



## Volume Knob V2 (USB-C)



Max Siebenschläfer

[VIEW IN BROWSER](#)

updated 13. 5. 2024 | published 13. 5. 2024

### Summary

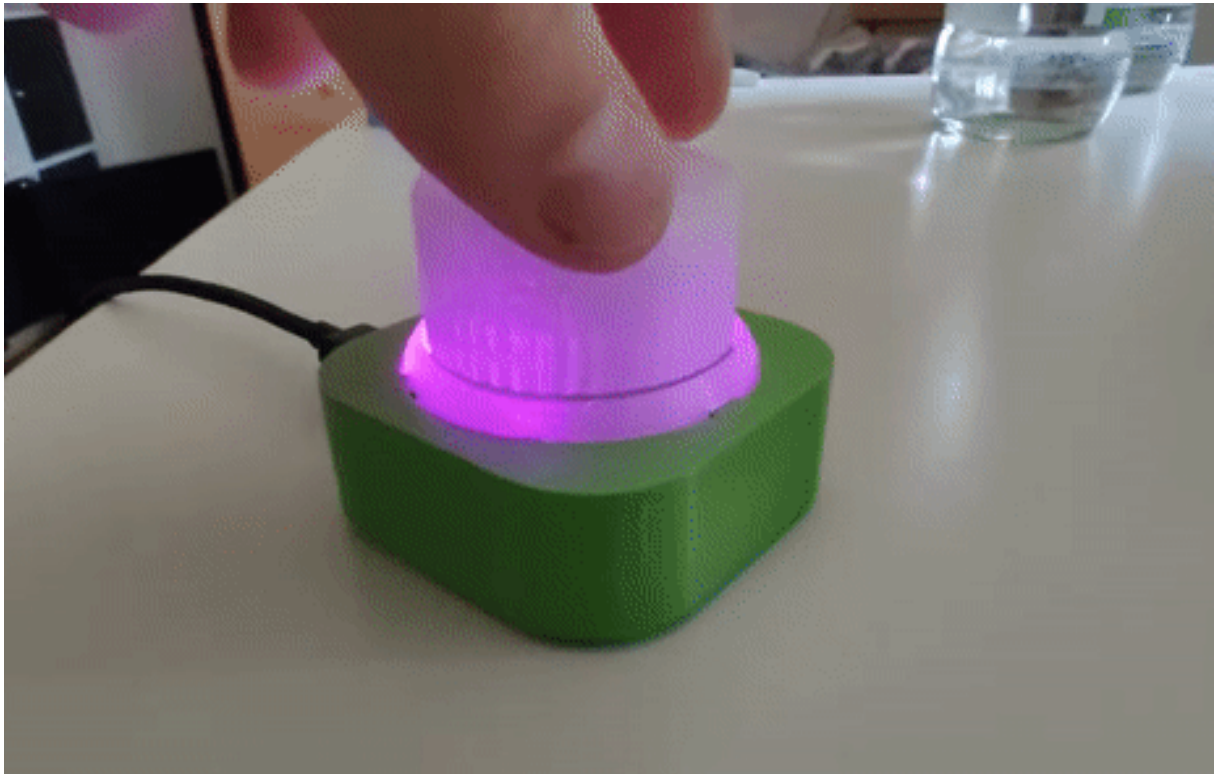
An Arduino pro micro project to control volume or mute the sound. You also use rgb to have a nice experience.

[Gadgets](#) > [Computers](#)

Tags: [led](#) [arduino](#) [leds](#) [media](#) [mediacontrolvolume](#)  
[rotaryencoder](#) [volumeknob](#) [arduino promicro](#) [multimedia](#)

An Arduino pro micro project to control volume or mute the sound. You also use rgb to have a nice experience.

**Regular Mode:**



### Animation-Modes:



### Assembly of the Volume Knob V2

This instruction is for those interested in assembling the Volume Knob V2, as shown in Figure 1. Firstly, gather all the necessary components from the sourcing list below. The enclosure parts should be created by 3D printing. You can download the files from the provided [link](#).

- 1 x [Arduino Pro Micro\(USB-C\)](#) blue version

- 8 x **SK6812 LEDs**(5050 Chip)
- 1 x **ALPS Rotary Encoder with integrated Switch**
- 1 x **10k Ohm Resistor**
- 4x M3X16 screws
- 4x **M3 threaded inserts**
- 1x **PCB**
- 1x **soldering iron**

It will cost around €20 per Volume Knob. If you want to find more information, you can go to my [GitHub page](#).

### 3D Printing:

You can print all the parts from PLA besides the transfuser that should be transparent. (Optional) use the special feature to edit the bottom pattern. I liked the Octagram Spiral that



### Assembly:

The exploded drawing displays the parts that are included in the assembly. The grey components (1,2,3,5) represent the 3D printed components. The green component (4) is the assembled PCB, which can be bought.

Solder the rotary encoder, Arduino Pro Micro and the 10k ohm resistor onto the PCB (4) at the designated locations.

Component 3 features four inner holes. Use a soldering iron and embed the four M3 threaded inserts into these four openings.

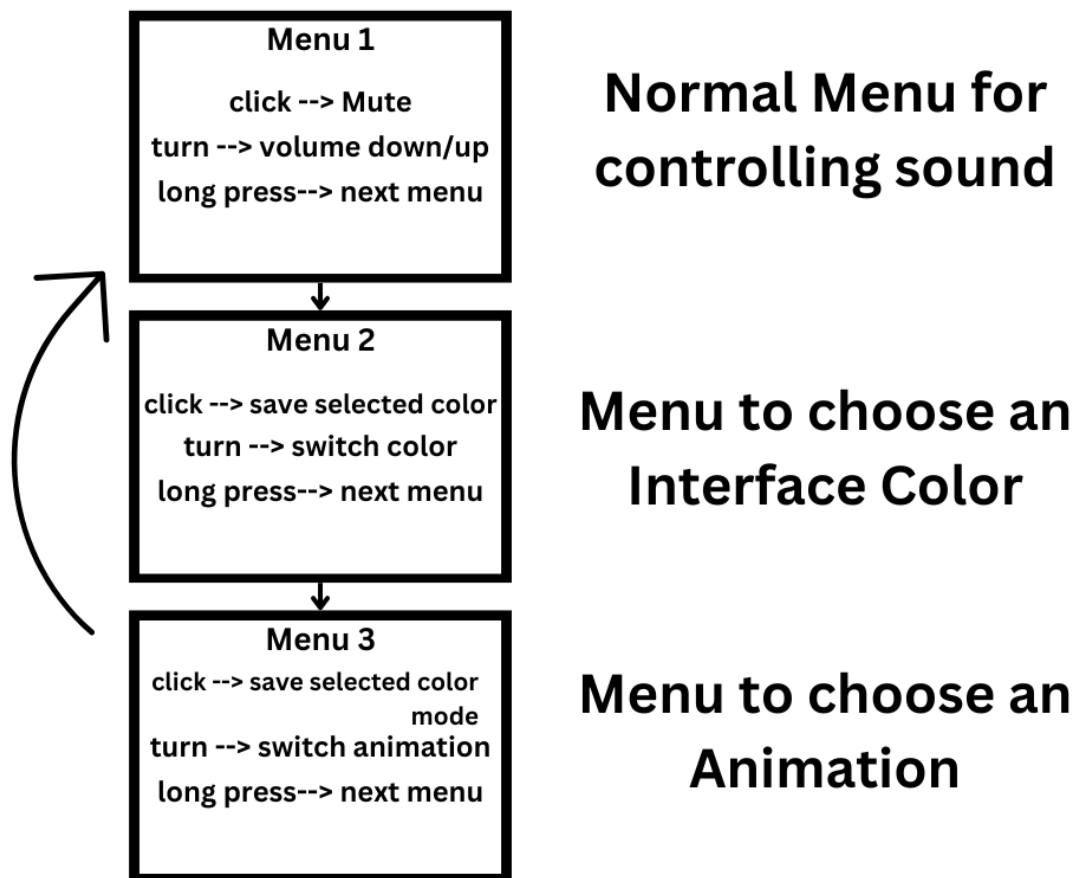
Insert the transparent ring (2) firmly into the three elongated holes on component 3, connecting the components with a snap-fit mechanism. Apply gentle pressure to ensure a secure fit.

Assemble the PCB (4) by inserting it in from the bottom with the USB-connector facing towards the cutout. Insert component 5 into the middle-part, then use four M3 screws to secure the bottom-part in place.

Lastly, push the tubular part of the control knob with the Play/Pause symbol (1) onto the cylindrical rotary encoder. This connection should be press-fit, requiring no adhesive.

## Software:

You have a limited amount of control. The software is arranged in a system like on the picture.



You have the selection of 25 different colors and 5 different animation modes to choose from.

Program the Arduino with the [Arduino IDE](#)

```
#include <HID-Project.h> #include <Encoder.h> #include  
<Adafruit_NeoPixel.h> #include <OneButton.h> #include <EEPROM.h>  
#define PIN 7 //Datapin for the LEDs #define NUMPIXELS 8 //number of  
LEDs in the row //Color Options byte color[25][3] = {{255,0,0},  
{255,64,0},{255,128,0},{255,191,0},{255,255,0},{191,255,0},  
{128,255,0},{64,255,0},{0,255,0},{0,255,64},{0,255,128},{0,255,191},  
{0,255,255},{0,191,255}, {0,128,255},{0,64,255},{0,0,255},  
{64,0,255},{128,0,255},{191,0,255},{255,0,255},{255,0,191},  
{255,0,128},{255,0,64},{255,0,0}}; //Array with the different color-values  
would be better to do this with a function int row = 0; //pointer, which  
color is selected ##das is kein Pointer //Rotary Encoder const int CLK =  
6;//Clockpin Encoder const int DT = 5; //Datepin Encoder const int SW = 2;  
//Button of the Encoders int pos = 0; //the variable for the Encoder-function  
to give back long oldPosition = -999; //position for the rotary encoder bool
```

```

turnLeft = false; //State for turning left //I use this variables to check that
you have to move two incereements to get one move in the volume bool
turnRight = false; //State for turning right //variable for the first mode int
newPos = 0; //the selected LED that should light up int oldpos = 0; //the
last lit up LED, You need this to turn the LED off after you jump to the next
one ##Name nicht eindeutig, oldPosition-oldpos //Rainbow-Mode uint16_t
r, m; //Florian-Mode byte florian[3][3] = {{0,255,0},{255,255,0},
{255,0,0}}; //green yellow red for the traffic light mode int
volumelnPercent = 0; //Settings-menu int menu = 0; //0 = Sound-Mode, 1
= color selection, 2 = modi-selection int mode = 1; //1-7 = Rotating-Pixel,
9-15 = rgb-wheel fade, 17-23 = rgb-wheel with the moving colors, 25-31 =
percent red, yellow, green for different noise levels, 33-39 = similar to the
first mode but you have 3 LEDs and the pixels stay on int brightness =
255; //of LEDs from 0-255 //Generall Objects OneButton
EncoderSwitch(SW,true); //Library that adds functions for LongPress,
DoubleClick etc. Encoder myEncoder(DT,CLK); Adafruit_NeoPixel
pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800); void setup(); void
loop(); void volumeControl(); void colorSelection(); void modeSelection();
void showPixels(); void onePixelFade(); void rgbWheelFade(); void
rgbWheelMoving(); void pixelFadeStay(); void clicked(); void longPress();
int TurnEncoder(int len); uint32_t Wheel(byte WheelPos); void resetLEDs();
void lightAll(int r, int g, int b); void setup(){ Serial.begin(9600);
Serial.println("Consumer setup..."); Consumer.begin(); //Rotary Encoder
with Switch Serial.println("Rotary Encoder setup..."); pinMode(SW, INPUT);
EncoderSwitch.attachClick(clicked);
EncoderSwitch.attachLongPressStop(longPress); //LEDs pixels.begin(); //
Read saved settings row = EEPROM.read(0); mode = EEPROM.read(1);
for(int i = 0; i<50;i++){ //for a special mode doesn't work till now ##geht
hier ein direktes auf 0 setzten nicht ? Consumer.write(MEDIA_VOL_DOWN);
} Consumer.write(MEDIA_VOL_UP); Consumer.write(MEDIA_VOL_UP);
volumelnPercent = 4; } void loop(){ if(menu == 0){ volumeControl(); }
else if (menu == 1) { colorSelection(); } else if (menu == 2) {
modeSelection(); } showPixels(); EncoderSwitch.tick(); //the library reads
the encoder value and gives back an int delay(10); } void volumeControl()
{ int i = TurnEncoder(NUMPIXELS); if(i != -1){ newPos = i; brightness =
255; if(newPos > oldpos && newPos != NUMPIXELS-1 && oldpos != 0 ||
newPos == 0 && oldpos == NUMPIXELS-1){ //solves the issue from 0 to
NUMPIXELS-1 and NUMPIXELS-1 to 0 Consumer.write(MEDIA_VOL_UP);
volumelnPercent -= 2; if(volumelnPercent < 0){ volumelnPercent = 0; } }
else if(newPos < oldpos && newPos != 0 && oldpos != NUMPIXELS-1 ||
newPos == NUMPIXELS-1 && oldpos == 0){
Consumer.write(MEDIA_VOL_DOWN); volumelnPercent += 2;
if(volumelnPercent>100) { volumelnPercent = 100; } } } void
colorSelection(){ int i = TurnEncoder(25); //turns the Encoder with the
length of the array if(i != -1){ row = i; lightAll(color[row][0],color[row]
[1],color[row][2]); } } void modeSelection(){ int i =

```

```

TurnEncoder(NUMPIXELS*4); //6 Positions for every mode with a blank spot
between the modes if(i != -1){ mode = i; brightness = 255; } } void
showPixels(){ //function for the different Modis you could select if((menu
== 0 || menu == 2 )&& mode > 0 && mode < NUMPIXELS){ //The first
mode with only one pixel lit up that fades the color after some time
onePixelFade(); } else if((menu == 0 || menu == 2)&& mode >
NUMPIXELS && mode < NUMPIXELS*2){ //the rgb-wheel fade
rgbWheelFade(); } else if((menu == 0 || menu == 2) && mode >
NUMPIXELS*2 && mode < NUMPIXELS*3){//the rgb-wheel with the moving
colors rgbWheelMoving(); } else if((menu == 0 || menu == 2) && mode >
NUMPIXELS*3 && mode < NUMPIXELS*4){ //similar to the first mode but
you have 3 LEDs and the pixels stay on. pixelFadeStay(); } if(mode ==
NUMPIXELS || mode == NUMPIXELS*2 || mode == NUMPIXELS*3 || mode
== NUMPIXELS*4 || mode == NUMPIXELS*5 || mode == 0 && menu == 2)
{ //The blank spots in the mode-menu where no animations should be
shown resetLEDs(); } pixels.setBrightness(brightness); pixels.show(); //
Only time I update the pixels } void onePixelFade(){ int value = 0; if(menu
== 0){ value = newPos; } else if(menu == 2){ value = mode; } if(value !=
-1 && (menu == 2 || menu == 0)){ pixels.setPixelColor(oldpos,
pixels.Color(0,0,0)); pixels.setPixelColor(value, pixels.Color(color[row]
[0],color[row][1],color[row][2])); oldpos = value; } if(brightness <= 1){
resetLEDs(); } for(int i = 0; i < NUMPIXELS; i++){
if(pixels.getPixelColor(i) != pixels.Color(0,0,0)){ brightness = brightness -
1; i = NUMPIXELS; } } } void rgbWheelFade(){ for(r = 0; r < NUMPIXELS;
r++) { pixels.setPixelColor(r, Wheel((r+m) & 255)); } m++; if(m >= 256)
{ m = 0; } oldpos = newPos; } void rgbWheelMoving(){ for(r = 0; r <
NUMPIXELS; r++) { pixels.setPixelColor(r,Wheel(((r * 256 /
pixels.numPixels()) + m) & 255)); } m++; if(m >= 256*5) { m = 0; }
oldpos = newPos; } void pixelFadeStay(){ int puffer = 0, value = 0;
if(menu == 0){ value = newPos; } else if(menu == 2){ value = mode -
NUMPIXELS*4; } if(value != -1 && (menu == 2 || menu == 0)){
pixels.fill(pixels.Color(0,0,0)); for(int i = 0; i<3;i++) { puffer = value+i;
if(puffer > NUMPIXELS-1){ puffer = i-1; } pixels.setPixelColor(puffer,
pixels.Color(color[row][0],color[row][1],color[row][2])); } oldpos = value;
} } void clicked() { //Select Button if(menu == 0){
Consumer.write(MEDIA_VOL_MUTE); brightness = 255;
pixels.fill(pixels.Color(color[row][0],color[row][1],color[row][2]));
delay(700); } else if(menu == 1){ //leave the color-selection menu and
select the color Serial.println("leave with selected color");
EEPROM.update(0,row); menu = 0; pos = 0; newPos = 0; oldpos = 0;
oldPosition = -999; delay(100); } else if(menu == 2){ //leave the mode-
menu and select the mode Serial.print("leave with selected mode: ");
Serial.println(mode); EEPROM.update(1,mode); pixels.clear(); menu = 0; r
= 0; m = 0; pos = 0; newPos = 0; oldpos = 0; oldPosition = -999;
delay(100); } } void longPress() { //Menu button menu++; //jump into the
next menu up pos = 0; newPos = 0; oldpos = 0; brightness = 255;

```



```

resetLEDs(); if(menu == 1){ lightAll(color[row][0],color[row][1],color[row]
[2]); } else if(menu == 2){ row = EEPROM.read(0); delay(100); } if(menu
>= 3){ menu = 0; mode = EEPROM.read(1); r = 0; m = 0; pos = 0;
newPos = 0; oldpos = 0; oldPosition = -999; } Serial.print(menu);
Serial.println("LongClick"); } int TurnEncoder(int len) { //The function that
reads the data from the encoder relative to the given size. You can put in
the len as the max number of points for one rotation long newPosition =
myEncoder.read(); //read the position of the encoder if (newPosition !=
oldPosition){ //compare it to the old position if(newPosition > oldPosition){
//if the wheel turns left if(turnRight){ //turnLeft/Right are for reducing the
number of points by half so you have to pass two points to get one,
because the encoder roation points are sometimes pretty small pos++;
if(pos == len){ pos = 0; } turnLeft = false; turnRight = false; } else {
turnRight = true; } } else { if(turnLeft){ pos--; if(pos < 0){ pos = len - 1; }
turnRight = false; turnLeft = false; }else{ turnLeft = true; } } oldPosition
= newPosition; return pos; } return -1; } uint32_t Wheel(byte WheelPos) {
//From the Adafruit Libyary a function for a rgb circle WheelPos = 255 -
WheelPos; if(WheelPos < 85) { return pixels.Color(255 - WheelPos * 3, 0,
WheelPos * 3); } if(WheelPos < 170) { WheelPos -= 85; return
pixels.Color(0, WheelPos * 3, 255 - WheelPos * 3); } WheelPos -= 170;
return pixels.Color(WheelPos * 3, 255 - WheelPos * 3, 0); } void
resetLEDs(){ pixels.clear(); } void lightAll(int r, int g, int b) {
pixels.fill(pixels.Color(r,g,b)); }

```

Use the Volume Knob and have fun! If you want, you can add more color patterns to the script.

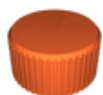
**If you like to, check out [my other designs](#) too!**

## Model files



**DrehreglerWithSymbol**

2 files



**drehreglerwithpause.stl**



**pause.stl**

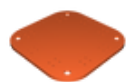


**PCB\_Assembly**

2 files



**pcb-holder.stl**



**stencil.stl**



**topusb-c.stl**



**bottomusb-c.stl**

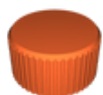


**transfuser.stl**



**transfuserdurchgehend.stl**





**drehregler.stl**

## Other files

**volumeknobcode.ino**

## License

This work is licensed under a  
**Creative Commons (4.0 International License)**



**Attribution—Noncommercial—Share Alike**

- 
- ✗ | Sharing without ATTRIBUTION
  - ✓ | Remix Culture allowed
  - ✗ | Commercial Use
  - ✗ | Free Cultural Works
  - ✗ | Meets Open Definition