## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Code</td>
<td>3</td>
</tr>
<tr>
<td>Wiring Diagram</td>
<td>8</td>
</tr>
<tr>
<td>Electronics Assembly</td>
<td>9</td>
</tr>
<tr>
<td>Sensor Calibration</td>
<td>14</td>
</tr>
<tr>
<td>Bottle Assembly</td>
<td>17</td>
</tr>
<tr>
<td>Use It</td>
<td>22</td>
</tr>
</tbody>
</table>
Introduction

Explore the magic of capacitive touch with this magical potion bottle. Place the key in the lock to switch it on, then touch the magical locks on the bottle to make it change colors. Touch the locks in the right combination and the potion will swirl in beautiful rainbow colors.

Professor Snape might actually give you a passing grade for this project, but you’d better hand it in on time.

Ingredients

- Teensy v 3.2 (http://adafru.it/2756)
- 5 to 8 neopixels (I used 30/m in white)
- 30awg silicone stranded wire (http://adafru.it/2051)
- Lithium Polymer 100mAh battery (http://adafru.it/1570)
- Female JST connector (http://adafru.it/1769)
- 3 x 1M resistors
- All metal luggage lock
- 3 metal scrapbooking locks or charms
- Small glass bottle
- Fantasy film
- Leather scraps

Tools

- Heat gun
- Soldering iron & accessories
- Hot glue gun
- Sharp scissors

Code

It’s a great idea to get your software all set up and loaded onto your Teensy right away, to make testing your connections easier later on.

To get the code running on the Teensy you’ll need:

1. Arduino IDE (1.6.5 or newer preferred)
2. Teensyduino Installer
3. FastLED Library

It's also helpful to install the Adafruit NeoPixel library, for testing purposes or in case FastLED is not working. (Sketch→Include Library→Manage Libraries...).

1. Arduino IDE

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.

2. Teensyduino Installer

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.

In the Arduino IDE, from the Tools menu, select Board→Teensy 3.2 and CPU Speed→72 MHz (Optimized). Confirm that you can compile and upload the classic “blink” sketch to the Teensy board. Be sure you can blink the light before continuing.

3. FastLED Library

Use the Library Manager in the Arduino IDE to install this (Sketch→Include Library→Manage Libraries...), or if you’re using an older version of the Arduino IDE, it can be downloaded and installed manually.

Once you've got everything installed, restart Arduino and select Teensy 3.1/3.2 from the Tools > Boards dropdown menu and upload the code.

```c
#include &lt;CapacitiveSensor.h&gt;
#include &lt;FastLED.h&gt;
#include &lt;elapsedMillis.h&gt;

#define DATA_PIN    23
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
#define NUM_LEDS    5        // Change this to reflect the number of LEDs you have

int HUE = 0;
int SATURATION = 255;
int BRIGHTNESS = 255;

int indexinc = 1;
elapsedMillis timeElapsed; //declare global if you don't want it reset every time loop runs

unsigned int interval = 2000;

CRGB leds[NUM_LEDS];
CRGBPalette16 gCurrentPalette;
```
CRGBPalette16 gTargetPalette;
CRGBPalette16 currentPalette;
TBlendType currentBlending;

long previousMillis = 0;

/************************************************************************/
* CapitiveSense Library Demo Sketch
* Paul Badger 2008
* Uses a high value resistor e.g. 10M between send pin and receive pin
* Resistor effects sensitivity, experiment with values, 50K - 50M. Larger resistor
* values yield larger sensor values.
* Receive pin is the sensor pin - try different amounts of foil/metal on this pin
* Modified by Becky Stern 2013 to change the color of one RGB Neo Pixel based on
* touch input
* */

CapacitiveSensor cs_4_1 = CapacitiveSensor(4,1);
CapacitiveSensor cs_4_2 = CapacitiveSensor(4,2);
CapacitiveSensor cs_4_3 = CapacitiveSensor(4,3);

void setup()
{
  delay(3000); // 3 second delay for recovery
  // tell FastLED about the LED strip configuration
  FastLED.addLeds<LED_TYPE,DATA_PIN,COLOR_ORDER>(leds, NUM_LEDS)
    .setCorrection(TypicalLEDStrip)
    .setDither(BRIGHTNESS &lt; 255);

  // set master brightness control
  FastLED.setBrightness(BRIGHTNESS);
  currentPalette = RainbowStripeColors_p;
  currentBlending = LINEARBLEND;
  Serial.begin(9600);
}

int mode=0;

void loop()
{
  elapsedMillis timeElapsed;
  if (timeElapsed &gt; interval)
  {
    mode=0;
  }

  checkpress();

  switch (mode){
    case 0: BRIGHTNESS=175; fairylights();break;
    case 1: BRIGHTNESS=255; Red(); break;
    case 2: BRIGHTNESS=255; Purple(); break;
    case 3: BRIGHTNESS=255; Blue(); break;
    case 4: BRIGHTNESS=255; Rainbow(); break;
    case 5: BRIGHTNESS=255; Green(); break;
    case 6: BRIGHTNESS=255; Cyan(); break;
    case 7: BRIGHTNESS=255; White(); break;
  }
}

void checkpress()
{
  long start = millis();
}
long total1 = cs_4_1.capacitiveSensor(30);
long total2 = cs_4_2.capacitiveSensor(30);
long total3 = cs_4_3.capacitiveSensor(30);

if (total1 &gt; 3000 &amp;&amp; total2 &gt; 3000 &amp;&amp; total3 &gt; 3000){
    mode=4; //rainbow
    Serial.print("4");
} else if (total1 &gt; 3000 &amp;&amp; total2 &gt; 3000){
    mode=5; //green
    Serial.print("5");
}
else if (total2 &gt; 3000 &amp;&amp; total3 &gt; 3000){
    mode=6; //cyan
    Serial.print("5");
} else if (total1 &gt; 3000 &amp;&amp; total3 &gt; 3000){
    mode=7; // white
    Serial.print("5");
} else if (total1 &gt; 3000){
    Serial.print("1");
} else if (total2 &gt; 3000){
    mode=2;
    Serial.print("2");
} else if (total3 &gt; 3000){
    mode=3;
    Serial.print("3");
}

const TProgmemPalette16 FairylightsPalette_p PROGMEM =
{
    CRGB::White,
    CRGB::Gray,
    CRGB::Gold,
    CRGB::Yellow,
    CRGB::Gray,
    CRGB::Yellow,
    CRGB::Gold,
    CRGB::Yellow,
    CRGB::Gray,
    CRGB::Yellow,
    CRGB::Gold,
    CRGB::Yellow,
    CRGB::Gray,
    CRGB::Yellow,
    CRGB::Gold,
    CRGB::Yellow,
    CRGB::Gray,
    CRGB::Yellow,
    CRGB::Gold,
    CRGB::Yellow,
};

void fairyLights()  //evaluate password. Also wait 3 seconds then go back to FairyLights
{
    FastLED.setBrightness(BRIGHTNESS);
    currentPalette = FairylightsPalette_p;
    static uint8_t startIndex = 0;
    startIndex = startIndex + 1;
    FillLEDsFromPaletteColors( startIndex);
}
FastLED.show();
FastLED.delay(25);

} void Wait(){
    if (timeElapsed &gt; interval)
    {
        mode=0;
        timeElapsed=0;
    }
}

void Solid()
{
    fill_solid(leds, NUM_LEDS, CHSV(HUE, SATURATION, BRIGHTNESS));
    FastLED.show();
    delay(20);
}

void Red()
{
    BRIGHTNESS=255; HUE=0; SATURATION=255;Solid();
    Wait();
}

void Purple()
{
    BRIGHTNESS=255; HUE=190; SATURATION=255;Solid();
    Wait();
}

void Blue()
{
    BRIGHTNESS=255; HUE=160; SATURATION=255;Solid();
    Wait();
}

void Cyan()
{
    BRIGHTNESS=255; HUE=120; SATURATION=255; Solid();
    Wait();
}

void Green()
{
    BRIGHTNESS=255; HUE=100; SATURATION=255; Solid();
    Wait();
}

void White()
{
    BRIGHTNESS=255; SATURATION=0; Solid();
    Wait();
}

void Rainbow()
{
    SATURATION=255;
    FastLED.setBrightness( BRIGHTNESS );
    currentPalette = RainbowColors_p;
    static uint8_t startIndex = 0;
    startIndex = startIndex + 1;
    FillLEDsFromPaletteColors( startIndex);
FastLED.show();
FastLED.delay(25);
}

void FillLEDsFromPaletteColors( uint8_t colorIndex)
{
  uint8_t brightness = BRIGHTNESS;
  for( int i = 0; i < NUM_LEDS; i++) {
    leds[i] = ColorFromPalette( currentPalette, colorIndex, brightness, currentBlending);
    colorIndex += 25;
  }
}

Wiring Diagram

Neopixel strip:

- Neopixel + to Teensy 3.3v
- Neopixel - toTeensyAGND
- Neopixel IN to Teensy 23
Power:

- Teensy G to battery G
- Teensy + to luggage lock
- Battery + to luggage key

Capacitive Sensors:

- Three 1M resistors to Teensy 4
- Other side of resistors split to individual sensors and Teensy pins 1, 2 and 3

Electronics Assembly

Assemble Battery Connector
Solder a red and black wire onto your female JST connector. Secure the individual wires with heat shrink. Then, add a 1/2 piece of heat shrink over the whole assembly. Fill it with hot glue and shrink it down to create a sturdy connector.

This is the bit that takes the most wear and tear so do a good job on this part.
Wire the LEDs

Solder 3 wires onto the "in" end of your neopixel strip. Attach the wires to Teensy at 3.3v, AGND and pin 23.

It's a great idea to upload the Neopixel Strandtest code to your Teensy and make sure all the LEDs are working before proceeding.
Power Switch

Solder the black G wire from your JST connector to G on your teensy.

I added a white 30 awg wire extension to my + wire, so it's a little easier to hide later on, since these wires will be outside the bottle. Connect this wire to the lock -- strip a whole bunch of wire and wrap it around the metal 3 or 4 times and solder it securely. The solder may not want to stick, but keep after it, it'll stick eventually.

Solder another power wire to Teensy's 3.3V pin and connect this to your key, soldering securely. Get a lot of metal-on-metal connection!

Plug in your battery. Slide the key into the lock and verify that your LEDs come on. If they don't, try adding more solder.
Prepare your Resistors

Twist your resistor legs around to form a loop. Solder one short wire onto one end, and one short and one long wire onto the other end of each resistor.

Cover the resistors with heat shrink.

Find the end of each resistor that has two wires. Take the shorter wire and solder one each into pins 1, 2, and 3 on the Teensy.
Now find the end of each resistor that has one wire. Take all 3 wires and twist them together, then solder into Teensy's pin 4.
For the 3 remaining long wires, solder each onto one of your capacitive touch charms. This may be tricky -- the solder doesn't always want to stick. Be sure your charms are clean, and if needed, sand or scrape the back to remove any coatings, and be sure you're getting a solid metal connection.

Sensor Calibration

Upload the test code below to your Teensy. While it's plugged into your computer, open your Serial monitor. You'll see a list of numbers scrolling down the screen.

With one finger on your switch lock, touch each charm and watch the numbers change. Get a sense of what the numbers are doing. You'll need a rough idea of the highs and lows you get when touching your charms to make the code work right.

```
//test code for Alohamora bottle
#include <CapacitiveSensor.h>
#include <FastLED.h>
#define DATA_PIN    23
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
#define NUM_LEDS    5        // Change this to reflect the number of LEDs you have

int HUE = 0;
int SATURATION = 255;
int BRIGHTNESS = 255;

CRGB leds[NUM_LEDS];
CRGBPalette16 gCurrentPalette;
CRGBPalette16 gTargetPalette;
```
/*
 * CapitiveSense Library Demo Sketch
 * Paul Badger 2008
 * Uses a high value resistor e.g. 10M between send pin and receive pin
 * Resistor effects sensitivity, experiment with values, 50K - 50M. Larger resistor
 * values yield larger sensor values.
 * Receive pin is the sensor pin - try different amounts of foil/metal on this pin
 * Modified by Becky Stern 2013 to change the color of one RGB Neo Pixel based on
 * touch input
 */

CapitiveSensor   cs_4_1 = CapitiveSensor(4,1);        // 10M resistor between
pins 4 & 2; pin 2 is sensor pin, add a wire and or foil if desired
CapitiveSensor   cs_4_2 = CapitiveSensor(4,2);        // 10M resistor between
pins 4 & 2; pin 2 is sensor pin, add a wire and or foil if desired
CapitiveSensor   cs_4_3 = CapitiveSensor(4,3);        // 10M resistor between
pins 4 & 2; pin 2 is sensor pin, add a wire and or foil if desired

void setup()
{
    delay(3000); // 3 second delay for recovery

    // tell FastLED about the LED strip configuration
    FastLED.addLeds<LED_TYPE,DATA_PIN,COLOR_ORDER>(leds, NUM_LEDS)
        .setCorrection(TypicalLEDStrip)
        .setDither(BRIGHTNESS &lt; 255);

    // set master brightness control
    FastLED.setBrightness(BRIGHTNESS);
    Serial.begin(9600);
}

void loop(){
    long start = millis();
    long total1 =  cs_4_1.capacitiveSensor(30);
    long total2 =  cs_4_2.capacitiveSensor(30);
    long total3 =  cs_4_3.capacitiveSensor(30);

    if (total1 &gt; 3000){
        digitalWrite(11, HIGH);
        BRIGHTNESS=255; HUE=0; Solid();
    } else if (total2 &gt; 3000){
        digitalWrite(13, HIGH);
        BRIGHTNESS=255; HUE=200; Solid();
    } else if (total3 &gt; 3000){
        digitalWrite(11, HIGH);
        BRIGHTNESS=255; HUE=160; Solid();
    } else {
        BRIGHTNESS=100; HUE=55; Solid();
    }

    Serial.print(millis() - start);        // check on performance in milliseconds
    Serial.print("\t");                    // tab character for debug window
    Serial.println(total1);                  // print sensor output 1
    Serial.println(total2);
    Serial.println(total3);
}
Serial.println(total2);                  // print sensor output 2
Serial.print("\t");
Serial.println(total3);                // print sensor output 3
delay(10);                             // arbitrary delay to limit data to serial
port
}

void Solid()
{
  fill_solid(leds, NUM_LEDS, CHSV(HUE, SATURATION, BRIGHTNESS));
  FastLED.show();
  delay(20);
}

In my serial monitor, I see the numbers jumping from the 20s-30s to the 10,000s when
I touch each charm. I chose a value of 3000 for the sensor cutoff, which works really
well with my charms.

Find the total1, total2 and total3 variables in the loop() portion of the test code and
play with the cutoff numbers to reflect what your charms are doing. When you've got
it right, the bottle will change colors when you touch each charm.

Once you have this working reliably while plugged into your computer, unplug your
arduino and try it with just the battery plugged in. To make it work reliably, keep one
finger on your on/off switch lock while you touch the charms.

Science Moment!

Capacitive touch sensors work great when they're plugged in to your computer or a
power grid of some kind. They need the tasty ground connection that comes with
plugged-in power.
Running capacitive touch sensors from a battery means no strong ground connection... like, to the ground beneath your feet. Capacitive touch sensors really want that connection.

When you touch the metal of your on/off switch lock with your finger, your body and your feet on the ground will provide that needed ground connection. Then the rest of the circuit is happy as a clam and will proceed to function properly.

Okay! Once you have your charms all working beautifully, it's time to assemble the bottle and make everything pretty.

---

**Bottle Assembly**

![Bottle Assembly Image](image)
Decorate the Bottle
Get out your heat gun and cut a bunch of strips of fantasy film. Layer the film over your bottle and gently hit it with the heat. Watch it shrink and bond around your bottle. (Man this is fun.) Put a bunch of layers on until you’re happy with the diffusion.

Cut a bunch of strips into your leather scraps and artfully wrap it around the bottle, tying and gluing as needed. I left a few long strips and a loop at the top so I can carry my bottle around with me easily, or hang it from my belt.
Once you're happy with the bottle, carefully stuff in the guts. Start with the LEDs (be sure they're facing inwards, for more diffusion / glow), then the teensy and resistors, then the battery and connector.

Leave the 3 charms and the lock and key outside.
Put the key in the lock and be sure everything's still working.

Hide the Wiring

The other purpose of the leather wraps are to hide all the wiring, so your bottle looks purely magical.

Place your charms on the outside of the bottle and glue them down, strategically placing the wires behind the leather.
Finally, add strain relief and hide the wires to the lock and key with more leather strips.
Use It

The bottle will start up in "fairy lights" mode, a shifting yellow and white twinkle.

Touching the locks in different combinations will result in different colors appearing on the bottle. If you touch all 3 locks at the same time, you'll get a swirling rainbow color pattern.

Customize the code to create your own color combinations!