### Table of Contents

**Overview**
- Prerequisite Guides
- Parts List
- YOU MUSTN'T CROSS THE STREAMS!

**3D Printing**
- Custom Enclosure
- Filament / Materials
- Support Material
- Tolerances, tweaks and differences
- Glue Halves
- Eye Lens (Not 3D Printed)

**Software**
- Changing Appearance, Wiring and Inputs

**Wiring**
- Connections
- Power
- Displays
- Analog Controls
- Buttons
- Next Steps...

**Assembly**
- Connect switch to Lipo Backpack
- Connect Lipo Backpack to Teensy
- Prep UV LED
- Connect Ribbon Cable to Display
- Connect Display Wires to Teensy
- Connect UV LED to Teensy
- Connect Wires to Pushbutton
- Mount Pushbuttons to Panel
- Connect Pushbutton to Teensy
- Block LED from Teensy
- Block Light from TFT Display
- Trim Battery Wire Short
- Apply Tack to Components
- Install Display to Panel
- Mount Teensy to Panel
- Install Switch to Face
- Mount Battery to Panel
- Mount Lipo Backpack to Panel
- Install Unicorn Horn
- Install Wings and Cabochon
- Install Face to Panel
- Secure Face to Panel
- Test Power Switch / Circuit
- Installing Doorbell
Overview

Here's another take on the electronic animated eye project by Phillip Burgess. Inspired by Sheb Wooley's 1958 hit novelty song, we present our one-eyed, one-horned, flying purple people eater...

![Purple People Eater](image)

Like the bowler hat project, this one uses a single eye, making it more affordable and easier to build than its two-eyed brethren. It can also be installed as a functional doorbell, perfect for Halloween!

Prerequisite Guides

We recommend following the guides below to get familiar with the components used in this project.

- Electronic Animated Eyes using Teensy 3.1/3.2
- Adafruit 1.44" Color TFT with Micro SD Socket
- Adafruit Pro Trinket LiPoly/Lilon Backpack
Parts List

- Adafruit 1.44" TFT Display
- Teensy 3.2 / Teensy 3.1
- Adafruit Lipoly Backpack
- 500mAh Lithium Polymer Battery
- Slide Switch
- 2x 16mm Momentary Pushbutton
- UV Purple 5mm LED
- 1x Acrylic Cabochron
- 4x #4-40 3/8 flat Phillips machine screws

YOU MUSTN'T CROSS THE STREAMS!

This project uses two push buttons, one triggers the eye blinking, and the other rings the doorbell. The wiring should never be crossed/mixed up. Doorbells are typically higher-voltage (10 to 20V) alternating current and will totally blue-smoke the electronics, if triggered by a single button.
3D Printing

Custom Enclosure

The parts in this project were specifically designed for the components listed in the bill of materials. Different micro-controller, display, switch and buttons most likely will not fit. If you’d like to source different parts, the CAD source files are available to download and modify.

Filament / Materials

The parts can be printed in different types of materials. For best effects, we used Ninajflex (TPE filament) for the wings, glow-in-the-dark PLA for the unicorn horn, and opaque purple PLA for the face pieces. You can however, use any type/color and paint them if you wish.

Download STL Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>face-bot.stl</td>
<td>Face bottom part</td>
<td>Purple PLA</td>
</tr>
<tr>
<td>face-top.stl</td>
<td>Face top part</td>
<td>Purple PLA</td>
</tr>
<tr>
<td>mplate.stl</td>
<td>Plate for mounting to wall</td>
<td>PLA</td>
</tr>
</tbody>
</table>
Support Material

Only two parts, face-bot and face-top will require support material. All of the parts are oriented in the optimal position printing. Depending on your slicing software, you will have to choose between automatically generated supports or custom supports if your using software such as Simplify 3D. For the most part, automatically generated supports will be suffice.

Tolerances, tweaks and differences

If you’re planning to use this project to replace your doorbell, you should check how your household doorbell is mounted. This may be different than the one modeled for this project. The 3D source files are available to modify and tweak if you’d like to fit it to your doorbell.
Glue Halves

Use E6000 adhesives to permanently bond the two face pieces together. A toothpick can help spread the glue across the surface. Be sure to allow the parts to dry for several hours before mounting the components to them.

Eye Lens (Not 3D Printed)

1.5 inch (38mm) cabochons (domes) magnify the screens slightly and give the eyes a cool 3D shape. I found mine at Tap Plastics, but any good plastics supplier should
Software

Before diving too deep into the software, there are some gotchas to be aware of...

- Do not install the Adafruit_GFX, Adafruit_SSD1351 or Adafruit_ST7735 libraries offered by the Teensyduino installer! Use the Arduino Library Manager or install these manually from Github code. The Teensyduino-installed libraries sometimes diverge from the latest Adafruit code and might prevent this project from compiling.
- When first building this project, please test initially with the canonical “Uncanny Eyes” sketch linked later in this guide, not anyone’s derivative code. This will help with any troubleshooting/support. Once the default code works, then you can try out variants that may be out there.

Teensy uses the Arduino environment for programming, so it’s pretty familiar and simple to work with, but it does require a little extra setup first...

If you’re not using a recent version of the Arduino IDE (1.6.5 or newer), this would be a good time to upgrade. Once you have that software installed and working, download and run the Teensyduino installer, which adds support for the full line of Teensy microcontroller boards in the Arduino IDE (but remember, don’t install the display/graphics libraries there...use the Arduino Library Manager instead).

From the Tools menu, select Board→Teensy 3.2 and CPU Speed→72 MHz (and whatever optimization settings you’d like). Confirm that you can compile and upload the classic “blink” sketch to the Teensy board. Don’t use the 96 MHz setting; the code actually performs a bit better at 72 MHz (due to SPI settings).

Do not continue until you have the Blink sketch working on the Teensy board.

Using the Arduino Library Manager (Sketch→Include Library→Library Manager...) install Adafruit_GFX, Adafruit_BusIO and Adafruit_ZeroDMA, plus the library compatible with your display: Adafruit_SSD1351 for the OLED display, Adafruit_ST7735 for TFT LCD.
(If you’re still using an oldschool version of the Arduino IDE, these libraries can be fetched from Github: Adafruit_GFX(), Adafruit_BusIO(), Adafruit_SSD1351(), Adafruit_ST7735() and installed manually.)

Finally, there’s the sketch code itself:

Download “Uncanny Eyes” Arduino code

The sketch is utterly ginormous. In addition to several hundred lines of code in the main sketch, arrays containing graphics take up most of the space in the microcontroller’s flash program space.

Before uploading to the board, check lines 56 and 57 in the file config.h:

```cpp
#include <Adafruit_SSD1351.h>  // OLED display library -OR-
#include <Adafruit_ST7735.h> // TFT display library (enable one only)
```

One line is enabled, the other is commented out. By default, OLED is used. Comment out the opposite line if using TFTs.

If using all the same wiring as the previous page, it should be possible to compile and upload to the board and see some results... you should at least see an eye doing something. (If you changed the wiring, skip ahead to the next section below to make the code match, then return here.)

The sketch doesn’t compile!

Either Teensy support has not been correctly installed with the Teensyduino installer, or one or more of the libraries is not installed (Adafruit_GFX, Adafruit_SSD1351 or Adafruit_ST7735). Please see the notes at the top of this page regarding IDE & library compatibility.

The code compiles and uploads but nothing happens!

Check the connections between the display and Teensy board.

- Did you enable the correct #include line for the display type (OLED vs TFT)?
- Are you following the correct order-of-wires for the display type (OLED vs TFT)?
- Are any wires off-by-one on the Teensy?
- Any cold solder joints, or solder bridges between pads?
It kinda works, but the display is glitchy!

- Keep your wires as short and as tidy as possible, check solder connections for good form. High speed SPI is really persnickety about connections.
- Is the right CPU speed selected?

Still having trouble? Start a new thread in the [Adafruit forums](https://www.adafruit.com/forum/) describing the symptoms. It's extremely helpful if you can provide in-focus and well-lit photos that clearly show all the connections between the display and Teensy.

Do not continue until you see an eye. If it's not doing exactly what you want, that's okay, just need an eye to start.

### Changing Appearance, Wiring and Inputs

Basic customization — enabling or disabling certain features, or changing pins assigned to functions — is done by editing the file `config.h` (it's the second tab when the code is open in the Arduino IDE).

First section of this file controls the appearance of things.

The software includes a few ready-made eyes that can be used by simply enabling the corresponding `#include` line — remove the comment (`//`) at the start of that line, and make sure all the others are commented out. The stock sketch has “defaultEye” enabled:

![Uncanny Eyes - config.h](attachment:image.png)
defaultEye.h is human-ish in design. Okay, so the iris is anime-sized, but I’m so proud of that iris-scaling code I had to show it off as much as possible.

Some animals have such huge irises you rarely see the sclera (the white part of the eye). noSclera.h is an example eye for these situations.

dragonEye.h because dragons. It's a moral imperative.
Goats (or is it Krampus?) have the weirdest pupils. goatEye.h is an attempt at simulating this. I designed this one to not move around, just to demonstrate how it's done (the sclera image is the same size as the screen...no room to move...normally the sclera image is a bit larger).

Just above the eye selection is this line, disabled by default:

```c
#ifndef SYMMETRICAL_EYELID
#define SYMMETRICAL_EYELID
#endif
```

Normally the software renders the two eyes with distinct left and right shapes (there's a caruncle — that corner area near the tear duct). Some projects only use a single eye and this shape just seems odd. Enabling SYMMETRICAL_EYELID uses a simpler “football shape” that's neither left nor right...it's a bit cartoonish but may look better on single-eyed creatures.

A little further down is a list indicating the pins used for each eye. The software can handle any number of eyes (it splits up time between them), but will usually be two (or, if compiling this for the Hallowing board, there's just one):

```c
eyeInfo_t eyeInfo[] = {
  #ifdef ADAFRUIT_HALLOWING
    { 39, -1, 2 }, // SINGLE EYE display-select and wink pins, rotate 180
  #else
    {  9, 0, 0 }, // LEFT EYE display-select and wink pins, no rotation
      { 10, 2, 0 }, // RIGHT EYE display-select and wink pins, no rotation
  #endif
};
```

Each line contains three items: the pin that's connected to the corresponding display's OC (OLED chip select) or TCS (TFT chip select) pin, and another pin where a button (connected to ground) makes that eye independently wink. If you don't want or need this wink control, the second pin value can be set to -1. Third item on each line is a rotation setting (0–3) for that screen:

- 0 is the default orientation, with the display's connector at the top.
- 1 rotates the graphics 90 degrees clockwise (compensating for a display that's installed 90° counterclockwise, with the connector at the left).
- 2 rotates the display 180° (as on the Hallowing board).
• 3 rotates 90 degrees counterclockwise (compensating for a display with the connector at right).

The stock code, as shown above, has two eyes. To work with just a single eye, comment out or delete one of the two lines. In the Hallowing case (if ADAFRUIT_HALLOWING is defined), this is already done (notice also the rotation value of “2,” because the Hallowing screen is oriented upside-down).

As previously explained, the next section decides whether to use the library for OLED or TFT displays, by #including one or the other:

```cpp
#include &lt;Adafruit_SSD1351.h&gt;  // OLED display library -OR-
#include &lt;Adafruit_ST7735.h;&gt;  // TFT display library (enable one only)
```

After this, two lines indicate the pins used for the displays’ DC (data/command) and RESET signals:

```cpp
#define DISPLAY_DC     7       // Data/command pin for ALL displays
#define DISPLAY_RESET  8       // Reset pin for ALL displays
```

If the project involves multiple eyes, these lines should fan out from the microcontroller to ALL displays; unlike the select pins above, which are unique to each display.

The Hallowing board has a different set of pins already defined, plus some extra items for backlight control.

The last section configures various controls:

```cpp
#ifndef ADAFRUIT_HALLOWING
#define LIGHT_PIN      A1 // Hallowing light sensor pin
#define LIGHT_CURVE   0.33 // Light sensor adjustment curve
#define LIGHT_MIN      30 // Minimum useful reading from light sensor
#define LIGHT_MAX     980 // Maximum useful reading from sensor
#else
#define LIGHT_PIN      A2 // Photocell or potentiometer (else auto iris)
#define LIGHT_PIN_FLIP  // If defined, reverse reading from dial/photocell
#define LIGHT_MIN      0 // Lower reading from sensor
#define LIGHT_MAX    1023 // Upper reading from sensor
#endif
```

```cpp
#define IRIS_SMOOTH         // If enabled, filter input from IRIS_PIN
#if !defined(IRIS_MIN)      // Each eye might have its own MIN/MAX
#define IRIS_MIN      120 // Iris size (0-1023) in brightest light
#endif
```
#define IRIS_MAX 720 // Iris size (0-1023) in darkest light

JOYSTICK_X_PIN and JOYSTICK_Y_PIN (here set to A0 and A1, respectively) state where the joystick inputs are connected. By default, these lines are commented out — the eye moves autonomously, without user input. If you have a joystick connected, enable these two lines. (X_FLIP and Y_FLIP reverse the input direction if needed)

TRACKING sets whether the upper eyelid follows the pupil (as actual eyes do, it’s a neat thing). You can turn this off by commenting out this line.

LIGHT_PIN states where the photocell or dial is connected for adjusting the size of the pupil/iris. LIGHT_PIN_FLIP reverses the direction, like the joystick settings (lower values should be darker). LIGHT_MIN and LIGHT_MAX are the minimum and maximum reliable readings from the sensor (it might occasionally go below or above these, but it's rare, probably noise that can be disregarded), while IRIS_MIN and IRIS_MAX control the dilation of the pupil in response to the light reading (smaller numbers = smaller pupil, range is 0 to 1023).

IRIS_SMOOTH filters the input from LIGHT_PIN so it's not twitchy. This slows the reaction time, but the movement is similar to real eyes, pretty nifty.

BLINK_PIN specifies a pin where a button is connected for blinking both eyes simultaneously (distinct from the individual left/right wink pins previously discussed). If you don’t have a button connected for this, the line can be commented out or the value set to -1.

AUTOBLINK (enabled by default) makes the eyes automatically blink randomly every few seconds. You can comment this out to make the eyes only blink with the buttons... or you can use both in combination.

If the example eyes don’t deliver quite what you need, it’s possible to generate new header files with custom graphics...
Wiring

Before diving in, give some thought to how you'll be using this.

At the very least, this project uses a Teensy microcontroller and a display or two. The animation effects are all capable of running autonomously if need be, or there are options to have them manually controlled. The method of assembly is also open to interpretation.

If you’re making a spooky prop to sit in the window on Halloween, but plan to disassemble it afterward and use the parts in other projects, you can use a breadboard and jumper wires for quick assembly and re-use.

For something portable, like jewelry or a costume piece, soldering wires directly between components is vital — both for space savings and for durability.

If creating eyes for a puppet, you probably want manual controls for nearly everything, as that’s the very nature of puppetry.

For a costume, I think autonomous works better. Good cosplay is all body language… but when electronics are added, fantastic characters are spoiled when the performer is focused on modes and buttons. But hey, it’s up to you.

Give it some thought. I’ll wait!
Half and half: these eyes are neatly assembled in 3D-printed enclosures while the rest of the circuit is a messy breadboard. Once everything’s tested and working, the breadboard side will be replaced with a more permanent solution and fitted inside a Halloween prop. Whatever works for your needs!

Connections

Many configurations of this project are possible, depending on the features you’re after. Rather than a single complex wiring diagram, a few subassemblies are illustrated here...pick and choose to match your needs.

What’s shown here are schematic diagrams — they indicate where to connect wires, but not necessarily their exact actual layout or lengths when you build the thing. You’ll need to think how everything fits in your own setting. Using color-coded wires helps a lot here!

We’ll be referring to several pins by name or number, so here’s a pin map for the Teensy 3.1 or 3.2 microcontroller:
This is a simplified pinout to clarify things specifically for this guide. If you plan to add your own advanced bells and whistles, a more complete pinout map is available on the PJRC web site.

Ground connections are vital for distributing power throughout the circuit. In addition to the two GND pins labeled here, there are two copper pads on the back of the board — one near the center, and a second large one near the USB port (next to the GND pin).

The AGND pin provides a cleaner ground reference specifically for analog inputs — this is not a current-carrying pin for distributing power, do not connect the displays here.

Some of the pin numbers we’ll be referencing are negotiable...if you find that a different pin would make routing wires easier, there’s usually a setting in the code that can be made for it. Anything related to power or SPI is not negotiable...those wires must go to the pins stated.

**Power**

If you have an opportunity to power everything from the Teensy’s USB port (running a USB cable to a power bank or wall charger), that’s easiest and reduces parts and steps.

The most compact, portable installations may optionally want a built-in Lithium-Polymer (LiPoly) battery. If your project uses one or two TFT LCDs, a 150 mAh battery may suffice (though a larger capacity will provide a longer run time). For one or two OLEDs, a 500 mAh battery is the minimum size.
To use the Teensy board with the Adafruit LiPoly Backpack (allowing USB charging), first two copper traces need to be cut: between the two pads next to the Teensy’s VUSB pin, and between the switch pads on the LiPoly Backpack (marked on back). Then add these three wires between the boards:

- LiPoly BAT to Teensy VIN/BAT+ (unmarked pin at corner)
- LiPoly G to Teensy GND
- LiPoly 5V to Teensy USB+ pin

Add power switch to pins on LiPoly backpack. These tactile on/off switches (http://adafru.it/1092) are my favorite!

If battery capacity is 500 mAh or larger, solder between the charge rate jumpers on the back of LiPoly backpack. Do not do this with small batteries!

When you cut the trace on the Teensy board, it won’t run from USB power until the LiPoly backpack is connected and switched on. This is normal. If a sketch won’t upload on a half-built project, this may be the reason why.
Power the display(s) from the BAT+ (corner) pin. If you're not using the LiPoly backpack (powering off a USB cable instead), that's okay — by default this pin also connects to USB+.

Displays

You have a choice of using one or two displays, either OLED or LCD. But you can't mix one of each — both must be the same type.

OLED displays have brighter colors and contrast. Downside is the price, and also that they flicker a little when captured on video. They also have a finite lifespan, albeit many thousands of hours.

TFT LCD displays are more affordable. Not as bright, but still a good effect. And they're rock-steady on video.

If making two eyes, both displays need to connect to the same SPI MOSI and CLK pins on the Teensy, plus a few other wires. The “OC” or “TCS” pins are unique to each display, left or right.
For OLED displays, make the following connections from the display breakout board to the Teensy:

- SI to SPI MOSI
- CL to SPI CLK
- DC to Digital Pin 7
- R to Digital Pin 8
- OC to Digital Pin 9 (left eye) or 10 (right eye)
- + to BAT+ (if using LiPoly Backpack) or USB+
- G to GND
- The remaining four pins are not connected.

For TFT LCD displays, use these connections:

- Vin to BAT+ (if LiPoly) or USB+
- Gnd to GND
- SCK to SPI CLK
- SI to SPI MOSI
- TCS to Digital Pin 9 (left eye) or 10 (right)
- RST to Digital Pin 8
- D/C to Digital Pin 7

“Left” and “right” eye in this case refer to the positions when looking at the eyes, not from their point of view. This nomenclature is used throughout this guide and in the software.

If you’re using them, the 3D-printed enclosures have a small notch that’s just wide enough for a 7-conductor ribbon cable to fit through. Space inside is really tight (I wanted them to fit inside a mask), so it’s necessary to route these wires to their pins very carefully so they lie as flat as possible on the back of the board, not all piled up. It’s delicate work that requires tweezers and patience.
If using a ribbon cable as shown above, write down your own legend that maps wire colors (or wire numbers if single-colored cable) to pin functions.

Because the cable colors are in a fixed order and there’s little space to reroute inside the case, certain wiring conventions used in electronics (such as using red wire for positive voltage and black for ground) no longer apply…you’re forced to take what you’re given. What’s more, with wires doubled back, the conductors along the ribbon don’t necessarily match the order along the display breakout header, it’s all jumbled now.

Do not use our photos for reference. Do not rely on the colors shown in any diagrams here. Write down the exact sequence for your cable and your routing, and use only that for reference, nothing else.

Write down your own wiring legend. Do not rely on the colors used in this guide.

The wiring for the TFT was even more “creative,” with some wires a straight shot and others doubled back. Again, notice the wiring legend. Do what works for you.

If you’re not using the 3D-printed enclosures, everything is much simpler. You can just connect to the pins in-order, there’s no space constraint that must be met.
Wanting to test with a breadboard first, I made the ribbon cables extra long (about 8 inches) and soldered row pin headers on the end.

Later, for permanent installation, I'll cut the ribbon cables down to size and solder wires directly to the Teensy board.

This is totally optional...maybe you only want to build it once. Just try to keep the wire lengths to a minimum...high-speed SPI can be very finicky about this. Even 8 inches is pushing it.

The displays arrive with a plastic sheet on them. Keep this in place when soldering, but remove it once you’re done. This isn’t like a phone screen protector, just for shipping and soldering protection.

The display breakout boards include microSD slots, but those are not wired up in this project and the code doesn’t reference them at all. Advanced users who want this capability can connect the SPI MISO and card select pins (you’ll need to modify the enclosure slightly to accommodate the extra wires) and make the required changes in the code.

## Analog Controls

Any analog controls that are used should include connections to the 3.3V and AGND pins. Don’t use the other power pins or there will be...trouble.

XOUT and YOUT from a joystick can connect to Analog Pins A0 and A1.

The eyes move autonomously by default — settings in the code enable the joystick instead.

If you need to mount the joystick in a different orientation, there are also settings to invert each axis. Swap the X and Y pins in the code to use the joystick sideways.
To have the pupils contract or expand in response to light, connect a photocell and 10K resistor in series. The midpoint connects to Analog Pin A2.

Analog input for the pupils (either photocell or the dial below) are enabled in the code by default. You can comment out IRIS_PN in the code to have this move autonomously.

For manual control of pupil dilation (instead of responding to light) a 10K potentiometer can be used. The center leg connects to Analog Pin A2 (same input as the photocell, just substituting a different analog control).

Buttons

The eyes normally blink autonomously, but you can also add one or more buttons to make them blink (or even wink individually) on command.

For all buttons, connect one leg of each to GND, and the opposite leg to a digital pin:

Digital Pin 0 is the left eye wink.
Digital Pin 1 blinks both eyes.
Digital Pin 2 winks the right eye.
If using our analog joystick breakout board, that stick includes a clicky button when you press down on it (on the SEL pin). This can optionally be used for manual blink control, or you can use a separate button for this (I find the joystick button a bit hamfisted).
Pro tip: a fully-equipped version of the circuit requires several ground connections, but there are only a few GND pins on the Teensy board. Two- or three-way splices are one option, but another way to provide extra ground connections is to repurpose unused I/O pins. Let's suppose you want to make Digital Pin 4 a spare ground pin. Add the following code in the setup() function:

```
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
```

OUTPUT LOW makes the pin function as an ersatz ground connection. This works perfectly for button connections like the blink controls, but don’t use it for heavy loads (like powering the displays) — it won’t work and might even damage the Teensy.

Next Steps...

After wiring everything up, proceed to the next page and test out the software.

Once it’s all tested and working...if using the 3D-printed enclosures, return to that page to complete the assembly. Otherwise, you’re on your own now, to install the electronics in your own finished design.
Assembly

Connect switch to Lipo Backpack

Let’s start off by setting up the power circuit. Measure and cut two 26AWG wires. Strip and tin the tips of each wire. Trim the leads from the slide switch short and solder the two wires to them. Apply two pieces of heat shrink tubing to insulate the exposed leads. Cut the traces with the label pwr switch on the Lipo backpack. Connect the two wires from the slide switch to the power switch pins on the Lipo backpack.
Connect Lipo Backpack to Teensy

Cut the trace on the pads of the Teensy noted in the wiring diagram. Measure and cut three 26AWG wires. Strip and tin the tips of each wire. Connect the following pins together.

5V from lipo backpack to VUSB on Teensy.
GND from lipo backpack to GND on Teensy.
BAT from lipo backpack to BAT+ on Teensy.
Prep UV LED

Trim the leads short and apply solder to tin them. Measure and cut two 30AWG wires. Strip and tin the ends of each wire. Solder wires to shortened leads. Add heat shrink tubing to insulate the exposed leads.
Connect Ribbon Cable to Display

Grab a set of jumper cables. Remove the connectors using wire cutters. Peel, strip and tin the ends of each wire. Tin the pins on the TFT display and solder the wires to them. Write the colors to each pin for reference.

Connect Display Wires to Teensy

Follow the wiring diagram and tin the pins on the Teensy. Solder the wires from the display to the Teensy using the wiring diagram and your color reference notes.
Connect UV LED to Teensy

Tin wires to 3.3V and GND pins on the Teensy board. Solder the positive wire from the UV LED to 3.3V and the negative wire to GND on the Teensy.

Connect Wires to Pushbutton

Measure and cut two 30AWG wires. Strip and tin the ends of each wire and solder them to the leads on the pushbutton.
Mount Pushbuttons to Panel

Unscrew the hex ring from the wired pushbutton. Insert the wires through the hole on the panel. Press the pushbutton into the hole and apply force to push it all the way through. Flip the panel over and thread wires through hex ring. Screw the ring tightly on to the pushbutton and secure it in place. Remove ring from second pushbutton and insert into the other hole. Thread the wires from pushbutton through the smaller hole.
Connect Pushbutton to Teensy

Secure the Teensy to a pair of helping third hands. Solder the wires from the pushbutton to #0 (wink left) and a ground pin on the Teensy.

Block LED from Teensy

To remove light leaking from the inside of the enclosure, paint the LED black using black acrylic paint using a tooth pick.
Block Light from TFT Display

Use a piece of black electrical tape to cover the bottom of the TFT display. This will help block any light leaking from side the enclosure.

Block light leakage from the components is only necessary if the enclosure is printed in opaque/transparent filament.
Trim Battery Wire Short

We’ll need to trim the cable short from the battery to allow space in the enclosure. Cut the red positive wire, then the black negative wire from the battery. Strip and tin the ends of each wire. Apply pieces of heat shrink tubing, then solder the wires back together.

Ensure they’re cut one at a time to avoid shorting the battery. The positive and negative wires mustn’t touch, ever!
Apply Tack to Components

To mount the Teensy and Lipo backpack PCB’s to the panel, we can use tack. Using tack allow you to shift the components around for optimal placement. You can optionally use epoxy or E6000 adhesives to permanently secure the boards to the panel.

Install Display to Panel

Fit the wires from the TFT display in between the bottom left walls (Peel apart the wires if needed). Lay the display over the two pushbuttons with the pins facing towards the top of the panel.
Mount Teensy to Panel

Place the Teensy PCB below the display, underneath the bottom wall. Press it down to secure the tack to the panel.

Install Switch to Face

Insert the slide switch through the hole near the bottom of the face piece (chin). The tolerances should be pretty tight to hold the switch in place, but you can optionally apply adhesives to permanently secure it.
Mount Battery to Panel
Add a piece of tack to the lithium polymer battery. Place it behind the slide switch and press it down to secure it onto the panel.

Mount Lipo Backpack to Panel
Stick the Lipo Backpack below the Teensy board. Press it down to secure it to the panel.
Install Unicorn Horn

Insert the unicorn horn through the opening near the top of the face piece. Apply force to push the unicorn through the opening. The tolerance should be tight, but if you find it loose, you can apply adhesive to secure it in place. Insert the LED through the hole on the bottom of the horn - If it’s too loose, add some tack/glue to hold it in place.
Install Wings and Cabochon
Place the wings over the sides of the panel. Line up the holes on the wings with the clips on the panel. Press them down to hold them in place. Lay the cabochon over the display.

Install Face to Panel
With the wings and cabochon installed, carefully lay the face part over the panel. Line up the lip of the panel with the inside of the face part and press them together - it should snap into place.
Secure Face to Panel

While holding the parts together, insert #4-40 3/8 size flat Phillips machine screws into the mounting holes on the back of the panel. Tightly hold the two parts together while fastening the screws all the way through until they’re flush with the surface.

Ensure wires are nice, tidy and completely enclosed inside the enclosure. Avoid kinking or clipping the wires when joining the halves together.

Make sure all of the wires are kept inside the enclosure and out of the way from the mounting holes!
Test Power Switch / Circuit

If everything fits nicely, flip the switch to power on the circuit. Slowly push the cabochon inward to actuate the pushbuttons. If everything is neatly in place, the eye should wink!

If the screen turns white when pressing the eye, some of the wires are probably shorting out. Take it apart and diagnose the wiring. Ensure the wires are neatly nested behind the display.

Installing Doorbell

Now it's time to install the doorbell. If you haven't already, you should poke around your existing doorbell and see how it's being mounted. If it's using the same mounting plate like in this project, congratulations! You've won the maker lottery.

Most doorbells probably aren't standardized, therefore they're all going to have different mounting holes.

Once you've figured out how to mount the plate, it's relatively easy to stick the enclosure to it - I used double-sided foam tape. The mounting plate has a large opening in the center to allow the two pushbuttons to fit into. The mounting plate itself is secured to a metal bracket thats fixed to the wall using two screws.

My original doorbell was a slightly larger, so you can see how the non-painted areas show through. There's only two wires that need to connect to the second pushbutton. Since the pushbutton doesn't have specific polarities, it didn't matter which lead is positive or negative. Just be careful that those wire DO NOT touch the wires from...
the pushbutton that triggers the eye wink. NEVER connect the doorbell wires to the same pushbutton - that would totally cook the Teensy board.