Creating Things That Think

-

Project 1

Eric Martin

September 14, 2021

IDEA 310L

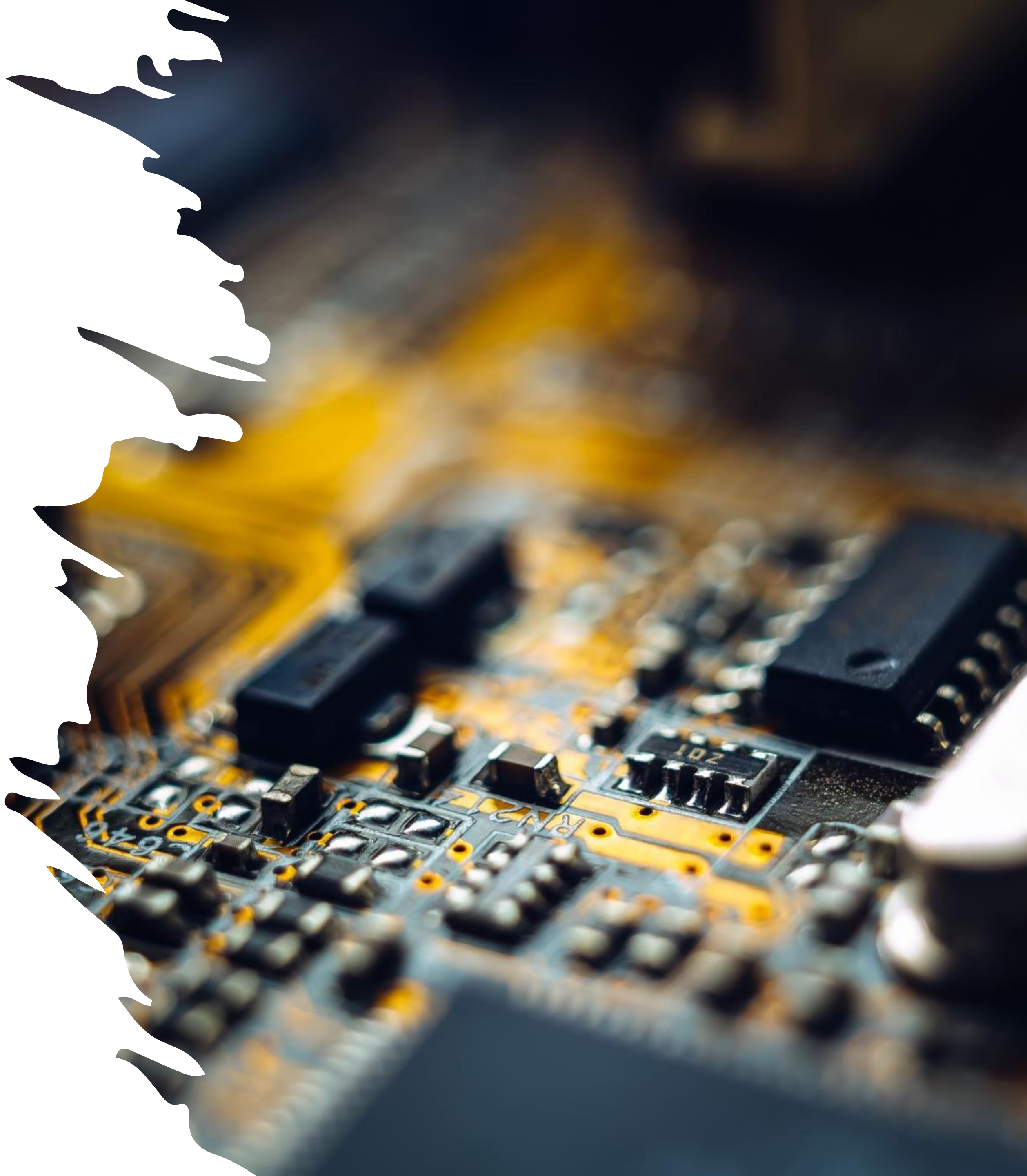
Description of Design Process

Design requirements :

- Using at least two inputs
- Using at least two outputs

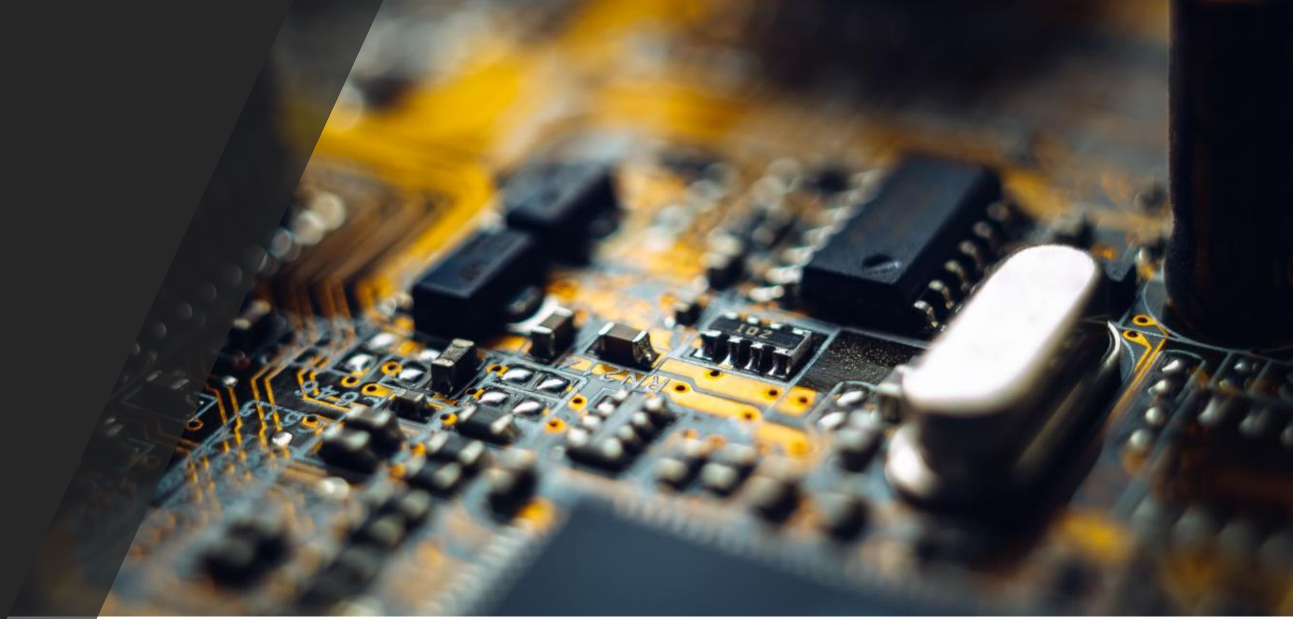
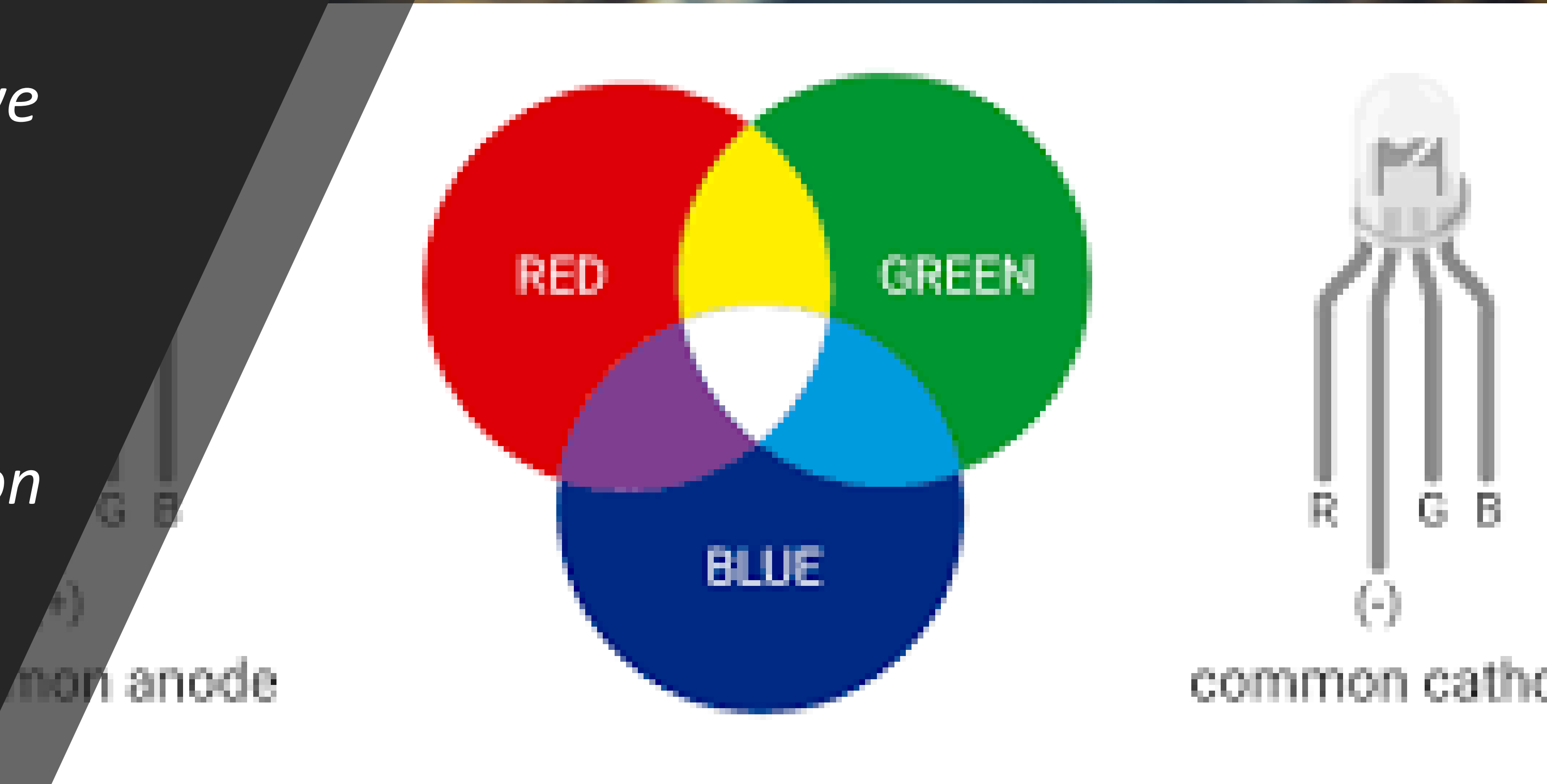
Design Process :

- Want to implement joystick as an analog input
- Want to implement LED’s
- Want to implement sound
- Want the design to look nice and compact



Description of Design

- *Utilizing the X-Y-Axis potentiometers found in the analog joystick, I have created a simple device that allows the user to manually change the color of an RGB LED depending on the position of the joystick.*

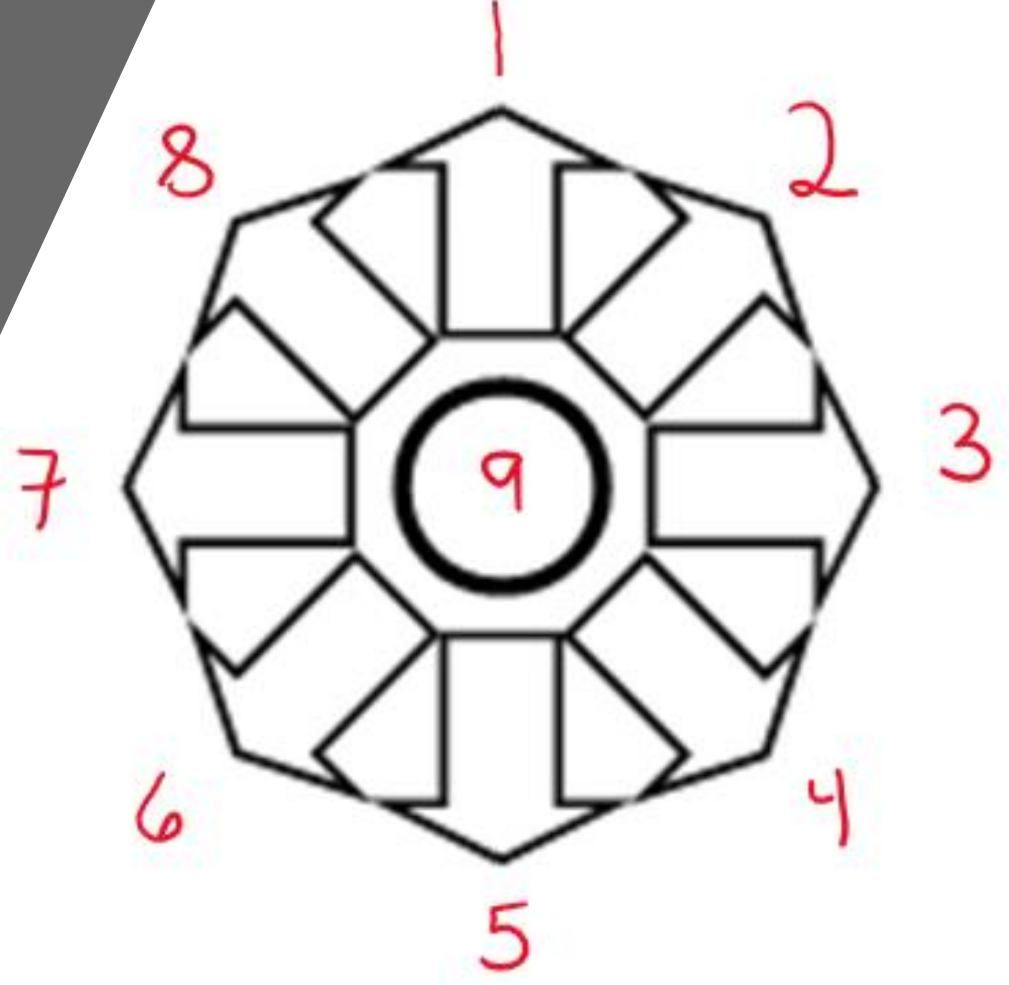
Description of Design

The joystick also has a push button input switch.

I have made it so that when the joystick is depressed, depending on the position when depressed, a unique output will occur.

1 – 8 will play a unique song

9 will display a random LED color



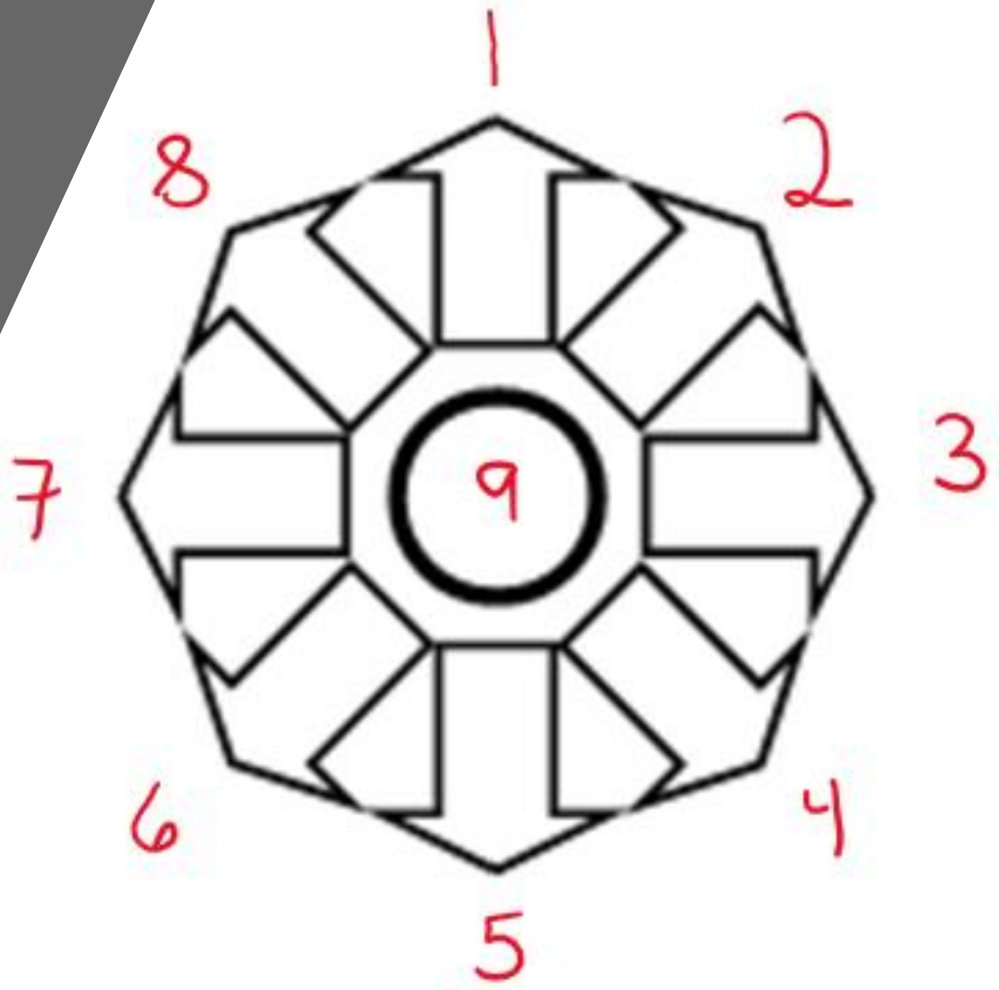
Description of Design

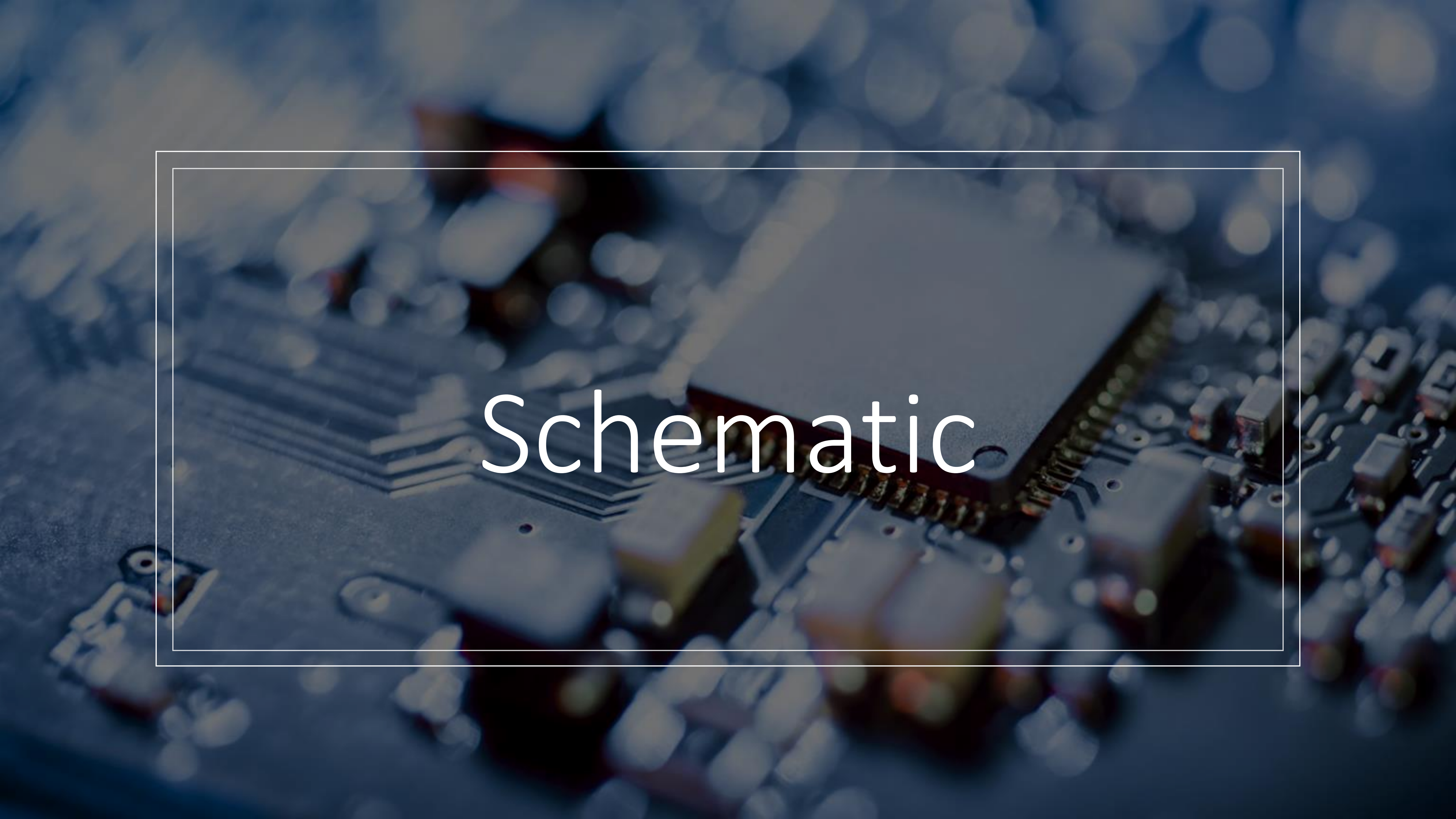
Songs:

1. Final Fantasy Victory Fanfare
2. Game of Thrones Theme
3. Hedwig's Theme – Harry Potter
4. Imperial March – Star Wars
5. Mario Theme Song
6. Jigglypuff's Song – Pokemon
7. Keyboard Cat Meme Song – Youtube
8. Greensleeves – Skyrim

Lights:

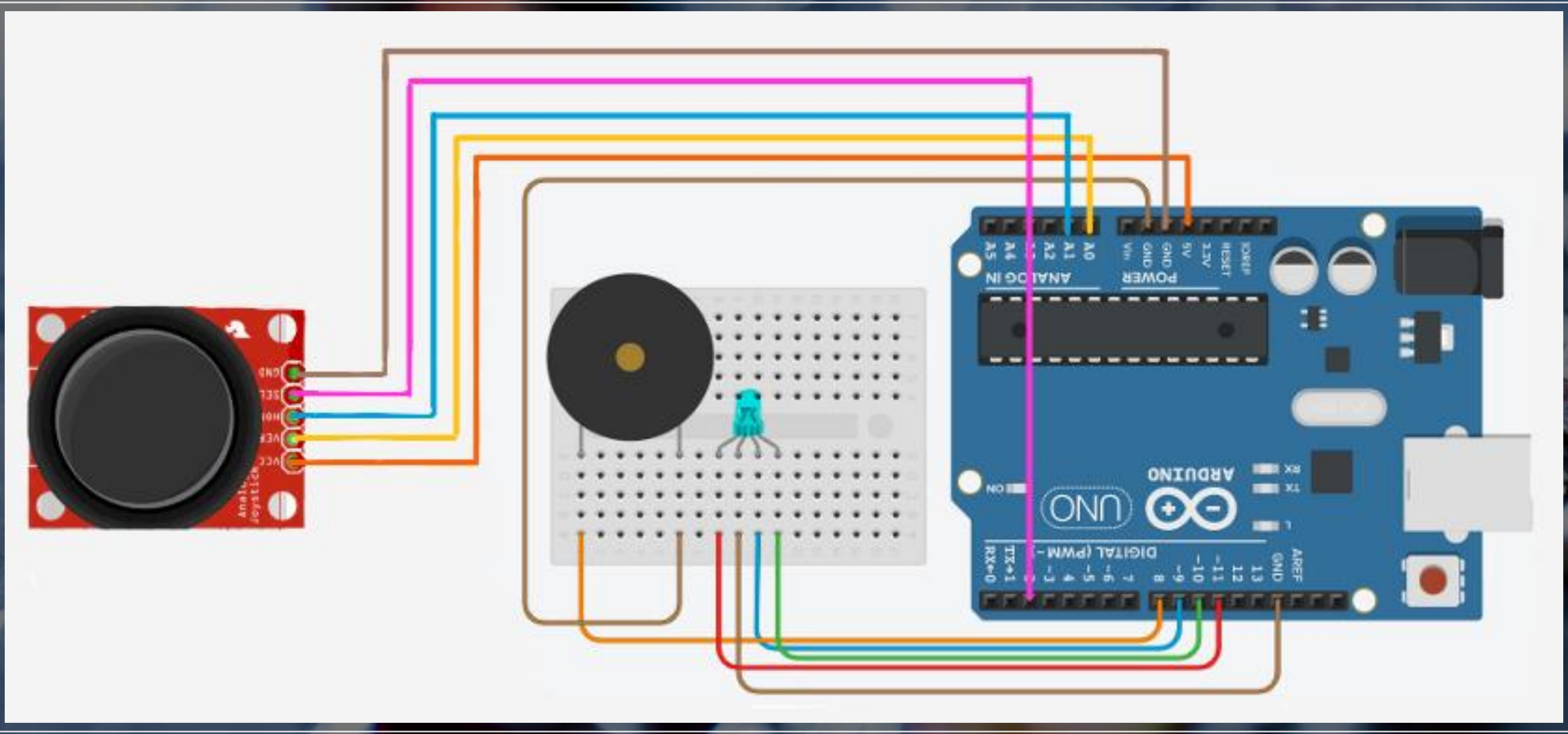
9.Random LED color





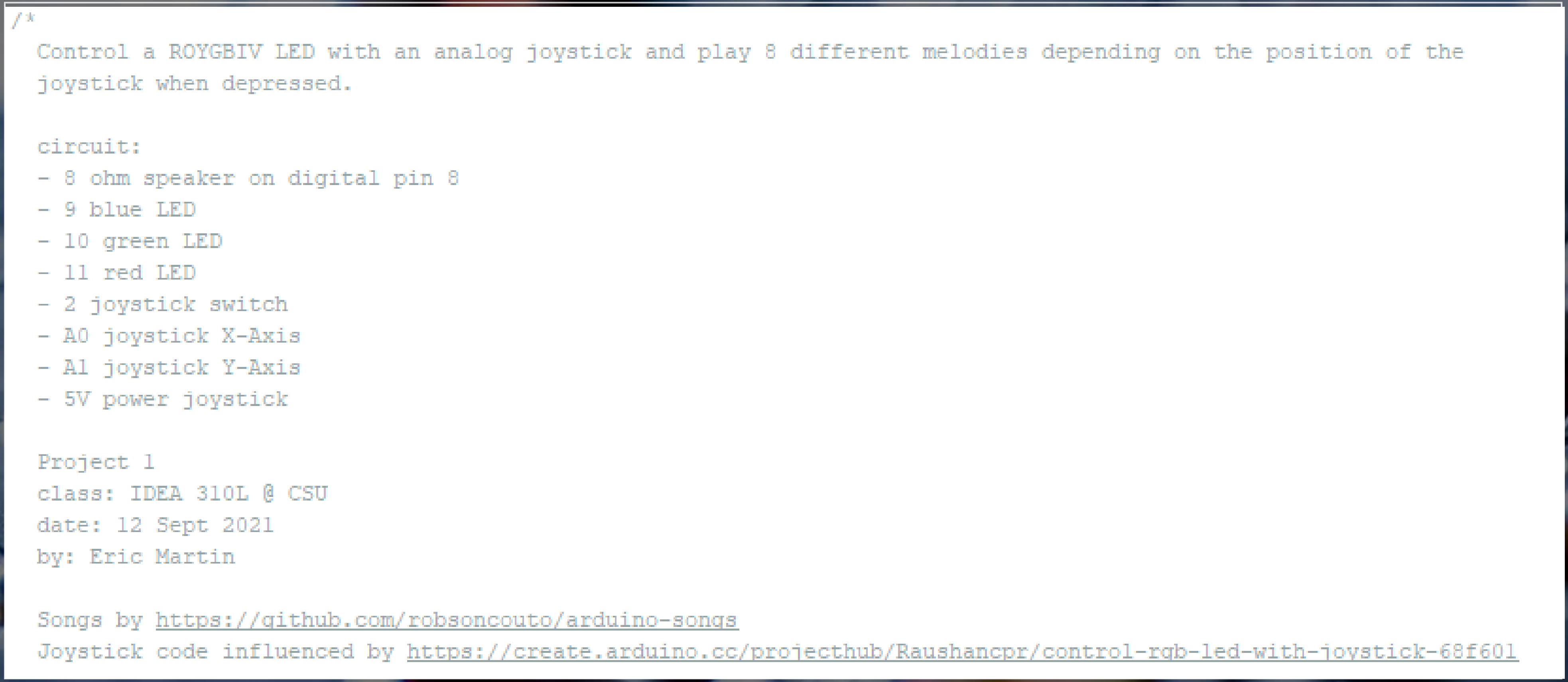
Schematic

Schematic

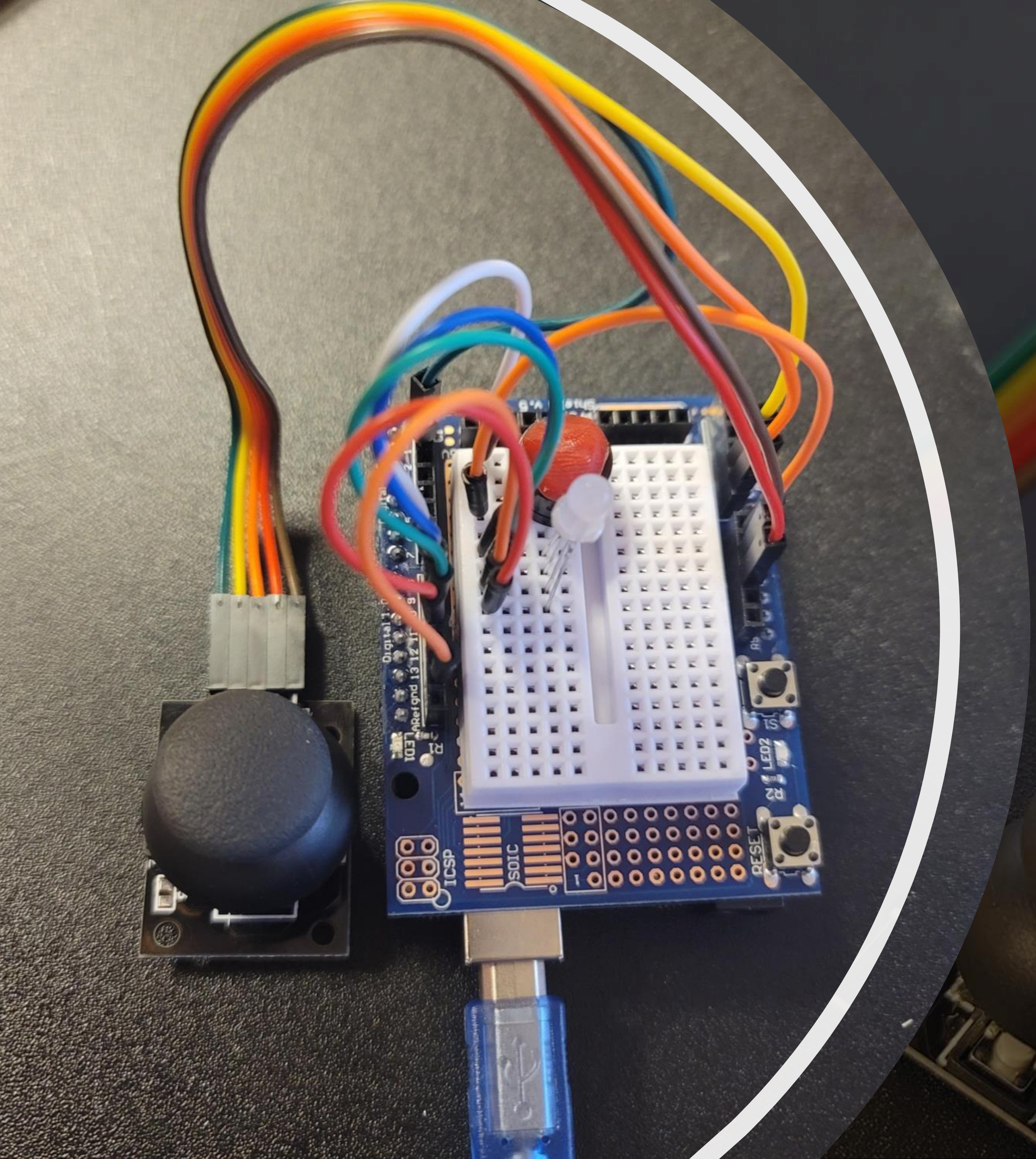


GND
SEL
HOR
VER
VCC
Analog Joystick
A5 A4 A3 A2 A1 A0
ANALOG IN
Vin GND GND 5V 3.3V RESET IOREF
POWER
ARDUINO
UNO
ON
RX
TX
L
DIGITAL (PWM ~)
RX←0 TX→1 2 ~3 4 ~5 ~6 7 8 ~9 ~10 ~11 12 13 GND AREF

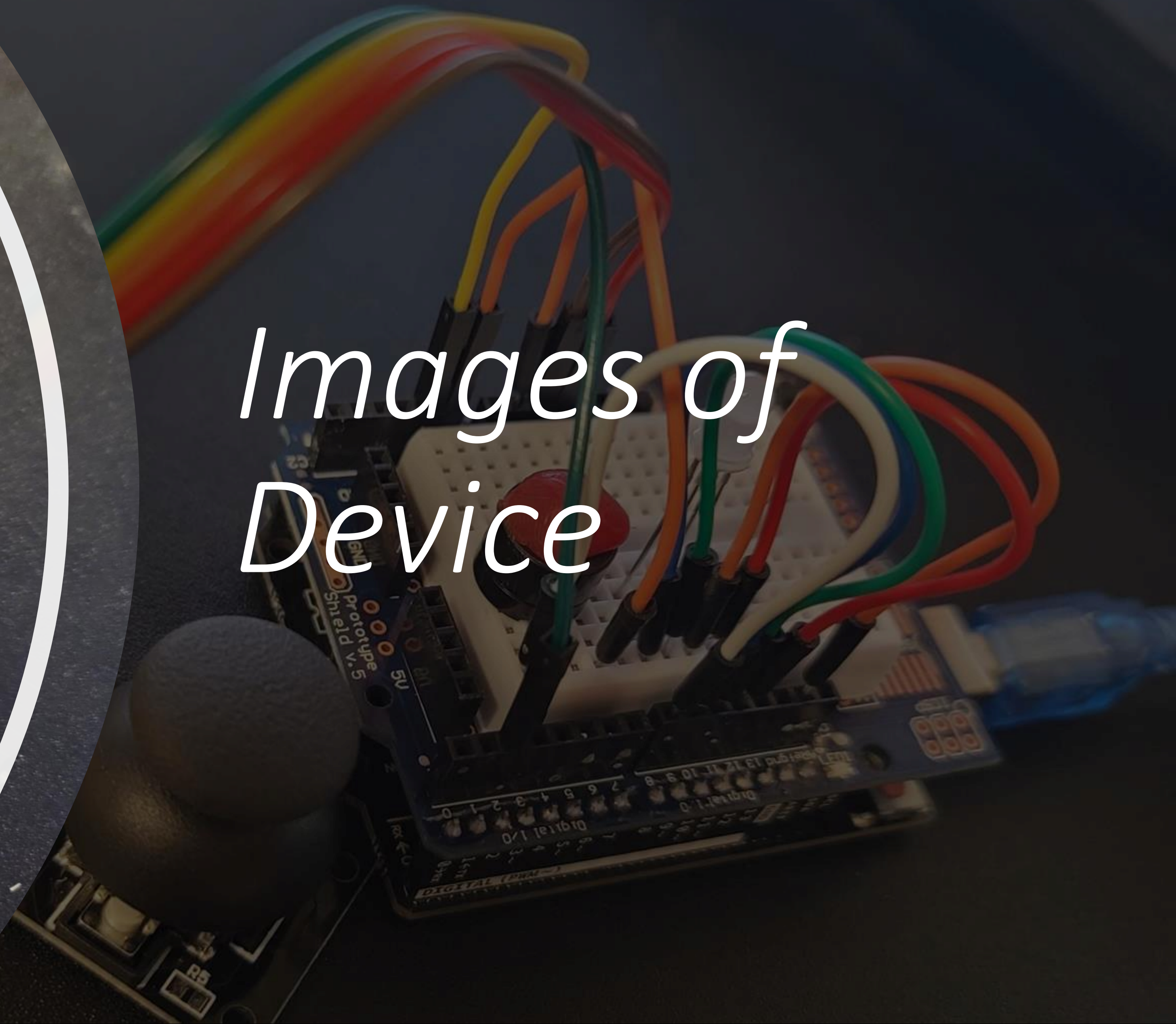
Schematic



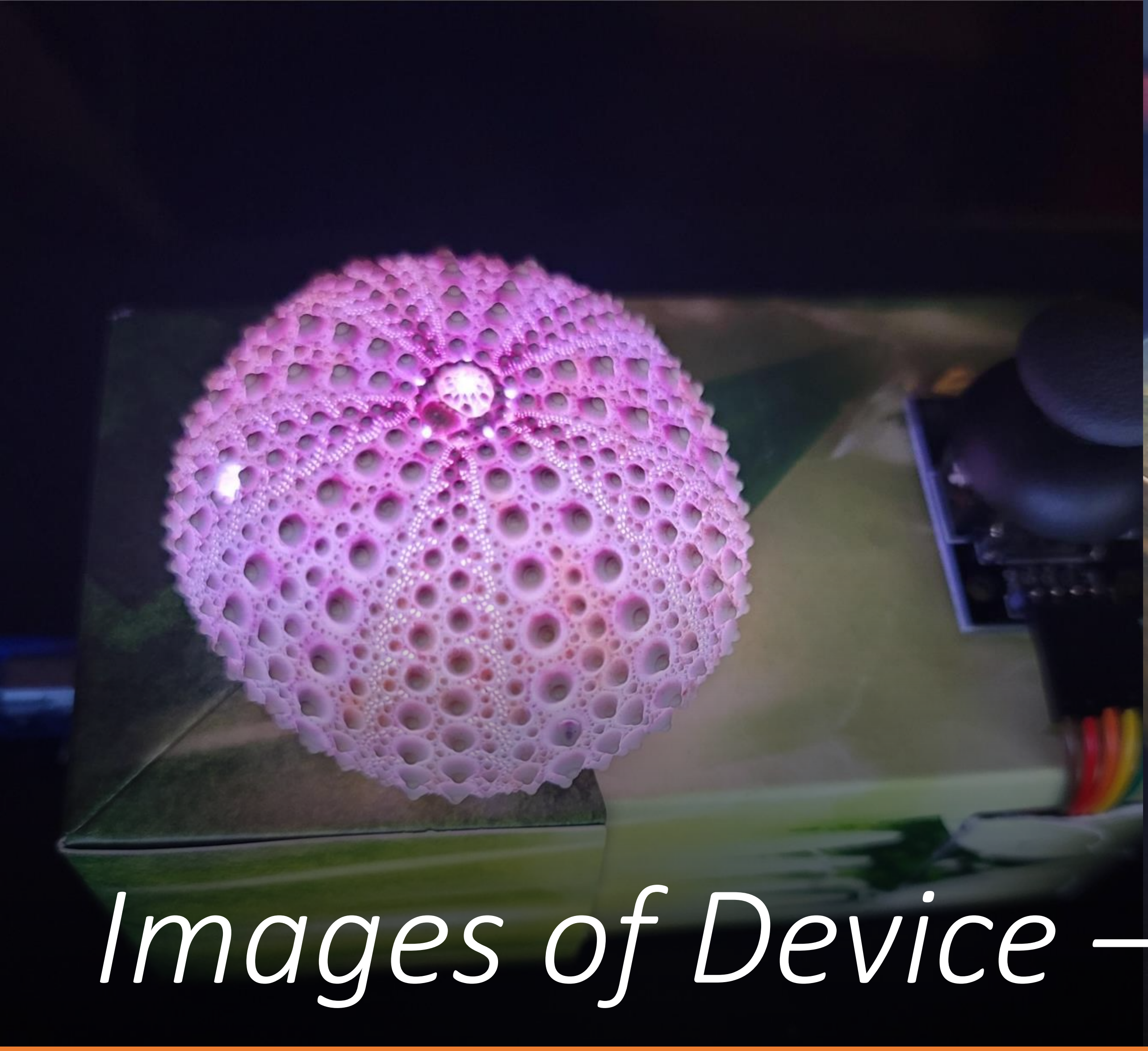
```
/*
  Control a ROYGBIV LED with an analog joystick and play 8 different melodies depending on the position of the
  joystick when depressed.

  circuit:
  - 8 ohm speaker on digital pin 8
  - 9 blue LED
  - 10 green LED
  - 11 red LED
  - 2 joystick switch
  - A0 joystick X-Axis
  - A1 joystick Y-Axis
  - 5V power joystick

  Project 1
  class: IDEA 310L @ CSU
  date: 12 Sept 2021
  by: Eric Martin

  Songs by https://github.com/robsoncouto/arduino-songs
  Joystick code influenced by https://create.arduino.cc/projecthub/Raushancpr/control-rgb-led-with-joystick-68f601
```



Images of Device



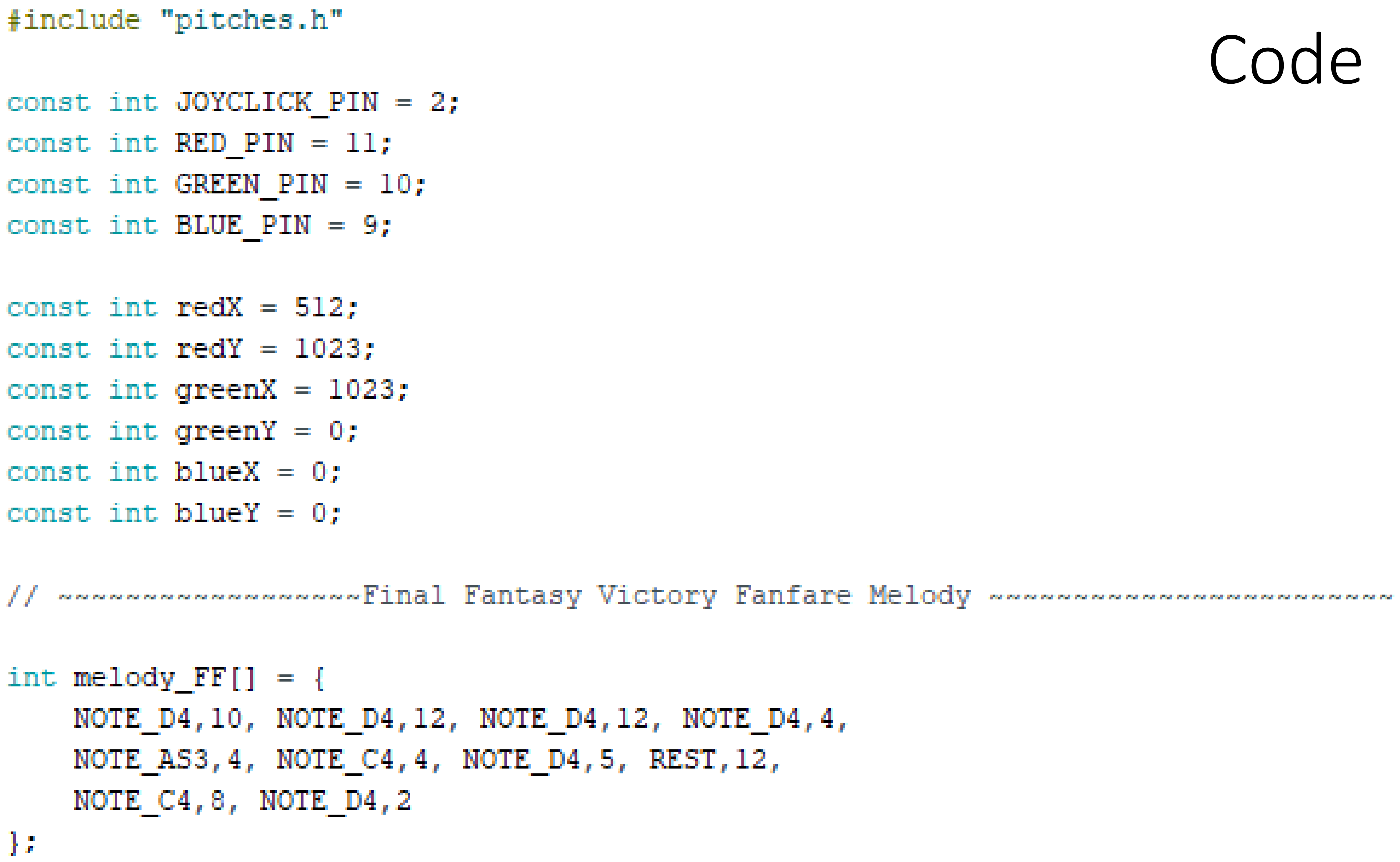
Images of Device – In Box



```
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

election at the end -add
 _ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifie
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

 OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
```

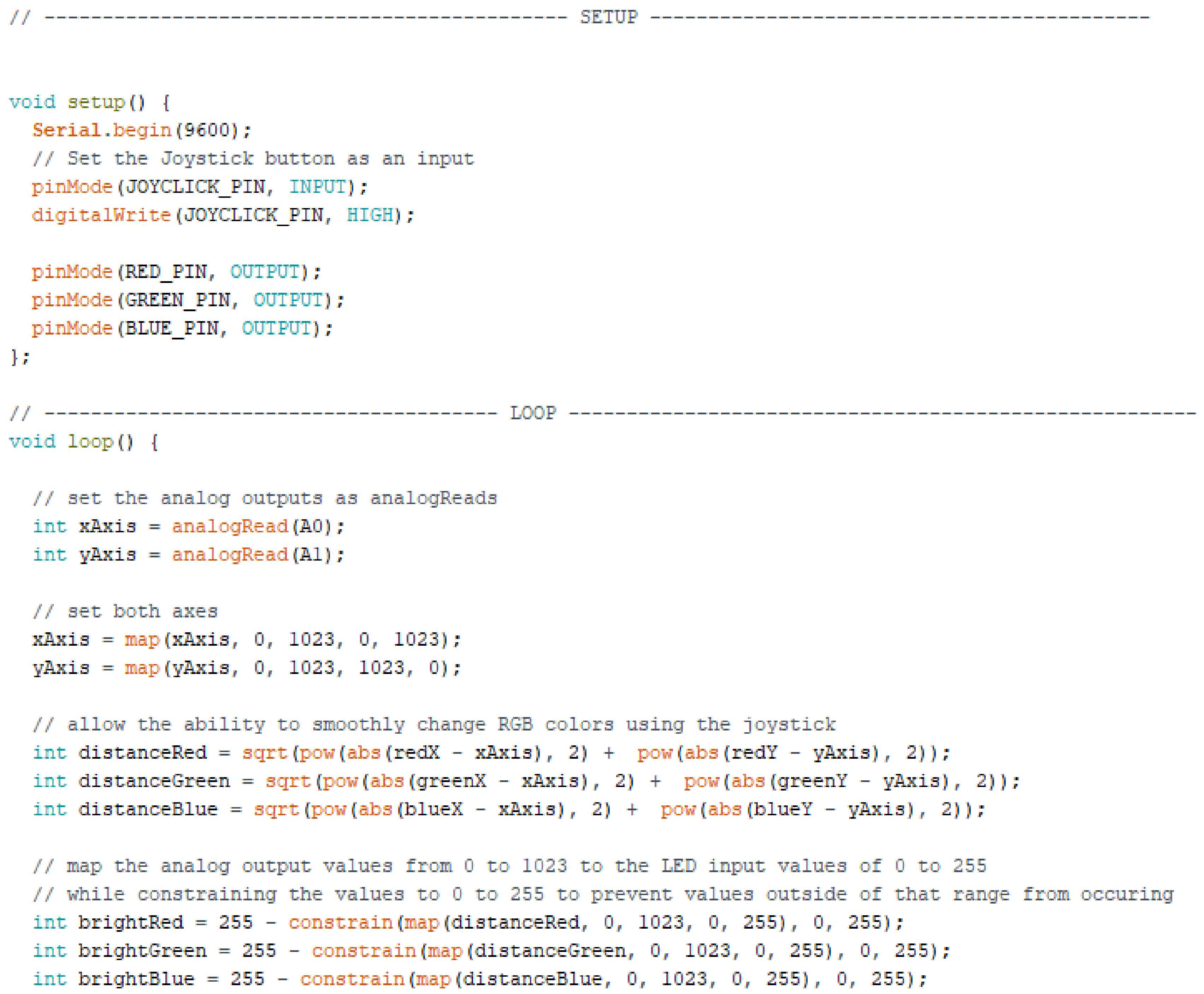
Code

Code



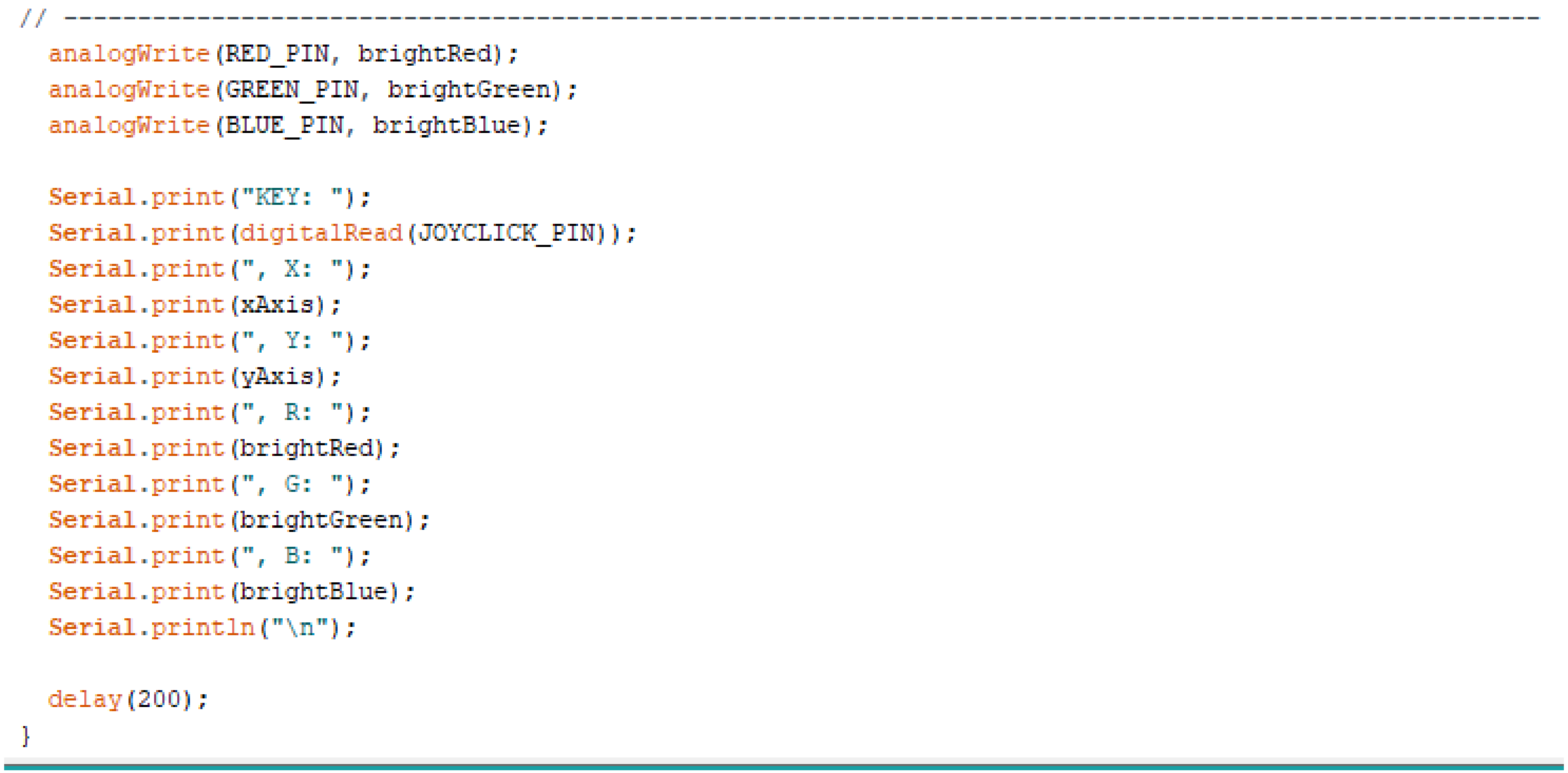
```
#include "pitches.h"

const int JOYCLICK_PIN = 2;
const int RED_PIN = 11;
const int GREEN_PIN = 10;
const int BLUE_PIN = 9;

const int redX = 512;
const int redY = 1023;
const int greenX = 1023;
const int greenY = 0;
const int blueX = 0;
const int blueY = 0;

// ~~~~~~~~~~~~~~~~~~Final Fantasy Victory Fanfare Melody ~~~~~~~~~~~~~~~~~~~~~~~~

int melody_FF[] = {
    NOTE_D4,10, NOTE_D4,12, NOTE_D4,12, NOTE_D4,4,
    NOTE_AS3,4, NOTE_C4,4, NOTE_D4,5, REST,12,
    NOTE_C4,8, NOTE_D4,2
};
```

Code



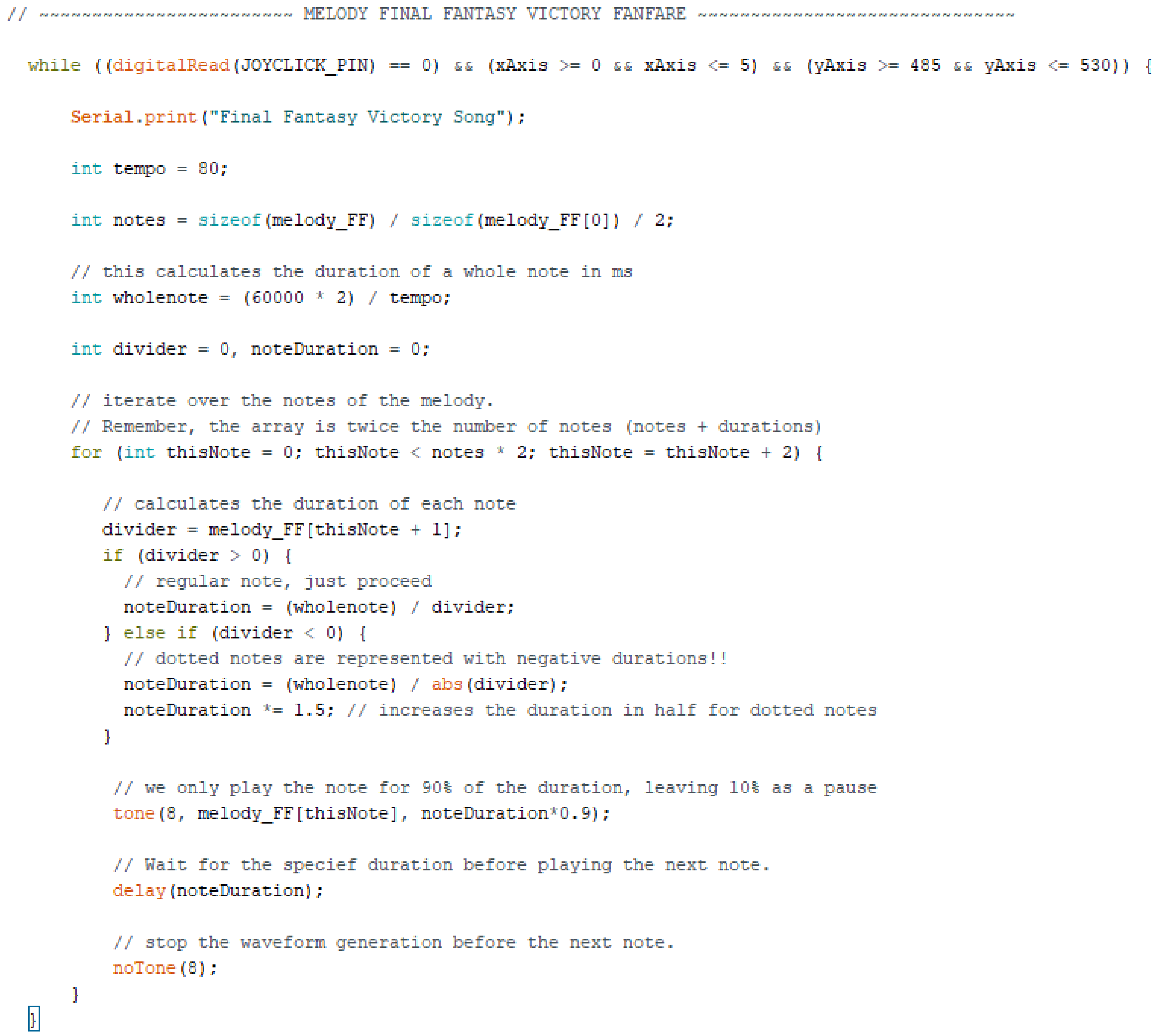
```
// ---------------------------------------------- SETUP -------------------------------------------


void setup() {
  Serial.begin(9600);
  // Set the Joystick button as an input
  pinMode(JOYCLICK_PIN, INPUT);
  digitalWrite(JOYCLICK_PIN, HIGH);

  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
};

// -------------------------------------- LOOP ------------------------------------------------------
void loop() {

  // set the analog outputs as analogReads
  int xAxis = analogRead(A0);
  int yAxis = analogRead(A1);

  // set both axes
  xAxis = map(xAxis, 0, 1023, 0, 1023);
  yAxis = map(yAxis, 0, 1023, 1023, 0);

  // allow the ability to smoothly change RGB colors using the joystick
  int distanceRed = sqrt(pow(abs(redX - xAxis), 2) +  pow(abs(redY - yAxis), 2));
  int distanceGreen = sqrt(pow(abs(greenX - xAxis), 2) +  pow(abs(greenY - yAxis), 2));
  int distanceBlue = sqrt(pow(abs(blueX - xAxis), 2) +  pow(abs(blueY - yAxis), 2));

  // map the analog output values from 0 to 1023 to the LED input values of 0 to 255
  // while constraining the values to 0 to 255 to prevent values outside of that range from occuring
  int brightRed = 255 - constrain(map(distanceRed, 0, 1023, 0, 255), 0, 255);
  int brightGreen = 255 - constrain(map(distanceGreen, 0, 1023, 0, 255), 0, 255);
  int brightBlue = 255 - constrain(map(distanceBlue, 0, 1023, 0, 255), 0, 255);
```

Code



```
// ------------------------------------------------------------------------------------------------
  analogWrite(RED_PIN, brightRed);
  analogWrite(GREEN_PIN, brightGreen);
  analogWrite(BLUE_PIN, brightBlue);

  Serial.print("KEY: ");
  Serial.print(digitalRead(JOYCLICK_PIN));
  Serial.print(", X: ");
  Serial.print(xAxis);
  Serial.print(", Y: ");
  Serial.print(yAxis);
  Serial.print(", R: ");
  Serial.print(brightRed);
  Serial.print(", G: ");
  Serial.print(brightGreen);
  Serial.print(", B: ");
  Serial.print(brightBlue);
  Serial.println("\n");

  delay(200);
}
```

Code



```
// ~~~~~~~~~~~~~~~~~~~~~~~~ MELODY FINAL FANTASY VICTORY FANFARE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  while ((digitalRead(JOYCLICK_PIN) == 0) && (xAxis >= 0 && xAxis <= 5) && (yAxis >= 485 && yAxis <= 530)) {

      Serial.print("Final Fantasy Victory Song");

      int tempo = 80;

      int notes = sizeof(melody_FF) / sizeof(melody_FF[0]) / 2;

      // this calculates the duration of a whole note in ms
      int wholenote = (60000 * 2) / tempo;

      int divider = 0, noteDuration = 0;

      // iterate over the notes of the melody.
      // Remember, the array is twice the number of notes (notes + durations)
      for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {

         // calculates the duration of each note
         divider = melody_FF[thisNote + 1];
         if (divider > 0) {
           // regular note, just proceed
           noteDuration = (wholenote) / divider;
         } else if (divider < 0) {
           // dotted notes are represented with negative durations!!
           noteDuration = (wholenote) / abs(divider);
           noteDuration *= 1.5; // increases the duration in half for dotted notes
         }

          // we only play the note for 90% of the duration, leaving 10% as a pause
          tone(8, melody_FF[thisNote], noteDuration*0.9);

          // Wait for the specief duration before playing the next note.
          delay(noteDuration);

          // stop the waveform generation before the next note.
          noTone(8);
      }
  }
```

Issues

- When activating some songs, the LED turns off.
  - Unsure about why this is occurring as each melody has the same outline of code.
- Program global variables use 79% of the dynamic memory on the heap which leaves less memory and could lead to stability issues.
  - This is caused by the large amount of melody variables declared
- Light transitions when using the joystick are sometimes jumpy rather than smooth.
  - This may be due to the limitations of the joystick input or the translation from analog to digital input.
- Box is flimsily and makes using the joystick harder than it needs to be