1)[0]
    time_out_sec = sum(int(x) * 60 ** i for i, x in
enumerate(reversed(time_out1.split(':'))))
    time_out_sec = float(time_out_sec) + float('.'+time_out.split('.',
1)[1].strip())
    time_out_sec
    return float(time_out_sec)

# Function for generating the program schedule.
def get_schedule(time_start,dir_channels,dir_os,sn,prog_min,time_delay_channel):
    # Initialize the lists to record the scheduled programming.
    prog_channel = []           # The channel that the program is played on
    prog = []                   # The program file name
    prog_dir = []               # The program file and directory name
    prog_len = []               # The length of the program in seconds
    prog_end = []               # The time that a scheduled program should end
    prog_after_start = []       # The time since start that the program should
begin in seconds
    prog_time = []              # The time that the program should begin playing
    prog_min_sec = prog_min * 60 * 60 # 60 seconds in a minute and 60 minutes in
an hour
    # Get the channels and programs to play.
    channels = os.listdir(dir_channels)
    channels.sort()
    for xx in channels:
        print('Processing channel: '+xx)
        # Get the programs in the channel
        temp_prog = os.listdir(dir_channels+'/'+xx)
        random.shuffle(temp_prog) # This line randomizes the lists when making the
schedule for each program.
        # Record the channel that this program is located on.
        temp_prog_channel = []
        for yy in temp_prog:
            temp_prog_channel.append(xx)
        # Get all the programs for the channel.
        temp_prog_dir = []
        for yy in temp_prog:
            def dir_prog(yy): return dir_channels+'/'+xx+'/'+yy
            temp_prog_dir.append(dir_prog(yy))
        # Get the runtimes for all of the mp4 files.
        temp_prog_len = []
        for yy in temp_prog_dir:
            print('Processing program: '+yy)
            temp_prog_len.append(get_length(yy))
        # Determine when the scheduled program should end (seconds).
        temp_prog_end = []
        time_end = 0
        i = 0
        for yy in temp_prog_len:
            # Determine the time in seconds that a scheduled program should end.
            time_end = time_end + temp_prog_len[i]
            temp_prog_end.append(time_end)
            i += 1

        # Determine how many seconds after start to play the program.
        temp_prog_after_start = []
        first = True
```

```
        return numbers2

# Function for getting potentiomter readings if using an arduino.
if board == 'ard':
        for yy in temp_prog_len:
            #print('Processing program start time for: '+xx+' '+yy)
            # The first item should be 0 seconds after start and the other items would
be the previous value + previous runtime.
            if first:
                first = False
                temp_prog_after_start.append(0)
            else:
                temp_prog_after_start.append(temp_prog_end[i-1])
            i += 1

        # Get the time that the show should start.
        temp_prog_time = []
        i = 0
        for yy in temp_prog_len:
            temp_prog_time.append(time_start + timedelta(seconds =
temp_prog_after_start[i]))
            i += 1
        # Save all of the variables to their respective master list.
        i = 0
        for yy in temp_prog:
            prog_channel.append(temp_prog_channel[i])
            prog.append(temp_prog[i])
            prog_dir.append(temp_prog_dir[i])
            prog_len.append(temp_prog_len[i])
            prog_end.append(temp_prog_end[i])
            prog_after_start.append(temp_prog_after_start[i])
            prog_time.append(temp_prog_time[i])
            i += 1
        # While the program schedule is less than the minimum time limit, keep
appending the existing content until the time is met.
        temp_last_time = temp_prog_end[-1]
        while temp_last_time < prog_min_sec:
            i = 0
            for yy in temp_prog:
                prog_channel.append(temp_prog_channel[i])
                prog.append(temp_prog[i])
                prog_dir.append(temp_prog_dir[i])
                prog_len.append(temp_prog_len[i])
                prog_end.append(prog_end[-1]+temp_prog_len[i])
                prog_after_start.append(prog_end[-2])
                prog_time.append(prog_time[-1] + timedelta(seconds=prog_end[-1]))
                temp_last_time = prog_end[-1]+temp_prog_len[i]
                i += 1
    # Make a dictionary.
    dict1 =
{'prog_channel':prog_channel,'prog':prog,'prog_dir':prog_dir,'prog_len':prog_len,'prog
_end':prog_end,'prog_after_start':prog_after_start,'prog_time':prog_time}
    # Creating a dataframe.
    df = pd.DataFrame(dict1)
    # Conver dataframe to csv.
    data = df.to_csv(dir_os+'/Program-Schedule/schedule_'+str(sn)+'.csv', index =
False)

# Function to determine which channel is selected:
def get_channel(num):
    lookup_table = {
        (0, 3): 3,
        (3, 12): 4,
        (12, 23): 5,
        (23, 33): 6,
        (33, 44): 7,
        (44, 55): 8,
        (55, 65): 9,
        (65, 75): 10,
        (75, 85): 11,
        (85, 95): 12,
        (95, 100): 13
    }
    for key, value in lookup_table.items():
        if key[0] <= num <= key[1]:
            return value

################################################################
# Initialization
################################################################
# Get the current time
time_start = datetime.now()

# Make new schedules if directed.
if make_new_schedule == True:

    # Make a new directory if it doesn't already exist.
    isExist = os.path.exists(dir_os+'/Program-Schedule')
    if not isExist:
      os.makedirs(dir_os+'/Program-Schedule')
    sched_num = list(range(0,sched_count,1))
    for sn in sched_num:
        get_schedule(time_start,dir_channels,dir_os,sn,prog_min,time_delay_channel)

# Randomly select one of the premade program schedules to run.
# Determine which schedules are available.
sched_list = os.listdir(dir_os+'/Program-Schedule')
random.shuffle(sched_list) # Randomly shuffle the schedules, then we will pick the
first in this randomized list.
sched = sched_list[0]
        i = 0

# Open the randomly selected schedule.
df = pd.read_csv(dir_os+'/Program-Schedule/'+sched)
data = read_csv(dir_os+'/Program-Schedule/'+sched)
prog_channel = data['prog_channel'].tolist()      # The channel that the program
is played on
prog = data['prog'].tolist()                      # The program file name
prog_dir = data['prog_dir'].tolist()              # The program file and directory
name
prog_len = data['prog_len'].tolist()              # The length of the program in
seconds
prog_end = data['prog_end'].tolist()              # The time that a scheduled
program should end
prog_after_start = data['prog_after_start'].tolist()      # The time since start
that the program should begin in seconds
prog_time = data['prog_time'].tolist()

x = 1
prog_time_end = [100] # Sets the scheduled end time for the program to a dummy
starting value, this will become the expected time (seconds) that the program is
scheduled to end.
prog_time_end_hold = 100
# Get the channels and programs to play.
channels = os.listdir(dir_channels)

# tv-on.txt must be present for the for loop to work. This is one of the fail
safes -- delete this
tv_on = os.path.exists(tv_on_file)
if tv_on == False:
    print('TV program set to stop because tv-on.txt file is not present at
location ' + str(tv_on_file))

# If the arduino is disconnected, then move to defauly mode.
if board == 'ard':
    if not os.path.exists(ard_path):
        tv_on = False
        print('TV program set to default mode because arduino is disconnected')

# Determine the starting channel (this will be labeled the "previous channel").
if tv_on == True:
    pots = get_pot()
    pot2 = pots[2]
    print('retrieved pot 2 before running script')
    channel_prev = 0 # Set the previous channel to 0 so that it will initialize
the first time through the main loop.

# Refresh the current time.
time_start = datetime.now()

################################################################
# Main script
################################################################
print('Executing main script')

while True == tv_on: # If tv_on is false, then tv-on.txt is not present and the
script should not be executed.
    # Wait the specified amount of time between taking potentiometer readings.
    time.sleep(time_delay_sensor)
    # tv-on.txt must be present for the for loop to work. This is one of the fail
safes -- delete this
    tv_on = os.path.exists(tv_on_file)
    if tv_on == False:
        print('TV program set to stop because tv-on.txt file is not present at
location ' + str(tv_on_file))
    # Get the current programming time (seconds).
    time_sec = datetime.now() - time_start
    time_sec = time_sec.total_seconds()
    # For computer health, reboot the television at least once a day
    if(time_sec > (time_reboot*60*60)): # This will reboot about 23 hours and 59
minutes into the running session (time in seconds calulcated here)
        os.system('sudo reboot')
    # Get the potentiometers' readings.
    print('\n==================Potentiometer Reading==================')
    pots = get_pot()
    pot0 = pots[0]
    print('Pot 0:',round(pot0,2))
    pot1 = pots[1]
    print('Pot 1:',round(pot1,2))
    pot2 = pots[2]
    print('Pot 2:',round(pot2,2))
    x += 1
    # Make brightness adjustments.
    bri = round(pot0/100,2)
    bri_command = 'DISPLAY=:0 xrandr --output HDMI-1 --brightness '+str(bri)
    os.system(bri_command)
    # Make pactl -- set-sink-pactl set-sink-volume @DEFAULT_SINK@ 1  adjustments.
    vol = round(pot1*vol_max,0)
    vol_command = 'pactl set-sink-volume @DEFAULT_SINK@ ' + str(int(vol)) + '%'
    os.system(vol_command)
    # Determine which channel is currently selected.
    channel_curr = get_channel(pot2) # There are 11 channels, so which
potentiometer reading is most close to a channel. This will give values 2 to 12
    # Determine if it is time for the next program on the same channel.
    next_prog = False
    if((time_sec+0.2) > prog_time_end_hold): # If it is about time for the
scheduled program to end, then begin playing the next program.
        #if((time_sec) > prog_time_end[0]): # If it is about time for the
scheduled program to end, then begin playing the next program.
        time.sleep(0.21)
```

```
# Make sure there is a location where the error logs can be written.
path = dir_os + "/Log-Files/"
if not os.path.exists(path):
    os.makedirs(path)

        time_sec = time_sec.total_seconds()
        next_prog = True
        channel_curr = 0 # To trigger the next program to play, set the channel to 0.
    # Determine if channel should be changed.
    if channel_curr != channel_prev:
        print('Channel changed from '+str(channel_prev)+' to '+str(channel_curr))
        # Set the brightness to 0.
        os.system('DISPLAY=:0 xrandr --output HDMI-1 --brightness '+str(bri_min))
        # Stop the current program.
        os.system('killall -9 vlc') # Stops the player from playing
        os.system('rm -r ' + dir_os + '/Scripts/temp.py')  # Prevents the player.py
from replaying the same movie.
        # If it is time to play the next program on the same channel, then reset to
the same channel.
        if next_prog == True:
            next_prog = False
            channel_curr = channel_prev
        # Check that the channel has not been changed for xxx seconds, this makes sure
that it does not skip to just the first channel
        change_time1 = datetime.now()
        change_time2 = datetime.now()
        time_sec2 = change_time2-change_time1
        while time_sec2.total_seconds() < time_delay_channel_input:
            time.sleep(0.1)
            pots = get_pot()
            pot2 = pots[2]
            channel_curr2 = get_channel(pot2)
            if(channel_curr2 != channel_curr): # If the channel has changed, then
reset the change_time1 to make the process start the timer again
                print(channel_curr)
                print(channel_curr2)
                channel_curr = channel_curr2
                change_time1 = datetime.now()
                print('Channel change detected again')
            change_time2 = datetime.now()
            time_sec2 = change_time2-change_time1
            print('Waiting to confirm channel')
        print('Channel confirmed')
        # Determine which channel directory corresponds with the inputted channel
number.
        temp = str(channel_curr).rjust(2,'0')
        prog_channel_curr = [match for match in channels if temp in match] # The input
channel will be a number, like 2, so this will find any directories that have 02 in
them, the first one that matches will be considered the channel to play.
        # Get the current programming time (seconds).
        time_sec = datetime.now() - time_start
        time_sec = time_sec.total_seconds()
        # For computer health, reboot the television at least once a day
        if(time_sec > (time_reboot*60*60)): # This will reboot about 23 hours and 59
minutes into the running session (time in seconds calulcated here)
            os.system('sudo reboot')
        # Determine which program should start.
        i = 0
        prog_i = 0
        play = False
        prog_dir_curr = []
        prog_time_start = []
        prog_time_end = []
        # Determine the list of programs that could be played (all programs from that
channel that are supposed to be aired before the current time (seconds)).
        print(prog_channel[i])
        print(time_sec)
        for yy in prog:
            if prog_channel_curr[0] == prog_channel[i]:
                #print('got channel')
                if(prog_after_start[i] <= time_sec): # Should the show have already
started?
                    #print('got start time')
                    if(prog_end[i] > time_sec):     # Should the program still be
playing?
                        #print('got end time')
                        prog_dir_curr.append(prog_dir[i]) # Save the directory for the
current program
                        prog_time_start.append(max(0,time_sec - prog_after_start[i]) -
(time_delay_channel*3))          # Save the time that the video should begin playing
from: This is adjusted by the time delay *3 so that it guarentees somethinhg plays
before the channel change finishes
                        prog_time_end.append(prog_end[i]) # Record the time that the
program is expected to end.
                        prog_i = i                          # Record which program
should be played.
                        prog_time_end_hold = prog_time_end[0] # This is in the loop to
ensure that the time a program should end is held constant and not overwritten as
blank so long as a channel end time has previosly been selected.
                        play = True # Indicate that a program has been found.
                        #print(i)
            i += 1
        # If there is not a program next, then play the static
        if play == False:

play_video(dir_os,'/home/andrewdmarques/Desktop/TV/Bin/Backgounds/static.mov',0,dateti
me.now())

            # Wait the specified time to allow VLC player to boot up the program.
        # Get the current programming time (seconds).
        time_sec = datetime.now() - time_start
                time_curr_channel1 = datetime.now() - time_curr_channel0
                time_curr_channel = time_curr_channel1.total_seconds()

                # Allow the volume to be changed while the brightness is ramping
up.
                pots = get_pot()
                pot1 = pots[1]
                # Make volume adjustments.
                vol = round(pot1*vol_max,0)
                vol_command = 'pactl set-sink-volume @DEFAULT_SINK@ ' +
str(int(vol)) + '%'
                print('In channel waiting loop')
                print(vol_command)
                os.system(vol_command)

            time.sleep(time_delay_channel)
            # Ramp up the brightness for the number of steps and rate.
            # Determine how many brightness steps there should be.
            bri_num_step = list(range(0,time_bri_steps*time_bri_rate+1))
            # Determine where the brightness should start and end at.
            bri_low = bri_min
            bri_high = max(bri,0.0001)
            # Determine how bright each step should be.
            bri_step = (bri_high - bri_low)/(time_bri_steps*time_bri_rate)
            # Determine how long to wait between steps.
            bri_wait = time_bri_steps/(time_bri_steps*time_bri_rate)
            for yy in bri_num_step:
                bri_step_command = 'DISPLAY=:0 xrandr --output HDMI-1 --brightness
'+str(bri_low + (bri_step*yy))
                #print(bri_step_command)
                os.system(bri_step_command)
                time.sleep(bri_wait)

                # Allow the volume to be changed while the brightness is ramping
up.
                pots = get_pot()
                pot0 = pots[0]
                print('Pot 0:',round(pot0,2))
                pot1 = pots[1]
                # Make volume adjustments.
                vol = round(pot1*vol_max,0)
                vol_command = 'pactl set-sink-volume @DEFAULT_SINK@ ' +
str(int(vol)) + '%'
                os.system(vol_command)
            play = False
        # Start the next program if there is one found.
        if play == True:
            play_video(dir_os,prog_dir_curr[0],prog_time_start[0],datetime.now())

            # Wait the specified time to allow VLC player to boot up the program.
            # Get the current time.
            time_curr_channel0 = datetime.now()
            time_curr_channel1 = datetime.now() - time_curr_channel0
            time_curr_channel = time_curr_channel1.total_seconds()
            while time_curr_channel < time_delay_channel: # While loop for
confirming the selected channel.
                time.sleep(0.1)
                # Check if enought time has passed.
                time_curr_channel1 = datetime.now() - time_curr_channel0
                time_curr_channel = time_curr_channel1.total_seconds()

                # Allow the volume to be changed.
                pots = get_pot()
                pot1 = pots[1]
                # Make volume adjustments.
                vol = round(pot1*vol_max,0)
                vol_command = 'pactl set-sink-volume @DEFAULT_SINK@ ' +
str(int(vol)) + '%'
                print('In channel waiting loop')
                print(vol_command)
                os.system(vol_command)

            # Ramp up the brightness for the number of steps and rate.
            tb = datetime.now()
            time_bri_1 = (datetime.now() - tb).total_seconds()
            bri_low = bri_min
            bri_high = max(bri,0.0001)
            while time_bri_1 < time_bri_steps: # While loop for ramping up
brightness.
                # Prepare for next loop.
                time.sleep(0.05)
                time_bri_1 = (datetime.now() - tb).total_seconds()

                # Determine the proportion of the ramp up time that has passed.
                bri_prop = time_bri_1/time_bri_steps
                # Calculate the current ramped up brightness base on time time in
this loop.
                pot0 = pots[0]
                bri_level = max(0.0001,((round(pot0/100,2))*bri_prop))
                bri_step_command = 'DISPLAY=:0 xrandr --output HDMI-1 --brightness
'+str(bri_level)
                #print(bri_step_command)
                os.system(bri_step_command)

                # Allow the volume to be changed while the brightness is ramping
up.
                pots = get_pot()
                pot1 = pots[1]
```

```
            # Get the current time.
            time_curr_channel0 = datetime.now()
            time_curr_channel1 = datetime.now() - time_curr_channel0
            time_curr_channel = time_curr_channel1.total_seconds()
            while time_curr_channel < time_delay_channel:
                time.sleep(0.1)
                # Check if enought time has passed.

                os.system(vol_command)

                # Check if the channel has been changed.
                pot2 = pots[2]
                channel_curr2 = get_channel(pot2)
                if(channel_curr2 != channel_curr):
                    time_bri_1 = time_bri_steps # This breaks the loop.
                    channel_curr = 0            # This initiates a channel change.
            play = False
    channel_prev = channel_curr



# Move to default mode if an arduino is indicated as connected but it is disconnected.
if board == 'ard':
    if not os.path.exists(ard_path):
        video_extensions = ['.mp4', '.avi', '.mov', '.wmv', '.flv', '.webm', '.mkv']
        # Set default brightness and volume to max.
        os.system('DISPLAY=:0 xrandr --output HDMI-1 --brightness 1')
        os.system('pactl set-sink-volume @DEFAULT_SINK@ ' + str(int(vol_max*100)) +
'%')
        tv_on = os.path.exists(tv_on_file)
        while True == tv_on: # While the tv-on.txt file is present, continue playing
the video.
            tv_on = os.path.exists(tv_on_file)
            time.sleep(0.1)
            # Check that vlc player is not running already.
            for filename in os.listdir(dir_default):
                # Determine the filepath to the video that should be played.
                filepath = dir_default+filename
                print(filepath)
                # Determine if the file is a video that should be played.
                ext = os.path.splitext(filename)[1].lower()
                if ext in video_extensions:
```

```
                # Make volume adjustments.
                vol = round(pot1*vol_max,0)
                vol_command = 'pactl set-sink-volume @DEFAULT_SINK@ ' +
str(int(vol)) + '%'
                print('In ramp up loop')
                print(vol_command)
                    # Determine how long the video is.
                    default_time = get_length(filepath)
                    dt = datetime.now()
                    default_curr = (datetime.now() - dt).total_seconds()
                    time.sleep(1)
                    # Stop any ongoing videos.
                    os.system('killall -9 vlc')
                    time.sleep(2)
                    # Initiate the video to be played.
                    play_video(dir_os,filepath,0,datetime.now())
                    # Wait while the video is being played.
                    while default_curr < default_time:
                        time.sleep(1)
                        default_curr = (datetime.now() - dt).total_seconds()
                        print('vlc playing')
                        # Check that the script should continue running.
                        tv_on = os.path.exists(tv_on_file)
                        if tv_on == False:
                            break
                            # Get the current programming time (seconds).
                        time_sec = datetime.now() - time_start
                        time_sec = time_sec.total_seconds()
                        # For computer health, reboot the television at least once
a day
                        if(time_sec > (time_reboot*60*60)): # This will reboot
about 23 hours and 59 minutes into the running session (time in seconds calulcated
here)
                            os.system('sudo reboot')
                    if tv_on == False:
                        break
    os.system('killall -9 vlc')
```

Bash code: **player.sh** is an auxiliary script that plays the selected video.

```
#!/bin/bash
sleep 1
while :
do
    sleep 0.1
    FILE=/home/andrewdmarques/Desktop/TV/Bin/Scripts/player-on.txt
    if [ -f "$FILE" ]; then
            rm -r /home/andrewdmarques/Desktop/TV/Bin/Scripts/player-on.txt
            FILE=/home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.py
            if [ -f "$FILE" ]; then
                    chmod +x /home/andrewdmarques/Desktop/TV/Bin/Scrripts/temp.py
                    # export DISPLAY=HDMI-1
                    python3 /home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.py > /home/andrewdmarques/Desktop/TV/Bin/Log-Files/bash_log.text
2>&1
                    # su -c /home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.sh andrewdmarques >
/home/andrewdmarques/Desktop/TV/bash_log.text 2>&1
                    #su -c /home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.sh andrewdmarques
                    # runuser -l andrewdmarques -c '/home/andrewdmarques/Desktop/TV/Bin/Scripts/temp.sh' >
/home/andrewdmarques/Desktop/TV/Bin/Scripts/bash_log.text 2>&1
            fi
    fi
done
```

Arduino code: **potentiometer-reader.ide.ino** is run on the arduino and sends the potentiometer data to the main computer.

```
// Set the pin for the toggle switch
int button = 8;

void setup() {
  // set up the serial connection
  Serial.begin(9600);

  // set the pin modes for the potentiometers
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(A4, INPUT);
```

```
  pinMode(A5, INPUT);

  // set the toggle switch
  pinMode(button,INPUT_PULLUP); // Unpressed button is high
}

void loop() {
  if (Serial.available() > 0) {
            // read the incoming command
            char command = Serial.read();

            if (command == 'r') {
            if(digitalRead(button) == true) {
            // read the values from the potentiometers using the front values.
            int val1 = analogRead(A0);
            int val2 = analogRead(A1);
            int val3 = analogRead(A2);

            // send the values back to the Linux computer
            Serial.print(val1);
            Serial.print(",");
            Serial.print(val2);
            Serial.print(",");
            Serial.println(val3);
            }
            if(digitalRead(button) == false) {
            // read the values from the potentiometers using the rear values.
            int val1 = analogRead(A3);
            int val2 = analogRead(A4);
            int val3 = analogRead(A5);

            // send the values back to the Linux computer
            Serial.print(val1);
            Serial.print(",");
            Serial.print(val2);
            Serial.print(",");
            Serial.println(val3);
            }
            }
  }
}
```

TROUBLESHOOTING: GENERAL

Your TV is not working correctly? Start with this checklist!

1. Reboot the system by disconnecting and reconnecting the main power supply to the unit.
2. Check that the 5A circuit breaker is not tripped. It should not require force to press it from the up to down position. Use caution, if the breaker is tripped, check that there are no shorts and there are no non-TV related items plugged into the GFCI outlets.
3. Check the GFCI outlets are reset. All outlets should have a solid green light to indicate that they have power. There are 3x outlets to check.
4. Check that all cables are fully connected. Cables that are most likely disconnected include:
    a. Jumper wires (rainbow cables) connected to the breadboard.
    b. Audio and power cables to the speaker.
    c. Power, display, and audio cables to the main computer.
    d. Power and display cables to the monitor.
5. Check that the system has power cycled to it at least 1x time per day. This is best done with an outlet timer that will cut power to the cabinet without human intervention.
6. Check that the main computer is powered on with its power button. DO NOT use the main computer power button to cycle power. Once it is pressed on, it should not be pressed again.

TROUBLESHOOTING: SPECIFIC

5A breaker keeps cutting power:

- Problem: Device does not have power when plugged in.
- Solution:
    - If the breaker is tripping, then immediately cut power to the system.
    - Check that there are no shorts.
    - Make sure no additional devices (vacuum cleaners, appliances, etc. are connected to the TV's GFCI outlets.
    - Consider using a surge protector that bypasses the GFCI outlets.
    - Contact Andrew Marques at email or phone on page 1.

No power when plugged in:

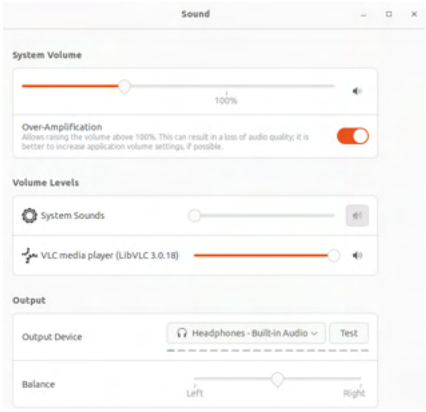
- Problem: Device does not have power when plugged in.
- Solution:
    - If using the surge protector, check if it is receiving power. It may need to be reset.
    - If using the GFCI outlets, check that the 5A breaker is not tripped, and the GFCI outlets are reset (there are 3x outlets to reset).

Some power when plugged in:

- Problem: Some but not all devices are receiving power.
- Solution:
    - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
    - Check that all power cables are plugged into outlets and their respective devices. Cables are labeled.
    - Check that all GFCI outlets are reset. There are 3x outlets to check.
    - Check that the power knob is working.

No audio or quiet audio:

- Problem: Despite changing controls, the audio remains too quiet to hear.
- Solution:
    - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
    - Check that all connections to the speaker and main computer are secure.
    - Check that the volume knob on the speaker itself is set to the designated position (indicated by arrows).
    - Check if the audio is louder in Default mode. If it is louder, there is likely an issue with the jumper wire connections. Check that all connections are secure. Check that the alternative knob (front or internal) are functioning). It may require a potentiometer to be replaced.
    - If the audio is too quiet in default mode, check the sound settings. the sounds for the video file itself may be too low.
        - Click the "Windows" button on the keyboard.
        - Search "Sound".
        - The settings should look like the image below.

- 

Audio too loud:

- Problem: The audio is playing too loud, typically this occurs in the Default mode.
- Solution:
  - Turn down the knob on the speakers themselves (not the knobs on the TV).
  - The max volume can be edited in the software under the vol_max variable, this accepts values from 0-1.

No picture:

- Problem: The screen is black or does not show a program.
- Solution:
  - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
  - Check that all connections to the monitor and main computer are secure.
  - Try default mode. There should be image and sound. If there is no image or sound, check that there is a "/home/andrewdmarques/Desktop/TV/Bin/Scripts/tv-on.txt" file. If there is no file here, then make a blank file with this "tv-on.txt" at that location.

Picture turns dark and fades in every few seconds:

- Problem: The TV is not operating correctly by showing an image briefly then cutting to black and fading back in. Typically the image is static.
- Solution:
  - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
  - Check that there are videos in the channel directory "/home/andrewdmarques/Desktop/TV/Channels". If there are no video files here, then make sure each channel has a video file.
  - Check that there are directories for channels 03-13 in "/home/andrewdmarques/Desktop/TV/Channels".
  - Generate a new set of program schedules.
    - Go to "/home/andrewdmarques/Desktop/TV/Bin/Program-Schedule/" and delete ALL the contents of this folder. They should be a set of CSV files.
    - Disconnect and reconnect the main power supply to the cabinet.
    - This will automatically generate a new list of schedules that now include the added program.
  - This may be a software issue if the problem persists. Try using the default mode.

Controls show no response:

- Problem: The dials are not changing volume, brightness, or channels.

- Solution:
    - This may be caused by a loose wire, damaged potentiometer, or the system is in a different setting.
    - Check that the desired system mode is selected. For example, the front knobs will only work if the front mode is selected. See the Quickstart Guide for details.
    - Check that there are no loose jumper wire. If there is a loose jumper wire, see the Loose Jumper Wire" section. Do not touch wires while system has power.
    - Try default mode. There should be image and sound. If there is no image or sound, check that there is a "/home/andrewdmarques/Desktop/TV/Bin/Scripts/tv-on.txt" file. If there is no file here, then make a blank file with this "tv-on.txt" at that location.
    - If default mode works, and all connections are made, contact Andrew.

Controls reversed:

- Problem: When turning a dial the response is the opposite of what's expected.
- Solution:
    - Check the Arduino wiring diagram, it is likely the 5V and GND are switched for that potentiometer.
    - If all dials are reversed compared to expectations, then 5V and GND to the breadboard as it connects to the Arduino may be switched.
    - If connections are loose or switches, correct them and ensure a tight connection. .

The power knob in the front of the TV set is not turning on or off:

- Problem: The power knob in the front of the unit is not turning on or off the TV set.
- Solution:
    - If your TV is plugged in using a surge protector strip bypassing the GFCI outlets, then this knob is designed to be bypassed too.
    - If your TV is plugged in using the outlets, check that plugs are connected to the correct outlets: only the main computer should be connected to "On Always". All other plugs should be connected to "On with Switch".
    - It is possible that the mechanism is broken. Contact Andrew.

Stuck in default mode:

- Problem: When the Arduino is connected, it does not allow the front or internal knobs to be connected.
- Solution:
    - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
    - Check that all connections are secure to the main computer, the Arduino, and the breadboard.

Monitor appears unlevel when viewed from the front:

- Problem: The picture appears askew when viewing from the front.
- Solution:
    - Readjust the monitor from the inside. With some minor adjustments it should sit level from the viewer's perspective in the front.
    - TIP: Make it so the bottom of the monitor is pressed as closely to the tan bakelite mount as possible.

Jumper wires (colored thin cables) disconnected:

- Problem: Knobs are not working.

- Solution:
    - Use caution when connecting the wires, it is important to connect the positive and negative leads to the appropriate pins. Cables are in groups of three, the outermost wires are the positive and negative wires, the middle wire should be wired to send data to the Arduino.
    - See the Arduino wiring diagram as a reference.

Popup windows on monitor:

- Problem: Notification popups on monitor.
- Solution:
    - Dismiss notification.
    - Try resetting the system by disconnecting and reconnecting the main power supply to the cabinet and rebooting the computer.
    - Try to disable notifications by clicking the "Windows" button, searching for "Notifications" and disabling notifications.

Date/Time incorrect error:

- Problem: The main computer has errors because the date and time are incorrect.
- Solution:
    - This is usually caused by being unplugged for more than a few weeks.
    - See the "Long-term Storage" section of the Quickstart Guide for instructions on resetting the internal clock.

DISCLAIMER

The television cabinet, electronics, and Vintage Television Operating System (VINTOS) described in this manual is provided "as is" with limited warranty at the discretion of the creator (Andrew D. Marques). This discretionary limited warranty includes but is not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. In addition, the user may return the device at any time. Replacement of the device can be discussed at the discretion of the creator.

The creator strongly recommends bypassing the optional internal power switch and using a UL certified, grounded surge protector. The creator cannot be held liable for any damages arising from the use of this product, including but not limited to direct, indirect, incidental, or consequential damages. The user assumes full responsibility for any risks associated with the use of this product. By accepting and using this product, the user agrees to these terms and conditions.

ABOUT THE CREATOR

Andrew D. Marques is a virologist at the University of Pennsylvania's Perelman School of Medicine. Some of his favorite pastimes include spending time with his wife and family, building projects, programming, film photography, amateur birding, cycling, maintaining and driving cars, and restoring antique electronics by finding ways to incorporate them into modern life. Some of his fondest memories are the drives he takes with his wife in their 1969 Volkswagen Beetle. At a young age, he has been interested in taking a holistic perspective of history: where historical and current events are closely connected, and having a physical and mental connection to our past can better shape our present and future.

ACKNOWLEDGEMENTS

The success of this project was made possible by the assistance of numerous friends, whose contributions I would like to acknowledge. I extend a special thank you to Scott Sherrill-Mix, Carter Merenstein, and Alex McFarland for their invaluable troubleshooting help with programming. I am also grateful to Aoife Doto for reproducing the front control labels and Orlando Ferreira for editing this manual.

Additionally, I owe a debt of gratitude to my wife Caitlyn Mierau-Marques, whose unwavering support was critical to the project's success. Caitlyn was endlessly patient and supportive, providing encouragement throughout the countless nights and many hours spent grinding away using the dremel or on the computer while we watched Netflix.

Their insights, feedback, and support were vital to the completion of this work.