VINTAGE TELEPHONE EMULATING LANDLINE

Contact: Andrew D. Marques

Website: www.AndrewDMarques.com

Email: _____
_____

Address: _____
_____

Phone: _____

Version: 4
Updated: 2/5/2024

INTRODUCTION

Vintage Telephone Emulating Landline (VinTEL) is the software and hardware designed to "reanimate" rotary dial phones to emulate interactions of an increasingly bygone era. This manual serves as a comprehensive guide for using and maintaining the telephone set. While it covers the hardware build process, the primary focus is on the accompanying software.

Key features of VinTEL include:

- Easy addition of new audio files to be dialed
- Easy addition of new "interactive" skits
- Graphic User Interface (GUI) to create new skits by selecting user input and outputs
- Functioning mechanical ringer
- Functioning original dial
- Functioning original handset speaker with authentic low fidelity quality
- Functioning original hook
- Optional compatibility with pay phones, sensing and responding to different coins inserted. Use with non-pay phone rotary phones also works. Functionality with some models of touch tone phones too.
- Motion-activated incoming calls
- User-generated incoming calls
- Extensive data logging for phone usage and coins deposited.

VinTEL is a non-invasive method that leaves the rotary phone intact to easily be converted back to it's original landline set up. The external control box houses the electronics and controls used to change the operation of the phone. Controls include:

- Disable paid outgoing calls/collect calls, making operation free
- System reset
- System sleep mode
- Activate incoming call by push of a button
- Activate incoming call by motion sensor

The emulation of landline use from the 1950s is achieved through the operation of outgoing calls that "dial" audio files to be played through the handset, and the interactive "skits" that can be designed to interact with the user (input through the dial and optionally through the microphone to have alternate responses for calls. Incoming calls that can be triggered by the presence of visitors (triggered stochastically such that not every detection of movement results in an incoming call).

TABLE OF CONTENTS

.

QUICKSTART GUIDE

Operating Steps:

1. Make sure all connections are tight.
2. Power on the control box.
    a. In about 20 seconds the status light should begin blinking.
    b. In 30 seconds the other indicator lights should begin illuminating then the bell should ring twice. This indicates the phone has initiated.
3. Make sure the standby switch is in the "On" position.
4. You can begin receiving or sending calls.

Change Modes:

- Motion-activated calls: switch with this label can disable the motion-activated calls, meaning that incoming calls can only be triggered by pressing one of the activation call buttons.
- Volume HI/LO: This allows control of the device to increase or decrease the volume from the handset.
- Activate incoming call: button (separate device or on the control box) can initiate the call on demand.
- If the standby mode is powered off, then the phone will not allow any further calls to occur, although the software will continue running in the background. The device will need to be unplugged to be fully powered off. When this switch is in the on position, then it can accept incoming calls.

Adding New Recordings -- Dialing Out:

1. Take the audio file and label the file so that the file name begins with the number to be dialed. For example "123.mp3" or "123_audio-file.mp3" are both acceptable formats.
2. Copy the audio files to the Audio directory.
3. NOTE: You can have multiple audio files with the same numbers
4. You're done!

Adding New Recordings -- Incoming Calls:

1. To make incoming calls, a skit must be made, first open the program skit-maker.py.
2. Name the skit according to whether the incoming call should occur with or without the docent present.
    a. "**incoming_call_xxx**" is the name for an incoming call that occurs when the docent is present, where "xxx" is the skit name you want to label it as. The program does not consider the xxx part.
    b. "**incoming_call_unattended_xxx**" is the name for an incoming call that occurs WITHOUT the docent present (motion-activated calls).
3. Make the current file the "ringer-us.py" file.
4. Check the box indicating that this ringer file is the first file to be played.
5. Select "hook" as the type of action required.
6. Select the audio file that should be played.

7. Add the selection using the "Add" button on the bottom left.
8. Submit the file to be generated, it will make a csv file with the name you created.
9. You're done!

Adding New Recordings -- Skits:

1. Open the program skit-maker.py.
2. Name the skit using the digits that will need to be dialed and a quick description name.
    a. For example, "123_skit-example" would be a good name for a skit example that should be played when dialed 123.
3. Determine the logic for the skit. Both python scripts and audio files can be used.
4. For the first file, add the current file and determine the action required to initiate the next call (for example, dial input or hook).
5. Check the box indicating that this file is the first file to be played.
6. Complete the rest of the information for one of the outcomes from the action item:
    a. Select the action type that should be required.
    b. Select the next file that should be played given that action type.
    c. Add the selection using the "Add" button on the bottom left.
7. Repeat step 6 for each of the potential outcomes (for example, if a different audio file is to be played with different action numbers for inputs).
    a. All inputs for the a given current file must be the same.
8. Repeat steps 6 and 7 for all subsequent files if there are multiple stages of actions desired.
9. When the logic of your skits are complete, submit the file to be generated, it will make a csv file with the name you created.
10. You're done!

Servicing the Phone -- Removing the Phone Front:

1. Make sure the main power is disconnected.
2. Remove the silver nut from the front of the phone.
3. Lean the phone body gently several inches, being careful to hold it from falling.
4. Disconnect all the dupont connectors.
    a. WAIT! Did you forget any connectors? Chances are you missed one or two. Proceed with caution.
5. Carefully pull the front of the body away from the back, taking note of any dupont connectors that might still be attaching the two.

Servicing the Phone -- Returning the Phone Front:

1. Make sure the main power is disconnected.
2. Gently place the front of the phone resting on the bottom/back -- make sure to give enough space to reach the wires.
3. Connect all the dupont connectors to their respective colors.
    a. WARNING: If incorrect connections are made, then the electronics may be destroyed.
    b. Connectors should remain together (do not plug a single wire into a socket that has room for 5 different wires).
4. Use velcro (hook and loop) to secure the cables together.
5. Make sure the following areas are clear:
    a. The silver coin shoot (wires commonly block this causing coins to jam).

   b. Brass coin shoot (wires might block the coins going to the coin return shoot).
   c. Cables along the outside of the housing (wires might be sticking out).
   d. Screw terminal connector (this commonly gets in the way of the coin return shoot, it is recommended to lay the screw terminal at the bottom of the phone, somewhat diagonally. It fits but it is tight).
   e. WAIT! Be strategic about where the dupont connectors are located, if they are too low or too high, they have been known to break off when being "jammed" into the phone body. proceed with caution. If there is resistance, check to make sure there is not too much pressure being applied to the connectors.
6. Carefully place the front of the phone onto the bottom/back component.
7. There is a knob at the top of the phone that the front piece rests on, then the bottom should swing back into place with LITTLE to no resistance.
8. If the phone is correctly pressed in place, then the front silver screw can be tightened.
9. Test that the coin shoots are not obstructed using the following coins:
   a. Nickel
   b. Dime
   c. Quarter
   d. Wrong coin shoot (using something like a dime in the nickel slot).
10. You're done!

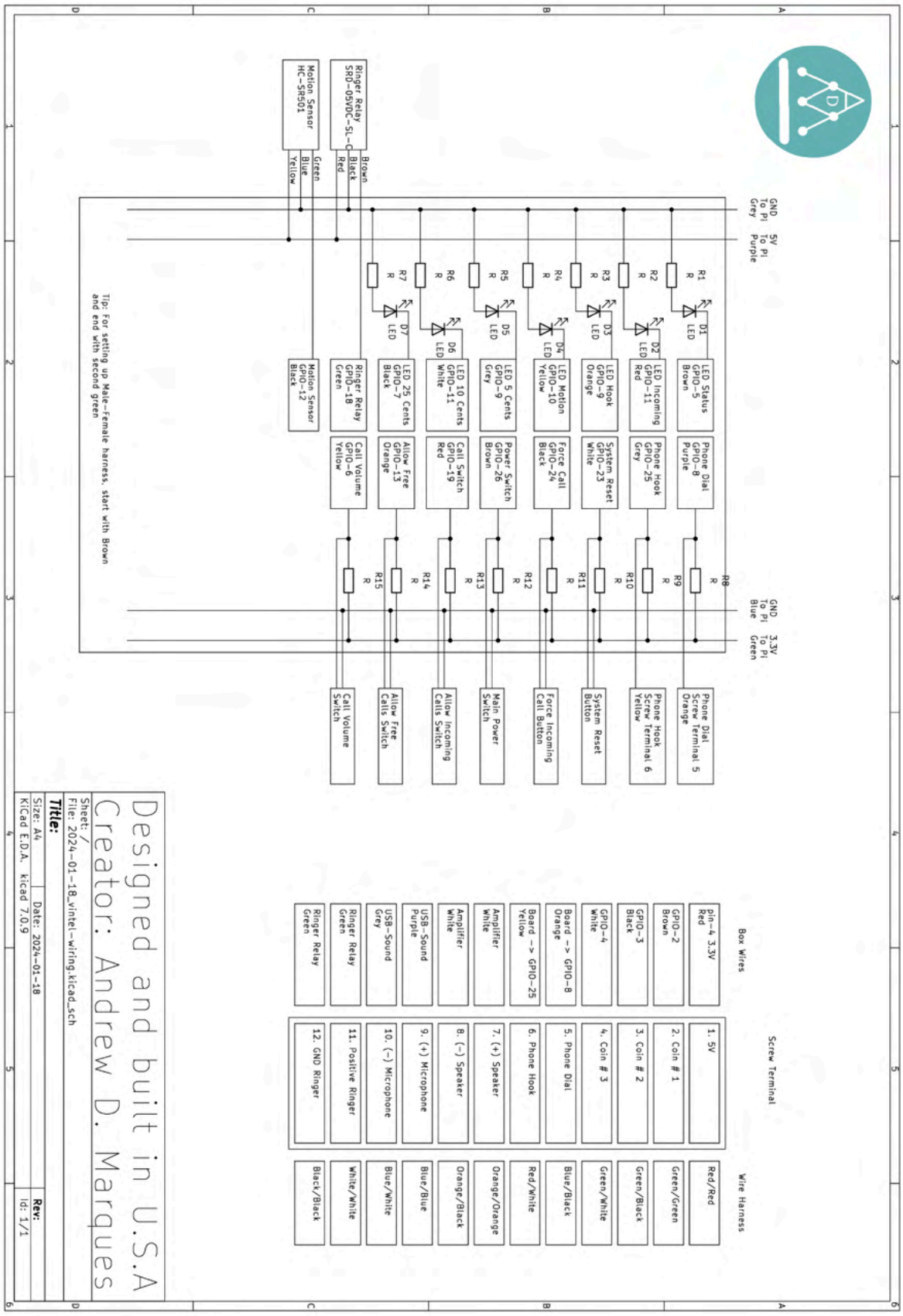
Moving the Telephone:

1. Moving the telephone should be done carefully -- first disconnect the main power.
2. Using the screw terminals, disconnect the cables so that the control box can be moved separately.
3. You may need to disconnect the phone unit from the harness cable which requires opening the phone:
   a. Unscrew the silver bolt in the center of the phone.
   b. Gently and slowly, lift the front of the phone from the back/bottom of the phone. CAUTION: cables are connecting the front of the phone to the phone back, these connecting cables in the phone can easily be damaged. Disconnect at the dupont connector joints. Once disconnected, the front of the phone can be removed.
4. When the phone has been moved, to a new locations, make sure to carefully reconnect the dupont connectors within the phone body.

Long-Term Storage:

1. The telephone should be stored in dry cool spaces without rodents. This unit should be thought of like a computer, so make sure that it is stored appropriately.
2. Check the integrity of the wiring connections.

DIAGRAMS


Designed and built in U.S.A
Creator: Andrew D. Marques
Sheet: /
File: 2024-01-18_vintel-wiring.kicad_sch
Title:
Size: A4
Date: 2024-01-18
Rev:
KiCad E.D.A. kicad 7.0.9
Id: 1/1
Tip: For setting up Male-Female harness, start with Brown and end with second green
GND To Pi Grey
5V To Pi Purple
GND To Pi Blue
3.3V To Pi Green
Motion Sensor HC-SR501
Ringer Relay SRD-05VDC-SL-C
Yellow
Blue
Green
Black
Red
Brown
Motion Sensor GPIO-12 Black
Ringer Relay GPIO-18 Green
LED 25 Cents GPIO-7 Black
Allow Free GPIO-13 Orange
Call Volume GPIO-6 Yellow
LED 10 Cents GPIO-11 White
Call Switch GPIO-19 Red
LED 5 Cents GPIO-9 Grey
Power Switch GPIO-26 Brown
LED Motion GPIO-10 Yellow
Force Call GPIO-24 Black
LED Hook GPIO-9 Orange
System Reset GPIO-23 White
LED Incoming GPIO-11 Red
Phone Hook GPIO-25 Grey
LED Status GPIO-5 Brown
Phone Dial GPIO-8 Purple
D1 LED
D2 LED
D3 LED
D4 LED
D5 LED
D6 LED
D7 LED
R1 R
R2 R
R3 R
R4 R
R5 R
R6 R
R7 R
R8 R
R9 R
R10 R
R11 R
R12 R
R13 R
R14 R
R15 R
Call Volume Switch
Allow Free Calls Switch
Allow Incoming Calls Switch
Main Power Switch
Force Incoming Call Button
System Reset Button
Phone Hook Screw Terminal 6 Yellow
Phone Dial Screw Terminal 5 Orange
Box Wires
Pin-4 3.3V Red
GPIO-2 Brown
GPIO-3 Black
GPIO-4 White
Board -> GPIO-8 Orange
Board -> GPIO-25 Yellow
Amplifier White
Amplifier White
USB-Sound Purple
USB-Sound Grey
Ringer Relay Green
Ringer Relay Green
Screw Terminal
1. 5V
2. Coin # 1
3. Coin # 2
4. Coin # 3
5. Phone Dial
6. Phone Hook
7. (+) Speaker
8. (-) Speaker
9. (+) Microphone
10. (-) Microphone
11. Positive Ringer
12. GND Ringer
Wire Harness
Red/Red
Green/Green
Green/Black
Green/White
Blue/Black
Red/White
Orange/Orange
Orange/Black
Blue/Blue
Blue/White
White/White
Black/Black

FILE DESCRIPTIONS:

- OS/Scripts: Contains the main system scripts and the configure.csv file for system configuration.
- OS/Status: Stores the status files indicating various system states.
    - main-on.txt: Indicates that the main switch is toggled to the on position.
    - hook-on.txt: Indicates that the handset was picked up and the hook is toggled to the on position.
    - hook-off.txt: Indicates that the handset was placed down and the hook is toggled to the off position.
    - call-on.txt: Indicates that there is already a call in progress.
    - dial-complete.txt: Indicates that a number has been completed being dialed.
    - Dial/number.txt: Contains the dialed numbers on the first line.
    - skit-on.txt: Indicates that a skit should be played, initiating skit-player.py.
    - call-initiated.txt: Indicates that a docent-triggered call has been initiated.
    - call-initiated-unattended.txt: Indicates that a motion sensor-triggered call has been initiated.
    - system-working.txt: Indicates the system status during skit playback.
- OS/Effects: Contains audio effects and prompts used in skits.
- OS/Skits: Holds the CSV files defining different skits and their sequences.
- OS/Audio: Stores the audio files used in skits and prompts.
- Hardware/Schematics: Contains detailed circuit diagrams and wiring instructions for hardware components.
- README.md: Provides information about the VinTEL system and its setup.
- LICENSE: The license under which the VinTEL system is distributed.

SCRIPT DESCRIPTIONS

1. **controller.py**: This script is responsible for controlling the overall VinTEL system. It monitors various input signals, such as button presses and sensor detections, and triggers corresponding actions, such as initiating calls, playing audio files, or controlling LEDs.
2. **coins.py**: This script handles coin detection and interaction in the VinTEL system. It detects when coins are deposited and keeps track of the deposited amount, which can be used to trigger specific actions or play corresponding audio files.
3. **dialer.py**: The dialer.py script emulates a dialing mechanism for the VinTEL system. It listens for input from a rotary dial or keypad and simulates the dialing process by generating corresponding tones or signals. It can be used to initiate calls or navigate through menus.
4. **audio-selector.py**: This script manages the selection and playback of audio files in the VinTEL system. It allows users to choose specific audio files for playback, control the volume, and handle audio-related functions.
5. **led-controller.py**: The led-controller.py script controls the LEDs (Light Emitting Diodes) in the VinTEL system. It manages the states and behavior of the LEDs, such as turning them on or off, blinking patterns, or adjusting brightness, to provide visual indications or feedback.
6. **make-skit.py**: This script is used to create skit files for the VinTEL system. It allows users to define a sequence of actions, audio files, and interactive elements to create custom skits or scenarios that can be played by the system.
7. **motion-controller.py**: The motion-controller.py script handles motion detection in the VinTEL system. It monitors motion sensors and detects movement or activity in the designated areas. It can be used to trigger specific actions, such as playing an audio file or initiating a call, when motion is detected.
8. **player.sh**: The player.sh script is a shell script that interacts with the audio player component of the VinTEL system. It executes the necessary commands to play audio files, control playback, and manage audio-related functionalities.
9. **skit-selector.py**: This script enables the selection and initiation of pre-defined skits in the VinTEL system. It provides options for users to choose from a list of available skits and triggers the playback of the selected skit.
10. **system-blink.py**: The system-blink.py script controls the system status indicator, typically an LED, in the VinTEL system. It continuously monitors the presence of a specific file and adjusts the LED's state accordingly. It provides visual feedback to indicate the operational status of the system.

DIRECTORY STRUCTURE

```
VinTEL
├── OS
│   ├── Scripts
│   │   ├── audio-selector.py
│   │   ├── coins.py
│   │   ├── controller.py
│   │   ├── dialer.py
│   │   ├── led-controller.py
│   │   ├── motion-controller.py
│   │   ├── skit-player.py
│   │   └── skit-selector.py
│   ├── Status
│   │   ├── Logs
│   │   │   ├── log-audio-selector.txt
│   │   │   ├── log-coins.txt
│   │   │   ├── log-controller.txt
│   │   │   ├── log-dialer.txt
│   │   │   ├── log-led-controller.txt
│   │   │   ├── log-motion-controller.txt
│   │   │   ├── log-skit-player.txt
│   │   │   └── log-skit-selector.txt
│   │   ├── main-on.txt
│   │   ├── hook-on.txt
│   │   ├── hook-off.txt
│   │   ├── call-on.txt
│   │   ├── dial-complete.txt
│   │   ├── Dial
│   │   │   └── number.txt
│   │   ├── skit-on.txt
│   │   ├── call-initiated.txt
│   │   ├── call-initiated-unattended.txt
│   │   └── system-working.txt
│   ├── Effects
│   ├── Skits
│   └── Audio
├── Hardware
│   ├── Schematics
│   └── Components
├── README.md
└── LICENSE
```

INSTALLATION GUIDE:

For first time installing VinTEL

1. Install Raspbian OS on the Raspberry Pi
    a. https://www.instructables.com/HOW-TO-INSTALL-RASPBIAN-OS-IN-YOUR-RASPBERRY-PI/
    b. OS Version 2020-02-14: https://downloads.raspberrypi.org/raspbian/images/raspbian-2020-02-14/
2. Using the command terminal, install the dependencies and prepare the files with the following commands:

```
sudo apt update
sudo apt install htop
sudo apt install thonny
sudo apt install firefox-esr
pip3 install PySimpleGUI
pip3 install pyaudio
sudo apt-get install libasound2-dev
pip3 install pyalsaaudio
```

3. Make the files in the VinTEL/Bin/Scripts executable
    a. Move files to be located in /home/pi/Desktop/VinTEL/Bin/Scripts/
    b. `chmod +x /home/pi/Desktop/VinTEL/OS/Scripts/*.py`
    c. `chmod +x /home/pi/Desktop/VinTEL/OS/Scripts/*.sh`
4. Make the crontab file so that the scripts can be played at reboot

```
a. @reboot python3
   /home/pi/Desktop/VinTEL/Bin/Scripts/audio-selector.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-audio-selector.txt
b. @reboot python3 /home/pi/Desktop/VinTEL/Bin/Scripts/coins.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-coins.txt
c. @reboot python3 /home/pi/Desktop/VinTEL/Bin/Scripts/controller.py
   1> /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-controller.txt
d. @reboot python3 /home/pi/Desktop/VinTEL/Bin/Scripts/dialer.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-dialer.txt
e. #@reboot python3
   /home/pi/Desktop/VinTEL/Bin/Scripts/led-controller.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-led-controller.txt
f. @reboot python3
   /home/pi/Desktop/VinTEL/Bin/Scripts/motion-controller.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-motion-controller.txt
g. @reboot python3 /home/pi/Desktop/VinTEL/Bin/Scripts/skit-player.py
   1> /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-skit-player.txt
h. @reboot python3
   /home/pi/Desktop/VinTEL/Bin/Scripts/skit-selector.py 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-skit-selector.txt
i. @reboot /home/pi/Desktop/VinTEL/Bin/Scripts/player.sh 1>
   /home/pi/Desktop/VinTEL/Bin/Status/Logs/log-bash-player.txt
```

5. Set the VLC audio output to the headphones
    a. Open VLC on the Raspberry Pi
    b. Go to Tools > Preferences > Audio > select All on the bottom left of the screen to show all settings
    c. Under Audio > Output modules > ALSA > select the VoD server module "bcm2835 Headphones, bcm2835 Headphones Default Audio Device
    d. Save the settings and exit
6. Add any audio files that you would like to add, following the information laid out in the quick start section of this manual.

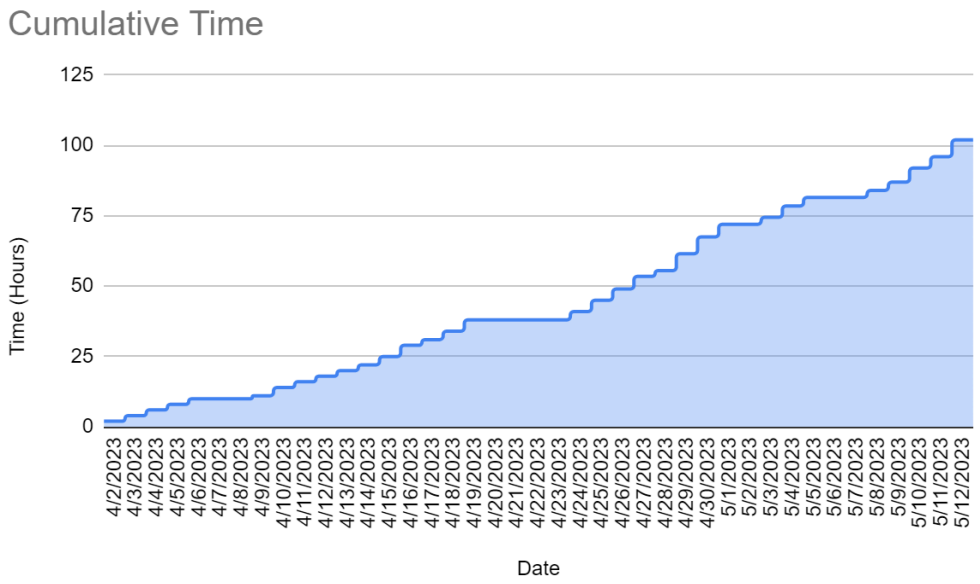
BUILD COSTS

Total: $196.72

- $0 Payphone donated by "Tiger" Tom Ehrhart.
- $0 Wiring harness donated by "Tiger" Tom Ehrhart and friends.
- $2.89 SR-501 Proximity sensor to detect new visitors (1x item at $2.89 each, $8.49 for 3x) https://www.amazon.com/gp/product/B07KZW86YR/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
- $8.89 12V 2A power supply for ringer https://www.amazon.com/gp/product/B01GD4ZQRS/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&th=1
- $3.90 5V Relay for ringer rated to control 30VDC 10A circuits using SMD optocoupler isolation (1x item at $3.90 each, $7.79 for 2x) https://www.amazon.com/gp/product/B00LW15A4W/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1
- $3.20 SPST toggle switch for system controls (4x items at $0.80 each, $7.99 for 10x) https://www.amazon.com/dp/B0799LBFNY?ref=ppx_yo2ov_dt_b_product_details&th=1
- $14.76 Jameco Valuepro WBU-301-R 400 points solderless breadboard (2x items at $7.38 each) https://www.amazon.com/dp/B00B8861R4?psc=1&ref=ppx_yo2ov_dt_b_product_details
- $70 Raspberry Pi 3B+, 32 GB SD card, 1GB RAM https://www.ebay.com/itm/145020177805
- $19.07 Mini surge protector 6ft cable and 3 outlets https://www.amazon.com/gp/product/B09CQDJJ98/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&th=1
- $11.20 Jumper cables (112x items at $0.10 each, $11.99 for 120)
- $4.40 Screw terminals (2x items at $2.20 each, $13.24 for 6)
- $21.49 Wooden control box https://www.amazon.com/gp/product/B0B4S9YGYR/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&th=1
- $29.99 Black Magic Telephone Ringer Transformer (5v DC to 86V 20Hz AC)
- $6.93 Lock with key to secure the control box https://www.amazon.com/gp/product/B085TC8KWX/ref=ppx_yo_dt_b_asin_title_o07_s00?ie=UTF8&th=1

LABOR HOURS

As of 5/12/2023, this project has benefited from 102 hours of work for planning, programming, execution, and troubleshooting.



Date	Daily Time (Hours)	Cummulative Time (Hours)	Description
5/12/2023	6	102	Phone: programming log files, adding skits, debugging
5/11/2023	4	96	Phone: debugging, adding new skits
5/10/2023	5	92	Phone: rewiring, audio amplifier, debugging
5/9/2023	3	87	Phone: programming skits
5/8/2023	2.5	84	Phone: installing new raspbian software and deubugging
5/5/2023	3	81.5	Phone: programming
5/4/2023	4	78.5	Phone: programming
5/3/2023	2.5	74.5	Phone: preparing motion sensor and remote button for calls
5/1/2023	4.5	72	Phone: generating incoming call skits, programming led status
4/30/2023	6	67.5	Phone: programming skit player, making skits, 3D printing parts
4/29/2023	6	61.5	Phone: programming GUI and skit player
4/28/2023	2	55.5	Phone: programing GUI for skits
4/27/2023	4.5	53.5	Phone: wiring control box, programming
4/26/2023	4	49	Phone: building control box and wiring
4/25/2023	4	45	Phone: building control box
4/24/2023	3	41	Phone: programming outgoing calls, simple format
4/19/2023	4	38	Phone: programming audio controller and sound player
4/18/2023	3	34	Phone: programming hook and dial
4/17/2023	2	31	Phone: programming main controller script
4/16/2023	4	29	Phone: programming, shortening cable
4/15/2023	3	25	Phone: programming, getting speaker to work
4/14/2023	2	22	Phone: programming, prototyping circuits
4/13/2023	2	20	Phone: prototyping circuits
4/12/2023	2	18	Phone: prototyping circuits
4/11/2023	2	16	Phone: preparing soldered connections of switches and lights
4/10/2023	3	14	Phone: working on ringer
4/9/2023	1	11	Phone: programming and planning
4/6/2023	2	10	Phone: restoring coin shoot, adapting coin sensors
4/5/2023	2	8	Phone: planning build, ordering supplies
4/4/2023	2	6	Phone: planning build, prototyping coin sensors
4/3/2023	2	4	Phone: planning build
4/2/2023	2	2	Phone: planning build

OPERATING SYSTEM CODE

Python code: **make-skit.py** executes a graphic user interface for users to design new skits to be played by VinTEL. It requires the input of (1) current file, (2) action that can be taken from the current file, i.e. dialing a number, and (3) the next file given that specific action taken. It spells out the logic for how to proceed and formats it in a way that VinTEL then follows..

```
#!/usr/bin/python3
##############################################################################
# Description
##############################################################################
# make-skit.py (v1.10) Helps organize the skit file based on the input files.
# The curator would fill out the forms for this.
##############################################################################
# Load libraries
##############################################################################

import time
from datetime import datetime
import csv
import os
import re
import random # For selecting a random file to play from.
import PySimpleGUI as sg # For GUI

##############################################################################
# User input variables
##############################################################################

config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

##############################################################################
# Initialize
##############################################################################
print('Initializing')

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']
        if row['variable'] == 'dir_effects':
            dir_effects = row['directory']
        if row['variable'] == 'dir_scripts':
            dir_scripts = row['directory']
        if row['variable'] == 'dir_audio':
            dir_audio = row['directory']
        if row['variable'] == 'dir_skits':
            dir_skits = row['directory']

# file_main     = dir_status + '/main-on.txt'       # The file that indicates the main switch is toggled to the on position.
# file_hook_on  = dir_status + '/hook-on.txt'       # The file that indicates the handset was picked up, the hook is toggled to the on position.
# file_hook_off = dir_status + '/hook-off.txt'      # The file that indicates the handset was placed down, the hook is toggled to the off
position.
# file_call_on  = dir_status + '/call-on.txt'       # The file that indicates there is already a call in progress.
# file_dialed   = dir_status + '/dial-complete.txt' # The file that indicates that a number is completed being dialed.
# file_number   = dir_status + '/Dial/number.txt'   # The file containing the dialed numbers on the first line.
# file_skit_on  = dir_status + '/skit-on.txt'       # The file that indicates that a skit should be played, beginning skit-player.py.
# file_player_on = dir_scripts + '/player-on.txt'   # The file that indicates that that vlc should be activated and begin playing the selected
audio file.
# file_temp     = dir_scripts + '/temp.py'          # The file that contains the commands to play the specified audio file.


##############################################################################
# Run
##############################################################################
print('Running...')

def get_user_input(values):
    if values['dial']:
        return 'dial', sg.popup_get_text('User dial input:')
    elif values['none']:
        return 'none', 'none'
    elif values['hook']:
        return 'hook', 'hook-on'
    elif values['audio']:
        return 'audio', sg.popup_get_text('User audio keywords input:')
    elif values['coins']:
        return 'coins', sg.popup_get_text('Charged amount:\n-Must be divisible by 5\n-Report as cents (i.e. 5, 10, 15 etc.)')
    else:
        return '', ''

layout = [
    [sg.Text('Name for skit file:\t'), sg.InputText(key='skit_file')],
    [sg.Text('')],
    [sg.Text('Current file:\t'), sg.InputText(key='from')],
    [sg.Checkbox('First file to be played in skit?', key='first_file')],
```

```
    [sg.Text('User input type:\t')],
    [sg.Radio('Dial', 'RADIO1', key='dial'), sg.Radio('Hook', 'RADIO1', key='hook'), sg.Radio('Audio', 'RADIO1', key='audio'), sg.Radio('Coins',
'RADIO1', key='coins'), sg.Radio('No Input', 'RADIO1', key='none')],
    [sg.Text('Next file:\t\t'), sg.InputText(key='to')],
    [sg.Table(values=[],
              headings=['Current File', 'First File', 'Input Type', 'Input Value', 'Next File'],
              display_row_numbers=False,
              auto_size_columns=True,
              num_rows=10,
              key='input_table')],
    [sg.Button('Add Interaction'), sg.Button('Delete Selected'), sg.Button('Submit')]
]

window = sg.Window('VinTEL Skit Creator', layout)

def create_csv_file(file_path, input_data):
    with open(file_path, 'a', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(['current_file', 'first_file', 'input_type', 'input_value', 'next_file'])  # Write header row
        for row in input_data:
            csvwriter.writerow(row)
    print(f'CSV file created with {len(input_data)} rows')

input_data = []

while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED:
        break
    if event == 'Add Interaction':
        if not all([values['from'], values['to'], any([values['dial'], values['none'], values['hook'], values['audio'], values['coins']])]):
            sg.popup_error('All boxes must be populated')
            continue

        input_type, input_value = get_user_input(values)
        if not input_value:
            continue

        first_file = 'yes' if values['first_file'] else 'no'
        new_input = [values['from'], first_file, input_type, input_value, values['to']]
        if first_file == 'yes':
            for row in input_data:
                if row[1] == 'yes' and row[0] != values['from']:
                    sg.popup_error("Multiple first files must have the same current file name")
                    continue

        input_data.append(new_input)

        window['input_table'].update(values=input_data)
        window['from'].update('')
        window['to'].update('')

    if event == 'Delete Selected':
        selected_item = values['input_table']
        if selected_item:
            index_to_remove = selected_item[0]
            input_data.pop(index_to_remove)
            window['input_table'].update(values=input_data)

    if event == 'Submit':
        skit_file = values['skit_file']

        if skit_file:
            file_path = dir_audio + '/' + skit_file + '.csv'

            if os.path.isfile(file_path):
                os.remove(file_path)

            create_csv_file(file_path, input_data)
            window['skit_file'].update('')
            window['from'].update('')
            window['to'].update('')
            window['first_file'].update(False)
            input_data = []
            window['input_table'].update(values=input_data)

window.close()
```

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

```
#!/bin/bash
sleep 10
while :
do
           sleep 0.1
           FILE=/home/pi/Desktop/VinTEL/Bin/Scripts/player-on.txt
           if [ -f "$FILE" ]; then
                      rm -r /home/pi/Desktop/VinTEL/Bin/Scripts/player-on.txt
                      FILE=/home/pi/Desktop/VinTEL/Bin/Scripts/temp.py
                      if [ -f "$FILE" ]; then
                                 chmod +x /home/pi/Desktop/VinTEL/Bin/Scripts/temp.py
                                 python3 /home/pi/Desktop/VinTEL/Bin/Scripts/temp.py > /home/pi/Desktop/VinTEL/Bin/Status/Logs/bash_log.text
2>&1
                      fi
           fi
```

```
done
```

Python code: skit-selector.py determines if the current skit is prompted with a docent present (call_initiated) or a docent absent (call_initiated_unattended) then determines which skit should be played given the situation..

```
#!/usr/bin/python3
########################################################################
# Description
########################################################################
# skit-selector.py (v1.00) This searches for motion detected or docent-
# initiated call and then randomly selects an appropriate interaction.
# There cannot be an ongoing call, it will ignore a message in that case.
########################################################################
# Load libraries
########################################################################

import time
from datetime import datetime
import csv
import os
import re
import random               # For selecting a random file to play from.

########################################################################
# User input variables
########################################################################

config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

########################################################################
# Initialize
########################################################################
print('Initializing')

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']
        elif row['variable'] == 'dir_effects':
            dir_effects = row['directory']
        elif row['variable'] == 'dir_scripts':
            dir_scripts = row['directory']
        elif row['variable'] == 'dir_audio':
            dir_audio = row['directory']
        elif row['variable'] == 'dir_skits':
            dir_skits = row['directory']

file_main     = dir_status + '/main-on.txt'       # The file that indicates the main switch is toggled to the on position.
file_hook_on  = dir_status + '/hook-on.txt'       # The file that indicates the handset was picked up, the hook is toggled to the on position.
file_hook_off = dir_status + '/hook-off.txt'      # The file that indicates the handset was placed down, the hook is toggled to the off position.
file_call_on  = dir_status + '/call-on.txt'       # The file that indicates there is already a call in progress.
file_dialed   = dir_status + '/dial-complete.txt' # The file that indicates that a number is completed being dialed.
file_number   = dir_status + '/Dial/number.txt'   # The file containing the dialed numbers on the first line.
file_skit_on  = dir_status + '/skit-on.txt'       # The file that indicates that a skit should be played, beginning skit-player.py.
file_call_initiated  = dir_status + '/call-initiated.txt' # The file that indicates that the docent triggered a call to be made.
file_call_initiated_unattended  = dir_status + '/call-initiated-unattended.txt' # The file that indicates that the motion sensor triggered a call
to be made.
file_player_on = dir_scripts + '/player-on.txt'   # The file that indicates that that vlc should be activated and begin playing the selected
audio file.
file_temp      = dir_scripts + '/temp.py'         # The file that contains the commands to play the specified audio file.
file_skit_wait = dir_status + '/skit-wait.txt'    # The file that indicates that a skit should waint until action is performed.
action         = 'unknown'                        # This must be reset between file types, it is used to trigger search for input type (dial,
hook, audio, coin).

########################################################################
# Run
########################################################################
print('Running...')

# Main on and the skit file must be present
while True:
    if os.path.isfile(file_main):
        if os.path.isfile(file_call_initiated):

            # Time to wait before initiating a call from the time that the button is pushed.
            time.sleep(5)

            # Remove the initiated call.
            os.system('rm -r ' + file_call_initiated)

            # If there is not an ongoing call already.
            if not os.path.isfile(file_call_on):
                # Report that the skit is ongoing.
                with open(file_call_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(str(datetime.now()))

                # Select from one of the "incoming_with_docent" skits.
                # Get all CSV files in the directory that start with "incoming_call_docent"
```

```
            csv_files = [f for f in os.listdir(dir_audio) if f.startswith("incoming_call") and f.endswith(".csv")]
            if csv_files:  # check if the list is not empty
                # Randomly select one of the CSV files
                file_skit = random.choice(csv_files)

                # Indicate to the skit script that should be played.
                with open(file_skit_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(file_skit)

    # Determine the skit to play if it is activated by the motion sensor.
    if os.path.isfile(file_call_initiated_unattended):

        # Time to wait before initiating a call from the time that the button is pushed.
        time.sleep(10)

        # Remove the initiated call.
        os.system('rm -r ' + file_call_initiated_unattended)

        # If there is not an ongoing call already.
        if not os.path.isfile(file_call_on):

            # Select from one of the "incoming_with_docent" skits.
            # Get all CSV files in the directory that start with "incoming_call_docent"
            csv_files = [f for f in os.listdir(dir_audio) if f.startswith("incoming_call_unattended") and f.endswith(".csv")]
            if csv_files:  # check if the list is not empty
                # Randomly select one of the CSV files
                file_skit = random.choice(csv_files)
                # Indicate to the skit script that should be played.
                with open(file_skit_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(file_skit)

time.sleep(0.2)
```

Python code: **skit-player.py** Plays the skit outlined by skit-selector.py

```
#!/usr/bin/python3
##############################################################################
# Description
##############################################################################
# skit-player.py (v1.03) when activated by an ongoing skit, this program
# searches for the skit csv instructions to initiate the skit.
##############################################################################
# Load libraries
##############################################################################

import time
from datetime import datetime
import csv
import os
import re
import random               # For selecting a random file to play from.
#import pandas as pd         # For opening the skit file to be read.
# from mutagen.mp3 import MP3 # For determining the length of the mp3 file.


##############################################################################
# User input variables
##############################################################################

config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

##############################################################################
# Initialize
##############################################################################
print('Initializing')

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']
        if row['variable'] == 'dir_effects':
            dir_effects = row['directory']
        if row['variable'] == 'dir_scripts':
            dir_scripts = row['directory']
        if row['variable'] == 'dir_audio':
            dir_audio = row['directory']
        if row['variable'] == 'dir_skits':
            dir_skits = row['directory']

file_main     = dir_status + '/main-on.txt'       # The file that indicates the main switch is toggled to the on position.
file_hook_on  = dir_status + '/hook-on.txt'       # The file that indicates the handset was picked up, the hook is toggled to the on position.
file_hook_off = dir_status + '/hook-off.txt'      # The file that indicates the handset was placed down, the hook is toggled to the off position.
file_call_on  = dir_status + '/call-on.txt'       # The file that indicates there is already a call in progress.
file_dialed   = dir_status + '/dial-complete.txt' # The file that indicates that a number is completed being dialed.
file_number   = dir_status + '/Dial/number.txt'   # The file containing the dialed numbers on the first line.
file_skit_on  = dir_status + '/skit-on.txt'       # The file that indicates that a skit should be played, beginning skit-player.py.
file_player_on = dir_scripts + '/player-on.txt'   # The file that indicates that that vlc should be activated and begin playing the selected
audio file.
file_temp      = dir_scripts + '/temp.py'         # The file that contains the commands to play the specified audio file.
file_skit_wait = dir_status + '/skit-wait.txt'    # The file that indicates that a skit should waint until action is performed.
action         = 'unknown'                        # This must be reset between file types, it is used to trigger search for input type (dial,
hook, audio, coin).
```

```
# Make sure the skit knows to move on regardless of the condition from the last skit status.
if os.path.isfile(file_skit_wait):
    os.system('rm -r ' + file_skit_wait)
########################################################################
# Run
########################################################################
print('Running...')

# Set a trigger so that the phone must be picked up and hung up for it to count as a needing to hang up.
hook_off_override = False
first_loop = True

# Main on and the skit file must be present
while True:
    if os.path.isfile(file_main):
        if os.path.isfile(file_skit_on):
            time.sleep(0.1) # Pause to make sure the file has enough time to be written.
            # Determine which skit file should be played, retrieve this from file_skit_on.
            with open(file_skit_on, 'r') as file:
                file_skit = dir_audio + '/' + file.readline().strip()

            # Determine if it is an incoming call as "incoming_with" -- for incoming_with and incoming_without will be detected by this search.
            if first_loop == True:
                first_loop = False
                if "incoming_call" in file_skit:
                    hook_off_override = True

            # Remove the file once the data has been extracted.
            if os.path.isfile(file_skit_on):
                os.system('rm -r ' + file_skit_on)

            # Determine all the data in the skit csv file.
            current_file = []
            first_file = []
            input_type = []
            input_value = []
            next_file = []

            with open(file_skit, 'r') as file:
                reader = csv.reader(file)
                next(reader) # skip header row
                for row in reader:
                    current_file.append(row[0])
                    first_file.append(row[1])
                    input_type.append(row[2])
                    input_value.append(row[3])
                    next_file.append(row[4])

            # Determine what the starting point is.
            # Loop through first_file column and assign corresponding current_file to file_active
            for i in range(len(first_file)):
                if first_file[i] == 'yes':
                    file_active = current_file[i]
                    break # Stop looping once first "yes" response is found

            # Remove the skit wait file if it is present.
            if os.path.isfile(file_skit_wait):
                os.system('rm -r ' + file_skit_wait)

            # Begin loop that determines which skit should be played and if the skit is done running.
            complete = False
            while complete == False:
                if not os.path.isfile(file_skit_wait):

                    # Make the skit wait until the actions are done.
                    with open(file_skit_wait, "w") as job_file: # xhost + local: # to give the permissions
                        job_file.write(str(datetime.now()))
                        print('wait file made')
                    action = 'unknown'

                    print(file_active)
                    # check if current_file is an mp3 file
                    if os.path.splitext(file_active)[1] in ('.mp3','.wav', '.m4a'):
                        os.system('killall -9 vlc')
                        print('playing ' + file_active)
                        # Initiate the audio file to begin being played.
                        with open(file_temp, "w") as job_file:
                            job_file.write("#!/usr/bin/python3\n")
                            job_file.write("import os\n")
                            #job_file.write("import time\n")
                            #job_file.write("media = '"+prog_dir_curr[0]+"'\n")
                            file = dir_audio + '/' + file_active
                            job_file.write("os.system('cvlc --no-xlib --aout=alsa " + file + "')\n")
                            #job_file.write("time.sleep(15)\n")
                            #job_file.write("os.system('if ! pgrep vlc > /dev/null; then rm skit_file; fi')\n")
                            #job_file.write("os.system('rm -r " + file_skit_wait + "')")
                        # Indicate to the player script that it should begin playing the program.
                        with open(file_player_on, "w") as job_file: # xhost + local: # to give the permissions
                            job_file.write(str(datetime.now()))
                        time.sleep(4)
                        #print('#1')
                    #print('#2')
                    # Check if the current file is a python script, then make it appropriately.
                    if os.path.splitext(file_active)[1] in ('.py'):
                        # Initiate the audio file to begin being played.
                        with open(file_temp, "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")
```

```
                    file = dir_audio + '/' + file_active
                    job_file.write("os.system('chmod +x " + file + "')\n")
                    job_file.write("os.system('python3 " + file + "')\n")
                    job_file.write("os.system('rm -r " + file_skit_wait + "')\n")
                # Indicate to the player script that it should begin playing the program.
                with open(file_player_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(str(datetime.now()))

            # If necessary actions are NOT met for the skit to move on, then wait until they are to continue.
            #print('#3')
            if os.path.isfile(file_skit_wait):
                #print('#4')
                print('skit_waiting for response')

                # Determine what type of action must be met.
                if action == 'unknown':
                    #print('#5')
                    for ii in range(0,len(current_file)):
                        if current_file[ii] == file_active:
                            action = input_type[ii]

                # If the action is still unknown, then wait for the phone to be hung up, it must be the last file to be played.
                if action == 'unknown':
                    while os.path.isfile(file_hook_on):
                        time.sleep(0.5)

                # If hook on is the action item.
                if action == 'none':
                    # Determine what the next current file should be.
                    for ii in range(0,len(current_file)):
                        if current_file[ii] == file_active:
                            if input_value[ii] == 'none':
                                while os.path.isfile(file_skit_wait):
                                    time.sleep(1)
                                os.system('killall -9 vlc')
                                # Update the active file to move on to the next item in the list.
                                file_active = next_file[ii]

                # If hook on is the action item.
                if action == 'hook':
                    # Determine what the next current file should be.
                    for ii in range(0,len(current_file)):
                        if current_file[ii] == file_active:
                            if os.path.isfile(file_hook_on):
                                # Tell the skit file to move on.
                                os.system('rm -r ' + file_skit_wait)
                                os.system('killall -9 vlc')
                                # Update the active file to move on to the next item in the list.
                                file_active = next_file[ii]

                # If number must be dialed.
                if action == 'dial':
                    print('Searching for dial response')
                    # Check if the file exists
                    if os.path.exists(file_number):
                        print('number.txt exists')
                        # Get the modification time of the file
                        mtime = os.path.getmtime(file_number)
                        # Check if the file was modified more than 3 seconds ago
                        if time.time() - mtime > 2:
                            os.system('rm -r ' + file_skit_wait)
                            os.system('killall -9 vlc')
                            # Read the contents of the file and save them in number_dialed
                            number_dialed = 1234567890
                            with open(file_number, "r") as f:
                                print('number.txt being opened')
                                # Extract the number that was dialed.
                                number_dialed = f.read().strip()
                            print('Number dialed: ',str(number_dialed))
                            # Determine which file is associated with the number.
                            file_active1 = file_active
                            for ii in range(0,len(current_file)):
                                if current_file[ii] == file_active:
                                    if number_dialed == input_value[ii]:
                                        file_active1 = next_file[ii]
                                        print('Number dialed and located: ' + str(number_dialed))
                            if file_active1 == file_active:
                                file_active = 'wrong-number.m4a'
                            else:
                                file_active = file_active1
                    # If audio must be captured.

                    # If coints must be deposited.

            # Wait until the phone is picked up before turning the override to the off position.
            if os.path.isfile(file_hook_on):
                hook_off_override = False

            if hook_off_override == False:
                if os.path.isfile(file_hook_off):
                    first_loop = True # Resets the the trigger needed to override the hook for the first time.
                    complete = True
                    os.system('rm -r ' + dir_status + '/skit*')
            time.sleep(0.2)

    time.sleep(0.2)
```

Python code: **audio-selector.py** determines what audio files should be played for outgoing calls.

```
#!/usr/bin/python3
########################################################################
# Description
########################################################################
# audio-selector.py (v1.05) determines what content to play and for how long.
# This script makes the selection for what to play.
########################################################################
# Load libraries
########################################################################

import time
from datetime import datetime
import csv
import os
import re
import random # For selecting a random file to play from.

########################################################################
# User input variables
########################################################################

config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

########################################################################
# Initialize
########################################################################
print('Initializing')

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']
        if row['variable'] == 'dir_effects':
            dir_effects = row['directory']
        if row['variable'] == 'dir_scripts':
            dir_scripts = row['directory']
        if row['variable'] == 'dir_audio':
            dir_audio = row['directory']
        if row['variable'] == 'dir_skits':
            dir_skits = row['directory']

file_main     = dir_status + '/main-on.txt'       # The file that indicates the main switch is toggled to the on position.
file_hook_on  = dir_status + '/hook-on.txt'       # The file that indicates the handset was picked up, the hook is toggled to the on position.
file_hook_off = dir_status + '/hook-off.txt'      # The file that indicates the handset was placed down, the hook is toggled to the off position.
file_call_on  = dir_status + '/call-on.txt'       # The file that indicates there is already a call in progress.
file_dialed   = dir_status + '/dial-complete.txt' # The file that indicates that a number is completed being dialed.
file_number   = dir_status + '/Dial/number.txt'   # The file containing the dialed numbers on the first line.
file_skit_on  = dir_status + '/skit-on.txt'       # The file that indicates that a skit should be played, beginning skit-player.py.
file_player_on = dir_scripts + '/player-on.txt'   # The file that indicates that that vlc should be activated and begin playing the selected
audio file.
file_temp     = dir_scripts + '/temp.py'          # The file that contains the commands to play the specified audio file.

dial_tone = False
call_in_progress = False

########################################################################
# Run
########################################################################
print('Running...')

# Main on and hook on must both be present
while True:
    if os.path.isfile(file_main):
        if os.path.isfile(file_hook_on):
            # If there is not a call already in process.
            if not os.path.isfile(file_call_on):
                print('Playing dial tone')
                dial_tone = True
                # Begin playing the program.
                with open(file_temp, "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")
                    job_file.write("os.system('cvlc --no-xlib --aout=alsa /home/pi/Desktop/VinTEL/Bin/Sound-Effects/dial_tone.mp3')")

                # Indicate to the player script that it should begin playing the program.
                with open(file_player_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(str(datetime.now()))
                # Indicate that there is a call ongoing.
                with open(file_call_on, "w") as job_file: # xhost + local: # to give the permissions
                    job_file.write(str(datetime.now()))
            # If a number is being dialed and the dial tone is on, then kill the dial tone.
            if os.path.isfile(file_number):
                print('A number is being dialed. Dial tone stopped.')
                if dial_tone == True:
                    dial_tone = False
                    os.system('killall -9 vlc')
            # If a number is completed being dialed, then find the appropriate audio file to play.
            if os.path.isfile(file_dialed):
                time.sleep(0.2)
                if call_in_progress == False:
                    call_in_progress = True
                    print('A number is completed being dialed. Searching for an appropriate skit or audio file.')
                    # Read in the number that was dialed.
                    num_dialed = 'wrong-number'
                    try:
                        with open(file_number, 'r') as f:
```

```
                    num_dialed = f.readline().strip()
                except FileNotFoundError:
                    # Handle the case where the file doesn't exist
                    num_dialed = 'wrong-number' # Set a default value or take some other action

                # Determine which file matches the numbers that were dialed.
                # Construct the search pattern based on the num_dialed
                pattern = "^" + num_dialed + "[_.]"
                print(num_dialed)

                # Search for files in the directory that match the pattern
                #matching_files = 'wrong-number.m4a'
                matching_files = [filename for filename in os.listdir(dir_audio) if re.search(pattern, filename)]
                if not matching_files:
                    matching_files.append('wrong-number.m4a')

                # Print the list of matching files
                print("Matching files:")
                for filename in matching_files:
                    print(os.path.join(dir_audio, filename))

                # Determine if there is a script to be played (indicated by a csv file being present in the list).
                csv_found = any(file.endswith('.csv') for file in matching_files)
                if csv_found:
                    # Determine which file is the skit file csv, to be saved in the file_skit_on file.
                    for file in matching_files:
                        if file.endswith('.csv'):
                            file_skit = file

                    # Indicate to the skit script that should be played.
                    with open(file_skit_on, "w") as job_file: # xhost + local: # to give the permissions
                        job_file.write(file_skit)

                else:
                    # Dial the matching number and play the audio file.
                    with open(file_temp, "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")
                        file = dir_audio + '/' + random.choice(matching_files)
                        job_file.write("os.system('cvlc --no-xlib --aout=alsa " + file + "')")

                    # Indicate to the player script that it should begin playing the program.
                    with open(file_player_on, "w") as job_file: # xhost + local: # to give the permissions
                        job_file.write(str(datetime.now()))


        else:
            # If the phone is recently placed on the hook, then stop all recordings
            if os.path.isfile(file_hook_off):
                call_in_progress = False
                # Cancel the skit if it was initiated.
                #if os.path.isfile(file_skit_on):
                #    os.system('rm -r ' + file_skit_on)
                # Get the time of the last modification of the file
                file_time = os.path.getmtime(file_hook_off)
                # Get the current time
                current_time = time.time()
                # Check if the file was modified in the last 5 seconds
                if current_time - file_time < 1:
                    print(f"The file {file_hook_off} was written to in the last 5 seconds.")
                    os.system('killall -9 vlc')
                    os.remove(file_call_on) if os.path.exists(file_call_on) else None
                # else:
                    # print(f"The file {file_hook_off} was not written to in the last 5 seconds.")
            else:
                print(f"The file {file_hook_off} does not exist.")
        time.sleep(0.2)

    else:
        time.sleep(0.2)
```

Python code: **temp.py** is a python script that is made by other python scripts to specify what action to take (play audio file or .py file). It is initiated by player.sh.

```
#!/usr/bin/python3
import os
os.system('cvlc --no-xlib --aout=alsa /home/pi/Desktop/VinTEL/Bin/Sound-Effects/dial_tone.mp3')
```

Python code: **coins.py** is script that tracks the coins that are inserted into the phone and sets them in the correct file location.

```
#!/usr/bin/python
##############################################################################
# Description
##############################################################################
# coins.py (v1.04) determines what coins were placed into the phone. After
```

```
# controller.py makes the files, then this script renames the files to have
# a time stamp added to the file name, and detects if the 25 cent and 10
# cent sensors went off, then this should just be a 10 cent.
#########################################################################
# Load libraries
#########################################################################

import os
import csv
import time
from datetime import datetime

#########################################################################
# Functions
#########################################################################

# Function that updates the lifetime coin count.
def increment_file_count(dir_path, increment,deposit_name):
    found_file = False
    for filename in os.listdir(dir_path):
        if filename.startswith(deposit_name + '-') and filename.endswith('.txt'):
            count_str = filename.split('-')[-1][:-4] # extract the count value from the filename
            try:
                count = int(count_str) # convert the count value to an integer
                new_count = count + increment # increment the count value
                new_filename = deposit_name + '-{}.txt'.format(new_count) # create the new filename
                os.rename(os.path.join(dir_path, filename), os.path.join(dir_path, new_filename)) # rename the file
                print("File '{}' renamed to '{}'".format(filename, new_filename))
                found_file = True
            except ValueError:
                print("Invalid count value in filename '{}': {}".format(filename, count_str))
    if not found_file:
        new_filename = deposit_name + '-{}.txt'.format(increment)
        with open(os.path.join(dir_path, new_filename), 'w') as f:
            print("Created new file '{}'".format(new_filename))

#########################################################################
# Initialize
#########################################################################

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']

# Make a directory that all the dated coins will go into.
dir_status_date = dir_status + '/Deposited-Coins-Temp'
if not os.path.exists(dir_status_date):
    os.makedirs(dir_status_date)

# Upon initial bootup, clear all the previous records of coin collections.
try:
    os.system('rm -r ' + dir_status_date + '/detected*')
except OSError:
    pass

try:
    os.system('rm -r ' + dir_status + '/deposited-daily*')
except OSError:
    pass

try:
    os.system('rm -r ' + dir_status + '/deposited-session*')
except OSError:
    pass

# Original file name.
file_5  = dir_status + '/detected-5-cent.txt'
file_10 = dir_status + '/detected-10-cent.txt'
file_25 = dir_status + '/detected-25-cent.txt'

# New file name.
file_5_new  = dir_status_date + '/detected-5-cent.txt'
file_10_new = dir_status_date + '/detected-10-cent.txt'
file_25_new = dir_status_date + '/detected-25-cent.txt'

#########################################################################
# Run
#########################################################################
print('running coins.py...')

tot = 0 # This will be a counter for the daily deposits.
tot_prev = 0
last_update_time = time.time() # This is required for updating the session deposits.

while True:
    if os.path.exists(file_5):
        os.rename(file_5, file_5_new[:-4] + '_' + datetime.now().strftime("%Y-%m-%d_%H-%M-%S") + '.txt')
        tot += 5
    if os.path.exists(file_10):
        file_10_time = os.path.getctime(file_10)
        os.rename(file_10, file_10_new[:-4] + '_' + datetime.fromtimestamp(file_10_time).strftime("%Y-%m-%d_%H-%M-%S") + '.txt')
        time.sleep(0.75)
        if os.path.exists(file_25):
            file_25_time = os.path.getctime(file_25)
            if (file_10_time - file_25_time) < 0.75:
```

```
                os.remove(file_25)
            tot += 10
        if os.path.exists(file_25):
            os.rename(file_25, file_25_new[:-4] + '_' + datetime.now().strftime("%Y-%m-%d_%H-%M-%S") + '.txt')
            tot += 25
        if tot != tot_prev: # If there is a new deposit, then update the deposit total file.
            # Update the daily total deposits.
            try:
                os.system('rm -r ' + dir_status + '/deposited-daily-' + str(tot_prev) + '.txt')
            except OSError:
                pass
            with open(dir_status + '/deposited-daily-' + str(tot) + '.txt', 'w') as f:
                pass

            # Update the lifetime total deposits.
            increment = tot - tot_prev
            increment_file_count(dir_status, increment,'deposited-lifetime-total')

            # Determine if it has been 5 minutes since the last deposite has been made. If so, start a new session.
            if (time.time() - last_update_time) > 300:
                try:
                    os.remove(dir_status + '/deposited-session-total.txt')
                    print("Deleted deposited-session-total.txt")
                except OSError:
                    pass
                tot_prev = 0
                tot = 0
                last_update_time = time.time()
            # Update the last deposited time to current.
            last_update_time = time.time()
            # Update the current session deposits.
            increment_file_count(dir_status, increment,'deposited-session-total')

            # Update the previous.
            tot_prev = tot

        time.sleep(0.1)
```

Python code: **controller.py** monitors the status of the phone and communicates to the other scripts what the state of the device is.

```
#!/usr/bin/python3
##############################################################################
# Description
##############################################################################
# controller.py (v1.04) determines what "status" the device is in based on
# the control box setting. Depending on the switches that are flicked, it
# will generate the status text files to communicate to the other scripts
# what the status is. It will also control some of the LED lights.
##############################################################################
# Load libraries
##############################################################################

import RPi.GPIO as GPIO
import csv
import time
from datetime import datetime
import subprocess
import threading
import re
import os

##############################################################################
# User input variables
##############################################################################

file_config = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

##############################################################################
# Initialize
##############################################################################

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
dir_dial   = '/home/pi/Desktop/VinTEL/Bin/Status/Dial'
with open(file_config) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']
            dir_dial = dir_status + '/Dial'

##############################################################################
# Define functions
##############################################################################

# Function that updates the configuration file to have the correct directories in the commands.
def update_config_file(csv_file_path):
    # Create an empty dictionary to store the mapping between variables and directories
    directory_mapping = {}

    # Open the CSV file and read its contents
    try:
        with open(csv_file_path, 'r') as csvfile:
```

```
            config_reader = csv.DictReader(csvfile)

            # Loop through each row in the CSV file
            for row in config_reader:
                # Store the mapping between the variable and the directory
                directory_mapping[row['variable']] = row['directory']
    except Exception as e:
        print("Error reading CSV file: " + str(e))
        return

    # Update the command_on and command_off columns
    updated_rows = []
    with open(csv_file_path, 'r') as csvfile:
        config_reader = csv.DictReader(csvfile)

        # Loop through each row in the CSV file
        for row in config_reader:
            # Update the command_on and command_off columns
            updated_row = {}
            for key, value in row.items():
                if key == 'command_on' or key == 'command_off':
                    # Replace the variable substring with the corresponding directory
                    for var, dir in directory_mapping.items():
                        value = value.replace(var, dir)
                updated_row[key] = value

            # Add the updated row to the list
            updated_rows.append(updated_row)

    # Save the updated config file
    output_file_path = os.path.join(os.path.dirname(csv_file_path), 'configure_temp.csv')
    try:
        with open(output_file_path, 'w', newline='') as csvfile:
            fieldnames = updated_rows[0].keys()
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()
            writer.writerows(updated_rows)
    except Exception as e:
        print("Error writing CSV file: " + str(e))
        return

# Function to check the state of an input pin.
def check_input_pin(pin, command_on, command_off, variable_map):
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    last_state = GPIO.input(pin)

    while True:
        state = GPIO.input(pin)

        if state != last_state:
            time.sleep(0.01)
            state = GPIO.input(pin)
            if state != last_state:
                last_state = state
                if state == GPIO.LOW:
                    # Perform command_off
                    # subprocess.run(command_off.split())
                    os.system(command_off)
                    print(f'Pin {pin} is LOW, performed command_off: {command_off}')
                else:
                    # Perform command_on
                    #subprocess.run(command_on.split())
                    os.system(command_on)
                    print(f'Pin {pin} is HIGH, performed command_on: {command_on}')
        time.sleep(0.01)

###############################################################################
# Initialize
###############################################################################

# Sleep to allow edits to be made before device powers on.
print('Initialization Complete. Entering waiting period...')
time.sleep(30)

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Correct the configuration file.
update_config_file(file_config)
file_config_temp = os.path.dirname(file_config) + '/configure_temp.csv'

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['variable'] == 'dir_status':
            dir_status = row['directory']

system_working_file = dir_status + '/system-working.txt'
dir_dial            = dir_status + '/Dial'

# Make directories if they do not already exist.
if not os.path.exists(dir_dial):
    os.makedirs(dir_dial)

# Indicate that initialization is complete.
with open(system_working_file, "w") as job_file: # xhost + local: # to give the permissions
    job_file.write(str(datetime.now()))
```

```
###############################################################################
# Run
###############################################################################

print('Beginning script..')

# Read in configure.csv
with open(file_config_temp, 'r') as csvfile:
    config_reader = csv.DictReader(csvfile)
    variable_map = {}  # Dictionary to map variables to directory paths
    threads = []
    for row in config_reader:
        if not row['pin']:
            print(f'Missing value in "pin" column for row {config_reader.line_num}')
            continue

        try:
            pin = int(row['pin'])
        except ValueError:
            print(f'Invalid value in "pin" column for row {config_reader.line_num}')
            continue

        gpio_type = row['type']
        command_on = row['command_on']
        command_off = row['command_off']

        # Iterate over the variable_map dictionary and perform variable substitution on command_on and command_off strings
        if variable_map:
            for variable, directory in variable_map.items():
                command_on = re.sub(variable, directory, command_on)
                command_off = re.sub(variable, directory, command_off)

        # If both the 'variable' and 'directory' values are not empty, replace the variable in the command_on and command_off strings with the
corresponding directory path
        variable = row['variable']
        directory = row['directory']
        if variable and directory:
            command_on = command_on.replace(variable, directory)
            command_off = command_off.replace(variable, directory)
            variable_map[variable] = directory

        if gpio_type == 'in':
            # Create a thread for this input pin check
            t = threading.Thread(target=check_input_pin, args=(pin, command_on, command_off, variable_map))
            threads.append(t)
            t.start()

    # Wait for all threads to finish before exiting
    for t in threads:
        t.join()
```

Python code: **motion-controller.py** Determines the state of the motion sensor and initiated incoming unattended calls if it is appropriate (a long enough duration of inactivity followed by activity, with an element of stochasticity to randomize if the calls will be made).

```
#!/usr/bin/python3
###############################################################################
# Description
###############################################################################
# motion-controller.py (v1.01) reads the data from the motion sensor. If
# there is motion detected, and the switch for motion activated skit is on,
# and it has been more than 5 minutes (as an example) since the last
# detection of motion, then give it a 1/3 random chance that there will
# be an unattended incomming call prompted.
###############################################################################
# Load libraries
###############################################################################

import os
import csv
import time
import RPi.GPIO as GPIO
import random # To determine if the motion should randomly call a new skit.
from datetime import datetime # For writing the files that motion was detected.

###############################################################################
# User input variables
###############################################################################

time_of_inactivity = 5/60                # Time (minutes) that the system would have to wait without detecting motion before there is a chance
of an incoming call with newly detected motion.
probability_of_incoming_call = 3/3    # Probability (0-1) that the newly detected motion prompts an incoming call. Values of 1 would result in a
call every time there is newly detected motion.

###############################################################################
# Inititalization
###############################################################################

# Open the configuration file and read the data
config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'
with open(config_file) as f:
```

```
    reader = csv.DictReader(f)
    for row in reader:
        if row['description'] == 'motion sensor':
            sensor_motion = int(row['pin'])
        elif row['variable'] == 'dir_status':
            dir_status = row['directory']


file_motion   = dir_status + '/detected-motion.txt' # The file that indicates that motion was detected.
file_motion_on = dir_status + '/motion-on.txt'
file_call_initiated_unattended  = dir_status + '/call-initiated-unattended.txt' # The file that indicates that the motion sensor triggered a call
to be made.
system_working_file = dir_status + '/system-working.txt'
file_main_on = dir_status + '/main-on.txt'

# Set up the input and output pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_motion, GPIO.IN)

# Reset the environment the first time the skit is set to run.
if os.path.isfile(file_motion):
    os.remove(file_motion)
if os.path.isfile(file_call_initiated_unattended):
    os.remove(file_call_initiated_unattended)

###############################################################################
# Run
###############################################################################

# Main on and motion sensor activeted incoming calls must be on
while True:
    if os.path.isfile(file_main_on):        # If the main switch is on
        if os.path.isfile(file_motion_on):  # If the switch to enable motin detected initiated calls is on.
            #if GPIO.input(sensor_motion):
            if os.path.isfile(file_motion):
                # Motion detected
                #print('Motion detected!')
                # Write motion detection file
                #with open(file_motion, "w") as job_file: # xhost + local: # to give the permissions
                #      job_file.write(str(datetime.now()))
                #     print('Motion detection file written!')
                # Check time since last motion detection file was created
                time_since_last_motion = time.time() - os.path.getmtime(file_motion)
                print(time_since_last_motion)
                if not os.path.exists(file_motion) or  time_since_last_motion > (time_of_inactivity*60):
                    num_rand = random.random()
                    print('Random number generated: ',str(num_rand))
                    # Reset the motion detected file so that it would have to wait another 5 minutes.
                    if os.path.exists(file_motion):
                        os.remove(file_motion)
                    # Last detection was more than 5 minutes ago, so initiate skit with 1/3 chance
                    if num_rand < probability_of_incoming_call:
                        #print(num_rand)
                        with open(file_call_initiated_unattended, "w") as job_file: # xhost + local: # to give the permissions
                            job_file.write(str(datetime.now()))
                            #print('Initiated skit unattended!')
            else:
                motion = 'not detected'
                #print('No motion detected')
        else:
            if os.path.isfile(file_call_initiated_unattended):
                os.remove(file_call_initiated_unattended)
    else:
        if os.path.isfile(file_call_initiated_unattended):
            os.remove(file_call_initiated_unattended)


    time.sleep(0.5)
```

Python code: **dialer.py** decodes the rotary dial to record what numbers were dialed.

```
#!/usr/bin/python3
###############################################################################
# Description
###############################################################################
# dialer.py (v1.06) decodes the pulses from the phone to determine what
# number was dialed. It generates numbers.txt file that contain the numbers
# dialed. It waits 0.3 seconds before it knows a dialed digit is completely
# returned to home. It waits 4 seconds before it calls a string of numbers
# completely dialed. After 10 seconds it will remove the dial to make sure
# dials are fresh. Also hanging up should remove the files.
###############################################################################
# Load libraries
###############################################################################

import os
import time

###############################################################################
# User input variables
###############################################################################

file_config = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'
# define the directory to monitor
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
```

```
dir_dial = '/home/pi/Desktop/VinTEL/Bin/Status/Dial'
file_dial_complete = dir_status + '/dial-complete.txt'
file_dial_number = dir_status + '/Dial/number.txt'
file_main = dir_status + '/main-on.txt'
file_hook = dir_status + '/hook-off.txt'

########################################################################
# Initialize
########################################################################
print('Initializing dialer.py...')

# initialize variable to count dial-on files
dial_on_count = 0
number_complete = False
time_active1 = time.time()
time_active2 = time.time()
file_deleted = False
hook_delete = False
residual_delete = True

########################################################################
# Run
########################################################################
print('Running dialer.py...')

# continuously monitor the directory for new files
while True:
    # Only run the script when the switch is flicked.
    if os.path.isfile(file_main):
        # If the phone is placed on the hook, then all records of numbers should be removed.
        if os.path.isfile(file_hook):
            if hook_delete == True:
                hook_delete = False
                residual_delete = True
                if os.path.isfile(file_dial_complete):
                    # if it exists, remove the file
                    os.remove(file_dial_complete)
                if os.path.isfile(file_dial_number):
                    # if it exists, remove the file
                    os.remove(file_dial_number)
                    print('Deleted dial file')
        if not os.path.isfile(file_hook):
            hook_delete = True
            if residual_delete == True:
                residual_delete = False
                # get the list of files in the directory
                file_list_del = os.listdir(dir_dial)

                # check if the directory is already empty
                if len(file_list_del) > 0:
                    for file in file_list_del:
                        os.remove(os.path.join(dir_dial, file))
            time_active1 = time.time()
            # get the list of files in the directory
            file_list = os.listdir(dir_dial)
            # filter the list to only include files containing "dial-on"
            dial_on_files = [f for f in file_list if "dial-on" in f]
            # if there are no dial-on files, reset the dial-on count and file creation time
            if not dial_on_files:
                dial_on_count = 0
                if file_deleted == False:
                    time_inactive = time_active1 - time_active2
                    if time_inactive > 4: # The time since inactive dialing to delete all current logs.
                        file_deleted = True
                        if os.path.isfile(file_dial_complete):
                            # if it exists, remove the file
                            os.remove(file_dial_complete)
                        if os.path.isfile(file_dial_number):
                            # if it exists, remove the file
                            os.remove(file_dial_number)
                            #print('Deleted dial file')

                # Determine if a number is complete from being dialed.
                if number_complete == True:
                    # Determine the current time.
                    time_now = time.time()
                    # Calculate how long ago the last file was created.
                    time_wait = time_now - time_last_file
                    if time_wait > 4:
                        # get the current time and format it as a string
                        current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
                        # create the new file and write the current time to the first line
                        with open(file_dial_complete, 'w') as f:
                            f.write(current_time + '\n')
                        number_complete = False

            # if there are dial-on files, check if they were created in rapid succession
            else:
                time_active2 = time.time()
                number_complete = True
                file_deleted = False
                # Determine the time of the last file that was created.
                sorted_files = sorted(file_list, key=lambda x: os.path.getctime(os.path.join(dir_dial, x)), reverse=True)
                time_last_file = os.path.getctime(os.path.join(dir_dial, sorted_files[0]))
                # Determine the current time.
                time_now = time.time()

                # Calculate how long ago the last file was created.
                time_wait = time_now - time_last_file
```

```
                # If it has been more than 0.3 seconds since the last file was generated, then calculate the number that was dialed.
                if time_wait > 0.3:
                    dial_on_count = len(dial_on_files) - 1 # This is the length - 1 because there is one extra pulse for each number to be
accounted for.
                    # The number 0 will create 10 pulses, so we will want to account for this.
                    if dial_on_count == 10:
                        dial_on_count = 0
                    for file in dial_on_files:
                        os.remove(os.path.join(dir_dial, file))
                    # check if number.txt exists
                    if os.path.isfile(os.path.join(dir_dial, "number.txt")):
                        # if it exists, append the dial-on count to the first line
                        with open(os.path.join(dir_dial, "number.txt"), "r+") as f:
                            contents = f.readlines()
                            contents[0] = contents[0].rstrip() + f"{dial_on_count}\n"
                            f.seek(0)
                            f.writelines(contents)
                    else:
                        # if it doesn't exist, create it and add the dial-on count to the first line
                        with open(os.path.join(dir_dial, "number.txt"), "w") as f:
                            f.write(f"{dial_on_count}\n")
    # wait for 0.1 seconds before checking again
    time.sleep(0.1)
```

Python code: **led-controller.py** reports the current state of the phone and displays it on the LED indicator lights of the control box.

```
#!/usr/bin/python3
##############################################################################
# Description
##############################################################################
# led-controller.py (v1.00) determines the current state of the system to
# have it turn on/off the LED lights appropriately.
##############################################################################
# Load libraries
##############################################################################

import time
from datetime import datetime
import csv
import os
import re
import RPi.GPIO as GPIO

##############################################################################
# User input variables
##############################################################################

config_file = '/home/pi/Desktop/VinTEL/Bin/Scripts/configure.csv'

##############################################################################
# Initialize
##############################################################################
print('Initializing')

# Define the GPIO pins to be used (these are the default if there is an issues with the configure file. Expect them to be optentially
overwritten.
# incoming_call_led = 11
# hook_led = 9
# motion_detected = 10
# cent_25_led = 17
# cent_10_led = 27
# cent_5_led = 22

# Open the configuration file and read the data
dir_status = '/home/pi/Desktop/VinTEL/Bin/Status'
with open(config_file) as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['description'] == 'system status':
            system_led_pin = int(row['pin'])
        elif row['description'] == 'incoming call':
            incoming_call_led = int(row['pin'])
        elif row['description'] == 'hook and dial':
            hook_led = int(row['pin'])
        elif row['description'] == 'motion detected':
            motion_detected = int(row['pin'])
        elif row['description'] == '5 cent':
            cent_5_led = int(row['pin'])
        elif row['description'] == '10 cent':
            cent_10_led = int(row['pin'])
        elif row['description'] == '25 cent':
            cent_25_led = int(int(row['pin']))
        elif row['variable'] == 'dir_effects':
            dir_effects = row['directory']
        elif row['variable'] == 'dir_scripts':
            dir_scripts = row['directory']
        elif row['variable'] == 'dir_audio':
            dir_audio = row['directory']
        elif row['variable'] == 'dir_skits':
            dir_skits = row['directory']
        elif row['variable'] == 'dir_status':
            dir_status = row['directory']
```

```
file_main     = dir_status + '/main-on.txt'       # The file that indicates the main switch is toggled to the on position.
file_hook_on  = dir_status + '/hook-on.txt'       # The file that indicates the handset was picked up, the hook is toggled to the on position.
file_hook_off = dir_status + '/hook-off.txt'      # The file that indicates the handset was placed down, the hook is toggled to the off position.
file_call_on  = dir_status + '/call-on.txt'       # The file that indicates there is already a call in progress.
file_dialed   = dir_status + '/dial-complete.txt' # The file that indicates that a number is completed being dialed.
file_number   = dir_status + '/Dial/number.txt'   # The file containing the dialed numbers on the first line.
file_skit_on  = dir_status + '/skit-on.txt'       # The file that indicates that a skit should be played, beginning skit-player.py.
file_player_on = dir_scripts + '/player-on.txt'   # The file that indicates that that vlc should be activated and begin playing the selected
audio file.
file_temp     = dir_scripts + '/temp.py'          # The file that contains the commands to play the specified audio file.
file_call_intitiated = dir_status + '/call-initiated.txt'
file_motion   = dir_status + '/detected-motion.txt'
file_5  = 'detected-5-cent'
file_10 = 'detected-10-cent'
file_25 = 'detected-25-cent'
dir_coin = dir_status + '/Deposited-Coins-Temp'

# Make a directory that all the dated coins will go into.
dir_coin_final = dir_status + '/Deposited-Coins'
if not os.path.exists(dir_coin_final):
    os.makedirs(dir_coin_final)

# Set up the GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(incoming_call_led, GPIO.OUT)
GPIO.setup(hook_led, GPIO.OUT)
GPIO.setup(motion_detected, GPIO.OUT)
GPIO.setup(cent_25_led, GPIO.OUT)
GPIO.setup(cent_10_led, GPIO.OUT)
GPIO.setup(cent_5_led, GPIO.OUT)

# Define a function to turn on a GPIO pin
def turn_on(pin):
    GPIO.output(pin, GPIO.HIGH)

# Define a function to turn off a GPIO pin
def turn_off(pin):
    GPIO.output(pin, GPIO.LOW)

# Define a function to check for the presence of a file
def file_exists(file_path):
    return os.path.exists(file_path)

######################################################################
# Run
######################################################################
print('Running...')



# Define the time interval for turning off the call-initiated GPIO pin
CALL_INITIATED_OFF_INTERVAL = 15

# Define the duration for turning on the detected-motion GPIO pin
DETECTED_MOTION_DURATION = 15

call_initiated_time = time.monotonic()

# Loop indefinitely
while True:
    # Check for the presence of call-initiated.txt
    if file_exists(file_call_intitiated):
        turn_on(incoming_call_led)
        call_initiated_time = time.monotonic()
        print('Call initiated')
    else:
        if time.monotonic() - call_initiated_time > CALL_INITIATED_OFF_INTERVAL or file_exists(file_hook_on):
            turn_off(incoming_call_led)

    # Check for the presence of hook-on.txt and hook-off.txt
    if file_exists(file_hook_on):
        turn_on(hook_led)
        print('Hook on')
    elif file_exists(file_hook_off):
        turn_off(hook_led)
        print('Hook off')

    # Check for the presence of detected-motion.txt
    if file_exists(file_motion):
        turn_on(motion_detected)
        print('Motion detected')
        #time.sleep(DETECTED_MOTION_DURATION)
        turn_off(motion_detected)

    # Check for the presence of coin files
    for coin_file in os.listdir(dir_coin):
        if coin_file.startswith(file_25):
            print('25 cents deposited')
            turn_on(cent_25_led)
            os.system('mv ' + dir_coin + '/' + coin_file + ' ' + dir_coin_final + '/' + coin_file)
        elif coin_file.startswith(file_10):
            print('10 cents deposited')
            turn_on(cent_10_led)
            os.system('mv ' + dir_coin + '/' + coin_file + ' ' + dir_coin_final + '/' + coin_file)
        elif coin_file.startswith(file_5):
            print('5 cents deposited')
            turn_on(cent_5_led)
            os.system('mv ' + dir_coin + '/' + coin_file + ' ' + dir_coin_final + '/' + coin_file)

    # Wait for a short time before checking again
```

```
time.sleep(0.1)
```

REQUIREMENTS:

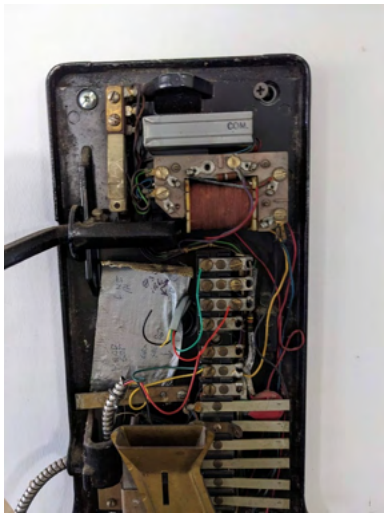
- Hardware Requirements
  - To set up the VinTEL system, you will need the following hardware components:
  - Raspberry Pi board (Raspberry Pi 3 or newer recommended)
  - Handset (microphone and speaker)
  - Motion sensor
  - LED indicator
  - Resistors, wires, and other necessary electronic components
  - Refer to the Hardware/Schematics directory for detailed circuit diagrams and wiring instructions.

- Software Requirements
  - The VinTEL system relies on the following software components:
    - Raspbian OS
    - Python 3
    - RPi.GPIO library
    - VLC media player

LICENSE:

BUILD DOCUMENTATION

4/2/2023

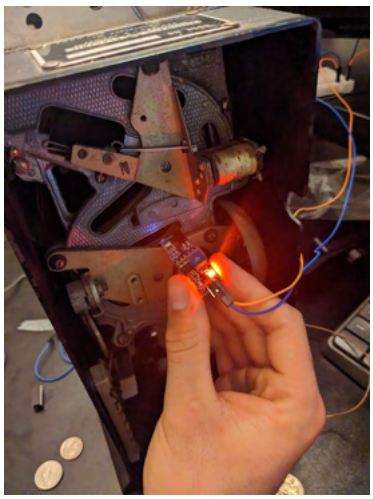


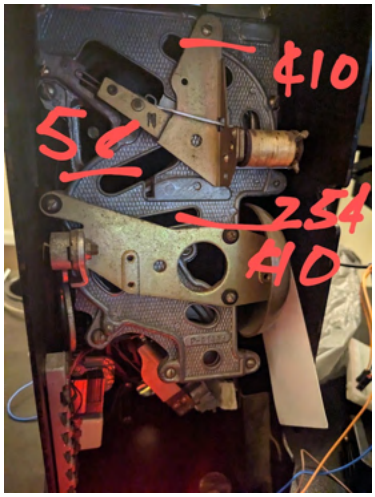
Wiring from when the phone was attached to "Tiger" Tom Ehrhart's wall.

4/3/2023

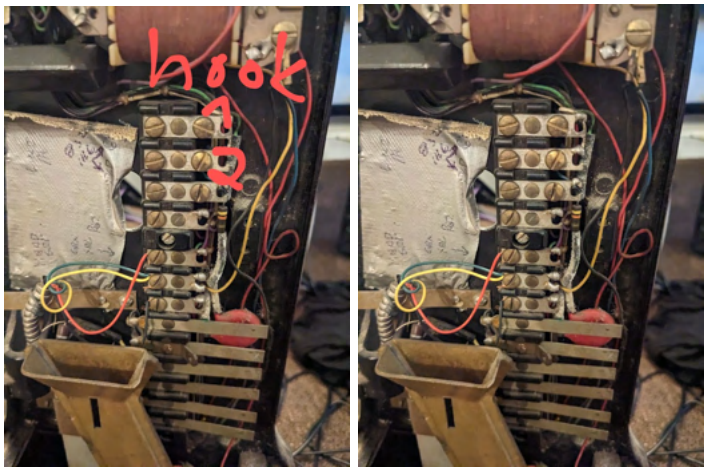
Video of path coins follow. Planning circuits and ordering parts.

4/4/2023



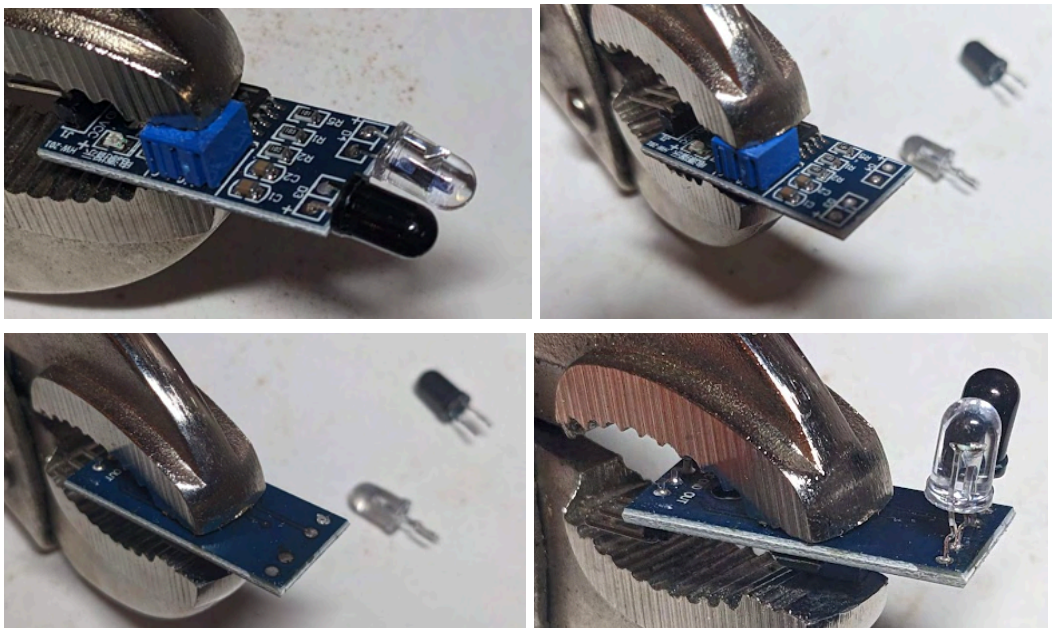


Testing IR sensors. The nickel and dime have unique paths, but the quarter follows a path in line with the dime.

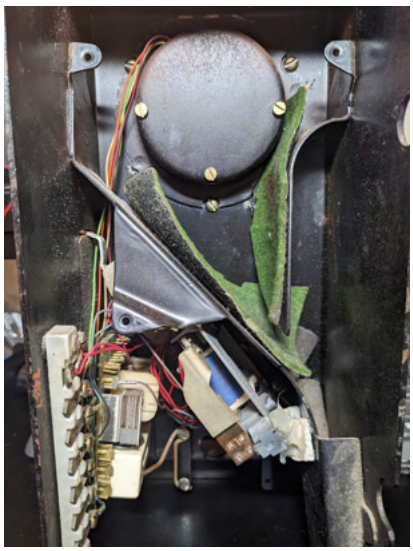


A continuity tester was used to determine which contacts will convey information if the phone is off the hook or not.

4/6/2023



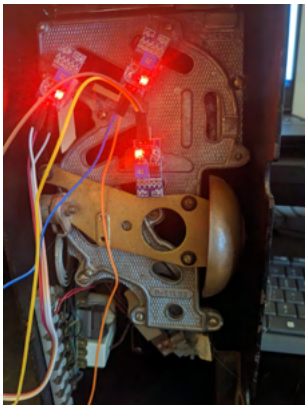
The HW-201 IR sensors are ideal for detecting coins, however, it is clear that the sensors are not oriented correctly to be mounted in the phone. IR light and sensor are desoldered, removed, and mounted on the reverse side of the control bracket. This will allow a secure site to mount while having the potentiometer sensitivity controller to be outward facing after being mounted.




Rejected coins call down the this hoot. It was clear that the felt backing was peeling off. Surfaces were cleaned and new glue was applied to keep the felt from peeling off. An additional piece of black fabric was added at the top left of the shoot to prevent coins from getting stuck at the edge of the green felt.

4/7/2023



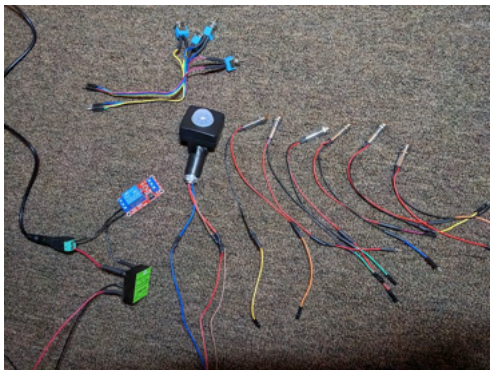



IR Sensors are mounted and tested for functionality. Additionally, a piece of wood is added to the shoot to keep nickels from getting dislodged and stuck in the shoot. It appears that after years of use there is enough wear that allows for some nickels to get stuck.

4/10/2023

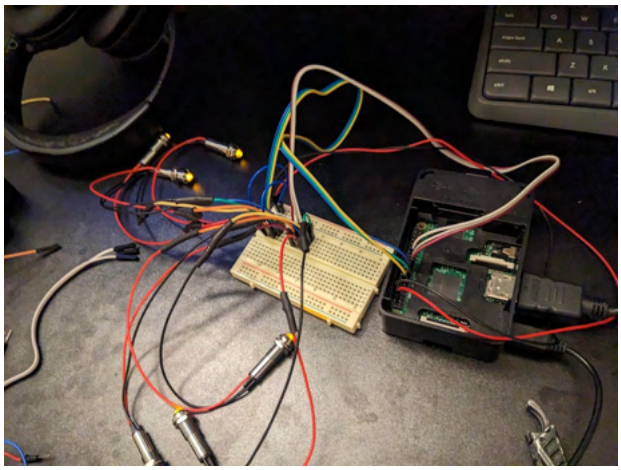
Ringer was made functional using Black Magic transformer.

4/11/2023



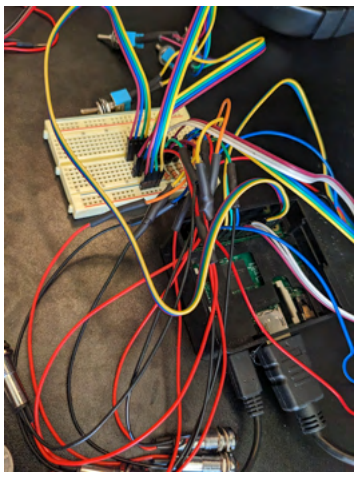
Soldering connections for LED indicator lights, proximity sensors, and switches.

4/12/2023



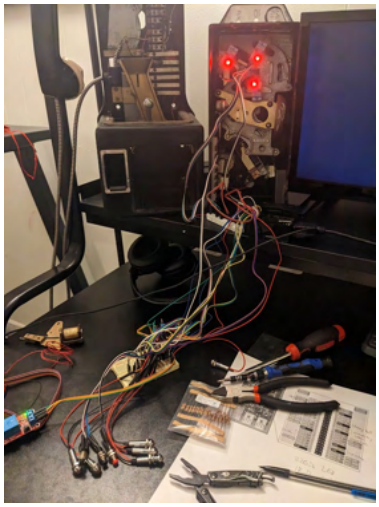
Testing configurations for LED indicator lights.

4/13/2023



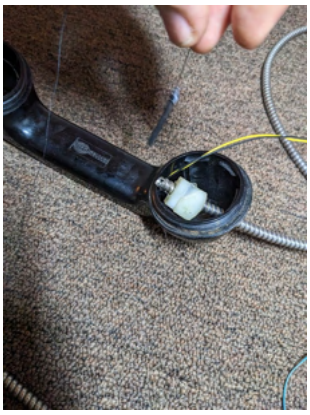
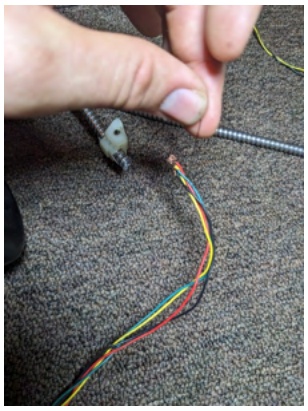
Assembling the wiring harness for the switches.

4/15/2023

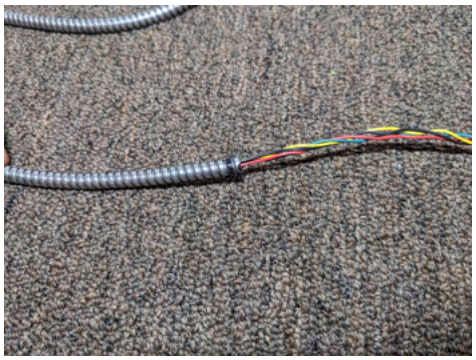
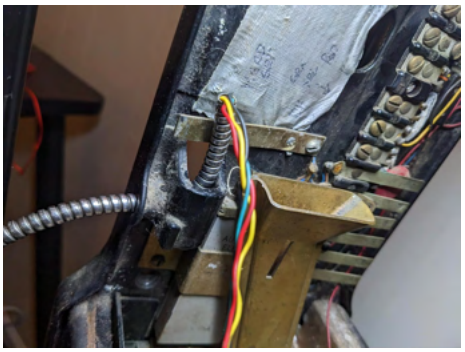


Testing wiring configuration for sensors and indicators/

4/16/2023

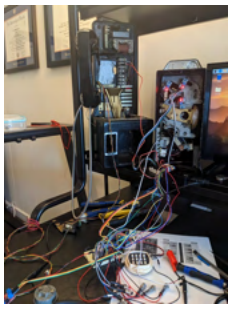




The metal tubing connecting the handset to the phone was too long (>9ft) which would hit the floor if dropped. The wire was cut to 3ft (Allowing for a 6ft remaining section to change the length in the future if needed). Center shows the process of re-threading the wire through the metal cable housing using a weighted rubbed attached to a thread. The wires are attached to the opposite end of the thread by knotting. When the weighted thread make it through the cable housing, the wires can then be pulled through threading them into their housing.

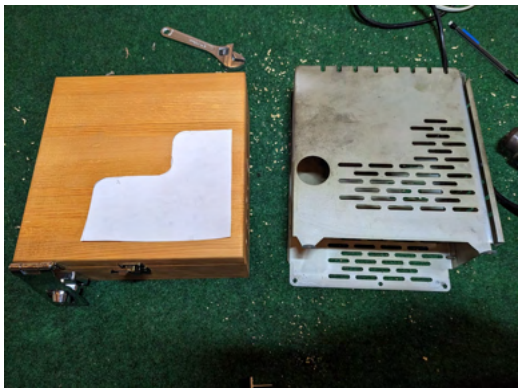
The threaded wires and housing are mounted to the phone body.

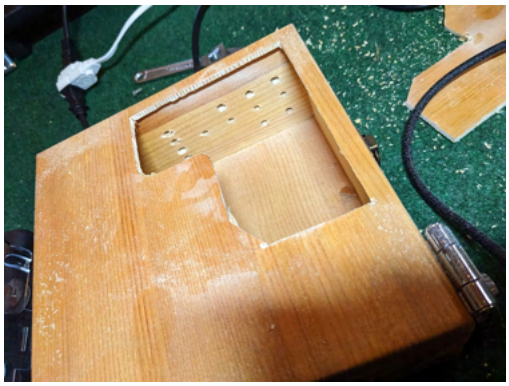
4/18/2023



4/25/2023

After measuring and planning, holes are drilled for the LED lights, switches, and button.



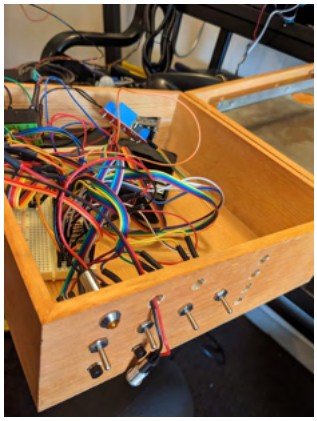
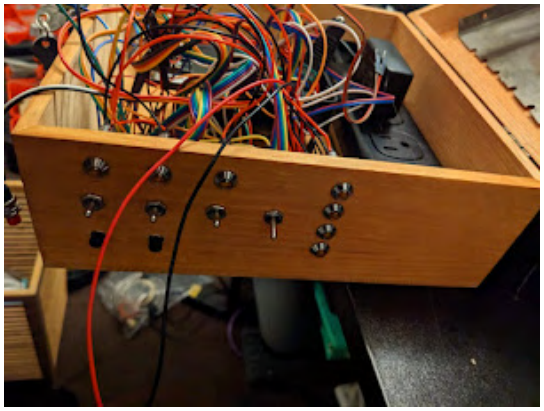


The top heat vent for the control box is cut. Remaining pieces from the AACA RCA TV are used as the vent guard.


Cuts for the hole for the power cable and the data cables are made into the side of the box. Also, GOOP is used to attached the heat vent guard.

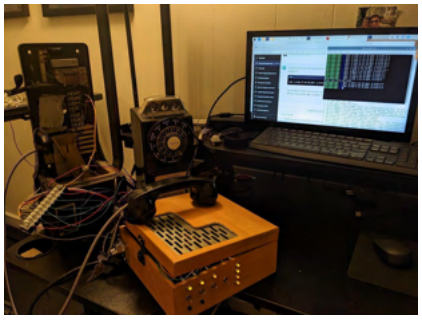
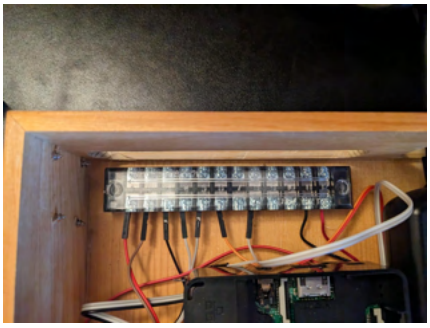
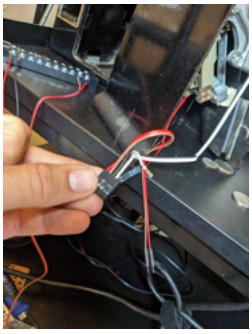
4/26/2023


A lock is mounted to the box to protect the electronics within and reduce tampering. Also pieces are sanded and finshed.

Installation of "Black Magic" to get the ringer to ring on demand. Center, installation of LEDs and switches in the box.Right, installation of switches and lights before cable management.

4/27/2023

Testing the LEDs in the first image. Installation of the screw terminals in the second image, and installation of Dupont connectors in the third image.

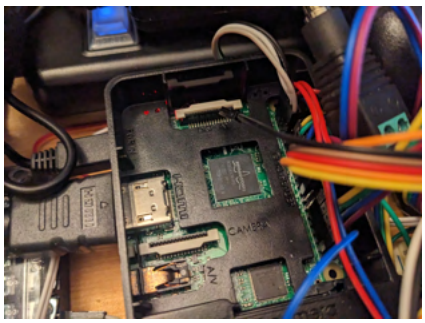
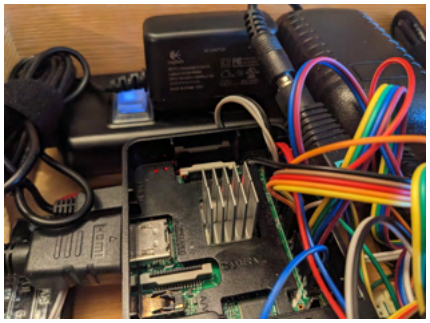
4/28/2023

Using photo editing software to adapt cartoons from old Western Electric advertisements into period-looking "phone book"

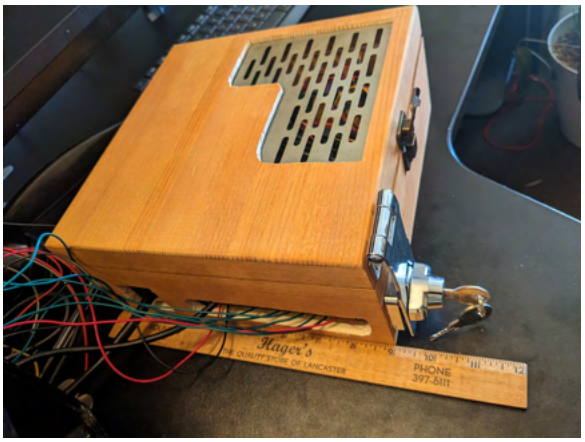
4/30/2023

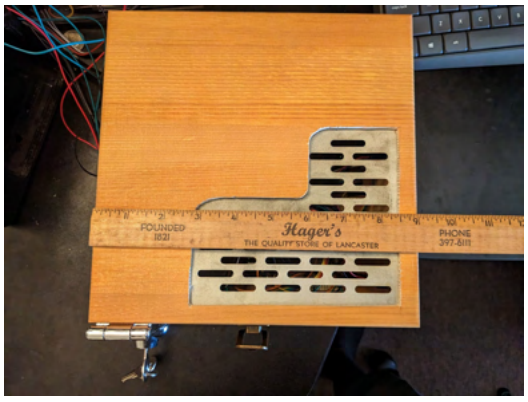
3D printing the motion sensor housing.

5/1/2023

Addition of thermal sink to the computer's CPU. This is adhered using thermal tape. The connection seems to be stronger than the thermal glue option that was tried previously.

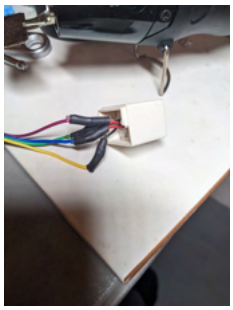


Bow without additions:
4x9x9

Box with additions:
4.5x10x10 without key
4.5x10x11.5 with key inserted

5/3/2023



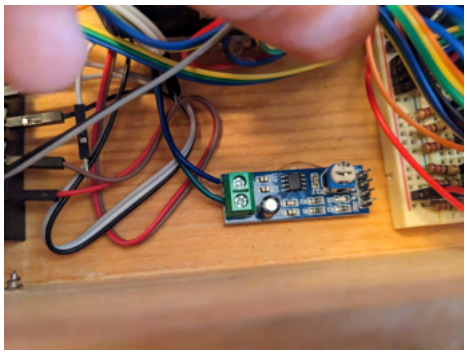


Preparing the motion sensor for adding the wiring connection plug by soldering, gluing, and painting the connection.

5/4/2023

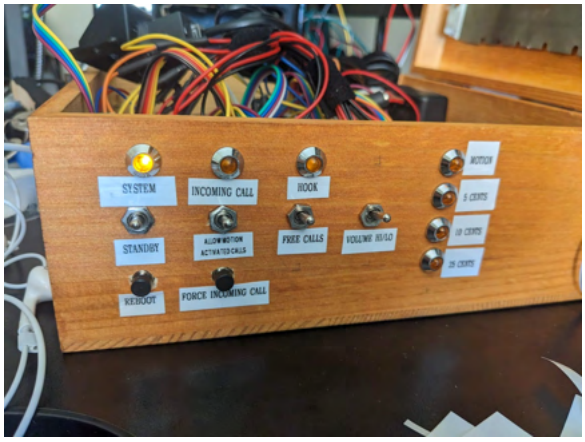
Soldering connections and building the housing for the remote switch.

5/9/2023

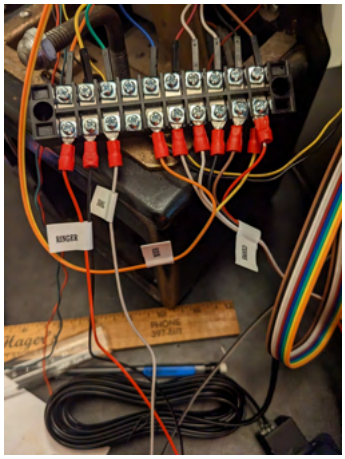


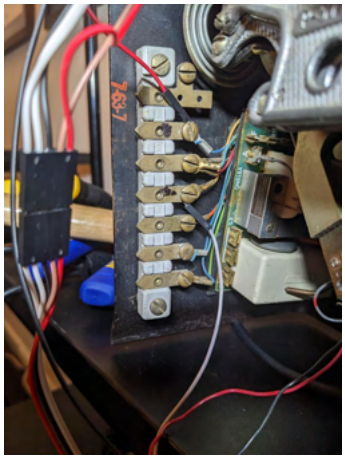
Preparing audio amplifier.

5/10/2023



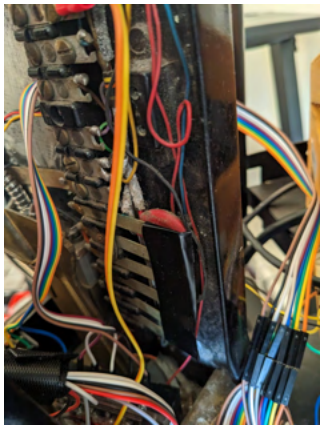
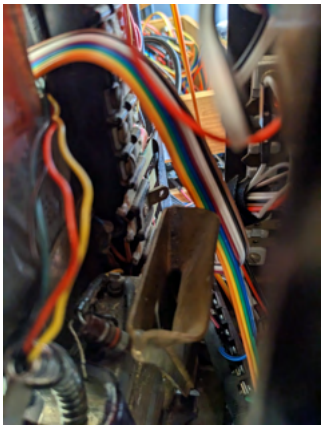
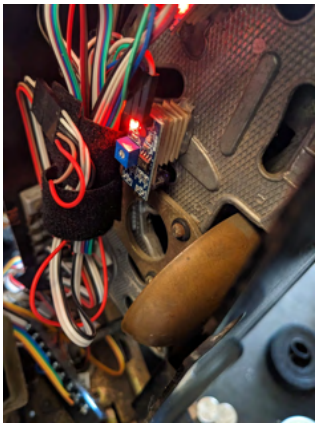
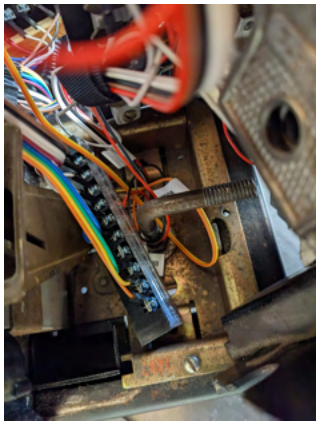
Controls were labeled.





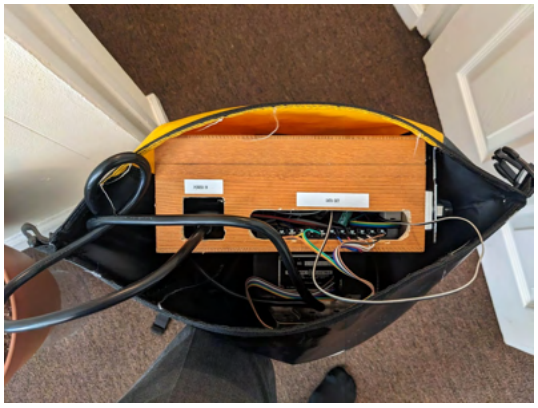
Inside the telephone body, fork connectors are used to connect wires to the screw terminals.
Center are the wires for the rotary dial. Right shows the screw driver turning the dial for the amplifier inside the control box.

5/11/2023

Wiring between the front of the phone and the back/bottom of the phone. Cable must be tightly velcroed in place, cables must be out of the coin shoots, screw terminals must be placed at the bottom and slightly angled, contactors must be taped (keep the electrical tape as-is shown in the first image blocking the contacts from being made). and dupont connector points are not in the way of being destroyed when closing.

2023-05-12




The telephone was biked into work where it was tested by students and technicians.

TROUBLESHOOTING: GENERAL

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

1. Reboot the machine using the reboot button on the control box. If the issue persists, continue on with this list.
2. Check that all connections are tightly made.
3. Unplug the surge protector from the outlet, wait 10 seconds, plug it into the outlet again.

If this does not solve the problem, go to the "Troubleshooting: Specific" section.

TROUBLESHOOTING: SPECIFIC

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.
    - Check that all power bricks are securely plugged into the surge protector.

Coin bell not ringing loudly:

- Problem: The bell is quiet when the coins hit them.
- Solution: It's likely that stray wires are dampening the bell. Open the phone body and move them to not be in contact with the bell. Consider using velcro (hook and loop) for cable management. To open this section, see the quick guide section "Serving the Phone -- Removing the Phone Front" and "Serving the Phone -- Returning the Phone Front".

Phone not picking up:

- Problem: Removing the phone from the hook has no response.
- Solution:
    - Check that the device standby mode is switched in the "on" position.
    - Reboot the machine using the reboot button on the control box. If the issue persists, continue on with this list.
    - Check that all connections are tightly made (nothing is clearly disconnected).
    - Check that the electrical tape on the connector within the phone is properly placed (if this tape is removed, the hook will not respond to changes).

Indicator lights not turning on:

- Problem: The LED lights on the control box are not responding.
- Solution:
    - Check that all connections are tightly made.
    - If it is one LED light that is not responding, it is likely a loose connection, a faulty sensor, or the LED is out. Troubleshoot these areas.

Mechanical ringer not ringing:

- Problem: Mechanical ringer is not sounding when expected.
- Solution: Check that the connections are tight and that the mechanical ringer power supply is connected.

Coins are jammed:

- Problem: The coin shoots are jammed.
- Solution: This could be caused by a foreign object obstruction or a wire in the way. Open the phone body and move them to not be in contact with the bell. Remove the foreign object if it is in the way. Consider using velcro (hook and loop) for cable management. To open this section, see the quick guide section "Serving the Phone -- Removing the Phone Front" and "Serving the Phone -- Returning the Phone Front".

Phone keeps ringing:

- Problem: The phone will ring without being prompted.
- Solution:
    - This is likely due to a faulty motion sensor.

- First check that all connections are tight to elimate this as a source and reboot the telephone.
- If the motion sensor continue to go off without expecting it, disable it by turning the toggle switch for "Motion activated calls" to the off position or unplugging the motion sensor.

Phone emits a pulsing sound:

- Problem: The phone makes a pulse sound every few seconds (or produces grey noise continuously).
- Solution:
  - The pulsing sound is a common feature of this device, it reminds visitors that the phone is there ready to be used, but it was not planned for.
  - The source of the pulsing probably comes from an unclean power supply disrupting the audio amplifier board. Consider swapping out the audio amplifier board if this becomes an issue.
  - There is a screw on the amplifier board that allows the gain to be adjusted, turn this up and down to trade off between (1) loud audio and (2) background gray noise to find a setting that is most acceptable.

DISCLAIMER

The telephone, electronics, and Vintage Telephone Emulating Landline (VinTEL) described in this manual are 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 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 "Tiger" Tom Ehrhart for the donation of the phone body, donation of wiring harness, consultation, and extensive mentorship. Orlando and Liz Ferreira contributed through financial donations and Orlando by editing this manuscript. Also a special thanks to the AACA Museum for supporting this work and facilitating the preservation of not only items but experiences.

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 television.

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