Computer Vision Project

This project is all about computer images, from simple still photography to detecting faces in a live video stream. You will make a record of your experiments in the form of a video. You will be working as a team, so please take turns at the keyboard, and make decisions as a group.

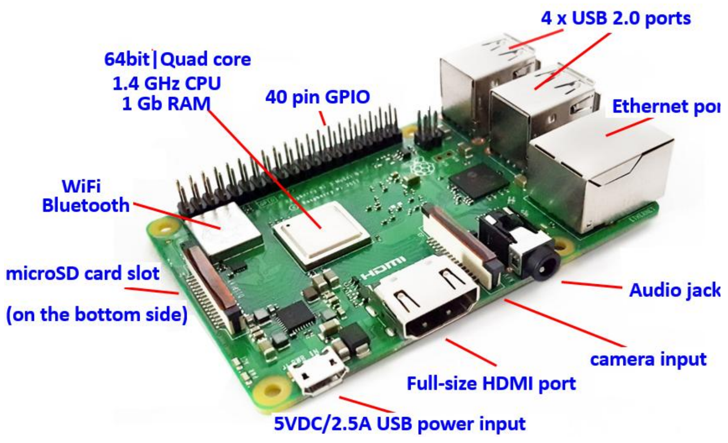
Along the way, you will also learn about:

- The free Linux operating system (pronounced “linnix” in the US)
- The Python programming language
- Some graphic and video editing methods

Equipment

A. Raspberry Pi

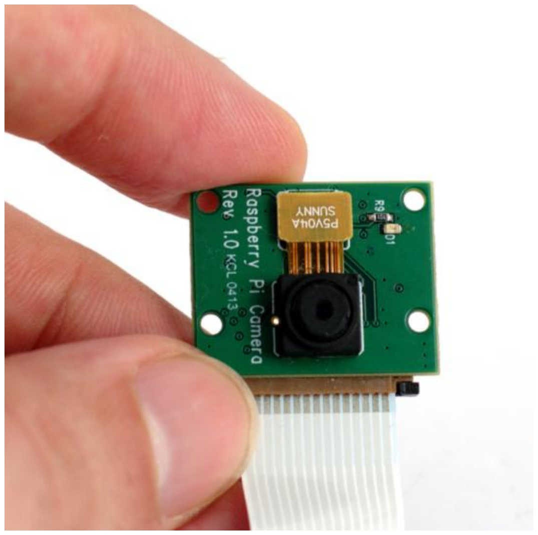
The Raspberry Pi is a “Single Board Computer”, that is, a complete computer built on a single circuit board.



The Raspberry Pi was developed by the Raspberry Pi Foundation, a non-profit organization founded in 2009 to promote the study of basic computer science. It was first available in 2012, and 25 million Raspberry Pis were sold by the end of February 2019.

B. Pi Camera

Still resolution	5 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90
Sensor	OmniVision OV5647
Sensor resolution	2592 × 1944 pixels
Fixed focus	1 m to infinity



C. Peripherals

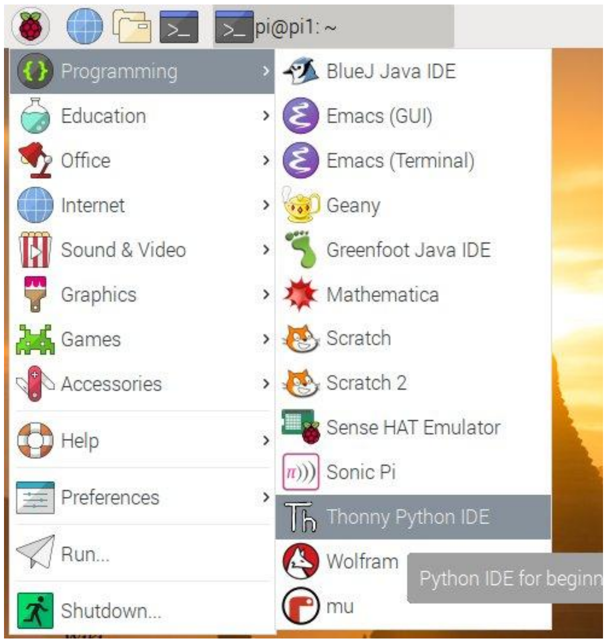
Monitor (20" LED-lit 1600 x 900 Resolution), keyboard, mouse.

Activities

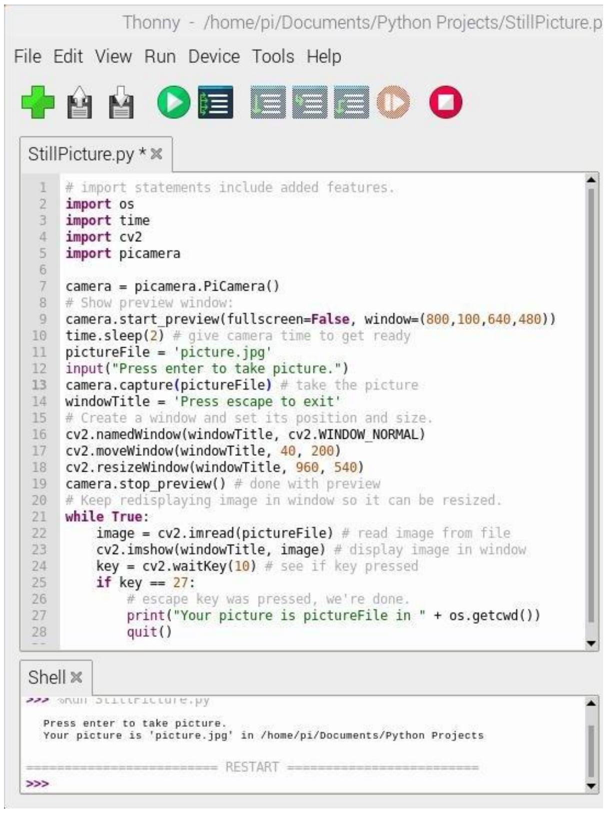
There are 9 activities total, **A** through **I**. You should try to average about two activities each day we meet, but it is up to you to set your own pace. There is no penalty if you do not do all the activities (other than perhaps wishing you had).

A. Take a picture

Our first activity will use the camera module to capture a still image. To do this, we will open and run a Python program.



1. Start "Thonny"[1]. To do this, click on the raspberry icon in the upper left corner of your screen, point at Programming, and click Thonny Python IDE, as shown.


Thonny - /home/pi/Documents/Python Projects/StillPicture.p
File Edit View Run Device Tools Help
StillPicture.py *
Shell

```
# import statements include added features.
import os
import time
import cv2
import picamera

camera = picamera.PiCamera()
# Show preview window:
camera.start_preview(fullscreen=False, window=(800,100,640,480))
time.sleep(2) # give camera time to get ready
pictureFile = 'picture.jpg'
input("Press enter to take picture.")
camera.capture(pictureFile) # take the picture
windowTitle = 'Press escape to exit'
# Create a window and set its position and size.
cv2.namedWindow(windowTitle, cv2.WINDOW_NORMAL)
cv2.moveWindow(windowTitle, 40, 200)
cv2.resizeWindow(windowTitle, 960, 540)
camera.stop_preview() # done with preview
# Keep redisplaying image in window so it can be resized.
while True:
    image = cv2.imread(pictureFile) # read image from file
    cv2.imshow(windowTitle, image) # display image in window
    key = cv2.waitKey(10) # see if key pressed
    if key == 27:
        # escape key was pressed, we're done.
        print("Your picture is pictureFile in " + os.getcwd())
        quit()
```

```
>>> %Run StillPicture.py

  Press enter to take picture.
  Your picture is 'picture.jpg' in /home/pi/Documents/Python Projects

========================== RESTART ==========================
>>>
```

2. Once Thonny starts, click the Open button ( ) and look for **`/home/pi/Python Programs/TakePicture.py`**. When it opens it should look something like the image at right.

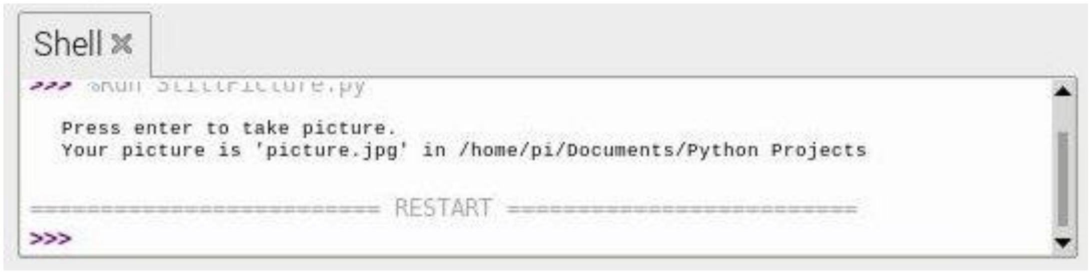
[1] Thonny is a simple Integrated Development Environment (IDE) for editing and running Python programs. "Integrated Development Environment" means you both edit and run your program in it. Usually it also does other helpful things like using different colors for different parts of your program.

3. Click the Run button ( ). The program puts up a preview window so you can see what will be in your picture.
4. When ready, type the Enter key to take the picture. The picture you just took will appear in a new window. (If the Enter key doesn't work, you may need to click on Thonny to give it focus.) Use the escape key to quit. Rerun the program as necessary to try again. We'd like to get a picture of everyone in your team. The easier way would be to get everyone in one picture. But if you want a separate picture for each person, you need to edit the Python code on line 11, where it says **`pictureFile = 'picture.jpg'`** to give different names for each person's picture. Later this picture or pictures will become the start of a video you put together using Flowblade Video Editor.

You just took a picture with the Raspberry Pi Camera using a Python program.
Next we're going to become more comfortable with finding our way around on the Raspberry Pi.

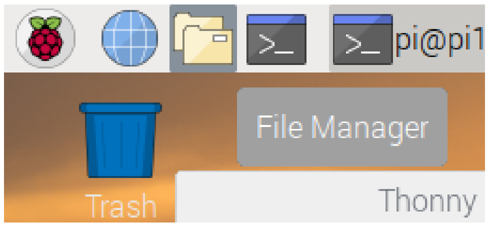
5. So that we can find your captured image later, we'll copy it now, while we have the information in front of us in Thonny's "Shell" window. The Shell window in Thonny is where the **`print`** statements in the Python code write to. (The Shell window is also where any error messages would appear.)

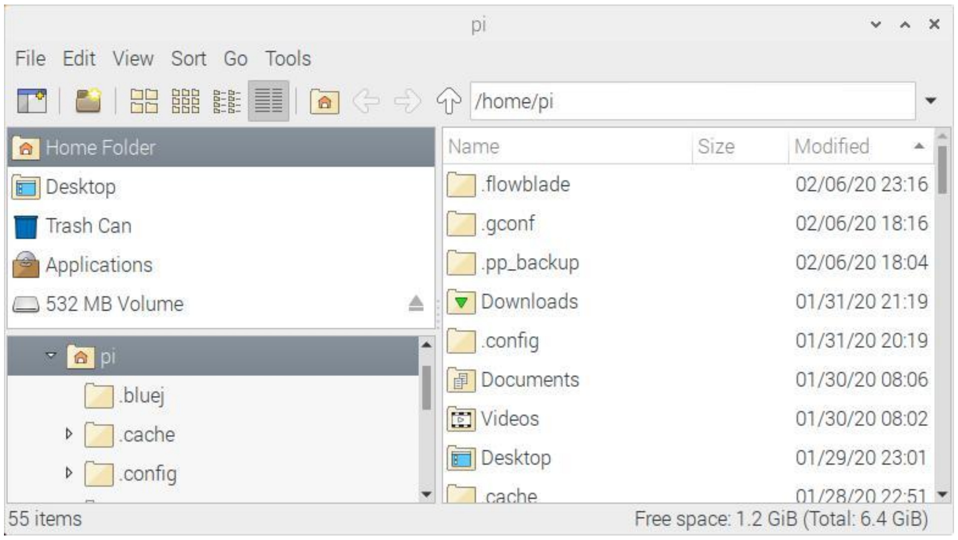
   It should look like this:



```
>>> %Run StillPicture.py

  Press enter to take picture.
  Your picture is 'picture.jpg' in /home/pi/Documents/Python Projects

============================ RESTART ============================
>>>
```

6. We'll use the Raspberry Pi's "File Manager" to copy the file. You can open it by clicking on the file folder icon as shown at right. When it opens, it will look like the image below.

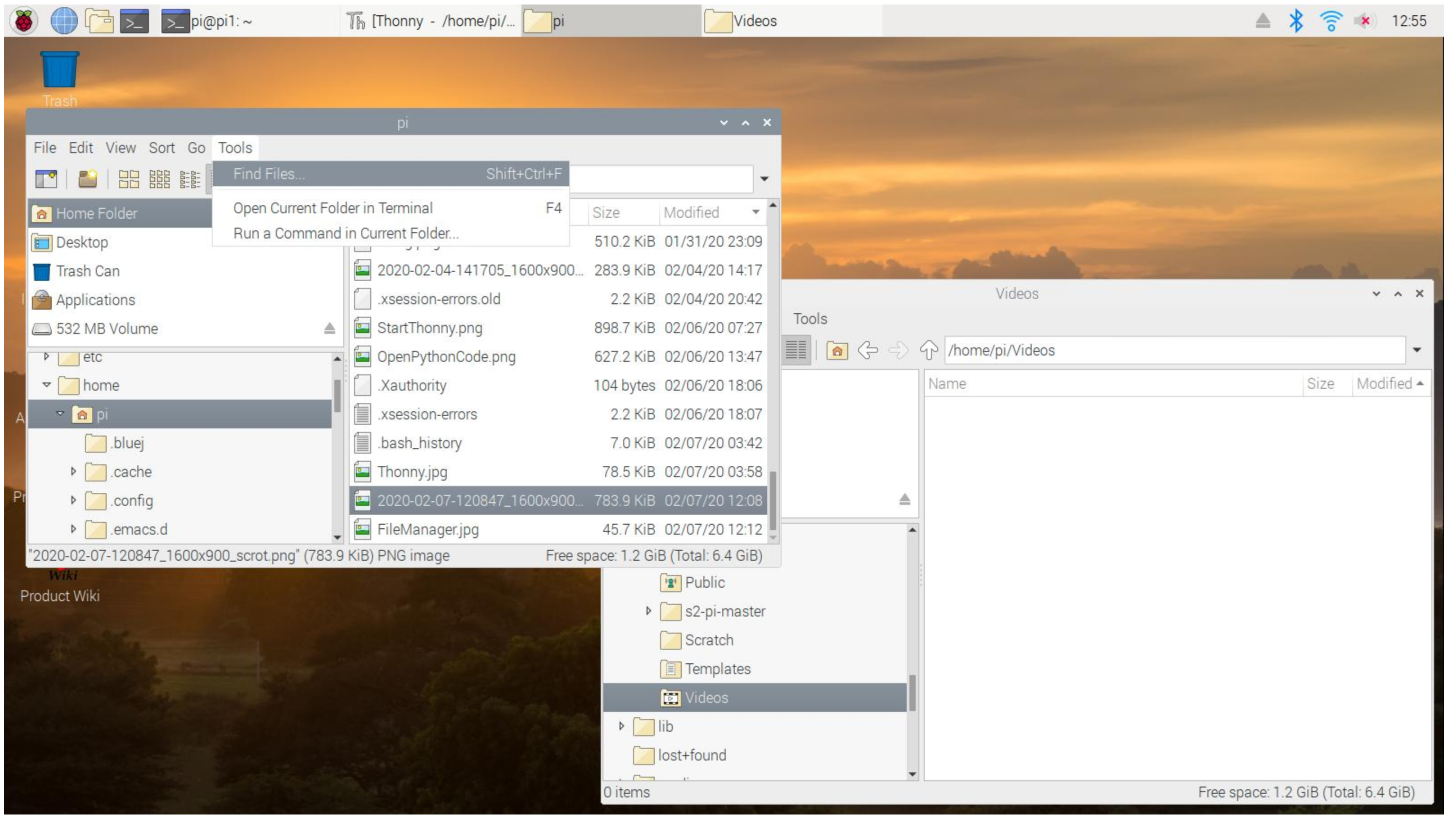




Name	Size	Modified
.flowblade		02/06/20 23:16
.gconf		02/06/20 18:16
.pp_backup		02/06/20 18:04
Downloads		01/31/20 21:19
.config		01/31/20 20:19
Documents		01/30/20 08:06
Videos		01/30/20 08:02
Desktop		01/29/20 23:01
.cache		01/28/20 22:51

55 items   Free space: 1.2 GiB (Total: 6.4 GiB)

7. To make it easy to copy from one folder to another, click the File Manager icon a second time so you have two such windows. The following image also shows the Find Files feature being selected from the Tools menu. You can use this, if necessary, to search for your picture file.

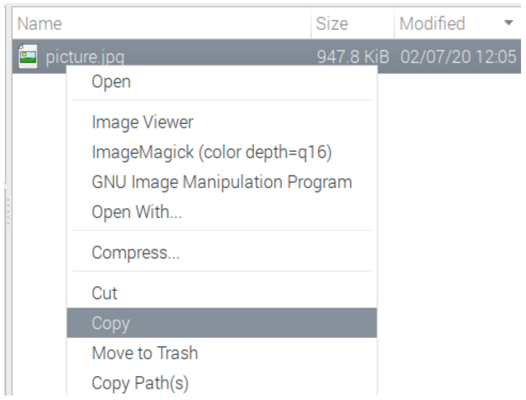


	Size	Modified
	510.2 KiB	01/31/20 23:09
2020-02-04-141705_1600x900...	283.9 KiB	02/04/20 14:17
.xsession-errors.old	2.2 KiB	02/04/20 20:42
StartThonny.png	898.7 KiB	02/06/20 07:27
OpenPythonCode.png	627.2 KiB	02/06/20 13:47
.Xauthority	104 bytes	02/06/20 18:06
.xsession-errors	2.2 KiB	02/06/20 18:07
.bash_history	7.0 KiB	02/07/20 03:42
Thonny.jpg	78.5 KiB	02/07/20 03:58
2020-02-07-120847_1600x900...	783.9 KiB	02/07/20 12:08
FileManager.jpg	45.7 KiB	02/07/20 12:12

"2020-02-07-120847_1600x900_scrot.png" (783.9 KiB) PNG image Free space: 1.2 GiB (Total: 6.4 GiB)

8. Locate your image file (or files) to copy. As in Windows, you can select more than one file by holding down the control key. To copy, right-click and select Copy.
9. We'll then paste into /home/pi/Videos, since we'll be using this and other files to create our video. Again, this can be done with the right-click menu.

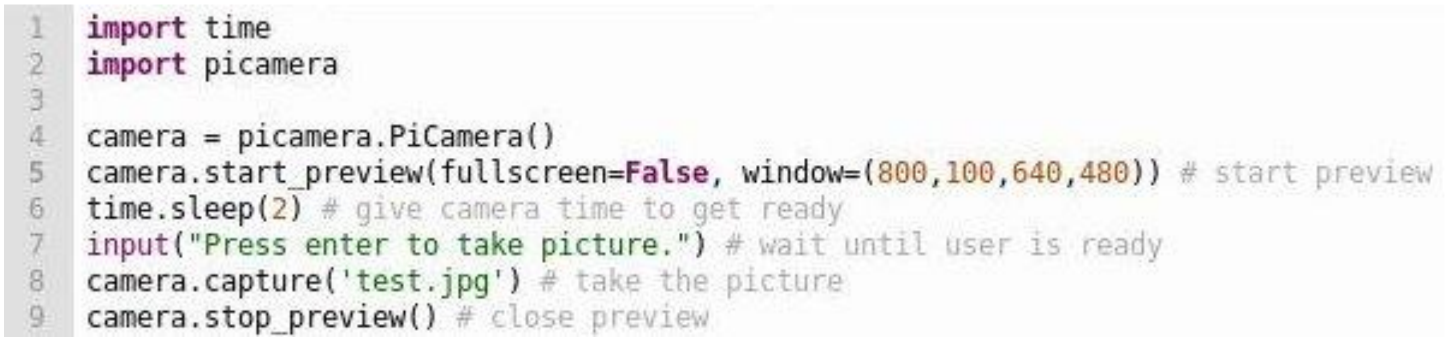
   Note: Linux uses forward slashes whereas Windows uses backslashes.



Name	Size	Modified
picture.jpg	947.8 KiB	02/07/20 12:05

You can double-click on your image file in File Manager to view it. (Or right-click and choose Open.)

The Python program you ran had some extra lines to do things like display the picture you took. If you just wanted to take the picture and open it later in File Manager, this would have been enough code:



```
import time
import picamera

camera = picamera.PiCamera()
camera.start_preview(fullscreen=False, window=(800,100,640,480)) # start preview
time.sleep(2) # give camera time to get ready
input("Press enter to take picture.") # wait until user is ready
camera.capture('test.jpg') # take the picture
camera.stop_preview() # close preview
```

You're done with this activity! If you have at least 20 minutes remaining, you can go on to **Record a Video**. Or see the following extra experiment with the camera. These extra activities are always short, and are recommended unless you don't have enough time.

Optional Activity Ax1:

Run the **TakePicture.py** Python program so that the preview window is present. Then shield the camera module from some of the available light with your hand or a book, and observe the background.

The camera is automatically adjusting its exposure (its sensitivity to light), as well as its "white balance" (how "warm" or "cool" the lighting appears).

B. Record a Video

1. If not already open, start Thonny (as in A.1) and open a file (as in A.2). This time, open **RecordVideo.py**.
2. Run the program (as in A.3). You will be prompted for a length of time and a filename. This is just a trial run, so you can leave the recording time at 10 seconds and the file name as movie.h264.
3. Locate movie.h264 in File Manager (as explained in A6), and view it. (It should open in VLC Player.)

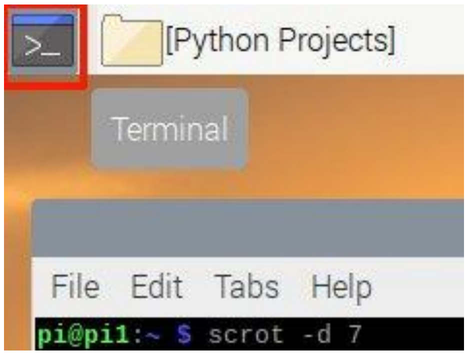
4. Ok, now let's take a more interesting movie for the video we'll be making. Aim the camera at your team and run the program, allowing 20 seconds this time. SLOWLY rotate the camera so that it eventually is looking right at the preview on the monitor. (You should get a picture of a picture of a picture…)

5. View the movie you recorded, and if you like it, copy it to /home/pi/Videos. (Refer to A7 if you need help copying.)

Great, you're done! Here are two extra activities you might find useful.
Note, the Video Editing activity will take you at least a half hour.

Optional Activity Bx1.

In this last program, **RecordVideo.py**, we used **os.system** to record the video. Python's **os.system("something")** is the same as entering "something" at a "Terminal" prompt. "Terminal" on a Raspberry Pi is like the "Command Prompt" on Windows. Open up a Terminal window now, and enter:



```
pi@pi1:~ $ scrot -d 7
```

```
cd /home/pi/Videos
```

Now (assuming your movie is named movie.h264), enter:

```
ffplay movie.h264
```

ffplay can open most video, still image, and audio formats. We'll use it later to play audio in our **Detect Motion** activity.

Optional Activity Bx2.

Add this line onto the very end of **`RecordVideo.py`** and rerun it. Be careful with the single and double quotes, and note that there must be a space after **`ffplay`**.

```
os.system('ffplay "' + outputFile + '"')
```

This takes advantage of the way os.system allows us to execute any command from our Python program.

C. Video Editing

As promised, we'll record our activities in the form of a video. We'll make a start on that now.

a) File conversion using the command line:

1. As in the first step of Optional Activity Bx1, open a "Terminal" window and enter the command:

```
cd /home/pi/Videos
```

2. We saved to the H264 format because it was quick and we wanted to get the best performance we could from the admittedly not-so-fast Raspberry Pi. Now we want to convert the video to the more common MP4 format before bringing it into Flowblade. We'll use ffmpeg to do this. The -i option specifies the input file, so if your video is named something other than movie.h264, use that name there instead. The ffmpeg command expects the last thing on the line to be the output file, so you can make that anything you want your MP4 file to be named:

```
ffmpeg –i movie.h264 movie.mp4
```

3. That's it; you can open your output file in VLC Player to make sure it worked.
4. Enter the **`ls`** command. This is the Linux "list files" command. You should see your JPEG (*.jpg) file(s) from activity A, and also your input and output from running **`ffmpeg`**. If not, copy these files now to **`/home/pi/Videos`** using File Manager, and repeat the **`ls`** command to make sure they are there.

b) Combining files using Flowblade:

1. From the main "Raspberry" menu, instead of Programming, go down to Sound and Video, and select Flowblade Movie Editor.
2. From Flowblade's Project menu, select Add Video, Audio, or Image….
3. In the left navigation pane, select Videos. You should see the same files you saw when you did the **`ls`** command above. Select all the files you want to use and click Open. You will probably get a dialog titled "Loaded Media Profile Mismatch". This is because Flowblade defaults to the PAL output format which is used in many other countries, whereas the United States uses NTSC. Just click "Change To File Profile".
4. Flowblade is arranged something like this diagram. You may need to refer to this in the following steps:

Project info	Your image and video files go here	You can preview the movie you're making here
Bins bin_1		
Sequences sequence_1		
	Media | Range Log | Compositors | Render	<| |> > []] [. . .

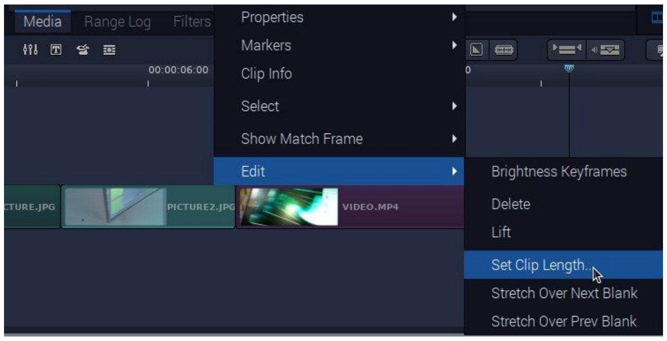
V3	This is your "timeline" showing the various pieces you arrange to make your movie
V2	
V1	
A1	
A2	

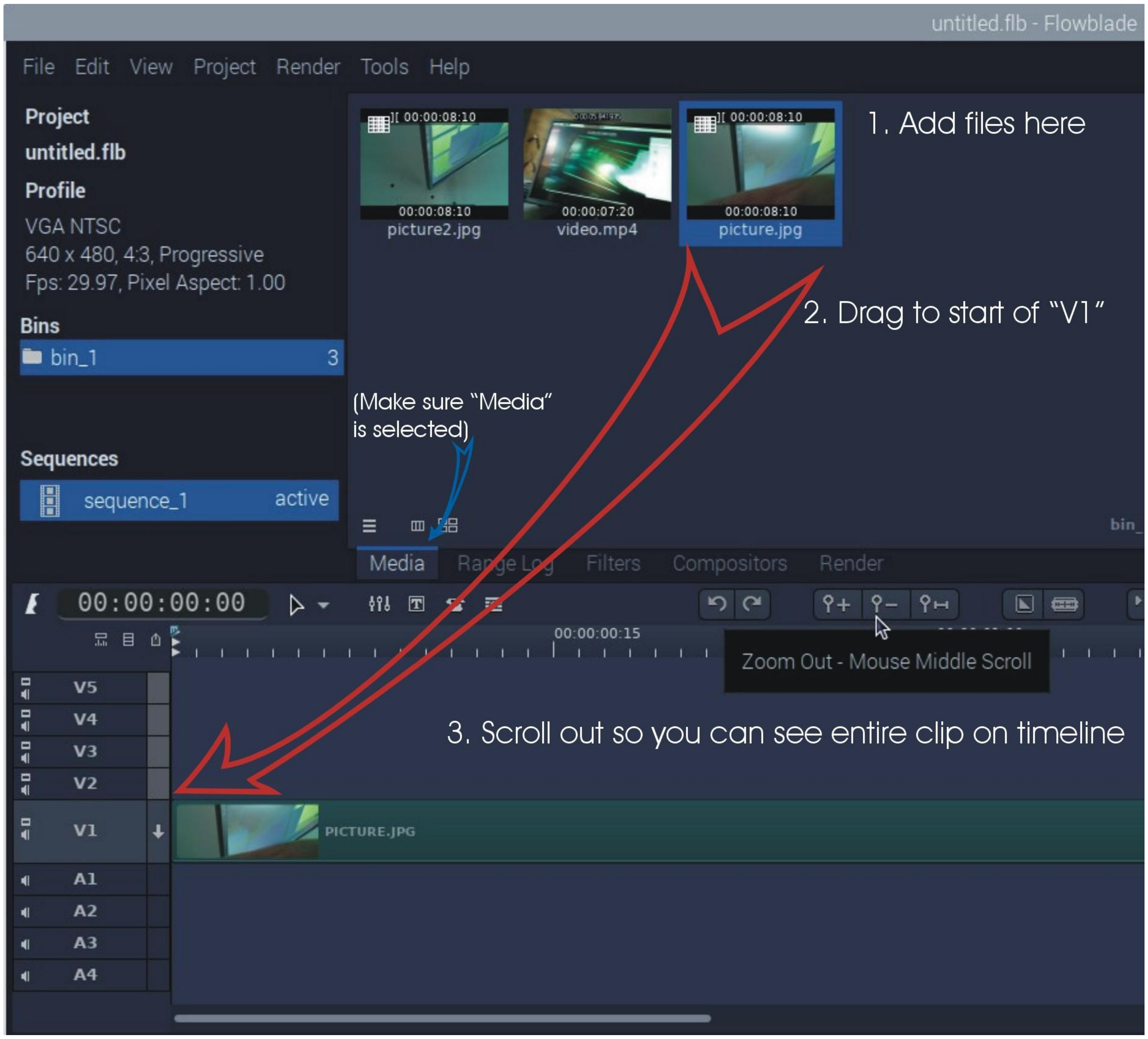
Note 1: Video editing is a complex activity and Flowblade has many complicated features. It is easy to get lost or to give Flowblade something to do that will take what seems forever on a Raspberry Pi. You are encouraged to explore Flowblade when you have free time, but it is best to follow instructions carefully for now.

Note 2: If at any point you don't see your image/video files, make sure "Media" is selected in the middle bar.

Note 3: When right-clicking an item in the timeline, first left-click to make sure you're working with the item you think you are; then right-click to bring up the desired menu.

5. Drag one file at a time to the start of the "**V1**" timeline. (See the following image.) This is your main video track. Slide them left so that the first item starts at time zero, and the next one starts right after the first one. Still images are given an arbitrary length, about 8 seconds. You may want to shorten that to 4 or 5 seconds. You can do this by dragging the right edge to the left, or by right-clicking and bringing up the Set Clip Length dialog as shown at right.

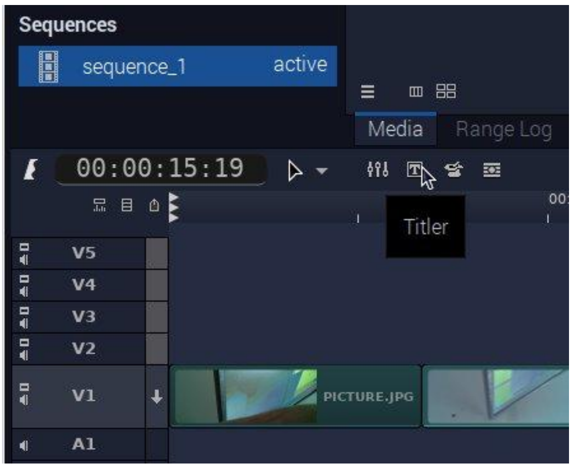


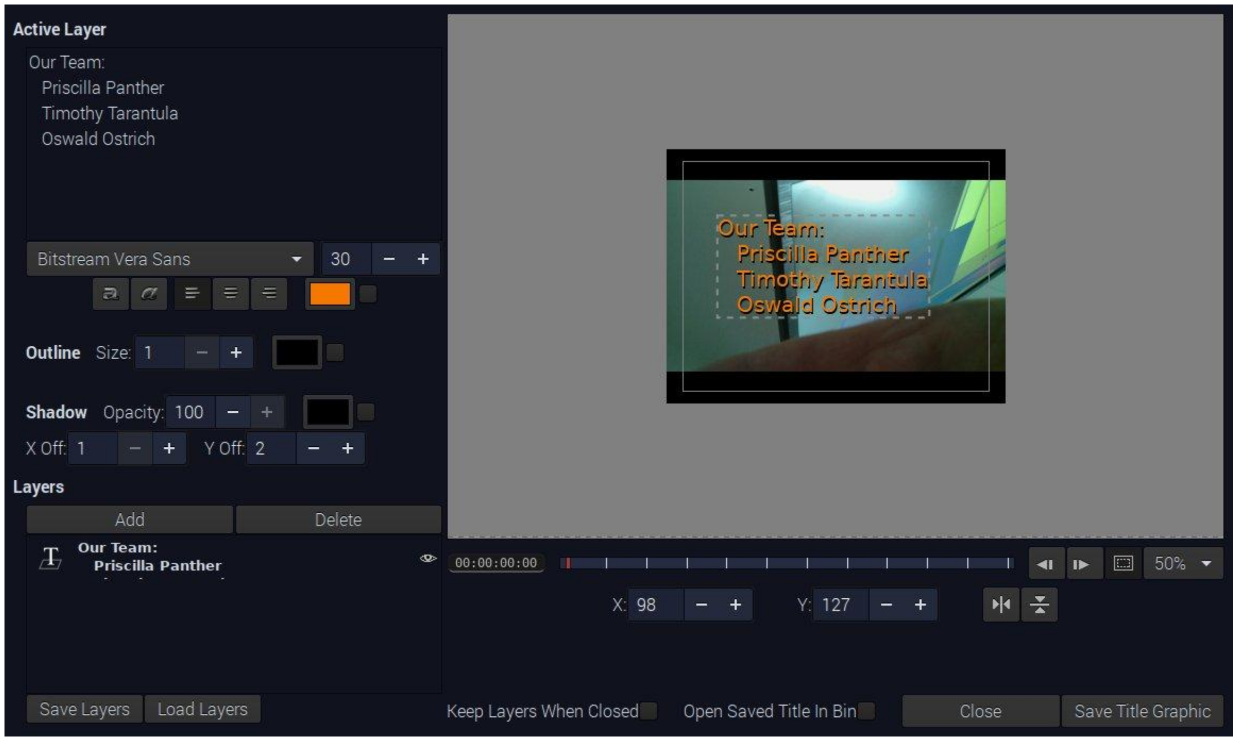

untitled.flb - Flowblade
File Edit View Project Render Tools Help
Project
untitled.flb
Profile
VGA NTSC
640 x 480, 4:3, Progressive
Fps: 29.97, Pixel Aspect: 1.00
Bins
bin_1 3
Sequences
sequence_1 active
picture2.jpg
video.mp4
picture.jpg
1. Add files here
2. Drag to start of "V1"
(Make sure "Media" is selected)
Media Range Log Filters Compositors Render
00:00:00:00
Zoom Out - Mouse Middle Scroll
3. Scroll out so you can see entire clip on timeline
V5
V4
V3
V2
V1
A1
A2
A3
A4
PICTURE.JPG

6. As each item is added, you may need to zoom out to see it entirely. You can't add something to the end of the timeline if you can't see where that is.

c) Adding titles in Flowblade:

1. Click the button that looks like a "T" in a box. This will launch the "Titler". (See image below.) You can enter your text in the upper left, change font, color, and add a shadow effect if you want. You can drag the result into position in the upper right preview pane. When you like what you have, click "Save This Graphic". It would be a good idea to save it to your Videos folder. Then click Close.
2. You will now have the image file you just created in your "bin" of media files. Drag it to the start of V2 (above V1).
3. You need to tell Flowblade to give the title image a transparent background so just the text appears "floating" above your main video track. To do this, right-click the title item you just dragged onto your timeline, and point to Add Compositor and click Blend.





Our Team:
Priscilla Panther
Timothy Tarantula
Oswald Ostrich

4. This will automatically switch you to "Compositors" in the middle bar. You can do some fancy things here, but to avoid getting lost, click "Media" to go back to your original view.
5. If you want, you can now right-click on your title block in V2 of your timeline, and select Add Fade, Fade In (or Fade Out). Drag the edge of your fade effects to set their length; usually you only want a few seconds.
6. Save your work! Go to File, Save. Give a name you will remember (ending with .flb), select the Videos folder, and click Save.
7. If you haven't already, preview your work by clicking the "play" triangle button below the upper-right preview pane. You may have to set your timeline cursor (the vertical line) to the beginning; the easiest way to do this is use the Home key.

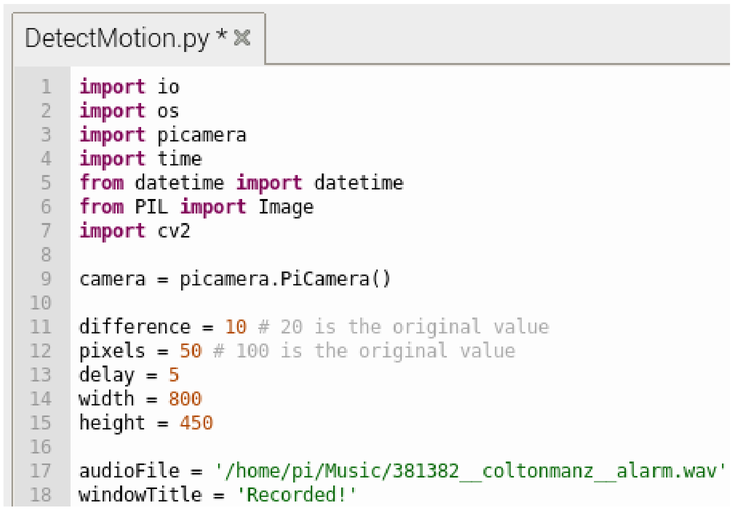
You did it! You may have a future in video editing!
The next optional activity is just for fun, you earned it!

Optional Activity Cx1.

If you have time, you can watch VIDEO_MOTAGE.mp4 (8 min) or MontageFromUp.mp4 (4 min) in your Videos folder.

D. Detect Motion

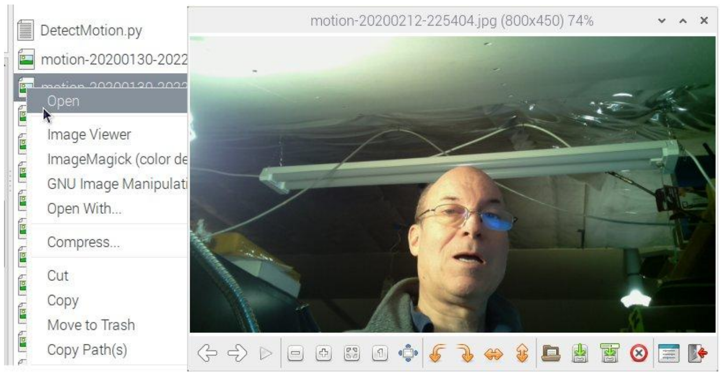
1. If not already open, start Thonny, and open **`DetectMotion.py`**.
2. Aim the camera so that nothing is moving in its field of view.
3. Run the program, wait several seconds, and walk in front of the camera. It should take a picture and display it, and sound an alarm.
4. It works but is not especially sensitive. So have someone sit in from of the camera about 2 feet (~25 cm) away and run the program. Observe what they need to do to trigger the alarm. Eye blink? Open mouth? Probably it will take a 90 degree head turn.
5. Now edit the program as shown, changing the variable **`difference`** to 10 and **`pixels`** to 50. Rerun the program.
6. Is opening a mouth or turning the head just a few degrees now enough to trigger the alarm?
7. Do some more experiments with **`difference`** and **`pixels`**.
8. The **`delay`** variable, set to 5 seconds, is to prevent one movement from creating multiple alarms. You can see what happens if you set it to 1 or 2.
9. Using File Manager, locate the other audio files in **`/home/pi/Music/SoundEffects.`** Try using a different one for the assignment of **`audioFile`**. (Rather than type the name, you can right-click on a file and select "Copy Path(s)" and then paste it into Thonny.)



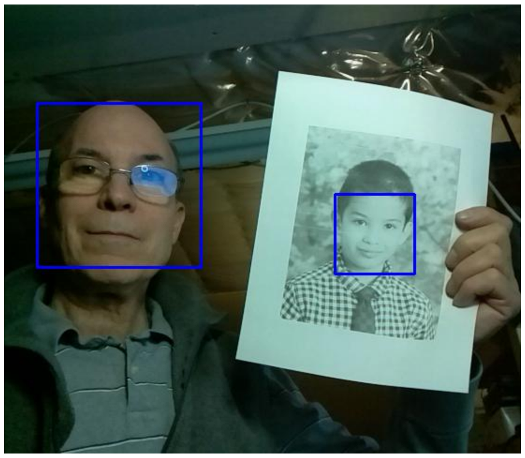
```
import io
import os
import picamera
import time
from datetime import datetime
from PIL import Image
import cv2

camera = picamera.PiCamera()

difference = 10 # 20 is the original value
pixels = 50 # 100 is the original value
delay = 5
width = 800
height = 450

audioFile = '/home/pi/Music/381382__coltonmanz__alarm.wav'
windowTitle = 'Recorded!'
```

10. Finally, using File Manager, go to **`/home/pi/Documents/Python Programs`**, and locate the **`motion-….jpg`** files which the DetectMotion.py program saves. Open the first one, and use the arrows to go through the images. Choose one or two to copy to your in **`/home/pi/Videos`** folder for use in the video you're making in Flowblade.

    Note: It would have been pretty easy to put a "timestamp" on the image. The next activity superimposes a rectangle.



E. Detect Faces



1. If not already open, start Thonny.
2. Open **`DetectFaces.py`**. It's down a level from the other programs, in **`/home/pi/Documents/Python Programs/faceDetection/`**.
3. Run the program in Thonny. It should display a live video of what the camera sees, with a blue rectangle around any faces it detects.
4. Get your whole team in the picture, and see if it can also detect a printed picture of a face. The person organizing this project should have provided some printed faces to choose from.
5. When you think you have a good demo, press the PrtScn key. It will capture an image of your entire screen, and put it in **`/home/pi`** with a name such as **`2020-03-13-151234_1600x900_scrot.png`**. If you like the way it came out, go on to the next step.

6. Close out of Thonny to conserve system resources, since you will do some graphic editing in the next step.
7. From the main "Raspberry" menu, select Graphics > GNU Image Manipulation Program (GIMP).
8. Drag your *_scrot.png file into GIMP, or use File > Open to open it.
9. In GIMP, do the following to crop the image down to just your "video" window:
    a. Tools > Selection Tools > Rectangle Select
    b. Drag from one corner to the opposite corner to select the "video" window. If you don't like it, just try again.
    c. Do: Image > Crop to Selection.
    d. Do: File > Export As... and save it in your Videos folder with a name such as FaceDetection.png. (You can use all the default Export settings. You have to use Export because Save saves as GIMP's own special file format.)

That is all for this activity, except for a couple notes:

- GIMP can do a LOT of other image processing besides cropping.
- The openCV package we're using can not only detect faces; it can *recognize* faces. That is, it can be taught to tell one person's face from another. Think: the CIA spotting a terrorist in an airport. Yeah, they have to use computers that are more powerful than a Raspberry Pi, but if you're not trying to process several hundred faces every second, you can do it with a Pi.

However, we have a few *other* things to try still if there's time.

Let's consider what time we have left for activities before the entire project session ends, and what we might do with it.

We're hoping to set up one Raspberry Pi in the school concourse which will cycle between detecting faces and displaying your videos. So it would really be good to get everything you can into your video. Aside from adding what you've done recently, a good plan would be to reserve at least the last half hour of the last day for this.

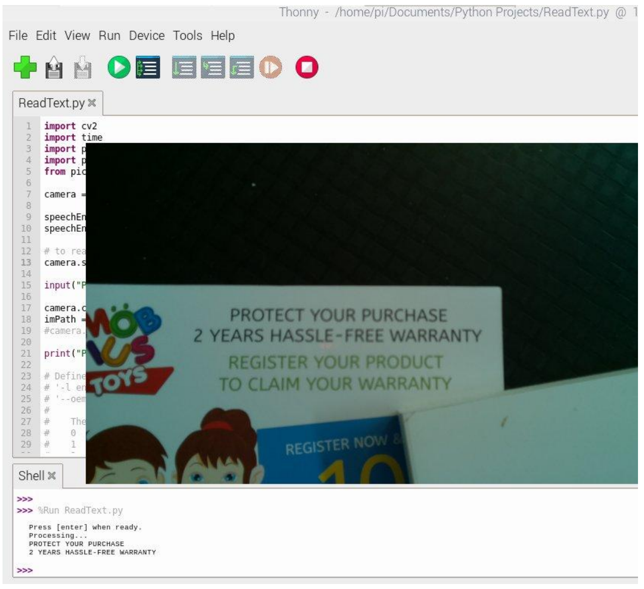
And there are 3 more activities we're about to describe, and they don't need to be done in order. So you may want to do them in the order that most interests you in case you can't get to them all.

Here's what we have:

- Optical character recognition (OCR) – read a sign and convert it to speech (read it out loud). This is pretty quick.
- Time lapse photography – condense an hour, day, or week of images into a few minutes of video. Once you test this out with a birthday candle, a good activity would be to point the camera out the window and record people, cars and buses coming and going for one whole week. This doesn't take too long to set up, but obviously you can't start that on your last week.
- "Invisibility Cloak" – Instead of doing the "green screen" thing where the background is removed, we record the background, then step into the picture and hold up a red cloth. Our program then removes all red from the foreground to give an "invisibility cloak" illusion.
- Finish up your video record of all you've done so it can become famous.

F. Optical Character Recognition (OCR)

1. If not already open, start Thonny.
2. Open **`ReadText.py`**, in **`/home/pi/Documents/Python Programs`**.
3. Run the program in Thonny. It should display a live video of what the camera sees. Carefully aim the camera straight down so that you can lay down a sheet of paper and see it with the camera.
4. Put some printed matter so it appears correctly to the camera, and type Enter as instructed in Thonny's "Shell" pane.
5. In the Shell pane, it will say "Processing…" and this will take several seconds. Then it will print what it thinks the printed text says, after which it will speak the text with a synthetic voice.
6. You can try different printed samples to see how well it does with each one.


Thonny - /home/pi/Documents/Python Projects/ReadText.py @
File Edit View Run Device Tools Help
ReadText.py
Shell
PROTECT YOUR PURCHASE
2 YEARS HASSLE-FREE WARRANTY
REGISTER YOUR PRODUCT
TO CLAIM YOUR WARRANTY
REGISTER NOW
MÖBIUS TOYS

```
import cv2
import time
import p
import p
from pic

camera =

speechEn
speechEn

# to rea
camera.s

input("P

camera.c
imPath =
#camera.

print("P

# Define
# '-l en
# '--oem
#
#    The
#    0
#    1
```

```
>>>
>>> %Run ReadText.py

  Press [enter] when ready.
  Processing...
  PROTECT YOUR PURCHASE
  2 YEARS HASSLE-FREE WARRANTY

>>>
```

Unfortunately, recording audio with a Raspberry Pi requires an additional microphone we don't have, and recording both audio and video at the same time involves some special set-up. So we're not going to have anything from this activity to add to our video record. But hopefully it was fun!

G. Make a Time-Lapse Movie

1. If not already open, start Thonny.
2. Open **`TimeLapse.py`**, in **`/home/pi/Documents/Python Programs`**.
3. Think about what you will be recording for your first short test. Do you need to change the **`delay`** or **`duration`** variables?
4. Run the program in Thonny. Again, it should display a live video of what the camera sees. Make sure you are set up to take your first test, and press Enter.
5. When it is done recording still images, it will print out the command you need to run in a Terminal window in order to combine the images into a video. You can replace "output.mp4" with something more meaningful, or rename it later.
6. Play the video you made, either with VLC or ffplay.

Notice that the Python program saves images to a subfolder named "CamOut" using filenames that start with the prefix "img". You can clear out all the *.jpg files from the CamOut folder (once you've made your .mp4 video from the *.jpg files) or you can edit the Python program to use a different prefix instead of "img". Or create another folder at the same level as CamOut, and edit the Python program to use that. The Python program will silently overwrite any files by the same name, but if you write fewer files than on a previous run, you'll have to figure out which ones to keep and which ones to delete before you run ffmpeg to make your movie, or your movie will, for example, start with a view outside and finish with a burning candle.

7. Figure out what you need to do to set up for a long capture. If it will be several days, what length should the resulting video be? It will be 24 frames per second (fps) where each still image becomes a frame.

For example, let's say you want to record for 6 days and condense it down to 1 minute. Given you want the video to be one minute long, that's 24 frames/second x 60 seconds = 1440 images to take over 6 days. Since 6 days is 518400 seconds, you'll want to take an image every 518400/1440 = 360 seconds. You'll need to edit your **`delay`** and **`duration`** variables accordingly.

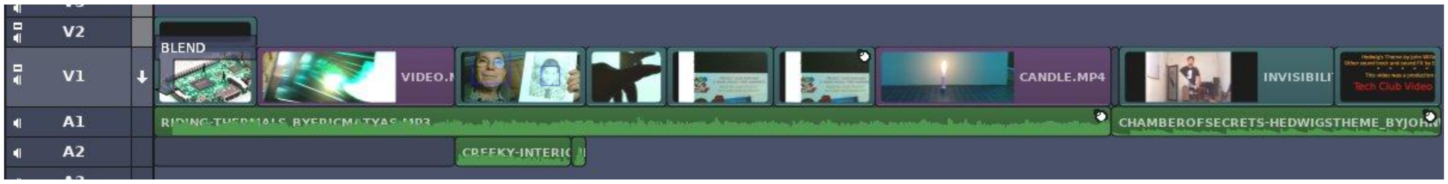
8. Decide your capture schedule, edit your Python variables, and set up your camera. Then start the process going, and carefully disconnect the monitor, keyboard, and mouse from the Pi. Good luck!
9. When you return, reconnect monitor, keyboard, and mouse. Run ffmpeg and view your movie.
10. Copy the movie you made to your Videos folder.

H. Invisibility Cloak

1. You will need the "magical" red coat that my auto mechanic gave me because someone left it in an old car. One could imagine spooky stories about why it is magical. I'll leave that part to you. Maybe add it to your video you're making.
2. Open **`/home/pi/Documents/Python Programs/InvisibilityCloak.py`** in Thonny.
3. Run the program. You will be prompted (by robot-voice) to "stay out of sight until the spell is ready." The program is taking an image of the background scene.
4. When you are told the "spell" is ready, simply get in front of the camera and hold up the magical red coat.
5. At a point where the effect looks especially good, press the Escape key. It will:
   a. Save a capture of the entire screen to **`cloak.jpg`** in your **`Python Programs`** folder.
   b. Stop the program.
6. You can rerun the program if you want to try again. If you don't remove cloak.jpg first, scrot will make up a new filename by adding a number after "cloak".
7. Refer to **E. Detect Faces**, starting at step 6, to crop cloak.jpg to show only the output window instead of the entire screen. Finish by exporting it to your video folder.

I. Final Video Editing

1. Start Flowblade again, as described in **C Video Editing, b) Combining files using Flowblade**.
2. Add all the media files you might use to the top-center Media pane.
3. Stack up your pieces from left to right on the V1 row of the timeline window (bottom pane).
4. You can also add audio tracks. A few are provided in **`/home/pi/Music/SoundEffects`** and **`/home/pi/Music/SoundTrack`**.



5. In the timeline image above, two music MP3 files were added to A1, each of them shortened and faded out. Two sound effect MP3s were added to A2 (the first one not starting until almost 12 seconds in).
6. The Titler was used (see **C Video Editing, c) Adding titles in Flowblade**) to create the credits at the end of V1. (Do be professional and give credit to work that is not your own.)

The demo movie shown in the timeline image above is /home/pi/Videos/FlowbladeMovieDemo.mp4.

7. The OCR.jpg image is used twice in a row on the timeline. The 2nd time it is used it is enlarged and offset. You can do this by right-clicking on an item in the timeline and selecting Add Filter > Transform > Translate.
8. To fade an audio track in or out, right-click on it and select Edit > Volume Keyframes.
9. When you have it all assembled the way you think you want it, you need to "render" it into a single video. From the main menus, select Render > Render Timeline. This will take about 3 minutes for every minute of video.

Please take a minute to complete the following survey

Please list any errors you found in this document:

Which activity did you find most difficult? Can you say why?

Which one or two activities did you like the most?

Please add any comments you wish to make about the activities:

A. **Take a picture**

B. **Record a Video**

C. **Video Editing**

D. **Detect Motion**

E. **Detect Faces**

F. **Optical Character Recognition (OCR)**

G. **Make a Time-Lapse Movie**

H. **Invisibility Cloak**

I. **Final Video Editing**