Halloween Ghost Images with IR motion sensor. (10/13/20)-1

The system makes ghostly images on a TV screen from objects that move in front of a camera.
If there is no movement, the image fads away to a blank screen
The system runs on a Windows 10 PC. The program ***ArtDisTest1***, written in the Processing language, was obtained from the Visual World Investigate Lab of the NC Museum of Natural Sciences. See Appendix 2 for the Processing code.

A camera mounted on a TV monitor feeds the Windows PC and the PC feeds the monitor. In this case a laptop computer with a built in camera is being used

As an enhancement to this system, we added sound to the TV images. The sound system runs on a Raspberry Pi. An Infra-Red motion sensor, HC-SR501, is connected to the Rasp Pi. The Rasp Pi monitors the output of the sensor and plays and audio clip when motion is detected. There are 25 audio clips of various short phrases such as “come closer”. The program cycles through these audio clips each time it receives a signal from the motion sensor.

The program is written is Python and can run on a Raspberry Pi-Zero.
The program is called ***ghostly_images.py*** and is in the ***/home/pi/distance*** directory. The audio effects files are stored in the ***/home/pi/distance/distance_audio_clips*** directory. The file names of the audio clips are 1.mp3, 2.mp3, ……., to 25.mp3. The program also continuously plays a background spooky audio track, ***eerie.mp3***.
The folder ***/home/pi/distance_audio_clips/list1*** contains all the audio files. There is an EXEL file that describes the audio in the numbered files

Additionally, the program also records the list of files played and the date and time in the ***logfile.txt*** text file. This file is in the ***/home/pi/ir_sensor*** directory.

The python code is in appendix 1

The Raspberry Pi Zero does not have an analogue audio output.
For this one will need an HDMI to VGA adaptor with an audio output and an HDMI micro to HDMI adaptor.

An alternative solution can be to add analog audio to the Pi Zero. See Appendix 3

To run the program automatically on startup:

Create a linux shell script ***autoexec.sh*** in the ***/home/pi*** directory with the following lines:

cd ghostly_images
sudo python ghostly_imaages.py

Note that the ***autoexec.sh*** must be an executable file. To make it executable, at the command enter:

chmod +x autoexec.sh

In order to have the system run automatically when power is applied, the file: **.desktop** must be in the folder **/home/pi/.config/autostart.**
Create the ***autostart*** folder in the ***.config*** directory using the ***mkdir*** command.
Create the ***.desktop*** file in the ***autostart*** directory using the ***sudo nano*** command.
Here are the three lines in the ***.desktop file***:

[Desktop Entry]
Type = Application
Exec = lxterminal -e ./autoexec.sh

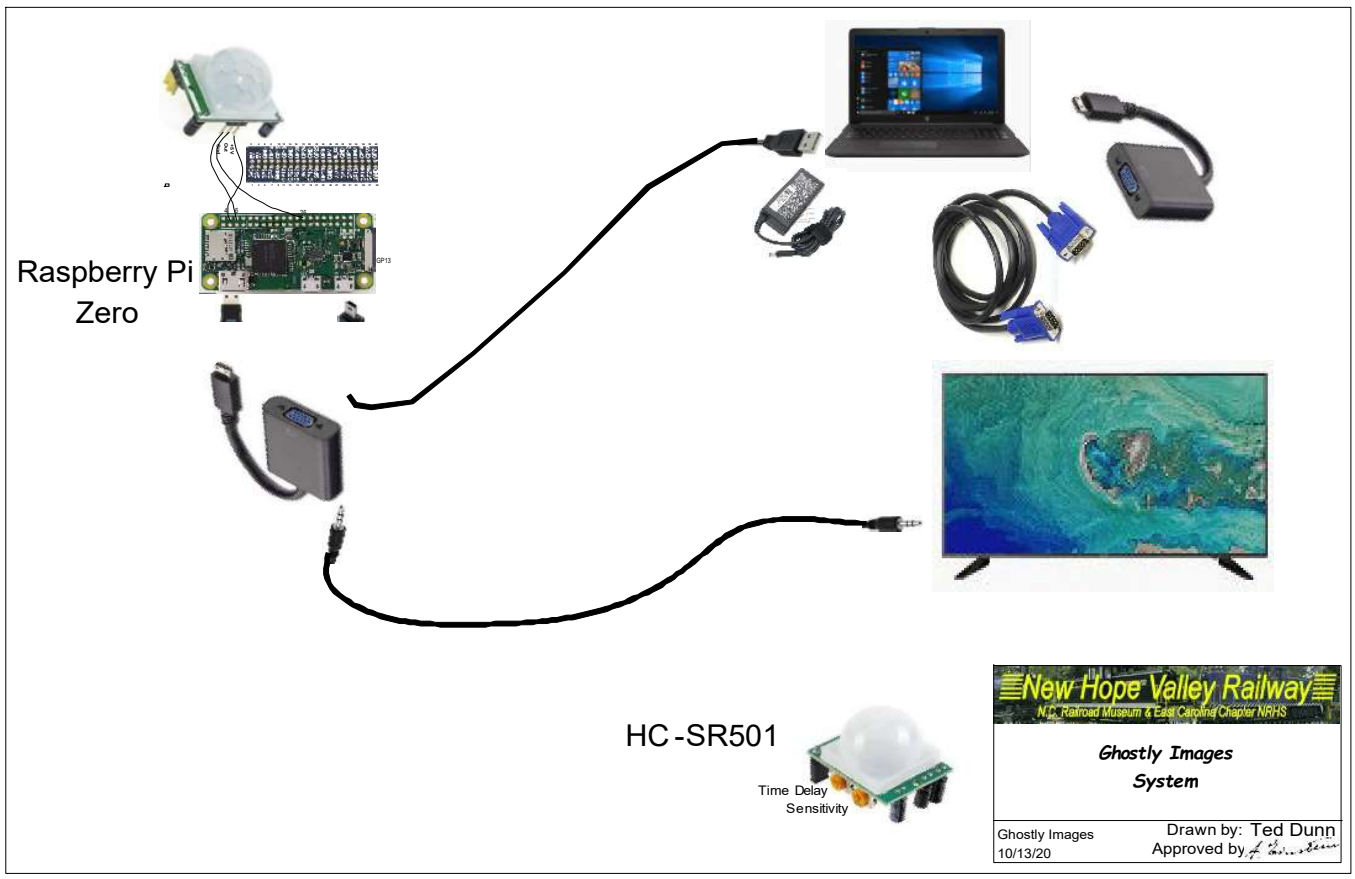
Note that to run an executable file, it must be preceded by the: **./** before the file name.

The 25 audio clips, see appendix 4, lists the audio files with the text contained in them.

The audio files of the special effect must be recorded by an announcer.
To convert wma files to mp3 files use *https://online-audio-converter.com/*. The system will play .mp3, .wav files as well as other formats.

The system will to play a background music track. The file is called ***eerie_sounds_7Hr.mp3*** and must be in the ***distance_audio_clips*** directory. The gain is set to -1800 and the loop option is selected.

This is done automatically using the threading function (see the code below) in Python.
To change the file, replace the ***eerie.mp3*** in the ***zzz =*** definition with a new file.
There is a utility program (***play_all.py***) in the distance directory which will play all of the audio files in the ***distance_audio_clips*** directory. To use it, at the command line enter ***python play_all.py***.


Raspberry Pi
Zero
HC-SR501
Time Delay
Sensitivity
New Hope Valley Railway
N.C. Railroad Museum & East Carolina Chapter NRHS
Ghostly Images
System
Ghostly Images
10/13/20
Drawn by: Ted Dunn
Approved by

Appendix 1

The Python file ***ghostly_images.py***

```
### PIR sensor

import RPi.GPIO as GPIO                    #Import GPIO library
import time                                #Import time library
import os
import threading
import datetime

gain = str(-1200)    #sets the gain of the audio output

GPIO.setmode(GPIO.BOARD)                   #Set GPIO pin numbering
pir = 26                                   #Associate pin 26 to pir
GPIO.setup(pir, GPIO.IN)                   #Set pin as GPIO in

########################  threading Plays the background music file eerie_sounds_7Hr.mp3

def play_clip():
   playing_clip = 1
   zzz = 'omxplayer ' + ' -o both --vol -1800 --loop ' +'/home/pi/distance/distance_audio_clips/eerie.mp3'
   print zzz
   os.system(zzz)

threading.Thread(target=play_clip).start() #Plays background music
#########################

j=1    #### Sound file index number Gets incemented after each playback
i=1    ### dummy variable for loop

while 1==1:          ##


   print "Waiting for sensor to settle ",j
   time.sleep(2)                           #Waiting 2 seconds for the sensor to initiate
   print "Detecting motion"

   while True:

     filename = str(j)+'.mp3'          # sets the file name to be played
     if GPIO.input(pir):                    #Check whether pir is HIGH
       ttime = datetime.datetime.now()       # saves the detection time
```


```
   print "Motion Detected!   ", filename,'  ',str(ttime)[0:19]  #
   xxx='omxplayer -o both --vol '+gain+ ' /home/pi/distance/distance_audio_clips/'+filename
   os.system(xxx)                        ## Play audio clip
   if j == 25:              #
      j=0
   j=j+1
   print "Detecting motion"
   file = open ('/home/pi/distance/logfile.txt','a')
   file.write(filename + '   '+str(ttime)[0:19]+'\n')
   file.close()



   time.sleep(2)                        #D1- Delay to avoid multiple detection
time.sleep(0.1)                         #While loop delay should be less than detection(hardware) delay
```

Appendix 2 Processing code

```
import processing.core.*;
import processing.data.*;
import processing.event.*;
import processing.opengl.*;

import gab.opencv.*;
import processing.video.*;
import java.awt.*;

import java.util.HashMap;
import java.util.ArrayList;
import java.io.File;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;

public class ArtDisTest1 extends PApplet {

Capture video;

OpenCV opencv;

int green = 0, red = 252, blue = 0, stage = 0, step = 7;

public void setup() {

  background(255);
  video = new Capture(this, 1280, 720);
  opencv = new OpenCV(this, 1280, 720);

  opencv.startBackgroundSubtraction(5,3,0.5f);

  video.start();
}

public void draw() {
  fill(0, 25);
  rect(-5, -5,1930,1090);
```

```
  scale(1.5f,1.5f);
  opencv.loadImage(video);
  opencv.flip(OpenCV.HORIZONTAL);
  opencv.updateBackground();

  opencv.dilate();
  opencv.erode();

 //shades
if(stage == 0) {
   if(green == 252) {
     stage = 1;
   }else{ green = green + step;}
 }else if(stage == 1) {
   if(red == 0) {
     stage = 2;
   }else{ red = red - step;}
 }else if(stage == 2) {
   if(blue == 252) {
     stage = 3;
   }else{ blue = blue + step;}
 }else if(stage == 3) {
   if(green == 0) {
     stage = 4;
   }else{ green = green - step;}
 }else if(stage == 4) {
   if(red == 252) {
     stage = 5;
   }else{ red = red + step;}
 }else if(stage == 5) {
   if(blue == 0) {
     stage = 0;
   }else{ blue = blue - step;}
 }

  noFill();
  stroke(red, green, blue);
  strokeWeight(3);
  for (Contour contour : opencv.findContours()) {
    contour.draw();
  }
}

public void captureEvent(Capture c) {
  c.read();
```

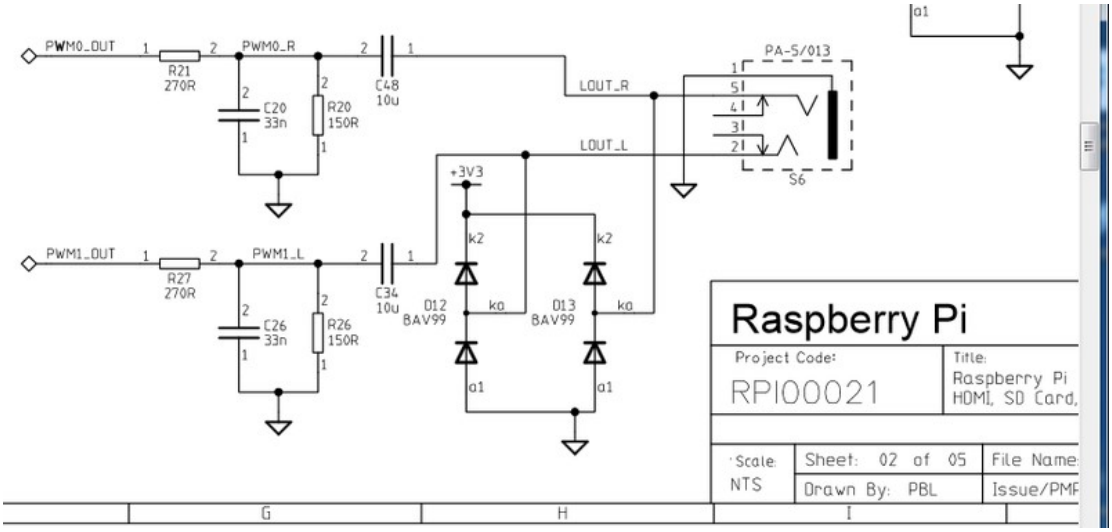

```
}
  public void settings() {  fullScreen(); }
  static public void main(String[] passedArgs) {
    String[] appletArgs = new String[] { "--present", "--window-
color=#666666", "--hide-stop", "ArtDisTest1" };
    if (passedArgs != null) {
      PApplet.main(concat(appletArgs, passedArgs));
    } else {
      PApplet.main(appletArgs);
    }
  }
}
```

Appendix 3 Pi-Zero analog audio output

How Other Pi's Create Audio

https://learn.adafruit.com/introducing-the-raspberry-pi-zero/audio-outputs

GPIO #18 is also known as PWM0 and in the original Pi was coupled with a very basic RC filter to create the audio output:

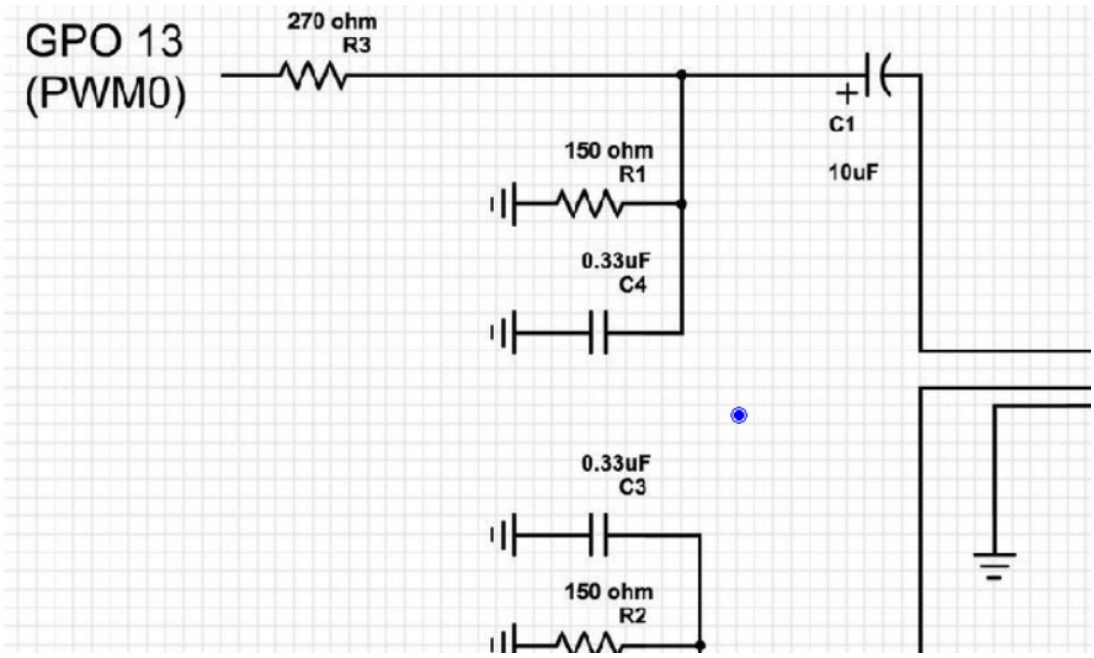


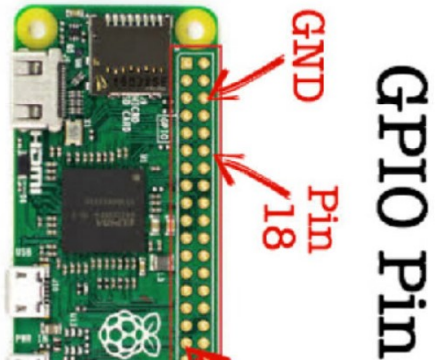
If you don't mind getting a few 150 and 270 ohm resistors, and two each of about 33nF (also known as 0.033uF) and 10uF capacitors, you can basically recreate those two filters.

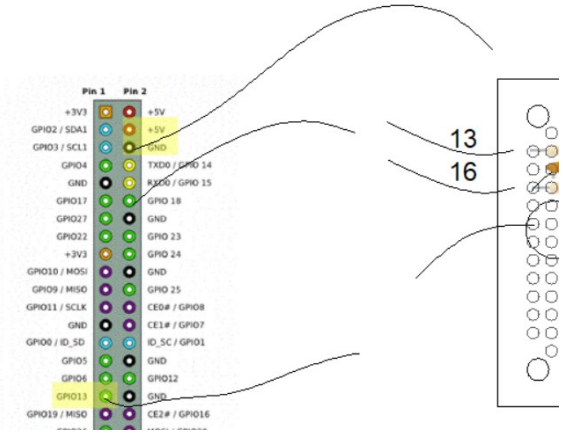
Now all you need is access to PWM0_OUT and PWM1_OUT, which are...on GPIO #40 and #45 and are not brought out on the Pi Zero. Tragedy? Give up? No! You can get to **PWM0** on GPIO #18 (ALT5) and **PWM1** on GPIO #13 (ALT0) or GPIO #**19** (ALT5)

*Simply adding the following line to your **/boot/config.txt** will reconfigure the pins at boot without any external software or services:*

dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4


GPO 13
(PWM0)
270 ohm
R3
150 ohm
R1
0.33uF
C4
0.33uF
C3
150 ohm
R2
C1
10uF


GND
Pin
18
GPIO Pin


13
16

Appendix 4 List of the audio files in the ***list1*** sub-directory.

(***/home/pi/distance/distance_audio_clips/list1***)

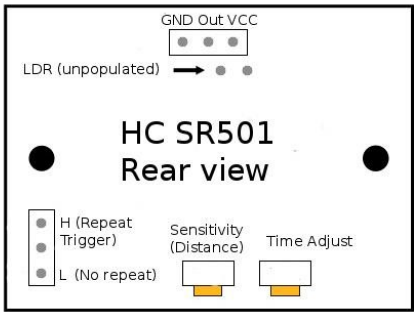
1.mp3 'you are too close!'
2.mp3 'stop moving'
3.mp3 'move right'
4.mp3 'move left'
5.mp3 'Look out'
6.mp3
7.mp3 'Be quiet'
8.mp3 'Wave your arm'
9.mp3 'Come closer'
10.mp3 'Who is that'
11.mp3 'Where are you'
12.mp3 'Go away'
13.mp3 'someone is watching you'
14.mp3 'The great pumpkin is after you'
15.mp3 'The witches are watching'
16.mp3
17.mp3 'proceed at your own risk'
18.mp3 'Be afraid'
19.mp3 'Beware of Dracula'
20.mp3 'Frankenstein is coming'
21.mp3 'Do not move'
22.mp3 'get help'
23.mp3 'no one can save you now'
24.mp3
25.mp3 'Please do not step on the rats'

Appendix 5

HC-SR501 PIR motion sensor on Arduino

HC-SR501

The HC-SR501 PIR motion sensor is one of the easiest to connect to an Arduino -or any other microcontroller for that matter- and can also be used as a stand alone motion detector.



The HC-SR501 board has 2 variable resistors: looking at the back, with the connections facing upwards and the variable resistors at the bottom, the left resistor is for sensitivity and the right one is for output timing.
for the sensitivity goes: Clockwise=>High sensitivity CCW=> low sensitivity (3-7 m). for the Output timing it is CW=>long, CCW=> short (3-300 sec)
The right prong of the connector is for Vcc (+5-20V), the middle one is signal out and the left one is ground. The output is either high (3.3V) or low (0v)

There are two versions of the board. One with a 3 prong jumper and one with solder pads instead of a jumper. If the jumper is put in its bottom position (with the board still facing as described) there is no reset. If it is in its top position (H) it is in auto reset mode. If set to Auto-reset the sensor will stay high until the motion stops. After motion is no longer detected the output will go low. If set to No reset (L) the sensor will stop sensing once it has triggered, and stays high for the preset time period.