Raspberry Pi Halloween Costume Detector

Created by Liz Clark

https://learn.adafruit.com/raspberry-pi-halloween-costume-detector

Last updated on 2025-11-25 01:30:05 PM EST

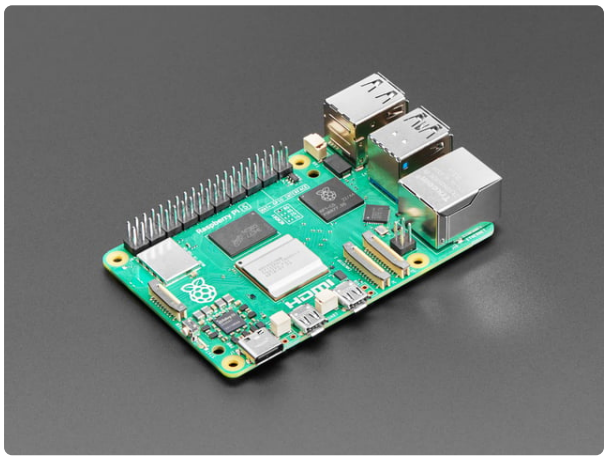
Table of Contents

Overview

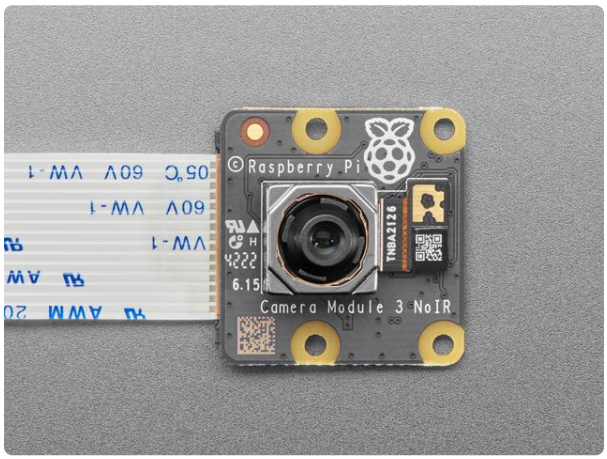
Build a wise cracking skull to MC your Halloween costume party. This project uses a Raspberry Pi 5 paired with a NoIR PiCamera and speaker bonnet. The Pi runs a Python script with OpenCV person detection. When a person crosses paths with the skull, a picture is sent to Claude (https://adafru.it/1auS) with a prompt requesting a kind hearted corny dad joke about the person's Halloween costume. The returned joke is sent through Piper text to speech (https://adafru.it/1auT) and played through a speaker attached to the speaker bonnet.

All of the electronics mount to the back of a 3D printed skull plate. A back plate with M3-sized standoffs encloses the build and helps it stand upright.

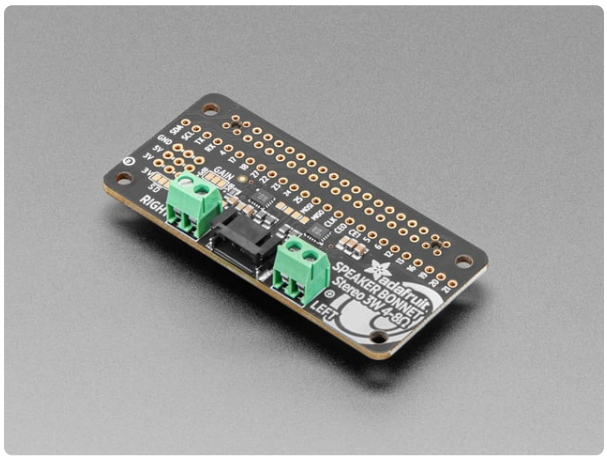
Parts



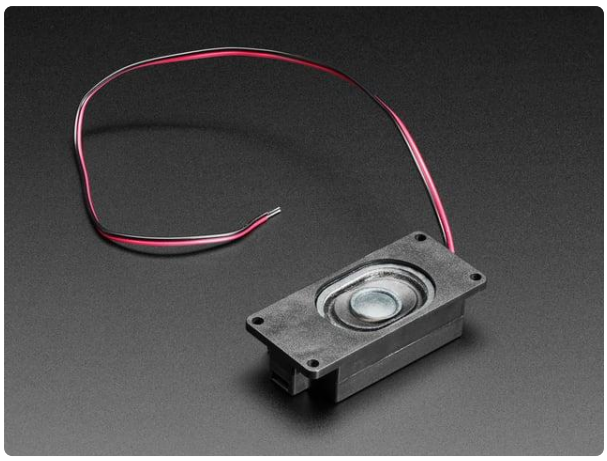
[Raspberry Pi 5 - 4 GB RAM]
The Raspberry Pi 5 is the newest Raspberry Pi computer, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 5 better than the...
https://www.adafruit.com/product/5812



[Raspberry Pi Camera Module 3 NoIR - 12MP 75 Degree Infrared Lens]
Raspberry Pi Camera Module 3 is a compact camera from Raspberry Pi. It offers an IMX708 12-megapixel sensor with HDR and features phase detection autofocus. Camera Module 3 is...
https://www.adafruit.com/product/5659



[Adafruit I2S 3W Stereo Speaker Bonnet for Raspberry Pi]
Hey Mr. DJ! Turn up that Raspberry Pi mix to the max with this cute 3W Stereo Amplifier Bonnet for Raspberry Pi. (It's not big enough to be an...
https://www.adafruit.com/product/3346

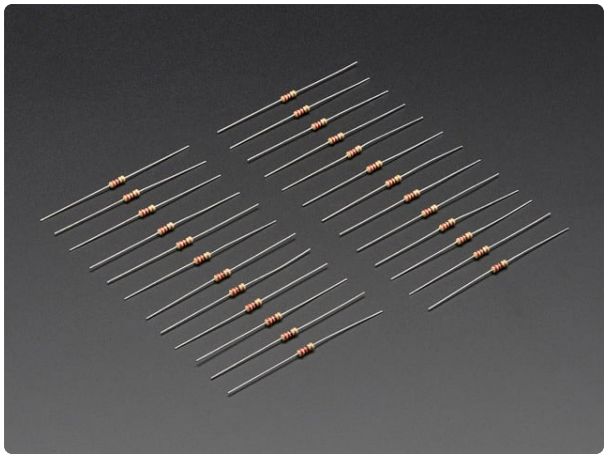


Mono Enclosed Speaker with Plain Wires - 3W 4 Ohm

Listen up! This single 2.8" x 1.2" speaker is the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We...
https://www.adafruit.com/product/4445

Diffused 10mm LED Pack - 5 LEDs each in 5 Colors - 25 Pack

Need some chunky indicators? We are big fans of these diffused LEDs. They are fairly bright, so they can be seen in daytime, and from any angle. They go easily into a breadboard and...
https://www.adafruit.com/product/4204



Through-Hole Resistors - 220 ohm 5% 1/4W - Pack of 25

ΩMG! You're not going to be able to resist these handy resistor packs! Well, axially, they do all of the resisting for you!This is a 25 Pack of...
https://www.adafruit.com/product/2780

1 x Raspberry Pi 5 FPC Camera Cable
22-pin 0.5mm to 15-pin 1mm
https://www.adafruit.com/product/5819

---

1 x Official Raspberry Pi Power Supply
5.1V 3A with USB C
https://www.adafruit.com/product/4298

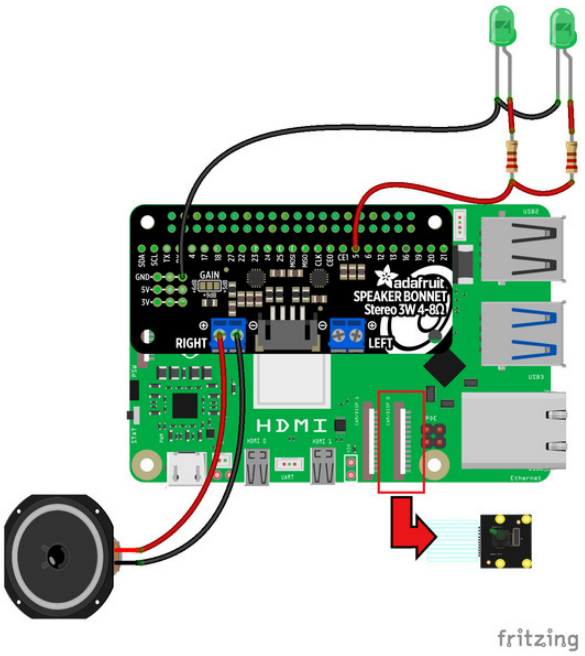
---

1 x Silicone Cover Stranded-Core Wire
30AWG in Various Colors
https://www.adafruit.com/product/2051

---

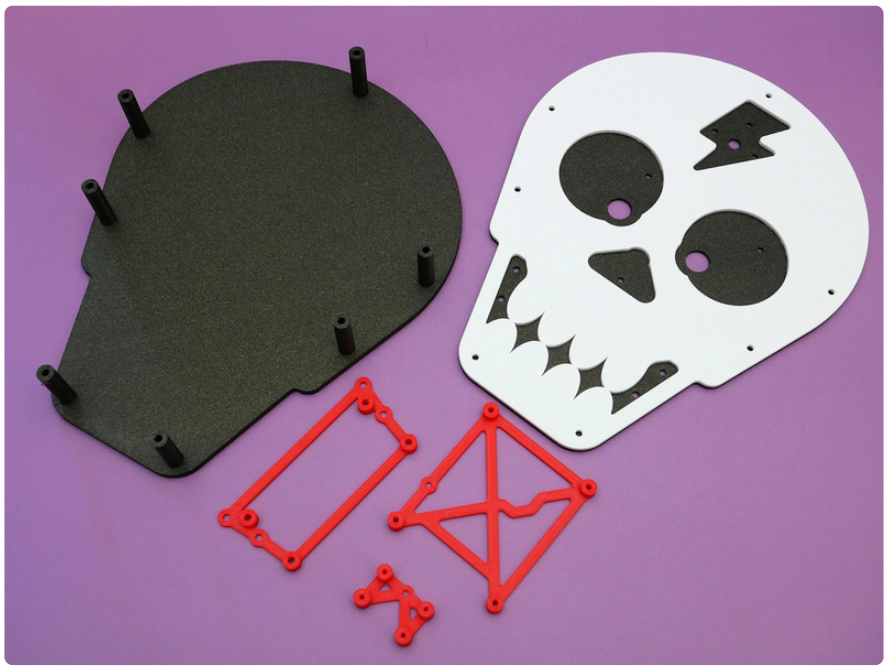
1 x Multi-Colored Heat Shrink Pack 3/32" + 1/8" + 3/16" Diameters	https://www.adafruit.com/product/1649
1 x Black Nylon Machine Screw and Stand-off Set - M3 Thread M3 thread	https://www.adafruit.com/product/4685
1 x Black Nylon Machine Screw and Stand-off Set – M2.5 Thread M2.5 Thread	https://www.adafruit.com/product/3299
1 x M2 Screws Assortment Kit	https://www.amazon.com/Hapric-Assortment-Machine-Stainless-Washers/dp/B0D3X33XGB/?th=1

Circuit Diagram



- Plug the Speaker Bonnet into the GPIO header on the Pi 5
- Plug the camera ribbon cable into camera port 0 on the Pi 5
- Speaker positive to speaker right + (red wire)
- Speaker negative to speaker right - (black wire)
- LED cathode to Pi GND (black wire)
- LED anode to 220Ω resistor (red wire)
- 220Ω resistor to Pi GPIO 5 (red wire)

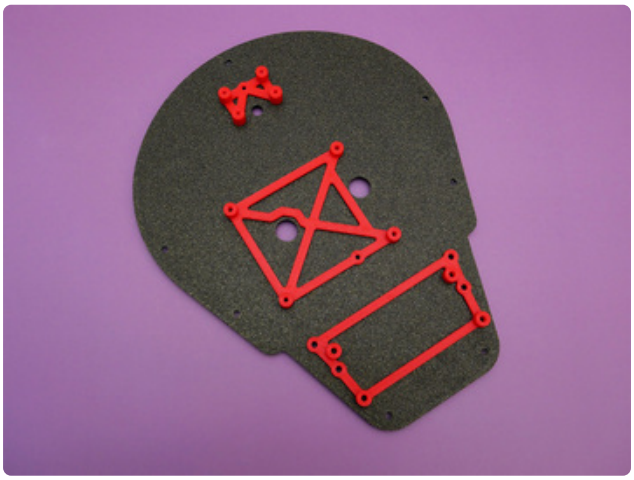
3D Printing



You can 3D print all of the parts for this project. The STL files can be downloaded directly below or from Printables.

https://adafru.it/1auU

https://adafru.it/1auV



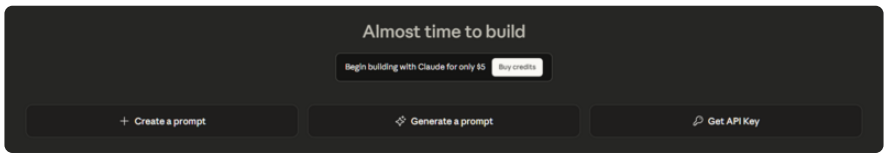
Three mounting plates secure the electronics hardware and attach them to the skull.

The back plate has standoffs to connect to the front skull plate with M3 screws and protect the electronics.

Claude API Key

This project uses the Claude API to analyze the image of the Halloween costume and generate a dad joke about the costume. You'll need an Anthropic Console account (https://adafru.it/1auW) to follow these steps.

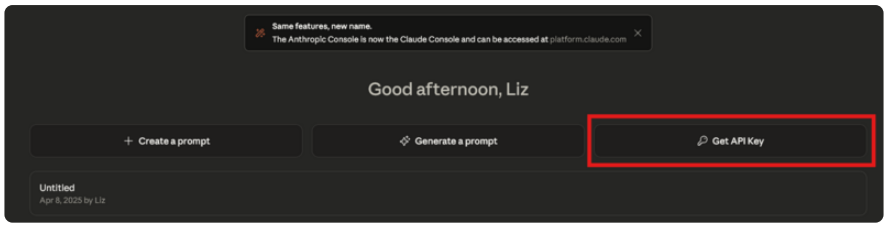
If you don't want to use the Claude API, the project has fallbacks to work without it. You just won't be able to generate custom jokes for each image.



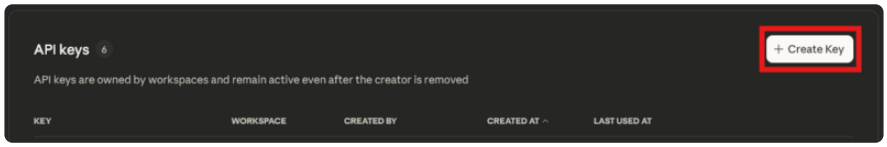
need to buy API credits before you can generate an API key.

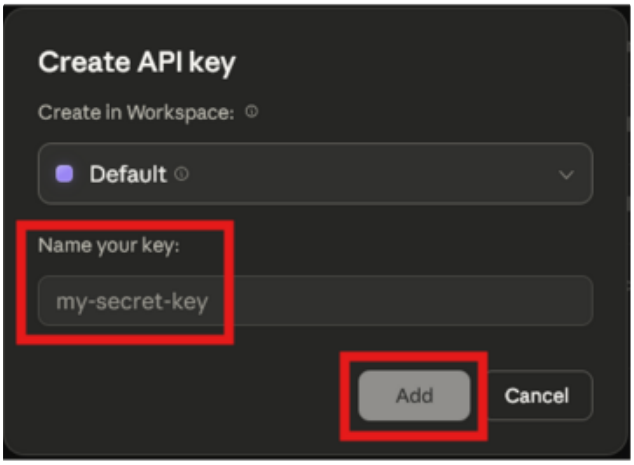
Navigate to https://console.anthropic.com (https://adafru.it/1auX) and log into your console account. If you haven't done so, you'll need to add API credits to your account by clicking the "**Buy credits**" button. The project is designed to use a minimal number of credits, you shouldn't need more than $1 to $2.

After adding credits, you can click the **Get API Key** button.



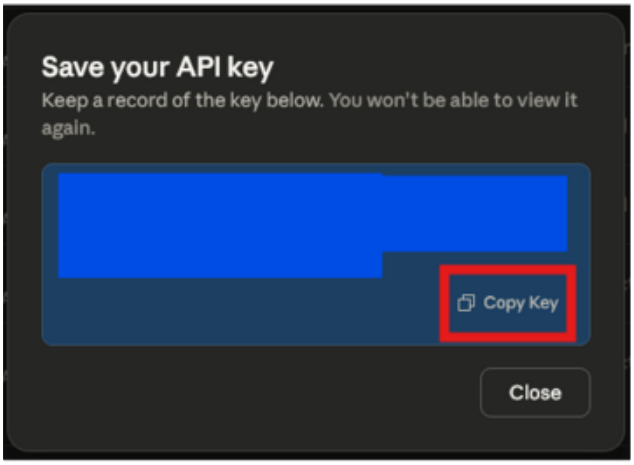
This brings you to the API keys page. Click the + **Create Key** button.





In the **Name your key: box**, give your API key a name. Then click **Add**.

cannot view your API key again after closing the "Save your API key" low! Make sure you copy the key and save it somewhere safe.



This opens a windows with your API key. Click the **Copy Key button** to copy your key text and then make sure you save it somewhere. **You cannot view the key again after closing this window!**

Speaker Bonnet Setup

> ıis time, Raspbery Pi linux kernel does not support mono audio out of the nterface, you can only play stereo, so any mono audio files may need ⁄ersion to stereo!

> ʼ-11-2 Raspbian PIXEL ('full') has broken something in volume control. I2S ‹s, but there's no software volume setup, if you need this, try Raspbian - will try to fix as soon as we figure out why :)

Setup Virtual Environment

If you are installing on the Bookworm version of Raspberry Pi OS or later, you will need to install your python modules in a virtual environment. You can find more information in the Python Virtual Environment Usage on Raspberry Pi (https://adafru.it/19a5) guide. To Install and activate the virtual environment, use the following commands:

```
sudo apt install python3-venv
python -m venv env --system-site-packages
```

To activate the virtual environment:

```
source env/bin/activate
```

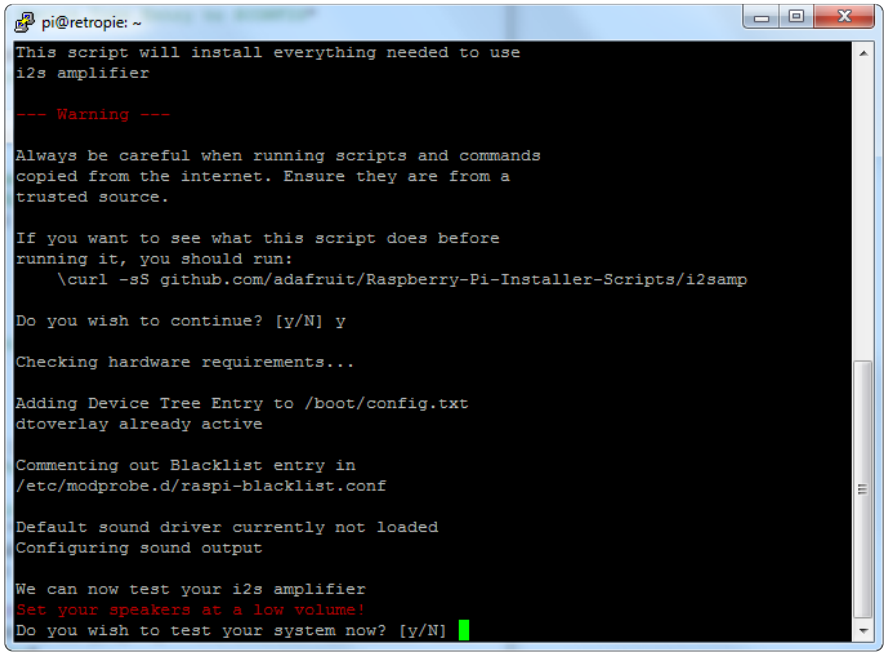
Installer script

Luckily its quite easy to install support for I2S DACs on Raspbian.

[These instructions are totally cribbed from the PhatDAC instructions at the lovely folks at Pimoroni!](https://adafru.it/nFy)

Run the following from your Raspberry Pi with Internet connectivity:

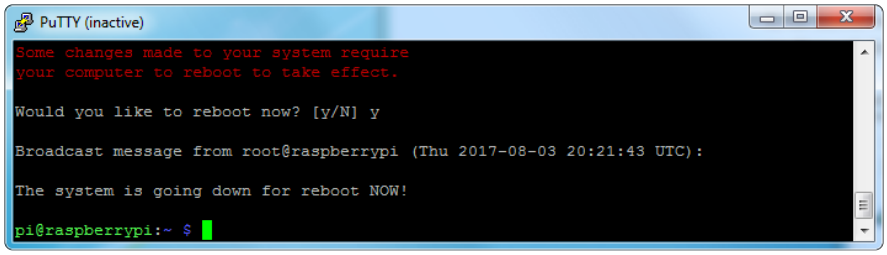
```
sudo apt install -y wget
pip3 install adafruit-python-shell
wget https://github.com/adafruit/Raspberry-Pi-Installer-Scripts/raw/main/i2samp.py
sudo -E env PATH=$PATH python3 i2samp.py
```



We've added an extra helper systemd script that will play quiet audio when the I2S peripheral isn't in use. This removes popping when playback starts or stops. It uses a tiny amount of CPU time (on a Pi Zero, 5%, on a Pi 2 or 3 its negligible). You don't need this on RetroPie because it never releases the I2S device, but it's great for Raspbian.

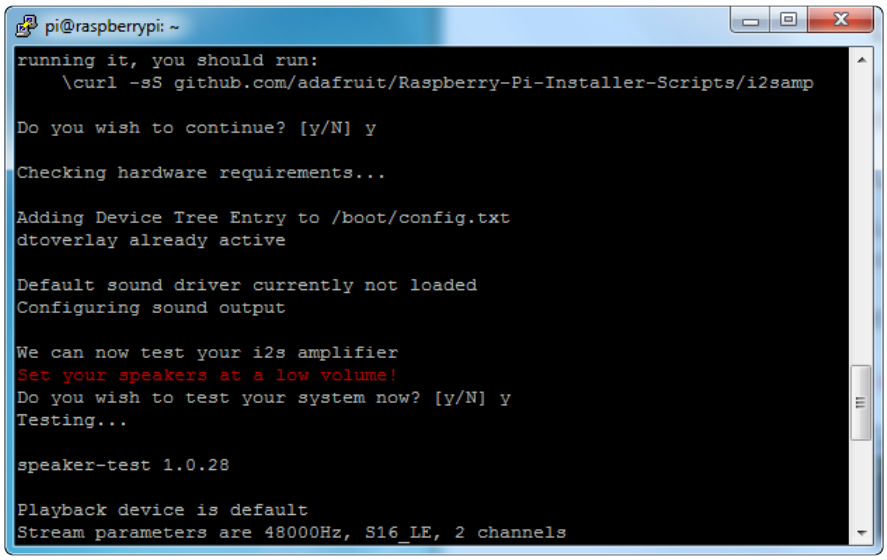
```
Installing aplay systemd unit
Removed /etc/systemd/system/multi-user.target.wants/aplay.service.

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N] y
```

You will need to reboot once installed.



must reboot to enable the speaker hardware!

After rebooting, log back in and re-run the script again...It will ask you if you want to test the speaker. Say **y**es and listen for audio to come out of your speakers...



If it sounds really distorted, it could be the volume is too high. However, in order to have volume control appear in Raspbian desktop or Retropie you must reboot a second time after doing the speaker test, with **sudo reboot**

> must reboot *twice* to enable alsamixer volume (really!)

Once rebooted, try running **alsamixer** and use arrow keys to lower the volume, 50% is a good place to start.

If you're still having audio problems, try re-running the script and saying **N** (disable) the `/dev/zero playback service`.

You can then go to the next page on testing and optimizing your setup. Skip the rest of this page on **Detailed Installation** if the script worked for you!

Detailed Install

If, for some reason, you can't just run the script and you want to go through the install by hand - here's all the steps!

Update /etc/modprobe.d (if it exists)

Log into your Pi and get into a serial console (either via a console cable, the TV console, RXVT, or what have you)

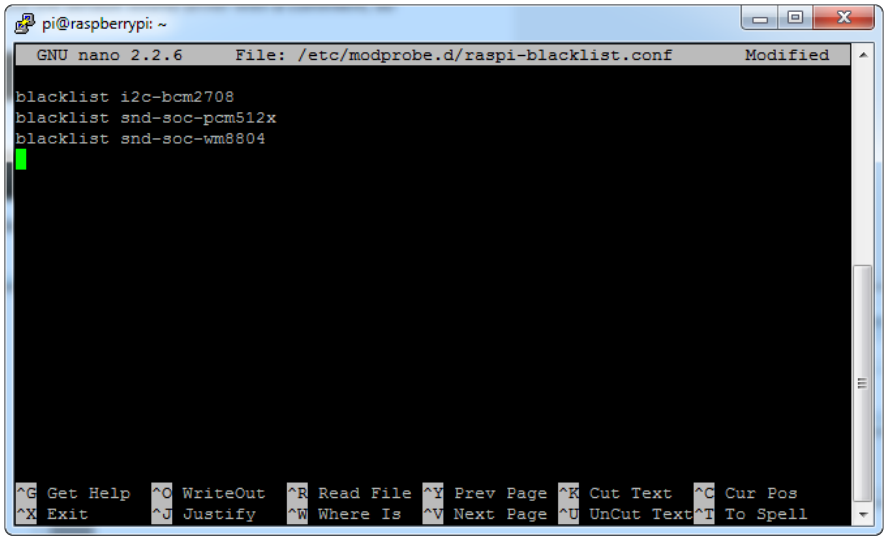
Edit the raspi blacklist with

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

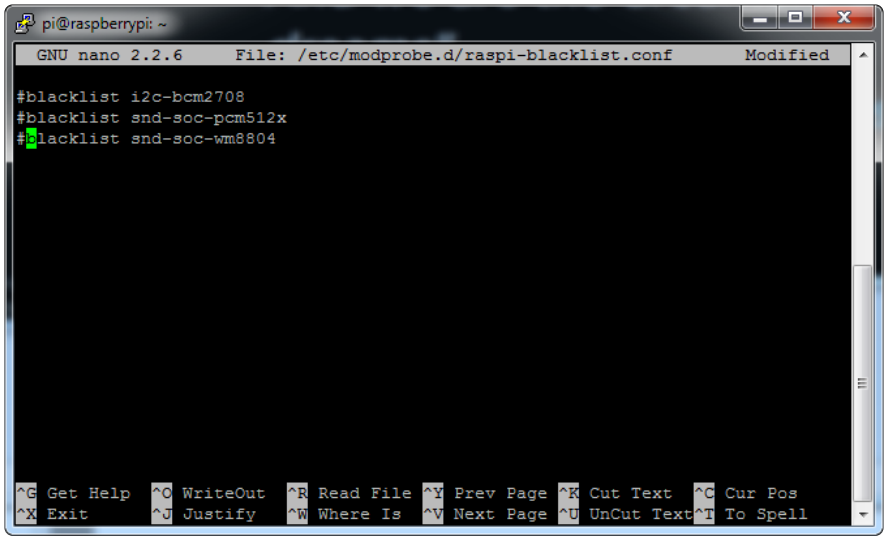
If the file is empty, just skip this step

However, if you see the following lines:

```
blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804
```



Update the lines by putting a # before each line



Save by typing **Control-X Y <return>**

Disable headphone audio (if it's set)

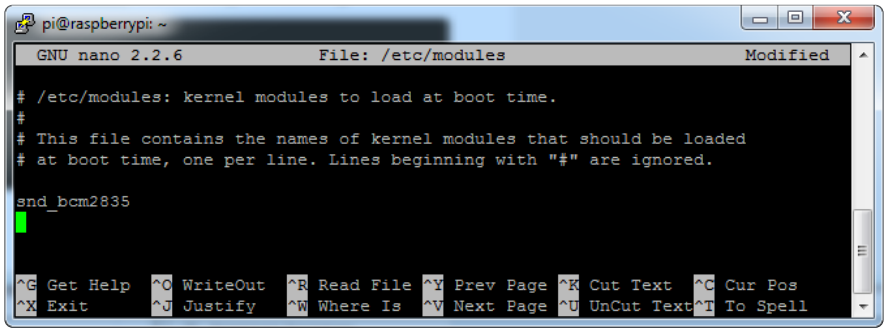
Edit the raspi modules list with

```
sudo nano /etc/modules
```

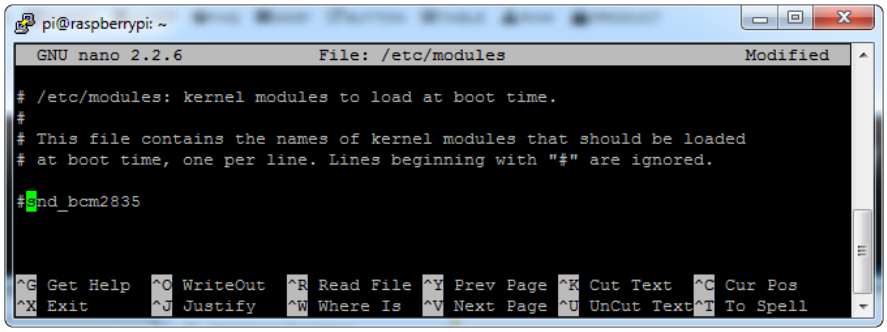
If the file is empty, just skip this step

However, if you see the following line:

`snd_bcm2835`



Put a # in front of it



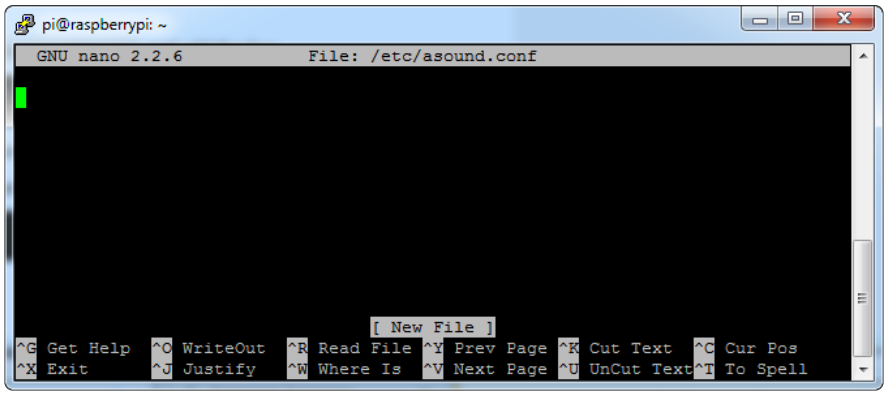
and save with **Control-X Y <return>**

Create asound.conf file

Edit the raspi modules list with

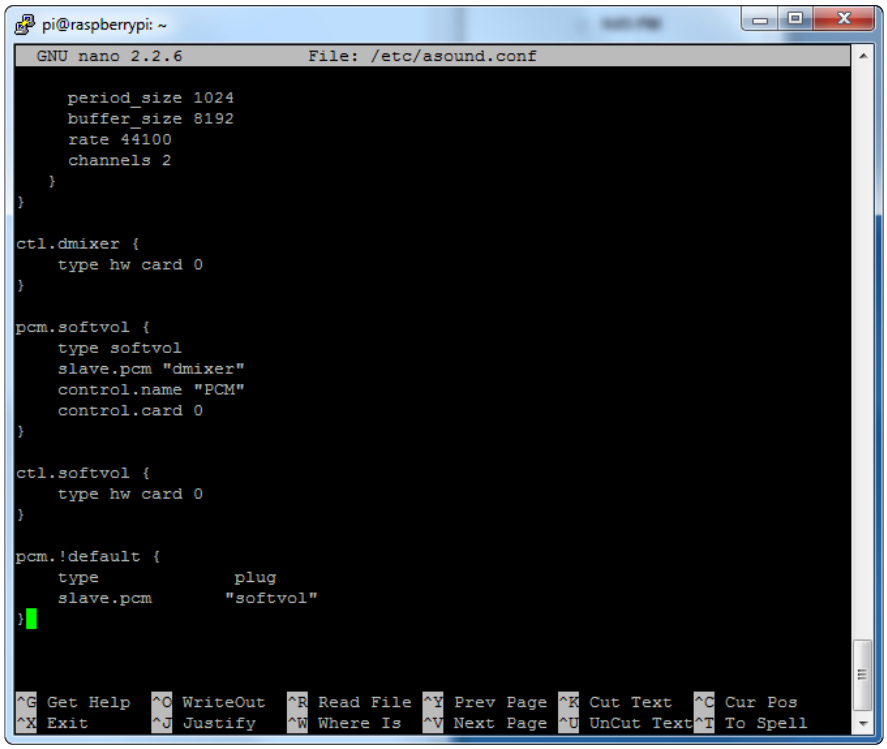
```
sudo nano /etc/asound.conf
```

This file ought to be blank!



Copy and paste the following text into the file

```
pcm.speakerbonnet {
   type hw card 0
}

pcm.dmixer {
   type dmix
   ipc_key 1024
   ipc_perm 0666
   slave {
     pcm "speakerbonnet"
     period_time 0
     period_size 1024
     buffer_size 8192
     rate 44100
     channels 2
   }
}

ctl.dmixer {
    type hw card 0
}

pcm.softvol {
    type softvol
    slave.pcm "dmixer"
    control.name "PCM"
    control.card 0
}

ctl.softvol {
    type hw card 0
}

pcm.!default {
    type             plug
    slave.pcm       "softvol"
}
```



Save the file as usual

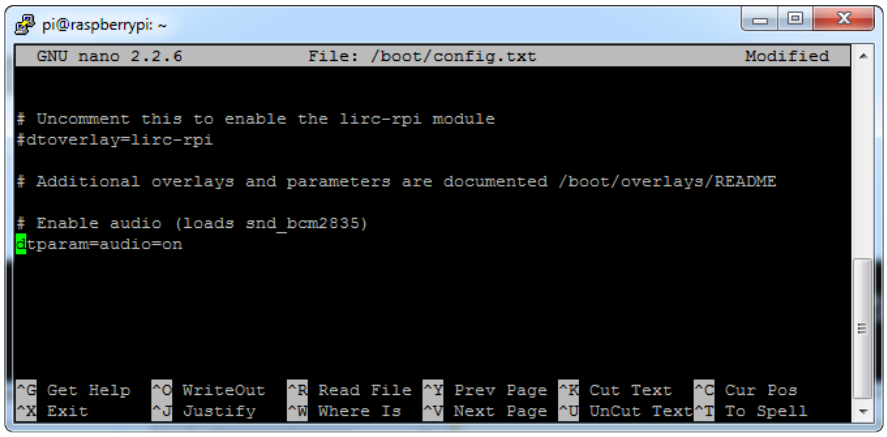
Add Device Tree Overlay

Edit your Pi configuration file with

> ɔlder versions of Raspberry Pi, edit /boot/config.txt instead.

```
sudo nano /boot/firmware/config.txt
```

And scroll down to the bottom. If you see a line that says: `dtparam=audio=on`



Disable it by putting a # in front.

Then add:

```
dtoverlay=max98357a
```

on the next line. Save the file and reboot your Pi with

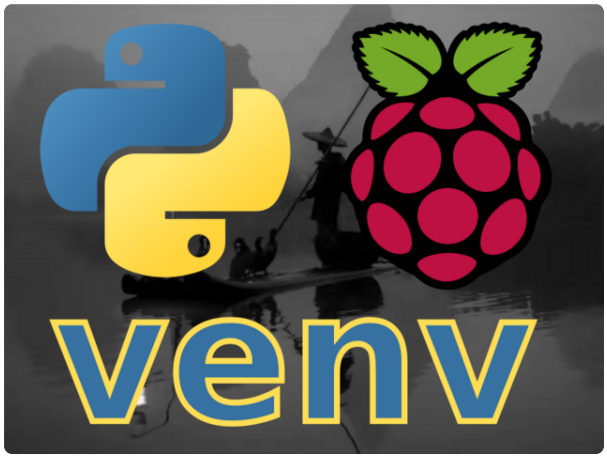
```
sudo reboot
```

Python Virtual Environment Prep

As Carter writes in his Python Virtual Environment Usage on Raspberry Pi guide (https://adafru.it/1a9O):

> Starting with the October 10, 2023 Bookworm release (https://adafru.it/199D) of the Raspberry Pi OS (https://adafru.it/XMd), the use of Python Virtual Environments (https://adafru.it/199E) (venv) when pip installing packages is required. **No more sudo pip**. This will break things and require learning new things. Yeah.

You will need to setup a Python virtual environment (venv) on your Raspberry Pi 5 for this project. Don't worry though! If you follow along with the guide step by step, you'll be just fine.



Python Virtual Environment Usage on Raspberry Pi
By Carter Nelson
Basic Venv Usage

https://learn.adafruit.com/python-virtual-environment-usage-on-raspberry-pi/basic-venv-usage

Always venv

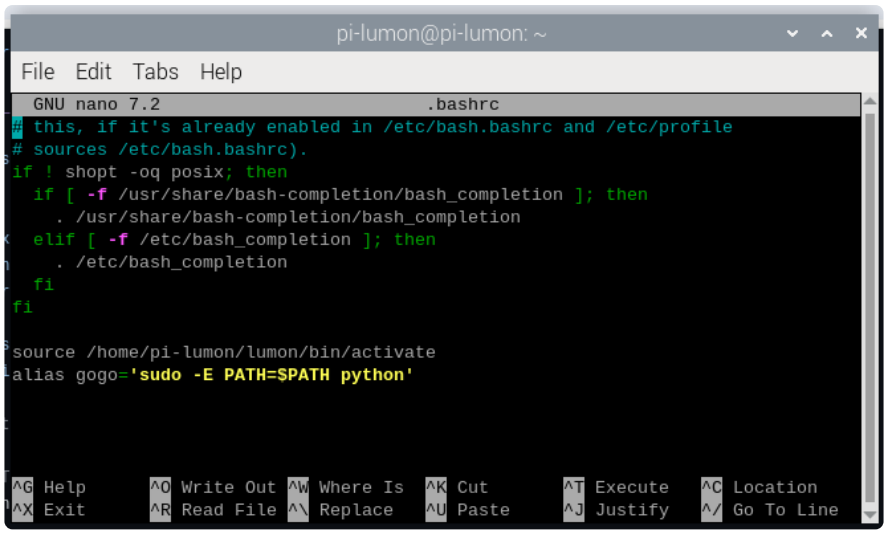
On the Other Ideas page (https://adafru.it/1a9P) in the venv guide, there is a tip for having the virtual environment enabled automatically at boot by editing the `.bashrc` file (`sudo nano .bashrc`) and adding this line to the bottom:

```
source home/user/venv/bin/activate
```

Where `venv` is the name of your virtual environment. You can take this a step further by adding the alias (https://adafru.it/1a9Q) for running Python scripts as `sudo` to your `.bashrc` file as well:

```
alias gogo='sudo -E env PATH=$PATH python'
```

You can change `gogo` to any command you want. This way, every time you boot up your Pi, you'll have your Python venv enabled and you'll be able to use your alias for running Python scripts.



Installing Blinka on Raspberry Pi

uitPython libraries and adafruit-blinka will work on any Raspberry Pi board! : means the original 1, the Pi 2, Pi 3, Pi 4, Pi 5, Pi Zero, Pi Zero 2 W, or even compute modules.

iis time, Blinka requires Python version 3.7 or later, which means you will d to at least be running Raspberry Pi OS Bullseye.

Prerequisite Pi Setup!

In this page we'll assume you've already gotten your Raspberry Pi up and running and can log into the command line

Here's the quick-start for people with some experience:

1. Download the latest Raspberry Pi OS or Raspberry Pi OS Lite (https://adafru.it/Pf5) to your computer
2. Burn the OS image to your MicroSD card (https://adafru.it/dDL) using your computer

3. Re-plug the SD card into your computer (don't use your Pi yet!) and set up your wifi connection by editing supplicant.conf (https://adafru.it/yuD)
4. Activate SSH support (https://adafru.it/yuD)
5. Plug the SD card into the Pi
6. If you have an HDMI monitor we recommend connecting it so you can see that the Pi is booting OK
7. Plug in power to the Pi - you will see the green LED flicker a little. The Pi will reboot while it sets up so wait a good 10 minutes
8. If you are running Windows on your computer, install Bonjour support so you can use **.local** names, you'll need to reboot Windows after installation (https://adafru.it/lPE)
9. You can then **ssh** into **raspberrypi.local** (https://adafru.it/jvB)

The Pi Foundation has tons of guides as well (https://adafru.it/BJY)

> really really recommend the latest Raspberry Pi OS only. If you have an r Raspberry Pi OS install, run "sudo apt-get update" and "sudo apt-get rade" to get the latest OS!

Update Your Pi and Python

Run the standard updates:

```
sudo apt-get update
sudo apt-get -y upgrade
sudo apt-get install -y python3-pip
```

and upgrade setuptools:

```
sudo apt install --upgrade python3-setuptools
```

> may need to reboot prior to installing Blinka. The raspi-blinka.py script will m you if it is necessary.

u are installing in a virtual environment, the installer may not work ectly since it requires sudo. We recommend using pip to manually install it at case.

Setup Virtual Environment

If you are installing on the Bookworm (released in 2023) or later version of Raspberry Pi OS, you will need to install your python modules in a virtual environment. You can find more information in the Python Virtual Environment Usage on Raspberry Pi (https://adafru.it/19a5) guide. To Install and activate the virtual environment, use the following commands:

```
cd ~
sudo apt install python3-venv
python3 -m venv env --system-site-packages
```

You will need to activate the virtual environment every time the Pi is rebooted. To activate it:

```
source env/bin/activate
```

To deactivate, you can use `deactivate`, but leave it active for now.

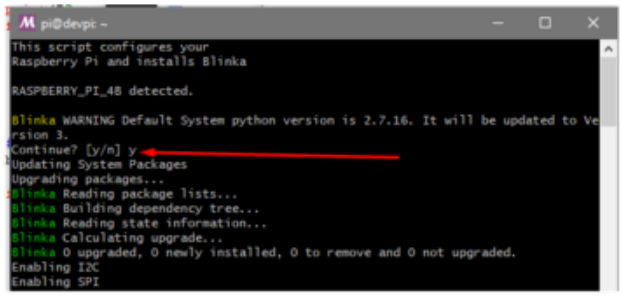
Automated Install

We put together a script to easily make sure your Pi is correctly configured and install Blinka. It requires just a few commands to run. Most of it is installing the dependencies.

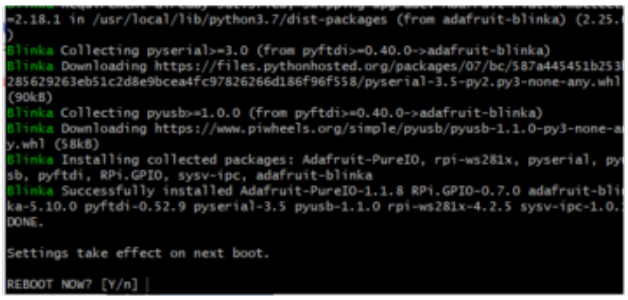
```
cd ~
pip3 install --upgrade adafruit-python-shell
wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/
master/raspi-blinka.py
sudo -E env PATH=$PATH python3 raspi-blinka.py
```

If you are installing on an earlier version such as Bullseye of Raspberry Pi OS and not using a Virtual Environment, you can call the script like:

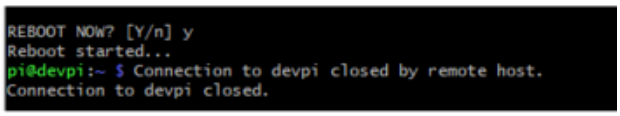
```
sudo python3 raspi-blinka.py
```



If you are installing an older version of Raspberry Pi OS, your system default Python is likely Python 2. If so, it will ask to confirm that you want to proceed. Choose **yes**.



It may take a few minutes to run. When it finishes, it will ask you if you would like to reboot. Choose **yes**.



Once it reboots, the connection will close. After a couple of minutes, you can reconnect.

Manual Install

If you are having trouble running the automated installation script, you can follow these steps to manually install Blinka.

Enable Interfaces

Run these commands to enable the various interfaces such as I2C and SPI:

```
sudo raspi-config nonint do_i2c 0
sudo raspi-config nonint do_spi 0
sudo raspi-config nonint do_serial_hw 0
sudo raspi-config nonint do_ssh 0
sudo raspi-config nonint do_camera 0
sudo raspi-config nonint disable_raspi_config_at_boot 0
```

Install Blinka and Dependencies

Blinka needs a few dependencies installed:

```
sudo apt-get install -y i2c-tools libgpiod-dev python3-libgpiod
pip3 install --upgrade adafruit-blinka
```

Raspberry Pi 5 Adjustments

At the moment, RPi.GPIO is installed, which causes issues. Just remove it with the following command:

```
pip3 uninstall -y RPi.GPIO
```

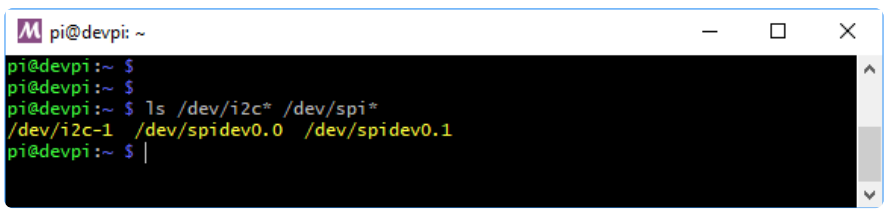
Check I2C and SPI

The script will automatically enable I2C and SPI. You can run the following command to verify:

```
ls /dev/i2c* /dev/spi*
```

You should see the response

`/dev/i2c-1 /dev/spidev0.0 /dev/spidev0.1`



Fixing CE0 and CE1 Device or Resource Busy Issue

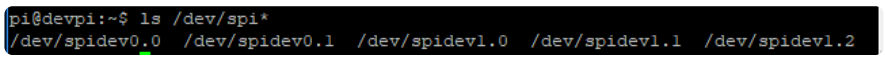
In order to use the CE0 and CE1 pins in Python, you will need to disable them from OS usage. To do so, check out the Reassigning or Disabling the SPI Chip Enable Lines (https://adafru.it/19fg) section of this guide.

Enabling Second SPI

If you are using the main SPI port for a display or something and need another hardware SPI port, you can enable it by adding the line

`dtoverlay=spi1-3cs`

to the bottom of **/boot/config.txt** and rebooting. You'll then see the addition of some /dev/spidev1.x devices:



Pi 5 : Cannot determine SOC peripheral base address

comment out this line :

```
#dtparam=spi=on
```

Blinka Test

ewire is enabled, you may need to use another digital input besides D4.

Create a new file called **blinkatest.py** with **nano** or your favorite text editor and put the following in:

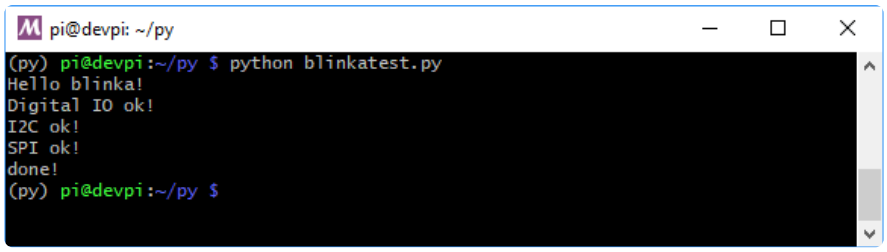
```
import board
import digitalio
import busio
```

```
print("Hello, blinka!")

# Try to create a Digital input
pin = digitalio.DigitalInOut(board.D4)
print("Digital IO ok!")

# Try to create an I2C device
i2c = busio.I2C(board.SCL, board.SDA)
print("I2C ok!")

# Try to create an SPI device
spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)
print("SPI ok!")

print("done!")
```

Save it, make sure your virtual environment is activated, and run at the command line with:

```
python3 blinkatest.py
```

You should see the following, indicating digital i/o, I2C and SPI all worked.



Code the Detector

Install the Required Libraries

You will need to install a few libraries and dependencies to run the Python script. In the terminal, enter:

```
sudo apt install -y python3-opencv
sudo apt install -y opencv-data
pip install piper-tts
python3 -m piper.download_voices en_US-joe-medium
pip install anthropic
```

This installs OpenCV, the Piper text to speech (TTS) local model, the Piper voice used in the project and the Anthropic Python API.

Download the Project Bundle

Once you've finished setting up your Raspberry Pi with Blinka and the library dependencies, you can access the Python code file by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download as a zipped folder.

```
# SPDX-FileCopyrightText: 2025 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT
'''Raspberry Pi Halloween Costume Detector with OpenCV and Claude Vision API'''

#!/usr/bin/python3
import os
import subprocess
import time
import base64
import wave
import cv2
from picamera2 import Picamera2
import anthropic
from piper import PiperVoice, SynthesisConfig
import board
import digitalio

ANTHROPIC_API_KEY = "your-api-key-here"

try:
    username = os.getlogin()
    print(f"The current user is: {username}")
except OSError:
    print("Could not determine the login name.")
    print("Consider checking environment variables like USER or LOGNAME.")
    # Fallback to environment variables if os.getlogin() fails
    username = os.environ.get('USER') or os.environ.get('LOGNAME')
    if username:
        print(f"The user from environment variable is: {username}")
    else:
        print("User information not found in environment variables either.")

# Initialize LED
led = digitalio.DigitalInOut(board.D5)
led.direction = digitalio.Direction.OUTPUT
led.value = False  # Start with LED off

# Initialize detectors
upperbody_detector = cv2.CascadeClassifier(
                     "/usr/share/opencv4/haarcascades/haarcascade_upperbody.xml")
fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows=False)

# Initialize camera
cv2.startWindowThread()
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(
                 main={"format": 'XRGB8888', "size": (640, 480)}))
picam2.start()
```

```
# Initialize Piper voice
voice = PiperVoice.load("/home/pi/en_US-joe-medium.onnx")
syn_config = SynthesisConfig(
    volume=1.0,
    length_scale=1.5,
    noise_scale=1.2,
    noise_w_scale=1.5,
    normalize_audio=False,
)

# Check for and create audio files if they don't exist
def create_audio_file_if_needed(filename, text):
    """Create audio file if it doesn't exist"""
    if not os.path.exists(filename):
        print(f"Creating {filename}...")
        with wave.open(filename, "wb") as wav:
            voice.synthesize_wav(text, wav, syn_config=syn_config)
        print(f"{filename} created!")
    else:
        print(f"{filename} already exists.")

# Create the hello and halloween audio files
create_audio_file_if_needed("hello.wav", "Hello? Who goes there?")
create_audio_file_if_needed("halloween.wav", "Happy Halloween!")

# Initialize Anthropic client
client = anthropic.Anthropic(api_key=ANTHROPIC_API_KEY)
# set API key via the terminal as an environmental variable:
# export ANTHROPIC_API_KEY="your-api-key-here"

# Detection parameters
MOTION_THRESHOLD = 100000
COOLDOWN_SECONDS = 15
last_capture_time = 0
capture_delay = 5
motion_frame_count = 0
MOTION_FRAMES_REQUIRED = 5

def encode_image(image_path):
    """Encode image to base64 for Claude API"""
    with open(image_path, "rb") as image_file:
        return base64.standard_b64encode(image_file.read()).decode("utf-8")

def get_costume_joke(image_path):
    """Send image to Claude and get dad joke about costume"""
    print("Analyzing costume with Claude...")

    # Encode the image
    image_data = encode_image(image_path)

    # Send to Claude API
    message = client.messages.create(
        model="claude-sonnet-4-5-20250929",
        max_tokens=250,
        messages=[
            {
                "role": "user",
                "content": [
                    {
                        "type": "image",
                        "source": {
                            "type": "base64",
                            "media_type": "image/jpeg",
                            "data": image_data,
                        },
                    },
                    {
                        "type": "text",
```

```
                        "text": """Look at this image.
                        Your response must follow these rules exactly:

                        1. If you see a person in a Halloween costume, respond with
ONLY a single-sentence cute,
                        family-friendly, encouraging, dad joke about the costume.
Nothing else.

                        2. If you do NOT see a clear Halloween costume (empty room,
unclear image, person in regular clothes, etc.),
                        respond with ONLY the character: 0

                        Examples of good responses:
                        - "Looks like that ghost costume is really boo-tiful!"
                        - "0"
                        - "I guess you could say that vampire costume really
sucks... in a good way!"

                        Do not add any explanations, commentary, or descriptions.
Just the joke or just 0.
                        """
                    }
                ],
            }
        ],
    )

    # Extract the joke from the response
    joke = message.content[0].text
    print(f"Claude's joke: {joke}")
    return joke
# pylint: disable=subprocess-run-check, broad-except
def play_audio_file(filename):
    """Play a pre-existing audio file"""
    print(f"Playing {filename}...")
    subprocess.run(["su", username, "-c", f"aplay {filename}"])

def speak_joke(joke_text):
    """Convert joke to speech and play it"""
    print("Generating speech...")

    wav_file = "joke.wav"

    # Generate audio with Piper
    with wave.open(wav_file, "wb") as wav:
        voice.synthesize_wav(joke_text, wav, syn_config=syn_config)

    print("Playing audio...")
    # Play the audio file (using aplay for Raspberry Pi)
    subprocess.run(["su", username, "-c", f"aplay {wav_file}"])

    # Optional: clean up the audio file after playing
    # os.remove(wav_file)

# Main loop
while True:
    im = picam2.capture_array()
    grey = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

    # Check for motion
    fgmask = fgbg.apply(im)
    motion_amount = cv2.countNonZero(fgmask)
    motion_detected = motion_amount > MOTION_THRESHOLD

    if motion_detected:
        motion_frame_count += 1
    else:
        motion_frame_count = 0
```

```
    if motion_frame_count >= MOTION_FRAMES_REQUIRED:
        # Detect upperbody
        bodies = upperbody_detector.detectMultiScale(grey, 1.1, 3)
        person_detected = len(bodies) > 0

        # Draw rectangles
        for (x, y, w, h) in bodies:
            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Process if person detected and cooldown passed
        current_time = time.time()
        if person_detected and (current_time - last_capture_time) >
COOLDOWN_SECONDS:
            print("Person detected!")

            # Turn on LED and play hello message
            led.value = True
            play_audio_file("hello.wav")

            im = picam2.capture_array()
            # Capture image
            timestamp = time.strftime("%Y%m%d-%H%M%S")
            file = f"costume_{timestamp}.jpg"
            cv2.imwrite(file, im)
            print(f"\nPicture saved: {file}")

            try:
                # Get joke from Claude
                the_joke = get_costume_joke(file)

                # If Claude returns 0, use the halloween fallback
                if the_joke.strip() == "0":
                    print("No costume detected, playing halloween.wav")
                    play_audio_file("halloween.wav")
                else:
                    # Speak the joke
                    speak_joke(the_joke)

            except Exception as e:
                print(f"Error: {e}")
                # Fallback to halloween.wav if something goes wrong
                play_audio_file("halloween.wav")

            # Turn off LED after processing
            led.value = False

            last_capture_time = current_time
            motion_frame_count = 0

    # Show motion amount
    cv2.putText(im, f"Motion: {motion_amount}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

    cv2.imshow("Camera", im)
    cv2.waitKey(1)
```

Anthropic API Key

You'll need to add your API key that you created on the Claude API Key guide page (https://adafru.it/1auY) to the **code.py** file. At the top of the file, update the `ANTHROPIC_API_KEY` string:

```
ANTHROPIC_API_KEY = "your-api-key-here"
```

Raspberry Pi Halloween Costume Detector
By Liz Clark
Claude API Key

https://learn.adafruit.com/raspberry-pi-halloween-costume-detector/claude-api-key

Run the Script

After downloading the Project Bundle, move the folder to your **/home/user** directory. Then, unzip the folder by **right-clicking** on the folder in the File Manager and selecting **Extract** or with your preferred command line tool. Keep the following file in the **/home/user** directory:

- **code.py**

To run **code.py** from the terminal in your Python virtual environment, you'll use this line in the terminal:

```
sudo -E env PATH=$PATH python code.py
```

How the Code Works

OpenCV is used to detect movement and identify when a person has entered the view of the camera. A preview screen is created to stream the camera feed to the desktop on the Pi.

```
# Initialize detectors
upperbody_detector = cv2.CascadeClassifier(
                     "/usr/share/opencv4/haarcascades/haarcascade_upperbody.xml")
fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows=False)

# Initialize camera
cv2.startWindowThread()
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(
                 main={"format": 'XRGB8888', "size": (640, 480)}))
picam2.start()
```

Piper

The Piper TTS model is used for voice synthesis. The `SynthesisConfig()` function can be adjusted to customize how the voice sounds. You can try out different voices from Piper with this browser tool (https://adafru.it/1auZ).

Two audio files are created before the loop, **hello.wav** and **halloween.wav** since these files will be used multiple times throughout the code. If these files already exist, then new files aren't created.

```
# Initialize Piper voice
voice = PiperVoice.load("/home/pi/en_US-joe-medium.onnx")
syn_config = SynthesisConfig(
    volume=1.0,
    length_scale=1.5,
    noise_scale=1.2,
    noise_w_scale=1.5,
    normalize_audio=False,
)

# Check for and create audio files if they don't exist
def create_audio_file_if_needed(filename, text):
    """Create audio file if it doesn't exist"""
    if not os.path.exists(filename):
        print(f"Creating {filename}...")
        with wave.open(filename, "wb") as wav:
            voice.synthesize_wav(text, wav, syn_config=syn_config)
        print(f"{filename} created!")
    else:
        print(f"{filename} already exists.")

# Create the hello and halloween audio files
create_audio_file_if_needed("hello.wav", "Hello? Who goes there?")
create_audio_file_if_needed("halloween.wav", "Happy Halloween!")
```

Functions

A few functions are used in the loop. The first, `encode_image()` encodes the JPEG image from the camera to base64 to send to the Anthropic API.

```
def encode_image(image_path):
    """Encode image to base64 for Claude API"""
    with open(image_path, "rb") as image_file:
        return base64.standard_b64encode(image_file.read()).decode("utf-8")
```

`get_costume_joke()` sends the image to the API and prompts the API to return a dad joke about the costume. You can tweak the prompt to customize the response from Claude.

```
def get_costume_joke(image_path):
    """Send image to Claude and get dad joke about costume"""
    print("Analyzing costume with Claude...")
```

```
    # Encode the image
    image_data = encode_image(image_path)

    # Send to Claude API
    message = client.messages.create(
        model="claude-sonnet-4-5-20250929",
        max_tokens=250,
        messages=[
            {
                "role": "user",
                "content": [
                    {
                        "type": "image",
                        "source": {
                            "type": "base64",
                            "media_type": "image/jpeg",
                            "data": image_data,
                        },
                    },
                    {
                        "type": "text",
                        "text": """Look at this image.
                        Your response must follow these rules exactly:

                        1. If you see a person in a Halloween costume, respond with
ONLY a single-sentence cute,
                        family-friendly, encouraging, dad joke about the costume.
Nothing else.

                        2. If you do NOT see a clear Halloween costume (empty room,
unclear image, person in regular clothes, etc.),
                        respond with ONLY the character: 0

                        Examples of good responses:
                        - "Looks like that ghost costume is really boo-tiful!"
                        - "0"
                        - "I guess you could say that vampire costume really
sucks... in a good way!"

                        Do not add any explanations, commentary, or descriptions.
Just the joke or just 0.
                        """
                    }
                ],
            }
        ],
    )

    # Extract the joke from the response
    joke = message.content[0].text
    print(f"Claude's joke: {joke}")
    return joke
```

`play_audio_file()` plays audio files, like the pre-made **hello.wav** and **halloween.wav**. `speak_joke()` takes the returned text from Claude, passes it through Piper and then plays the created audio file.

```
def play_audio_file(filename):
    """Play a pre-existing audio file"""
    print(f"Playing {filename}...")
    subprocess.run(["su", username, "-c", f"aplay {filename}"])

def speak_joke(joke_text):
    """Convert joke to speech and play it"""
```

```
    print("Generating speech...")

    wav_file = "joke.wav"

    # Generate audio with Piper
    with wave.open(wav_file, "wb") as wav:
        voice.synthesize_wav(joke_text, wav, syn_config=syn_config)

    print("Playing audio...")
    # Play the audio file (using aplay for Raspberry Pi)
    subprocess.run(["su", username, "-c", f"aplay {wav_file}"])
```

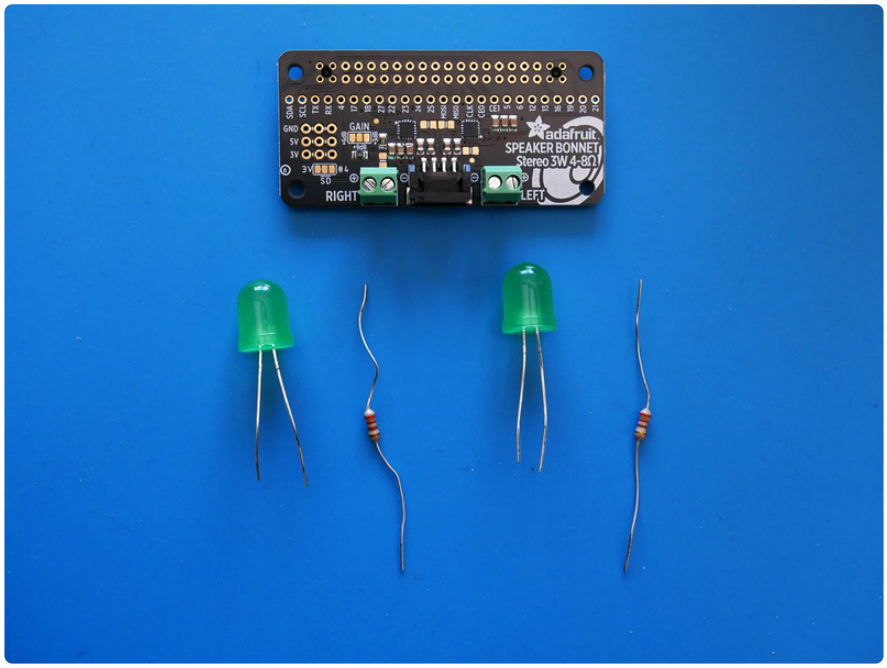
The Loop

In the loop, the camera feed is streamed to the OpenCV window. When motion it detected, the frame is checked for the person detection. If a person is detected, then the LEDs turn on and the **hello.wav** file is played. A picture is taken and its sent to Claude with the costume joke prompt. When the joke is returned, its run through Piper and the joke audio file plays. If for whatever reason no joke is returned (API connection issue, no costume identified, etc), then the **halloween.wav** file is played instead. After this process, the LEDs turn off.

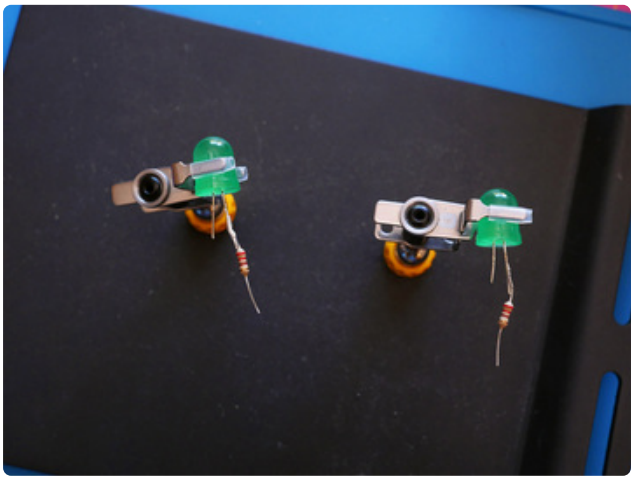
```
while True:
    im = picam2.capture_array()
    grey = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

    # Check for motion
    fgmask = fgbg.apply(im)
    motion_amount = cv2.countNonZero(fgmask)
    motion_detected = motion_amount &gt; MOTION_THRESHOLD

    if motion_detected:
        motion_frame_count += 1
    else:
        motion_frame_count = 0

    if motion_frame_count &gt;= MOTION_FRAMES_REQUIRED:
        # Detect upperbody
        bodies = upperbody_detector.detectMultiScale(grey, 1.1, 3)
        person_detected = len(bodies) &gt; 0

        # Draw rectangles
        for (x, y, w, h) in bodies:
            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Process if person detected and cooldown passed
        current_time = time.time()
        if person_detected and (current_time - last_capture_time) &gt;
COOLDOWN_SECONDS:
            print("Person detected!")

            # Turn on LED and play hello message
            led.value = True
            play_audio_file("hello.wav")

            im = picam2.capture_array()
            # Capture image
            timestamp = time.strftime("%Y%m%d-%H%M%S")
            file = f"costume_{timestamp}.jpg"
            cv2.imwrite(file, im)
            print(f"\nPicture saved: {file}")
```

```
            try:
                # Get joke from Claude
                the_joke = get_costume_joke(file)

                # If Claude returns 0, use the halloween fallback
                if the_joke.strip() == "0":
                    print("No costume detected, playing halloween.wav")
                    play_audio_file("halloween.wav")
                else:
                    # Speak the joke
                    speak_joke(the_joke)

            except Exception as e:
                print(f"Error: {e}")
                # Fallback to halloween.wav if something goes wrong
                play_audio_file("halloween.wav")

            # Turn off LED after processing
            led.value = False

            last_capture_time = current_time
            motion_frame_count = 0

    # Show motion amount
    cv2.putText(im, f"Motion: {motion_amount}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

    cv2.imshow("Camera", im)
    cv2.waitKey(1)
```

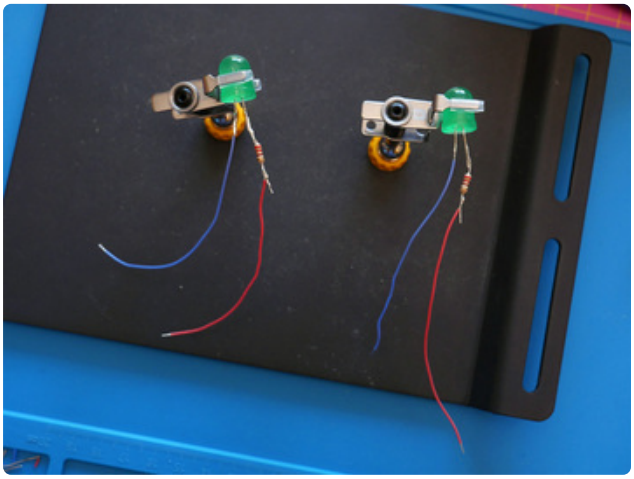
Wiring



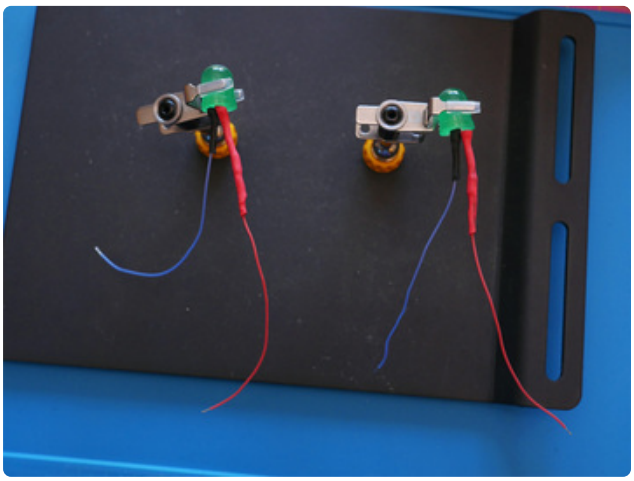
Two LEDs are used as a visual indicator that the skulls is recognizing the goblins and ghouls walking by. You'll connect the LED to the broken out GPIO pins on the speaker bonnet.



Solder a 220Ω resistors to each of the anodes on the LEDs. The anode is the longer leg of the two tegs of the LED.

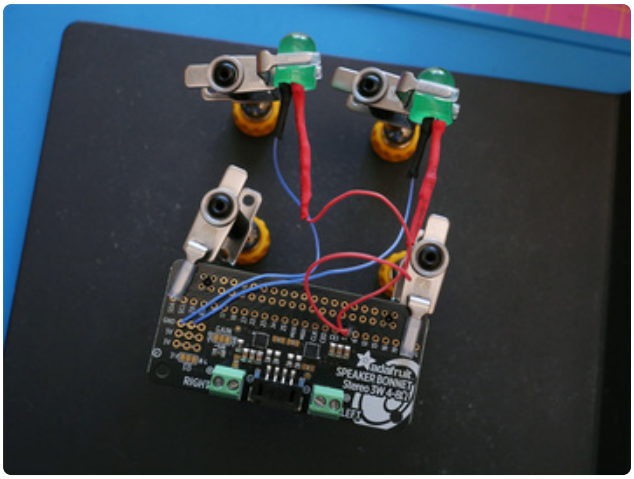


Solder wires to the cathodes of each LED. The cathode is the shorter leg of the LED. Solder wires to the other end of the resistors on each LED.



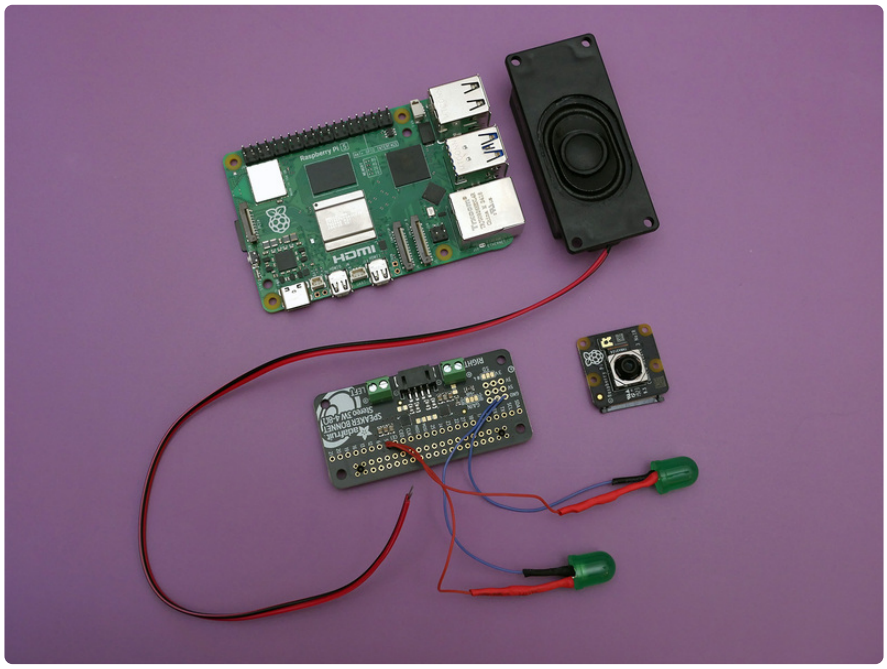
Cover the exposed LED connections with heat shrink.

Gently heat the heat shrink.

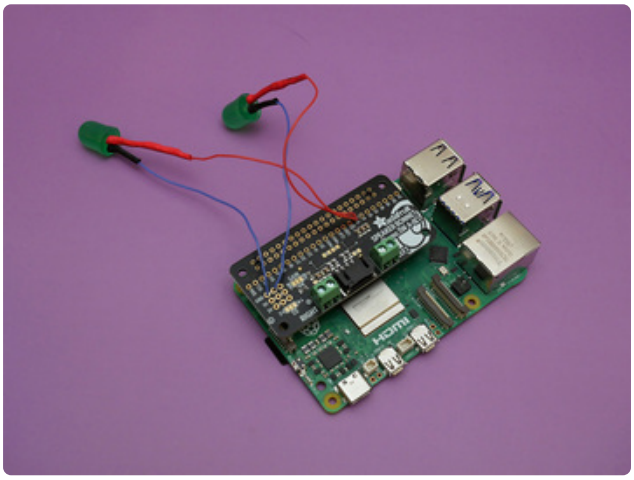


Solder the cathode wires from the LEDs to GND on the speaker bonnet. Solder the anode wires from the LEDs to GPIO 5 on the speaker bonnet.

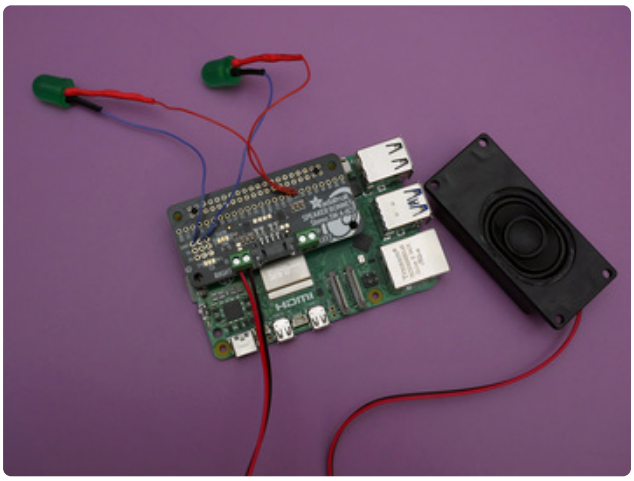
Assembly



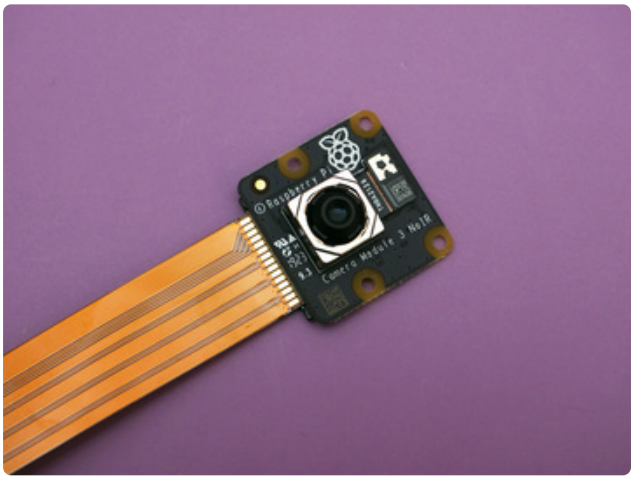
Connect the Hardware



Plug the speaker bonnet into the Raspberry Pi GPIO header.



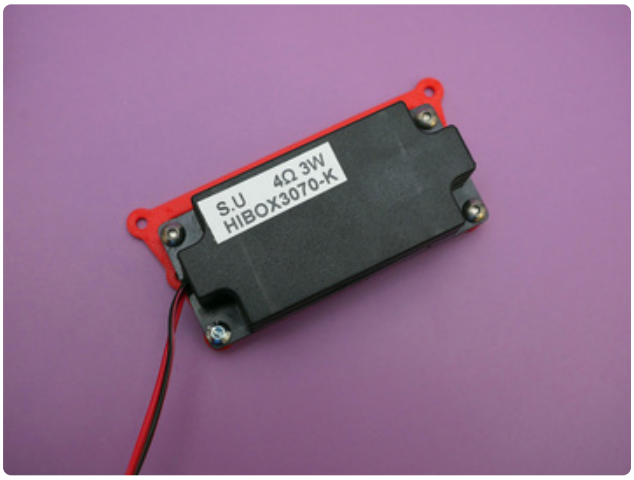
Insert the speaker wires into the right speaker terminal block. Speaker red wire to + and speaker black wire to -.



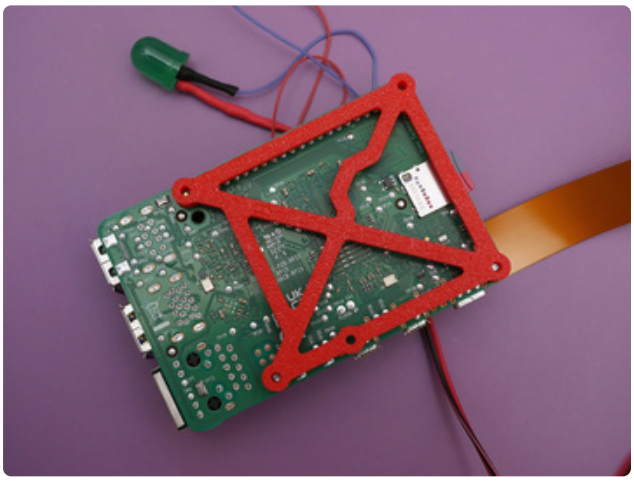
Insert the ribbon cable into the camera. The gold contacts should be facing up.

Insert the other end of the ribbon cable into the first MIPI port on the Raspberry Pi. The contacts should be facing towards the Ethernet port.

Mounting Plates



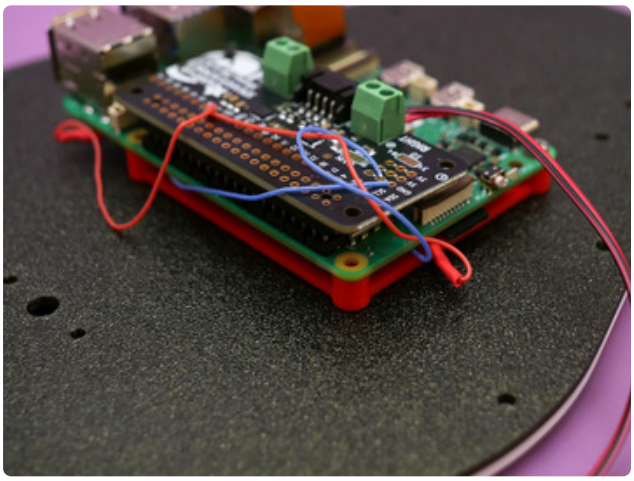
Attach the speaker to the speaker mount with four M3 screws.



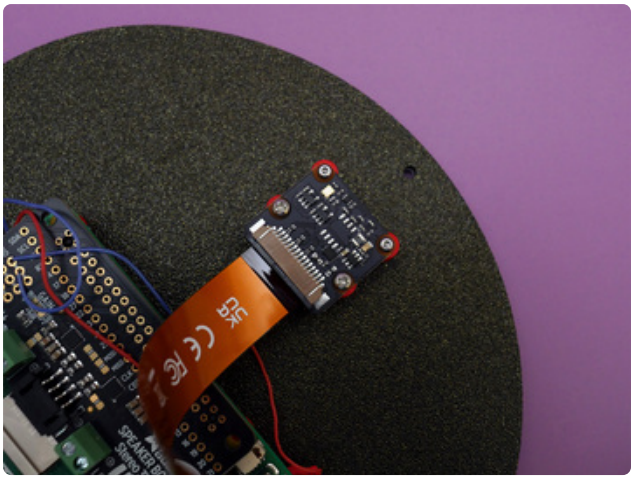
Attach the Raspberry Pi to the Pi mount using two M2.5 screws. Only attach it on the edge with the connectors opposite the GPIO header.

Attach the camera to the camera mount with four M2 screws.

Skull



Attach the Raspberry Pi mount to the skull with three M2.5 screws.



Attach the camera mount to the skull with three M2.5 screws. The camera lens will line up with the cutout in the lightning bolt.

Attach the speaker mount to the skull behind the mouth with two M3 screws.

Align the back plate stand-offs with the mounting holes on the skull. Connect the two together with seven M3 screws.

Use

Make sure you've followed all of the steps in the guide to:

- Get your Claude API key
- Setup the speaker bonnet audio output
- Setup your Python virtual environment
- Install Blinka

If you skip any of these steps the project will not work!

Power up the Raspberry Pi with a 5.1V/3A USB C power supply and run the script. You'll see the video preview appear on the desktop.

To avoid having to use an external display, you can use the Raspberry Pi Connect software (https://adafru.it/1av0) to remotely log into your Pi and start a screen sharing session.

When a person is identified, the LED eyes will light up to let you know that the script is running. After a moment, you'll hear the Claude-generated joke come through the speaker. When the joke is done playing, the LEDs will turn off.

If there are any problems with making a request to the Anthropic API (bad API key, connection error, unable to detect a costume, etc) then the "Happy Halloween" audio file will play instead. You'll be able to see what the actual error was in the terminal window for troubleshooting.

API Reference Links

For more information on the API's used in this project, check out the links below:

https://adafru.it/1auS

https://adafru.it/1auT