Raspberry Pi Face Recognition Treasure Box

Created by Abigail Torres

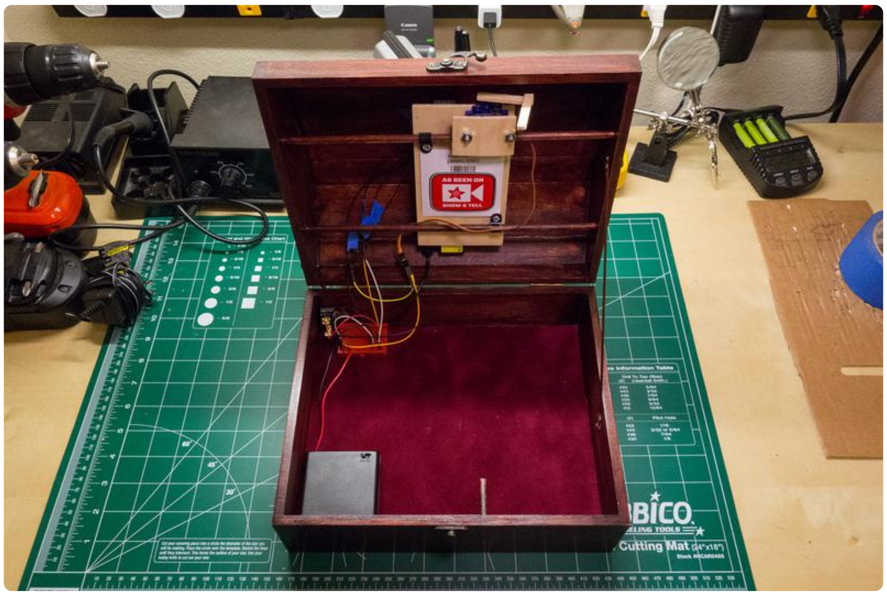
https://learn.adafruit.com/raspberry-pi-face-recognition-treasure-box

Last updated on 2023-08-29 02:28:29 PM EDT

Table of Contents

Overview

Face recognition is an exciting field of computer vision with many possible applications to hardware and devices. Using embedded platforms like the Raspberry Pi and open source computer vision libraries like OpenCV (), you can now add face recognition to your own maker projects! In this project I'll show you how to build a treasure box which unlocks itself using face recognition running on a Raspberry Pi.



Before you get started with this project, it will help to familiarize yourself with a few things:

- If you haven't setup or used a Raspberry Pi, go through the tutorials on learning the Raspberry Pi () like preparing an SD card (), network setup (), and using SSH () before attempting this project. Also take a look at the lesson on using a servo () for more background on how servos work.
- Skim the OpenCV tutorial on face recognition () for an idea of how face recognition algorithms work. Don't worry, you don't need to fully understand the math or code to build and learn from this project.

This is a learning project for fun and not suited for real security duty. In addition to the basics of computer vision, you'll learn vital concepts like the occasional false positive, false negative and spoofing. It's exciting that we can explore computer vision at the hobbyist level now...but it takes work to get right!

Continue on to learn about the hardware needed to build this project.

Hardware

You will need the following parts for this project:

- Raspberry Pi computer (). Nearly any model will suffice, old or new...provided it has the camera connector. That leaves out only the Pi 400 and very early Pi Zero units.
- Your Pi will need internet access to set up the software, so make sure you have either a wired or WiFi network connection for your Pi.
- Raspberry Pi camera (). There are several models now, and most any will do...this is not a demanding project, resolution-wise. Depending on where your camera and Raspberry Pi can be placed inside your box, you might need a longer (http://adafru.it/1648) or shorter (http://adafru.it/1645) camera cable. Pi Zero models use a smaller camera connector and require special cabling.
- Small box that can fit the Raspberry Pi and locking mechanism inside. I found an inexpensive plain wooden box at a craft store, and finished it with wood stain and polyurethane. Look for a box that has hinges which are screwed in from the outside of the box. Although not terribly secure, it will allow you to disassemble and open the box in case the locking mechanism fails to open.
- Small servo or lock solenoid for the locking mechanism, depending on how your box can be latched shut.
    - A servo (http://adafru.it/169) that rotates a latch can work with most boxes that open from the top or side. The software for this project is written to use a servo.
    - A lock solenoid (http://adafru.it/1512) can work with boxes that have a door or drawer. See this locking drawer project () for information on using a lock solenoid. As written, the code for this project does not work with a solenoid...this is a "learning opportunity."
- Momentary push button (http://adafru.it/1505) that can mount to the box. Depending on how thick your box is, you might need a smaller or larger push button. Improvise!
- Power supply for the Raspberry Pi () and servo or solenoid. Small hobby servos can be powered directly from the Pi's GPIO header. For larger servos, a 4x AA battery pack (http://adafru.it/830) is a simple option.
- Wood, wood glue, and fasteners for building a latch mechanism and frame to support the Pi inside the box. The exact material will depend on your box, but you can see further below how I used 1/4" dowel and thin bass wood to build the frame and latching mechanism for my box.
- Hookup wires (http://adafru.it/1311) to connect the switch, servo, and servo power supply. Female jumper wires (http://adafru.it/824) work well for

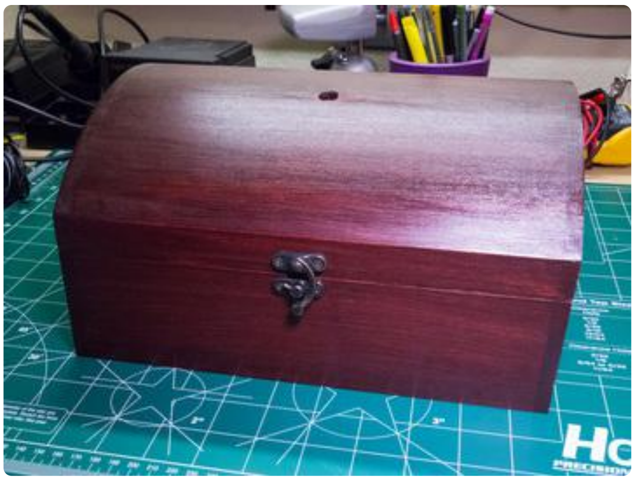
connecting directly to the Raspberry Pi GPIO pins, or you could use the Pi cobbler () with a small breadboard (http://adafru.it/64).

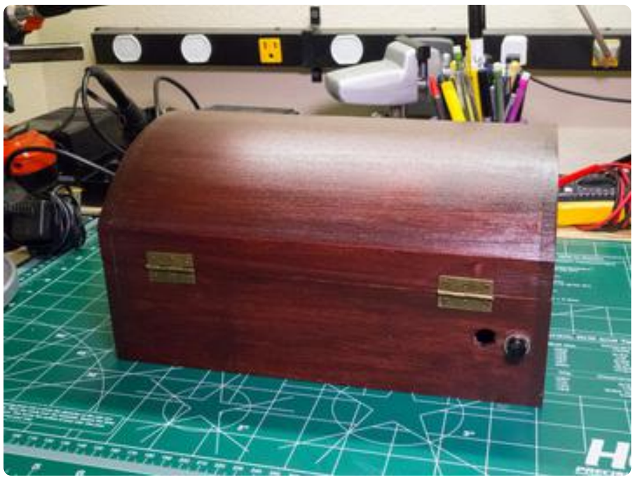
This project is not meant to be a highly secure container...it’s more a school science project. Face recognition is not perfect and is easily circumvented with a photograph. Don't store valuables in the box!

Assembly

The box you use for your project will dictate exactly how the Raspberry Pi, servo, and latching mechanism need to be mounted. Read the notes below for tips on how to construct your hardware, based on how I built mine:

Use a box which can fit the Raspberry Pi, servo, and latching mechanism.



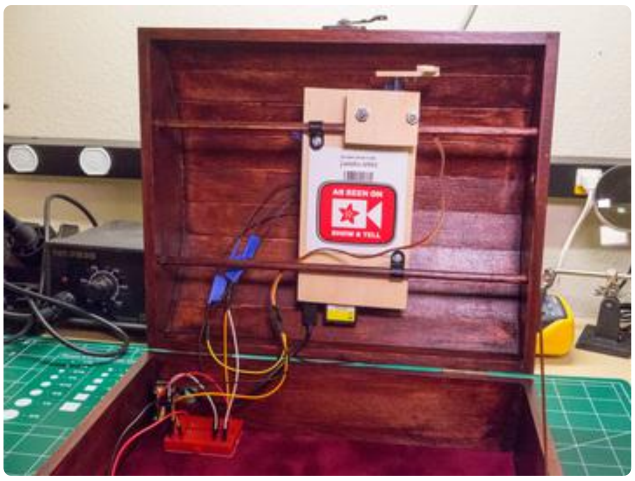
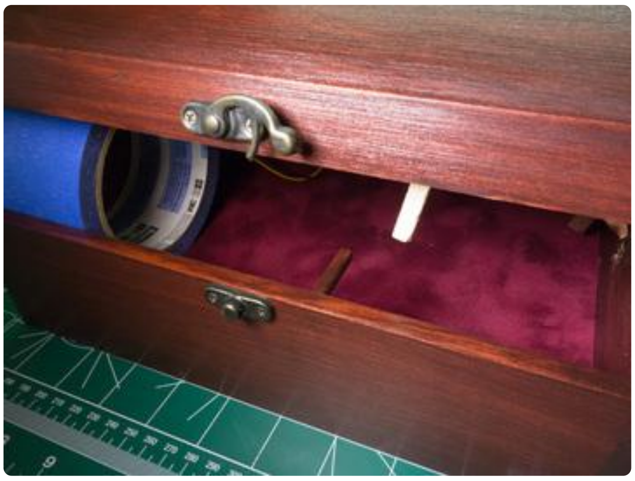
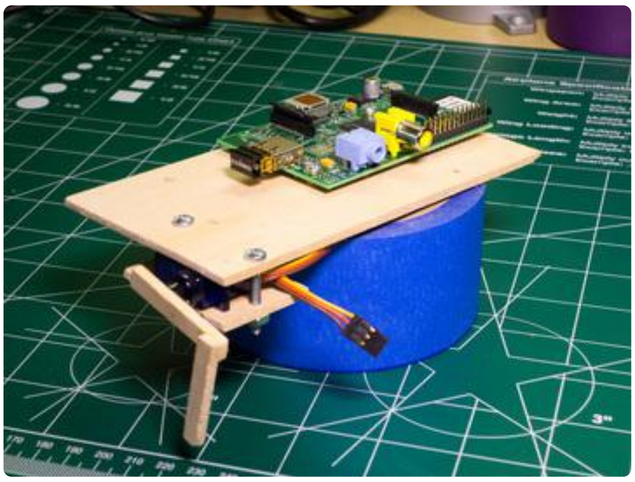
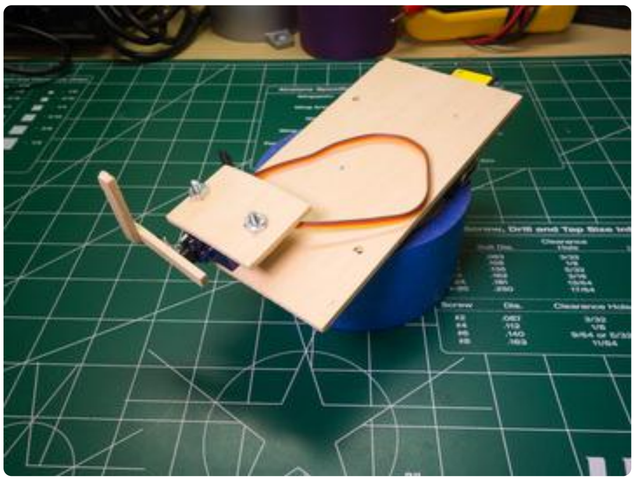


Drill a hole in the top or side for the Raspberry Pi camera. I found a 7/16" drill bit was enough for the camera to comfortably fit. Be careful when drilling large holes--work up from smaller bits so you don't crack the box. If you do damage the box, you can generally use wood filler to hide mistakes.

If they aren't mounted there already, consider moving the hinges to be mounted from the outside of the box. This will allow you to disassemble the box in case the hardware fails in a locked position.

Drill a hole in the box to mount the push button. You can see I mounted mine on the back of the box where it can be reached while a user looks into the camera on the top.

Drill another hole to allow power cables to come into the box for both the Raspberry Pi and servo. I found a 1/2" hole was enough to fit a micro USB cable through for powering the Pi. To the left, you can see the hole I drilled in my box next to the push button.

For the latching mechanism, mount a small dowel perpendicular to the side of the box. Attach a right angle of wood to a servo horn so it can act as a latch that swings down to catch the dowel, locking the box. See the pictures to the left to see the latch mechanism and dowel I built in my box.

If possible, mount the Raspberry Pi in the top of the box or near the hole for the camera. You can see how I mounted my Pi to a small board along with the locking servo, and then attached that board to dowels glued in to form a frame in the box top.

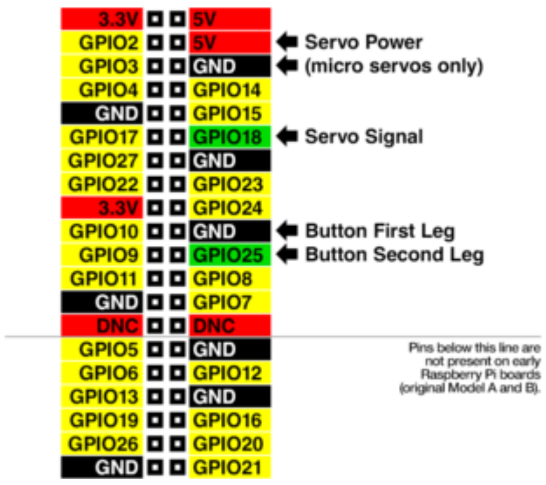
A few screws and a smaller board can act as a clamp to hold the servo in place.

Wiring

The electronics in this project are fairly simple and involve connecting a servo and push button to the Rasperry Pi. If you have never used these devices with a Raspberry Pi, read the following tutorials for a good overview of their usage:

- Raspberry Pi Lesson 4: GPIO Setup ()
- Raspberry Pi Lesson 8: Using a Servo Motor ()
- Bread Board Setup for Input Buttons ()

Other than the camera (connecting to the "CAMERA" connector on the Raspberry Pi board), only four or five connections are needed...

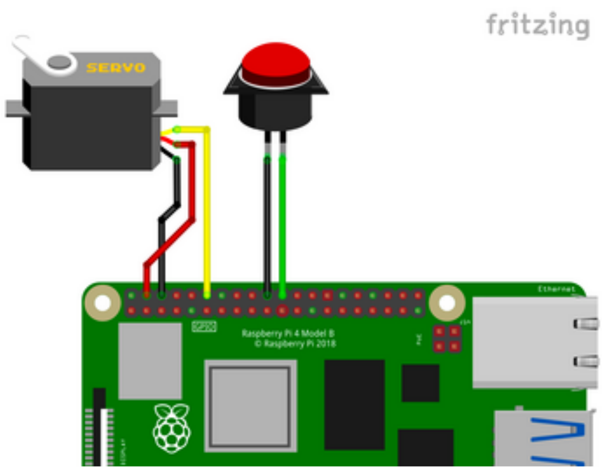


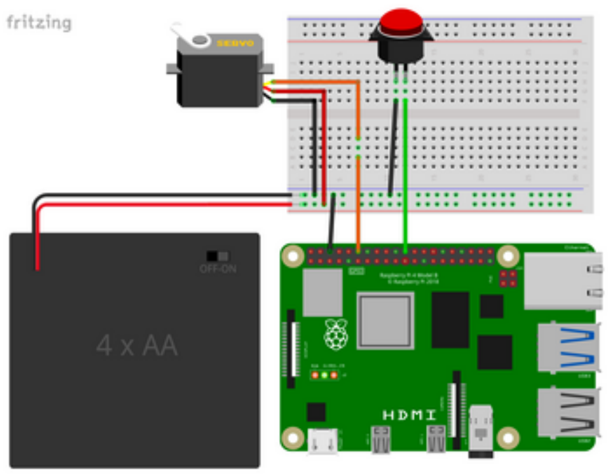
Here's a view of the Pi's GPIO header (40 pins on most boards, 26 on the early Model A and B). This is with the Pi turned "the tall way," with the two 5V pins near the top-right corner of the board.

The servo signal comes from GPIO18.

The button connects to GPIO25 and any ground pin (there are several, see diagram, any will do but there's one conveniently right next to GPIO25).

The software is easily reconfigured to use different pins, if you have other devices wired to the Pi that would conflict.





Small micro servos often work fine powered directly off a 5V pin on the Raspberry Pi.

For larger servos, or for best performance even on small servos, a separate 6V power source is best. In the second image here, notice that there's no 5V connection to the Pi…but it's vital that a ground connection is made between the Pi and battery "–" terminal.

These are merely schematic representations…you might use a breadboard or not, either way, whatever approach works best. It's the connections that are important.

Continue on to learn about how to set up the software for the project.

Software

Start with a fresh install of the latest 32-bit Raspberry Pi OS "Lite" on a flash card. You do not need the "Desktop" version (but no harm done if that's what you're most familiar with), and certainly not the 64-bit version (it's cutting edge and doesn't always play well with different hardware).

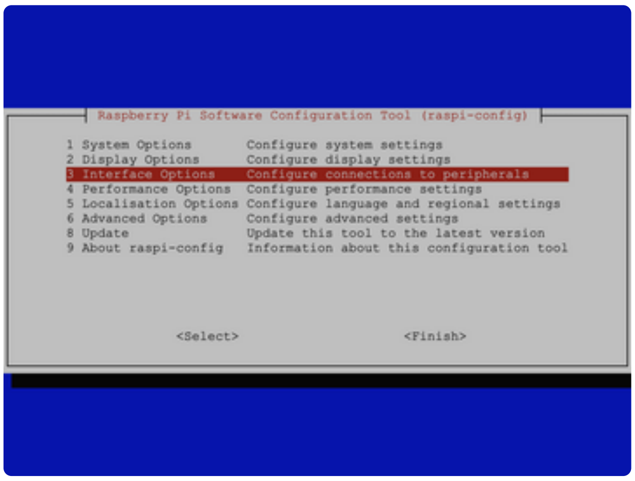
The operating system can be downloaded from the Raspberry Pi site () or selected in the Raspberry Pi Imager () application. Strongly recommend the Imager option, as this gives you the opportunity to set up an account, ssh and network credentials before booting the Pi for the first time.

So let's suppose at this point you have the Pi up and running and accessible on the network…

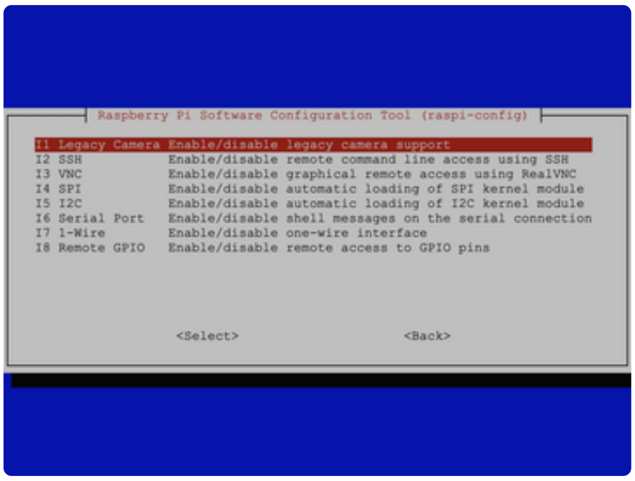
Prerequisite Configuration and Installation

First the Pi camera must be enabled with the raspi-config utility. From the command line…

```
sudo raspi-config
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

            <Select>                  <Finish>
```

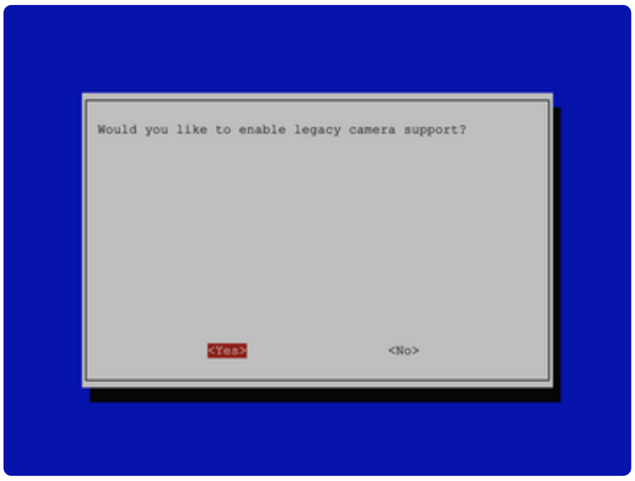


```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC           Enable/disable graphical remote access using RealVNC
I4 SPI           Enable/disable automatic loading of SPI kernel module
I5 I2C           Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins

            <Select>                  <Back>
```

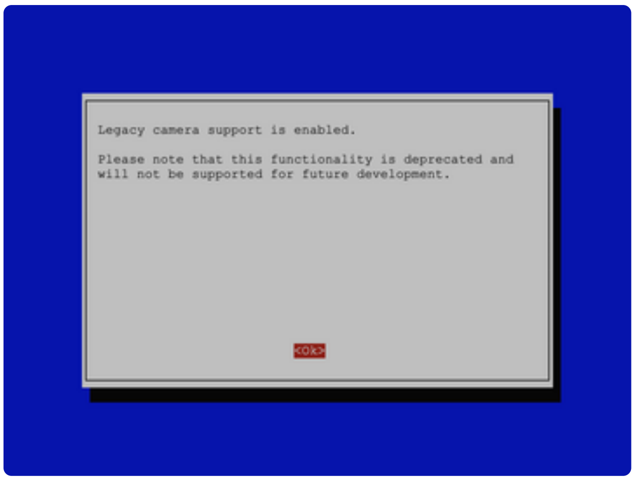
From “Interface Options,” select “Legacy Camera Enable/Disable.” We want this enabled.

You’ll see a warning about future deprecation. That’s okay, we can still use this at present.

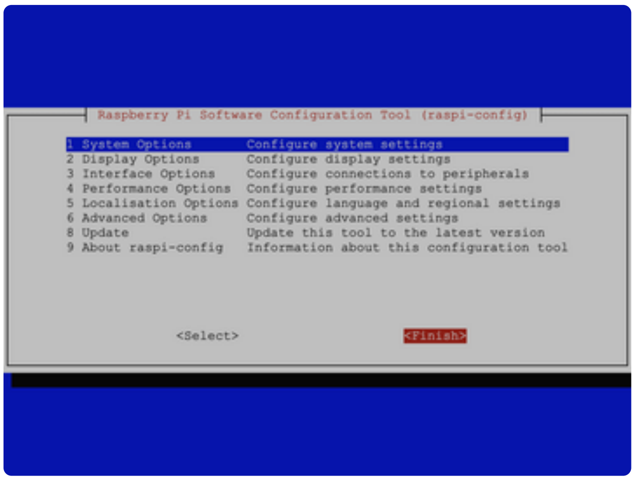
From the main menu, select “Finish” and “Yes” when asked to reboot. Then log back in and we’ll resume installing.



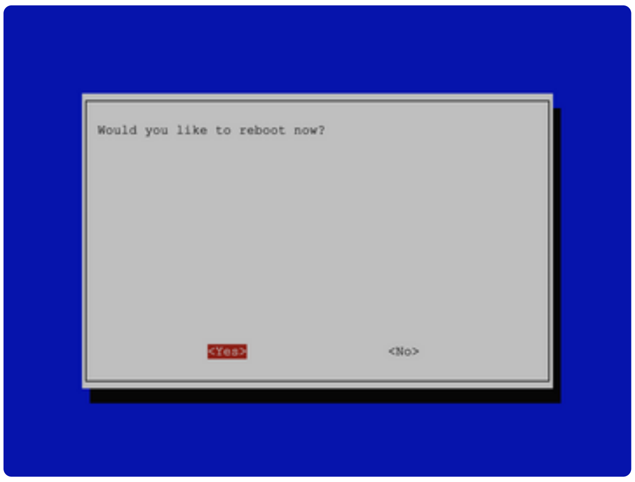
```
Would you like to enable legacy camera support?

          <Yes>               <No>
```



```
Legacy camera support is enabled.

Please note that this functionality is deprecated and
will not be supported for future development.

                          <Ok>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

              <Select>                  <Finish>
```



```
Would you like to reboot now?

               <Yes>                <No>
```

Back at the command line, the following commands then install most of the prerequisites…

```
sudo apt-get update
sudo apt-get install libatlas-base-dev libavcodec-dev libavformat-dev libgtk-3-dev
libopenjp2-7 libswscale-dev python3-pip
pip3 install numpy opencv-python picamera
```

(the second `apt-get` should be one long continuous line…it may appear wrapped in the browser, but if you click “Copy Code” this will grab these as whole lines)

You can ignore any messages about packages which are already installed or up to date. Also, there are a lot of packages here and very infrequently a server will fail to respond. If you encounter this, try again in a couple hours, it's usually just a hiccup at the other end.

There's one more prerequisite package to install…but we must note that this takes ext remely long to install! All the underlying face detection software must be downloaded and compiled from source code, a complex process.

Even on the latest Raspberry Pi, this could take a bit over an hour to run. On older models, especially single-core, several hours. You might want to start this before going to bed, or out for errands.

Ready? Here we go:

```
pip3 install opencv-contrib-python
```

Answer "Y" when prompted to start the lengthy install process.

Project Software Install

After prerequisite software is all loaded above, download the software for this project using:

```
cd
wget https://github.com/adafruit/pi-facerec-box/archive/refs/heads/master.zip
unzip master.zip
```

This creates a new subdirectory pi-facerec-box-master in your home directory, containing the project code. Let's go in there to start using the pieces…

```
cd pi-facerec-box-master
```

Continue on to learn about training the face recognition algorithm to recognize your face.

Training

This project uses the eigenfaces algorithm in OpenCV to perform face recognition. To use this algorithm one first creates a set of training data with pictures of faces that are and are not allowed to open the box.

Included in the project is a large set of face images that are tuned for training with the face recognition algorithm. This is the database of faces published by research AT&T Laboratories Cambridge in the mid 90's. These faces make up the set of negative images which represent faces that are not allowed to open the box. You can see these images in the training/negative subdirectory of the project.

To generate images of the person who will be allowed to open the box, or positive training images, you can use the included capture-positives.py script. This script will take pictures with the box hardware and write them to the training/positive sub-directory (which will be created by the script if it does not exist).

With the box hardware assembled and powered up (servo power is not necessary for this step), connect to the Raspberry Pi in a terminal session and navigate to the directory with the project software. Execute the following command to run the capture_positive script:

```
python capture_positives.py
```

After a moment for the script to load, you should see the following instructions:

Capturing positive training images.
Press button or type c (and press enter) to capture an image.
Press CTRL-C to quit.

The script is now ready to capture images for training. If you see an error message, make sure OpenCV and the software dependencies from the previous step () are installed and try again.

Point the box's camera at your face and press the button on the box (or send the letter 'c' in the terminal session) to take a picture. If the script detects a single face, it will crop and save the image in the positive training images sub-directory. If the script can't detect a face or detects multiple faces, an error message will be displayed.

If see error messages about not detecting a single face, you can examine the full picture that was captured by the camera to understand if it's picking up your face (or if other faces might be in view--be careful, even photos or posters with faces behind you can be detected!). Sometimes it’s just that the camera is upside-down. Open the file capture.pgm (located in the directory with the scripts) in an image editor to see the last image that was captured. It's easiest to use a tool like Cyberduck () to view the files on your Raspberry Pi over SFTP (SSH file transfer protocol).

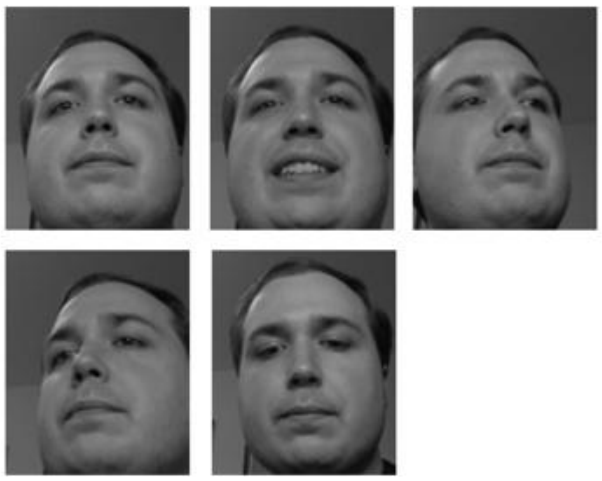
NOTE: As each image is captured, you might get an error message like the following:

imwrite_('capture.pgm'): can't write data: OpenCV(4.6.0) [...]

error: (-5:Bad argument) Portable bitmap(.pgm) expects gray image in function 'write'

This is non-fatal and can be ignored. The image is gray, something in an underlying library is just reporting wrong. Sorry!

Using the capture_positives.py script, capture some pictures of the face that is allowed to open the box. Try to take pictures of the face with different expressions, under different lighting conditions, and at different angles. You can see the images I captured for my positive training data below:



Training images I captured for my face. Try to get pictures with different expressions, in different light, etc. to build a better face recognition model.

Exit the capture_positives.py script by pressing CTRL-C in the terminal session.

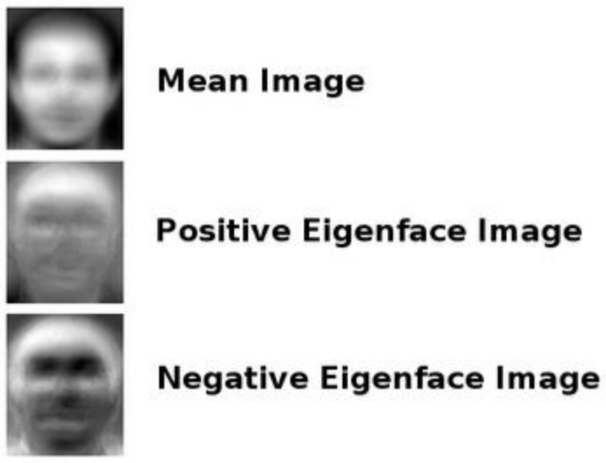
After capturing a number of positive training images, you can run the train.py script to perform the training. Run the following python script to train the face recognition model:

```
python train.py
```

You should see a message that the training images were loaded, and the text 'Training model...' to indicate the training calculations are being performed. Training the face recognition model on the Pi can take several minutes.

Once the training is complete you should see the message 'Training data saved to training.xml'. The training data is now stored in the file training.xml, which will be loaded by the box software to configure the face recognition model.

If you're curious you can examine images that are output by the training to visualize the eigenfaces of the model. Open the mean.png, positive_eigenface.png, and negative_eigenface.png files in an image viewer. You don't need to do anything with these images, but can learn more about their meaning from the OpenCV website, Wikipedia, or the original research paper on eigenfaces.



Eigenfaces generated from my training data.

The positive eigenface is a summary of the features that differentiate my face from the average or mean face. This face will be allowed to unlock the box.

The negative eigenface represents all the other faces in the training set. This face will not be allowed to unlock the box.

With the training.xml file generated, you're ready to move on to configure the servo latch and software for the project.

Configuration

Servo Configuration

In this step you'll determine the positions for the locked and unlocked servo positions of the box. Make sure the hardware is assembled, the software dependencies are installed, and power is applied to both the Raspberry Pi and servo before you continue. Connect to the Raspberry Pi in a terminal session, “cd” to the project directory and type the following command:

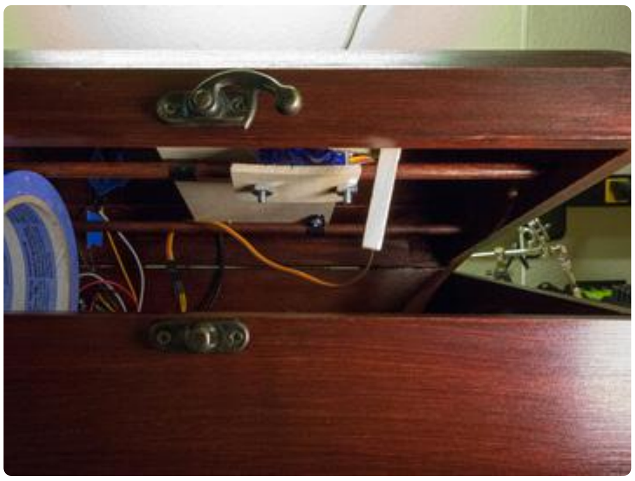
```
python servo.py
```

Servos operate on timed pulses, typically from 1000 to 2000 microseconds long (but a few servos may function a bit beyond this range), covering the full “sweep” of the servo (usually 180 degrees, sometimes 90 for high-torque units).

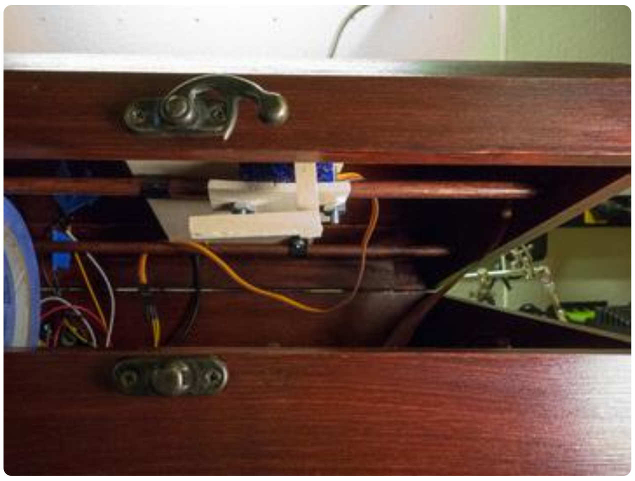
Using this script, you can interactively type in different timing values and see how this affects servo position. Idea is to progressively dial in on good values for “open” and “locked” positions.

If your servo is connected to a different pin than GPIO18, edit the file config.py and change the number associated with LOCK_SERVO_PIN.

You can see the values I found for my hardware below:



Latch in the unlocked position with a servo pulse width of 2000 microseconds.



Latch in the locked position with a servo pulse width of 1100 microseconds.

Take note of the pulse width values you determined for the locked and unlocked position as they will be used later in the configuration, then press CTRL-C to exit.

Software Configuration

In the directory with code for this project, open the file config.py in a text editor. This file defines the configuration for the project and might require a few changes for your hardware. Specifically you can adjust the following values:

- LOCK_SERVO_PIN - this should be the GPIO number of the Raspberry Pi which is connected to the signal line of the lock servo. This project is built to use GPIO 18 by default.
- LOCK_SERVO_UNLOCKED - this should be the pulse width value (in microseconds) for the unlocked position of the latch. Use the value you determined in the previous step.
- LOCK_SERVO_LOCKED - this should be the pulse width value for the locked position of the latch.

- BUTTON_PIN - this should be the GPIO number which is connected to the push button on the box. This project is built to use GPIO 25 by default.

The rest of the values in the configuration can be ignored. The only other value of interest is the POSITIVE_THRESHOLD configuration, which gives the boundary for how confident the face recognition has to be before it's considered a positive match. For now the default value of 2000 should suffice, but it can be adjusted later if the box has too many false positive or false negative recognitions.

Continue on to run the box software and finish the project.

Usage

To run the box software, make sure the hardware is assembled and the training () is complete. First you'll run the box with only power to the Raspberry Pi and not the lock servo--this will allow you to test the face recognition without locking yourself out of the box. Apply power to just the Raspberry Pi (and not the servo), then connect to the Pi in a terminal session, navigate to the folder with the software, and execute the following command:

```
python box.py
```

You should see the box software load and the message 'Loading training data...' appear. Allow a moment for the training data to load and the following message to appear:

```
Running box...
Press button to lock (if unlocked), or unlock if the correct face is detected.
Press CTRL-C to quit.
```

The box will lock itself (if no power is applied to the servo yet, the lock latch should not move) and be ready to try unlocking with face recognition. Aim the box camera at your face, like you did when training the face recognition model, and press the button on the box. You should see a message that the button was pressed, and after a moment if the face is recognized you will see a message like:

```
Predicted POSITIVE face with confidence 1321.35253959 (lower is more confident).
Recognized face!
```

If the face is not recognized you will see a message like:

```
Predicted NEGATIVE face with confidence 3987.76625152 (lower is more confident).
Did not recognize face!
```

If a single face couldn't be detected (because none is visible, or there are multiple faces detected) an error message will be displayed like with the training script.

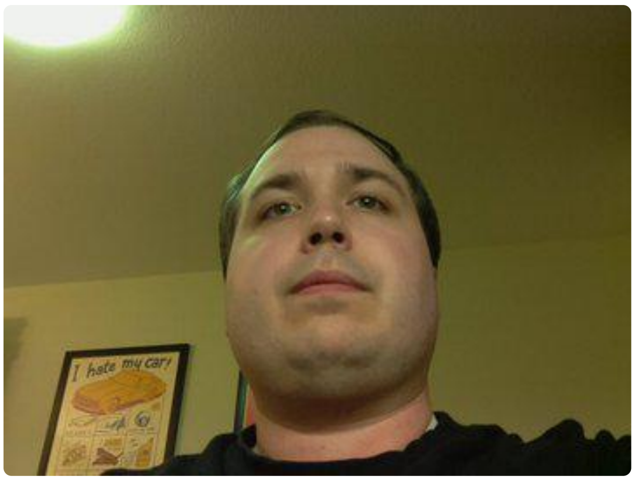
When a face is found, you'll notice the prediction message includes how the face was recognized (i.e. matching either the positive or negative training data), and the confidence of the recognition. The confidence is a value like 1321.35 or 3987.77 above and represents the distance of the captured face from the predicted face--a lower distance value means the captured face is more similar to the predicted face.

To unlock the box, a captured face must be a positive prediction with a confidence value below the POSITIVE_THRESHOLD configuration value. By default the POSITIVE_THRESHOLD is 2000, so the positive prediction with confidence 1321.35 was considered a match and would have unlocked the box.

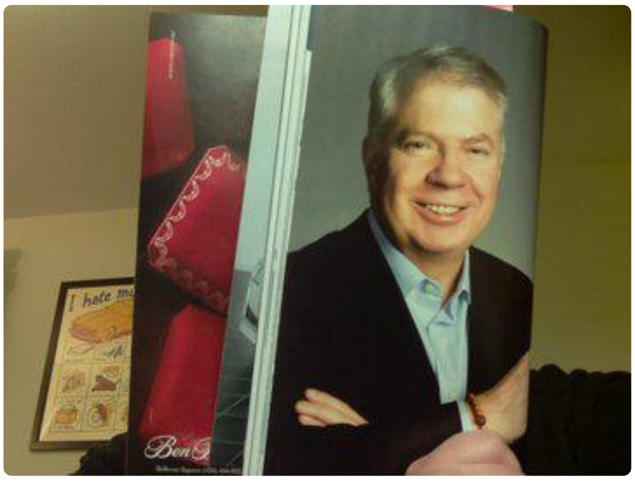
When the box is in an unlocked state, press the button to lock the box again. You do not need to have a face recognized to lock the box.

Play with unlocking and locking the box to see what prediction and confidence you get from your trained face recognition model. Ideally you want images of the positively trained face to be predicted as positive images with a low number for the confidence value. The eigenfaces recognition algorithm is sensitive to the lighting and orientation of your face, so try to have as similar conditions as when the positive training images were captured (or consider re-training your model to include more varied images).

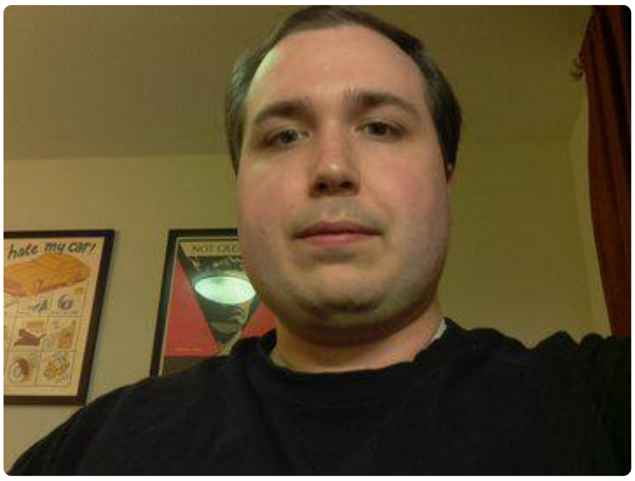
You can see examples of how my training data predicted faces below (remember you can look at the capture.pgm image in the software directory after pressing the button to attempt a face recognition):



This image unlocked my box because it was a match for the positive training images, and had a confidence of 1321.35 which was below the threshold of 2000.

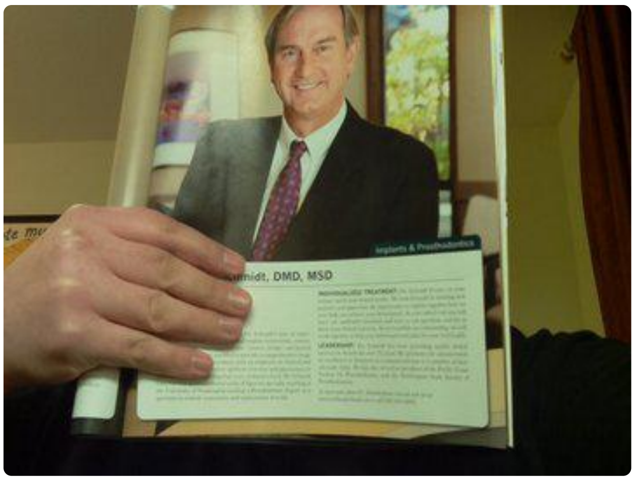


This image of a face from a magazine did not unlock my box because it was a match for the negative training data, with a confidence of 3987.77.



This image unfortunately did not unlock my box. Although it was predicted as a positive match, the confidence was 2269.04 which was slightly higher than the positive match threshold.

If you get a lot of false negative predictions like this, it's a sign you need better training data or should adjust the POSITIVE_THRESHOLD configuration value up.



This image matched the positive training data, but the confidence value of 4407.85 was too far above the threshold to unlock the box.

If you get a lot of false positive responses like this it's also a sign that you need better training data. Remember face recognition is far from perfect--there's always a chance someone who looks like the positively trained images will match close enough to unlock the box!

Based on how the face recognition responds, you might consider training it again or adjust the POSITIVE_THRESHOLD configuration to a higher/lower value.

When you're happy with how the recognition is working, apply power to the servo and try locking and unlocking the box for real. Congratulations on building the face recognition treasure box!

If for some reason your box is locked and will not recognize a face to unlock itself, you can manually set the servo position to an unlocked value by following the steps in the servo configuration (). If you don't have access to the Raspberry Pi terminal or the servo isn't moving, unfortunately you'll need to disassemble your box (this is why it's a good idea to use a box with hinges on the outside!).

Future Work

This project is a great example of how to use the Raspberry Pi and Pi camera with OpenCV's computer vision algorithms. By compiling the latest version of OpenCV, you can get access to the latest and most interesting computer vision algorithms like face recognition. You can consider extending this project in other interesting ways, such as:

- Adding LED's to the box which flash when a face is detected and recognized. This feedback would help you train and use the box without having to connect to the Pi in a terminal session.
- Make the main box.py script start up automatically when the Raspberry Pi powers on. Search the web for “systemd startup” for tips on automatically running a script at start up.

- Investigate using other face recognition algorithms in OpenCV. See the OpenCV tutorial on face recognition () for more information. Algorithms such as Fisherfaces might be more robust to varied lighting or other conditions.
- Have the box send you an email with a picture of whoever is attempting to open it.
- Add a microphone and look at adding speech recognition () as another means of unlocking the box.

Can you think of other ways to extend the project?