JAMsj Presence Activated Media Player

Designed by Sammy Pulos (Email: spulos@alumni.scu.edu)
April 13, 2020
Santa Clara University

Latest revision: August 16, 2020

Contents

Overall Description

This project aims to create a system that will detect when visitors are present and start playing a preselected video until no viewers are present, at which point it will begin to loop an idle video. To do this we use two sensors, one for motion and one for proximity, to detect presence in a determined area.

Developer

Part list

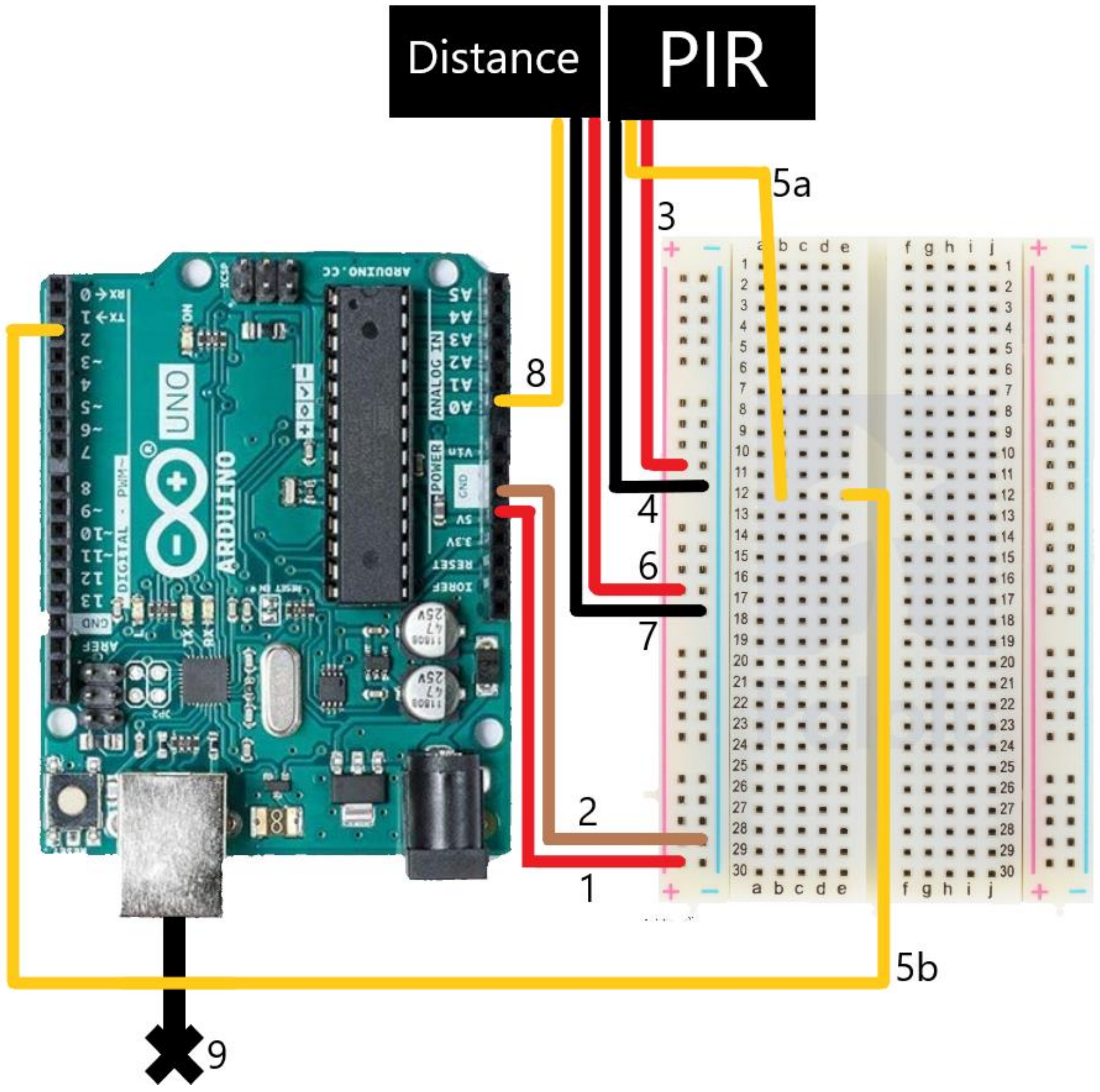
Raspberry Pi 3 Model 8+	(Other Raspberry Pi’s may be usable)
Arduino Uno - R3	(Other Arduino’s may be usable)
USB cable A to B	
Screen display	
HDMI cable	(Compatible with the display)
Speakers or headphones	(Optional, needed if audio output is desired)
Micro SDHC card	(May come with Raspberry Pi, see more about selection here)

Link 3 Male to Male jumper wires
Link Infrared Proximity Sensor Long Range - Sharp GP2Y0A02YK0F
Link Infrared Sensor Jumper Wire - 3-Pin JST
Link Adafruit Multiple Function Sensor Development Tools PIR (Motion) Sensor
Link Adafruit Accessories Half-size Breadboard

Connection diagram

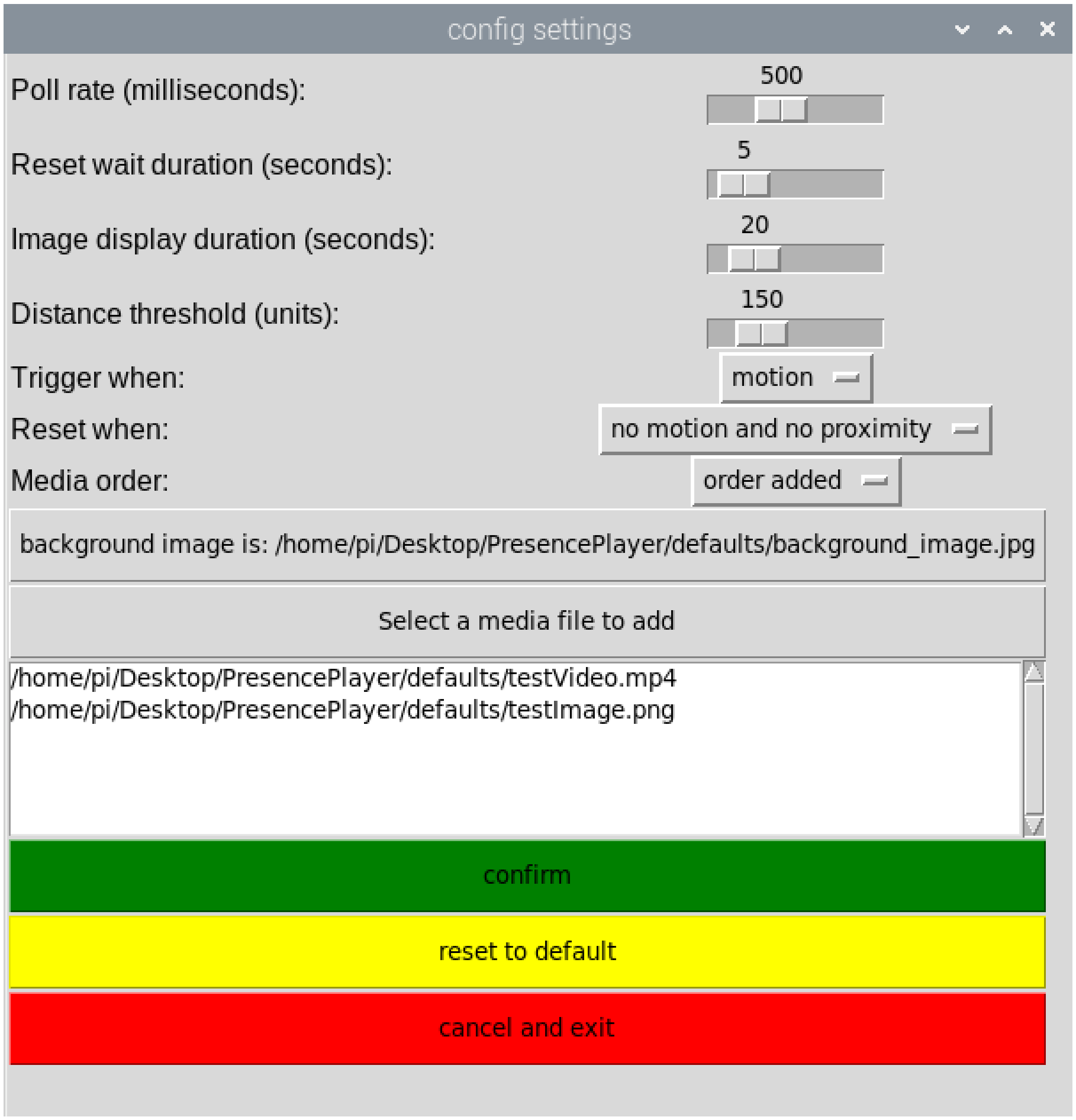
(Please ensure the Raspberry Pi & Arduino are not powered on when making the connections)

1. Arduino 5V Power pin to breadboard
2. Arduino GND Power pin to breadboard
3. PIR sensor VCC to 5V source on breadboard
4. PIR sensor ground to ground on breadboard
5. PIR sensor output to Arduino Digital Pin 2 (done on breadboard during development)
6. Distance sensor VCC to 5V source on breadboard
7. Distance sensor ground to ground on breadboard
8. Distance sensor output to Arduino Analog IN A0 (not done on breadboard in development)
9. Connect Arduino to Raspberry Pi using the USB cable listed previously



Software/GUI

The provided software programs consist of the video playing application (main.py, videoPlayer.py, arduino.py, and configParser.py) and a configuration GUI application (config.py) shown below


config settings

Poll rate (milliseconds): 500

Reset wait duration (seconds): 5

Image display duration (seconds): 20

Distance threshold (units): 150

Trigger when: motion

Reset when: no motion and no proximity

Media order: order added

background image is: /home/pi/Desktop/PresencePlayer/defaults/background_image.jpg

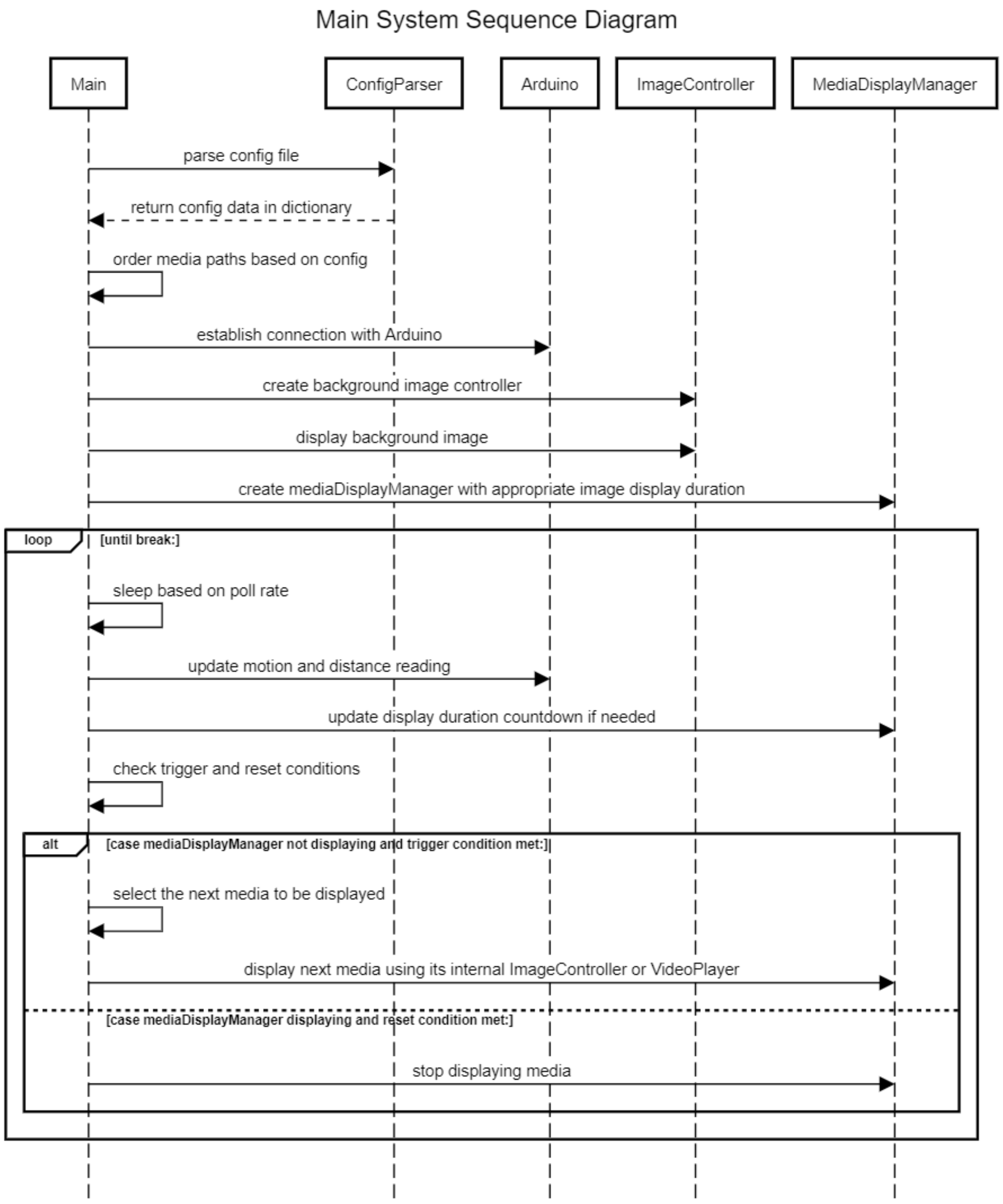
Select a media file to add

```
/home/pi/Desktop/PresencePlayer/defaults/testVideo.mp4
/home/pi/Desktop/PresencePlayer/defaults/testImage.png
```

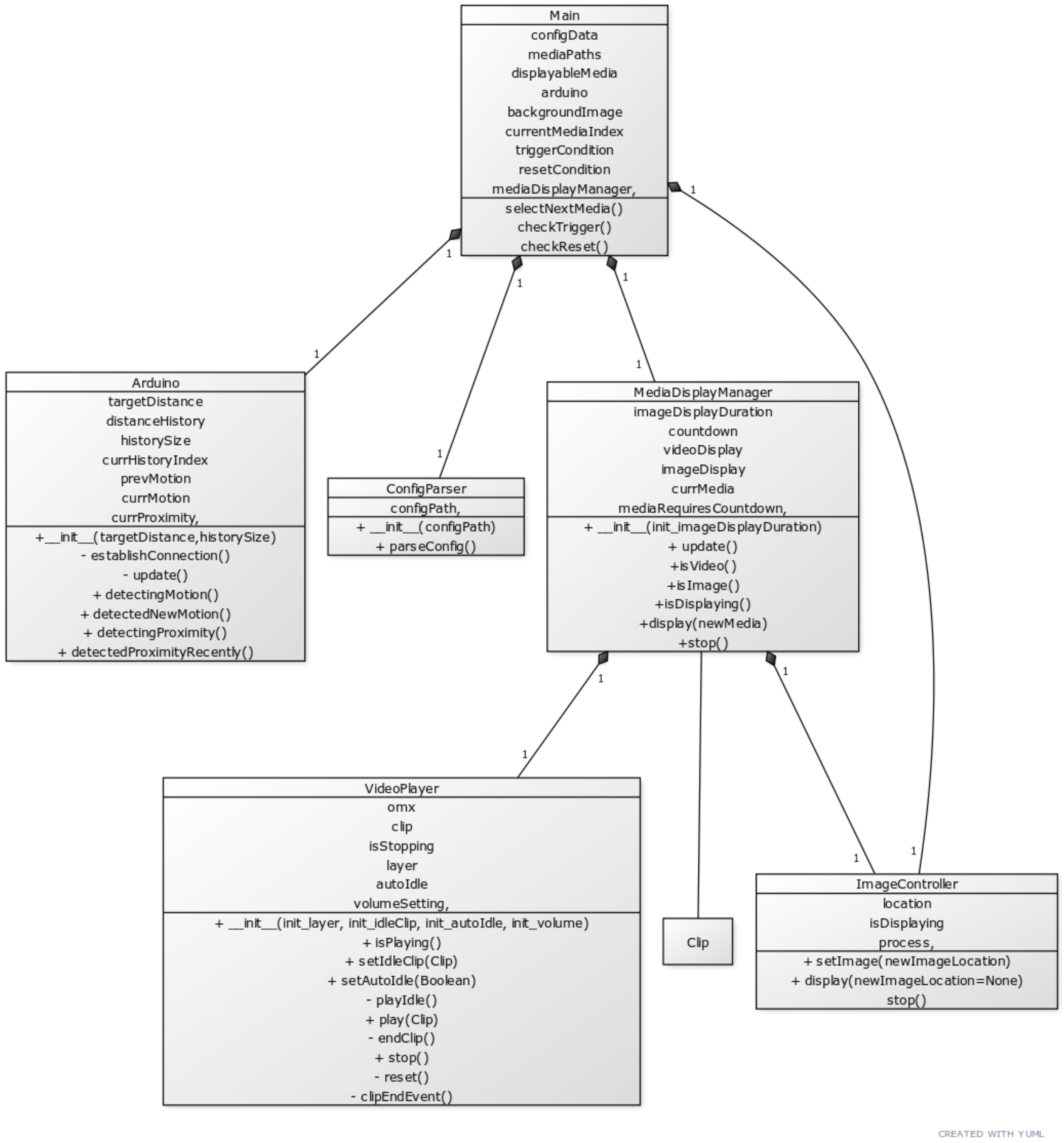
Code

The code can be found at: https://github.com/SammyPulos/PresenceActivatedMediaPlayer
The following diagrams overview the structure and behavior of the code

Sequence diagram: can be found here


Main System Sequence Diagram
Main
ConfigParser
Arduino
ImageController
MediaDisplayManager
parse config file
return config data in dictionary
order media paths based on config
establish connection with Arduino
create background image controller
display background image
create mediaDisplayManager with appropriate image display duration
loop
[until break:]
sleep based on poll rate
update motion and distance reading
update display duration countdown if needed
check trigger and reset conditions
alt
[case mediaDisplayManager not displaying and trigger condition met:]
select the next media to be displayed
display next media using its internal ImageController or VideoPlayer
[case mediaDisplayManager displaying and reset condition met:]
stop displaying media

Class diagram: can be found here



User

First time setup

Please note the Raspberry Pi must be connected to the Arduino which is connected to the sensors for the following steps to work.

1. Setting up Rasbian
   - Rasbian is needed for the Raspberry Pi to function, as such, many beginner Raspberry Pi's come with a micro SD card with Rasbian already installed. If not simply follow the tutorial found here to image Rasbian onto a microSD card. (During development an 8GB mircoSD card was used with Rasbian version 10 (buster), released 2020-05-27)
2. Setting up the program files
   - Copy all program files to the Raspberry Pi in a directory of your choice. This can be done conveniently using a USB. If you do not have the program files they can be found on github at: https://github.com/SammyPulos/PresenceActivatedMediaPlayer
3. Installing dependencies for the config program
   - Open a terminal window by pressing the icon on the taskbar of pressing ALT and F2 simultaneously.
   - Type:
     ```
     python3 -m pip install guizero
     ```
     then press enter and wait for the package to install
     (Note: installing the package this way requires an internet connection during the installation process)
4. Installing dependencies for the main program
   - Open a terminal window (or use the same window from step 3)
   - Type:
     ```
     python3 -m pip install omxplayer-wrapper
     ```
     then press enter and wait for the package to install (Note: installing the package this way requires an internet connection during the installation process)
   - Once the omxplayer-wrapper has been installed type:
     ```
     sudo apt-get install feh
     ```
     then press enter and wait for the install, if prompted with a Y/n option type Y and hit enter to proceed (Note: installing the package this way requires an internet connection during the installation process)
   - Type:
     ```
     python3 -m pip install system_hotkey
     ```
     then press enter and wait for the package to install (Note: installing the package this way requires an internet connection during the installation process)
   - Type:
     ```
     python3 -m pip install xcffib
     ```
     then press enter and wait for the package to install (Note: installing the package this way requires an internet connection during the installation process)

- Type:
  ```
  python3 -m pip install xpybutil
  ```
  then press enter and wait for the package to install (Note: installing the package this way requires an internet connection during the installation process)

5. Setting up the Arduino sketch
   - For the program to function properly it will need to communicate with the Arduino to find poll sensor values. This requires an Arduino sketch to be setup on the Arduino.
   - First be sure the pir and distance sensors are connected to the Arduino and that the Arduino is connected to the Raspberry Pi using a USB A to B cable.
   - Install the Arduino IDE on the Raspberry Pi by pressing the `ctrl` key, the `alt` key, and the `t` key all at the same time and then typing:
     ```
     sudo apt-get install arduino
     ```
     Then press the `Enter` key and wait for the installation to complete (pressing the `y` key and then the `Enter` key if/when prompted)
     (Note: installing the package this way requires an internet connection during the installation process)
   - Once the install is complete navigate to the arduino_sketch folder in the project directory, double click the arduino_sketch folder and then right click the enclosed arduino_sketch.ino file. A menu should appear, click the option “open with” then click the arrow next to the option “Electronics” then click the “Arduino IDE” option and finally click the “OK” button in the bottom right corner of the window.
   - Once the Arduino IDE has opened click the blue right-pointing arrow button near the top of the window and wait for the sketch to compile and upload. If you are prompted with a port warning simply click ok and click the blue right-pointing arrow button again.
   - Once the sketch has finished uploading the Arduino should be ready to go and you can close the IDE window.
6. Configuring the program
   - Once all dependencies are installed be sure to configure the program before running.
   - To do this first run Config.py (details on how to do this are included towards the end of the section “How to operate”).
   - Once open enter your desired configuration files (reference the section “settings” below as needed) or click the yellow button “reset to default”
   - Finally, click the green “confirm” button to confirm your changes and finally the red “cancel and exit” button to exit the Config program.
7. Setting up autoexec (optional, but recommended)
   - Following the following steps (based on those found here) will allow the program to start automatically when the Raspberry Pi is turned on, relieving the need for it to manually be started, a task which often requires a connected keyboard.
   - Open a terminal window and type:
     ```
     sudo nano /etc/profile
     ```
   - Navigate to the bottom of the file using the down arrow key then add the following line:
     ```
     python3 /path/Main.py &
     ```
     Where `/path` is the path to Main.py, for example when testing the added line was:
     ```
     python3 /home/pi/Desktop/PresencePlayer/Main.py &
     ```

- Finally hit the `ctrl` and `x` key together to exit, then the `y` key and then `Enter` twice
- Now, upon turning the Raspberry Pi on, the program should start automatically, the program can be exited using the `esc` key, allowing for normal Raspberry Pi use. Please note that if the project files are moved to a different directory the `path` in `python3 /path/Main.py &` will need to be updated to reflect the new file location

8. Disabling screen blanking
   - If the system is left idle for a set duration the background display will go blank/black and fail to display any selected images. To prevent this screen blanking must be disabled on the Raspberry Pi.
   - To disable screen blanking click on the Raspberry Pi icon in the top left corner of the screen and then click "preferences" from the drop-down menu and then click on the "Raspberry Pi configuration" option. This will pull up a new window for Raspberry Pi configuration. Click on the "Display" tab in this window then select "Disable" on the row for screen blanking and finally click the "OK" button in the bottom right corner. This will require a reboot of your Raspberry Pi to take effect.

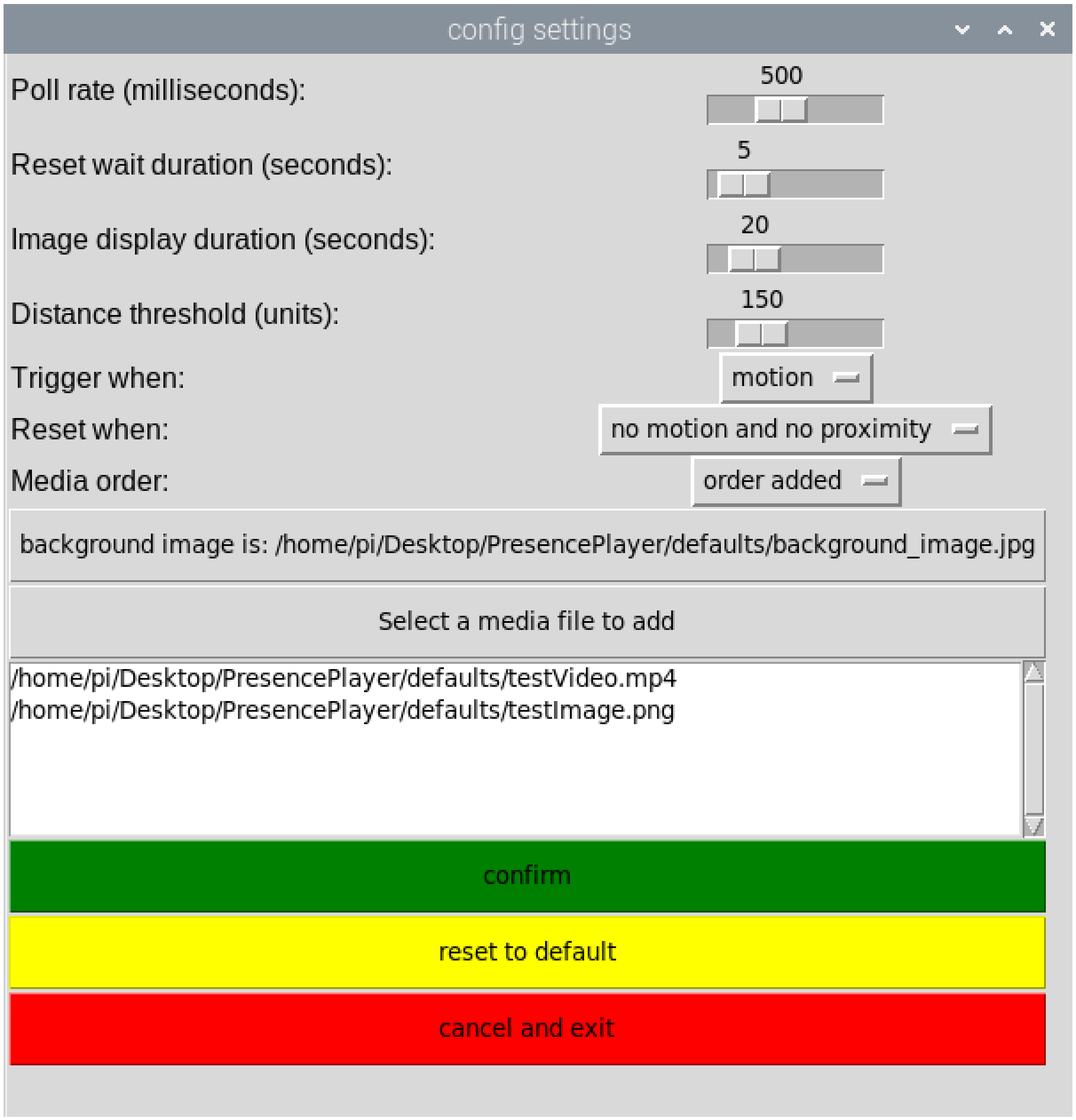
How to operate

If the program is set to start automatically when the Raspberry Pi is turned on, operating the device is as simple as turning the pi on when it should be active and off when it shouldn't.
If the program was not set to automatically start (step 7 of first-time setup) there are two options to manually start the program.

1. Terminal: Open a terminal window (you can do this by pressing the button on the tollbar at the top of the screen or by pressing the `ctrl` key, the `alt` key, and the `t` key all at the same time). Once the terminal window type:
   ```
   python3 /path/Main.py
   ```
   Where `/path` is the path to Main.py, for example when testing the typed line was:
   ```
   python3 /home/pi/Desktop/PresencePlayer/Main.py
   ```
   Then press enter and the application should start.
2. Thonny: To start the program simply navigate to the project directory and open the file main.py in Thonny by double clicking. Once open, press the green arrow at the top of the window to start the application.

To stop the program simply press the `esc` key or turn off the Raspberry Pi by cutting its power.
To configure the program run the `Config.py` file following either of the two methods (Terminal or Thonny) outlined above but be sure to replace instances of `Main.py` in the instructions with `Config.py` Once the configuration window is open you should be able to enter your configuration settings and then press the green "confirm" button to generate the config file, the yellow "reset to default button" to enter the default settings, or the red "cancel and exit" button to cancel the changes and close the window.

Settings


config settings

Poll rate (milliseconds): 500

Reset wait duration (seconds): 5

Image display duration (seconds): 20

Distance threshold (units): 150

Trigger when: motion

Reset when: no motion and no proximity

Media order: order added

background image is: /home/pi/Desktop/PresencePlayer/defaults/background_image.jpg

Select a media file to add

```
/home/pi/Desktop/PresencePlayer/defaults/testVideo.mp4
/home/pi/Desktop/PresencePlayer/defaults/testImage.png
```

- Poll rate:
  Scale of 200 to 1000 milliseconds
  Controls the rate at which the system checks its sensor values (checks if a viewer is present). Lower values should be more accurate but be more resource intensive.
- Reset wait duration:
  Scale of 1 to 30 seconds
  Controls the length of time required for no viewers to be detected before resetting. Lower values should allow the system to reset more readily, however low values may lead to the

system resetting unexpectedly if the viewer is not constantly being detected (note: this may happen if the viewer is on the edge of the detection area).

- Distance threshold:
  Scale of 50 to 500 units
  Controls the minimum distance a user needs to be from the distance sensor to be detected as being present. (**Note**: this distance measurement is on an unconventional, non-linear, scale in which 500 is right in front of the sensor, 200 is roughly 2 feet from the sensor, and 100 is roughly 4 feet from the sensor). Values should be tuned to detect slightly further than the expected view distance, but not so far that it will detect walls or other stationary items behind the viewer.
- Trigger when:
  Option of: motion, proximity, motion and proximity, motion or proximity
  Controls the condition that must be met for the system to trigger the video to start. It is recommended that motion is used for the initial trigger as it has a greater detectable area and thus is more likely to detect possible viewers more readily.
- Reset when:
  Option of: no motion, no proximity, no motion and no proximity, no motion or no proximity
  Controls the condition that must be met for the system to return to an idle state. It is recommended that “no motion and no proximity” is selected as it reduces the likelihood that the system will reset despite a viewer being present.
- Image display duration:
  Scale of: 1 to 120 seconds
  Controls the length of time that an image will be displayed on the screen once triggered before advancing to the next piece of active media.
- Media order:
  Option of: order added, alphabetic, shuffle, random
  Controls the order that the media is displayed. “Order added” will display the media in the order they were added during configuration. “Alphabetic” will display the media in alphabetic order based on their file path. “Shuffle” will randomly shuffle the order of the media and loop through them in that order. “Random” will randomly select the next media item after one has completed.
- Select idle image:
  Select the image you want to play in the background (Note: Failing to select an idle image leads to undefined behavior)
- Select playable media files:
  Select the media (videos and/or images) you want to play when a viewer is detected and present. Clicking an entry in the list of media will remove it from your selection. (Note: A video or image must be selected for the system to function)

Hardware interfaces

No external interfacing is currently supported through the main program as it has been tailored to a certain application. However, the portion of the system localized to the Arduino including its sensor setup and sketch file (arduino_sketch.ico) are encapsulated allowing for potential use in other applications. Systems will be able to poll the Arduino for distance and motion values through a serial USB connection. On top of this the Arduino.py file can be used in other Raspberry Pi python programs to facilitate the aforementioned Arduino communication.

Uploading videos and images

To upload media (videos/images)

1. Load media onto a USB drive
2. Plug that into the Raspberry Pi.
3. Copy the media from the USB into any directory of your choice
4. Eject the USB
5. Redo the configuration by running Config.py being sure to select the desired media in the section “Select a media file to add”