SOS Flashlight

John Poirier, Spencer Hart
UVM Department of Electrical Engineering

Table of Contents

Description

This project required the construction of a light source with two intensity levels and an SOS signal function. The materials allowed include two AAA batteries and two white LEDs that must be operated in series, all controlled by a single push button. These components are to be connected on a custom-designed PCB prototype. Additionally, there must be an assessment of the flashlight lifetime in SOS and idle modes, and the idle mode must be optimized to use the least amount of power possible.

Solution

System

Our approach for this project was to break it into sections that build off each other. First, we built the Buck-Boost circuit, which allows us to operate two series LEDs from a single 3.3V source, as shown in Figure 1. Using this circuit, we connected a function generator and tested values for period and frequency to find when the LEDs were bright and dim. Using an oscilloscope, we could visualize different nodes of the circuit, as shown below.

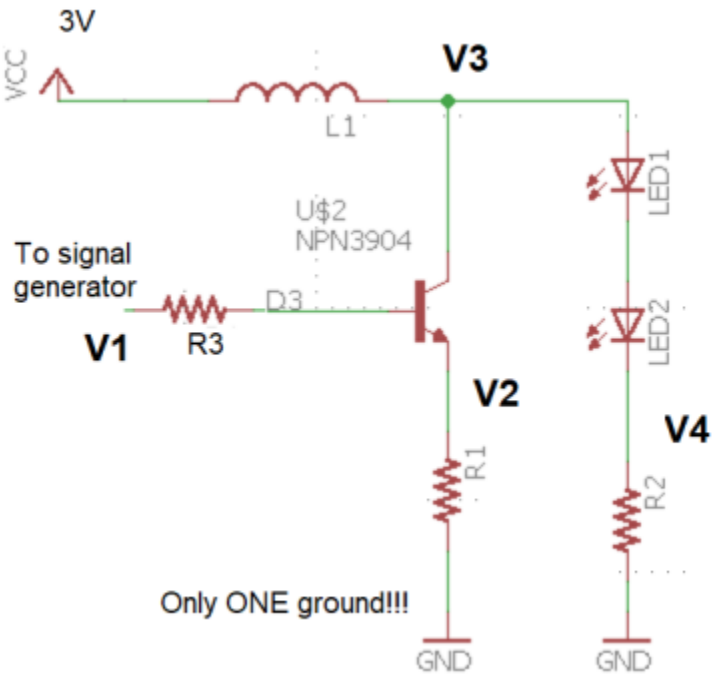


Figure 1: Buck-Boost Circuit

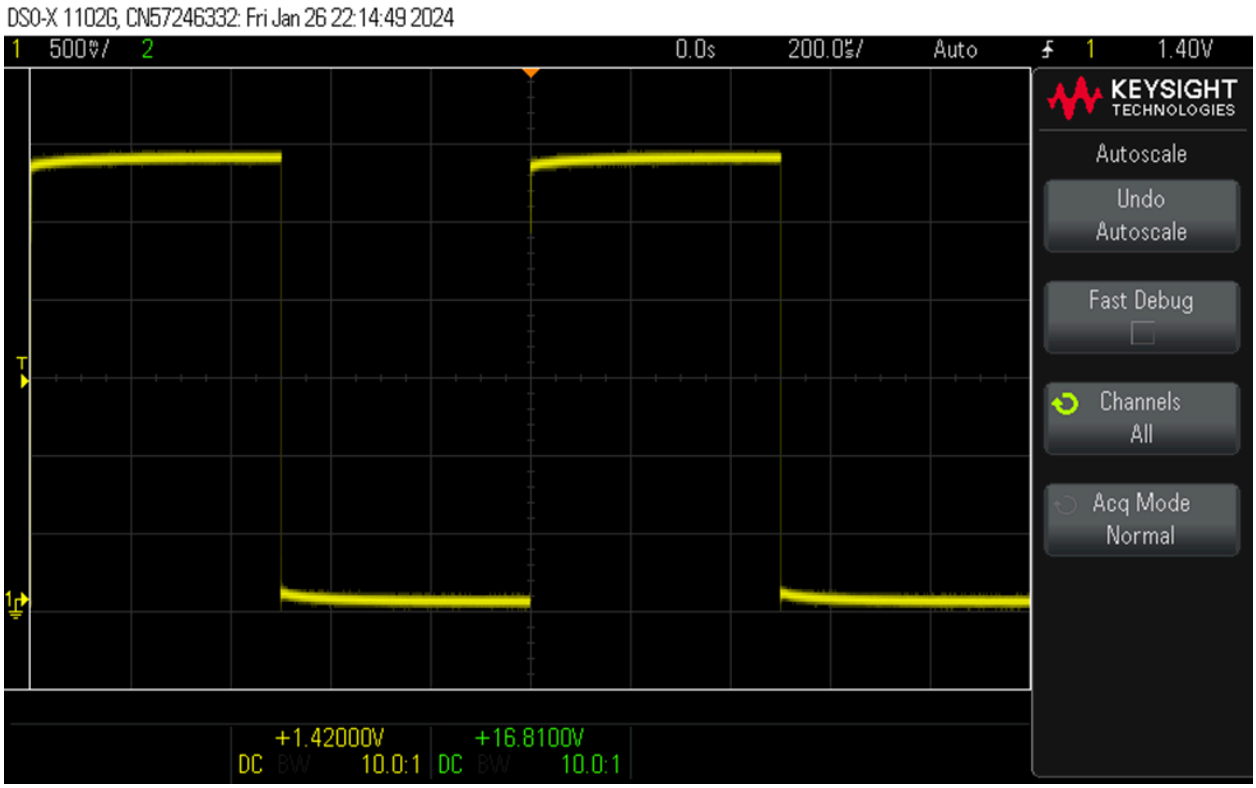


Figure 2: Voltage at V1 (See Figure 1)

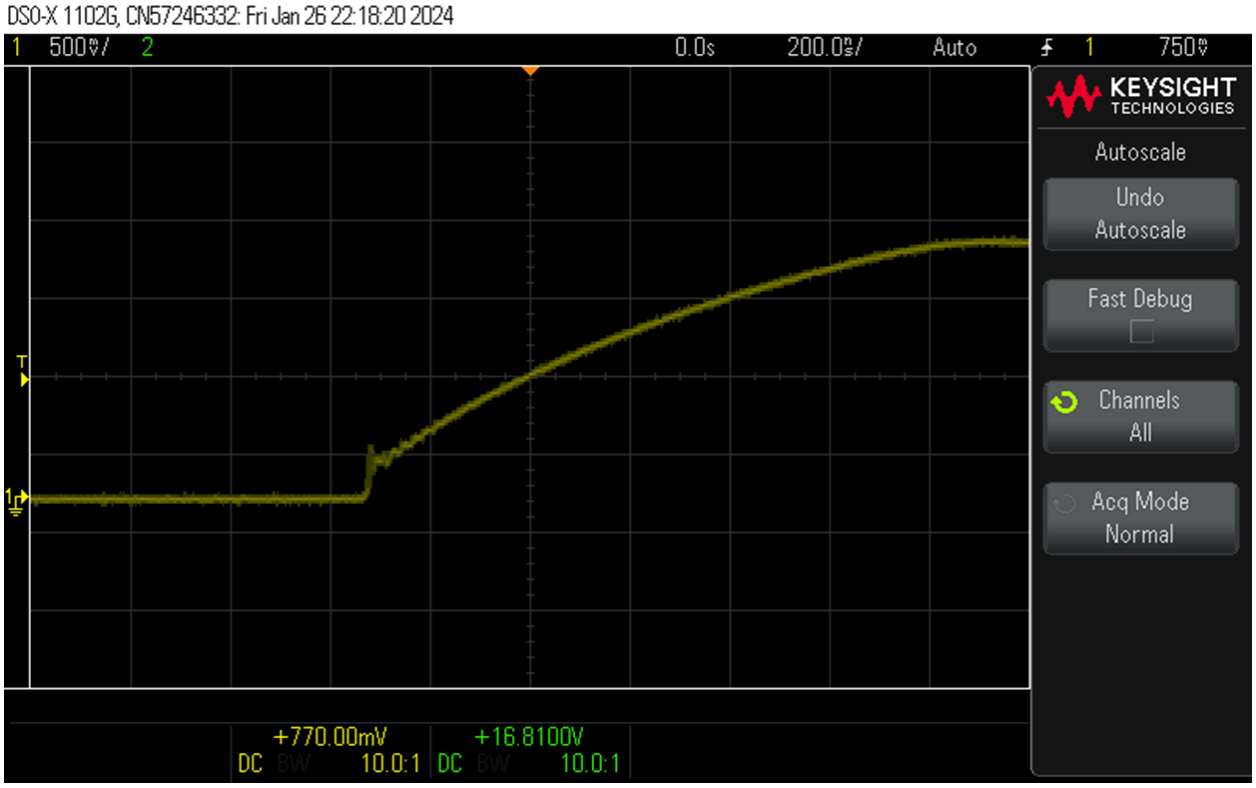


Figure 3: Voltage at V2 (See Figure 1)

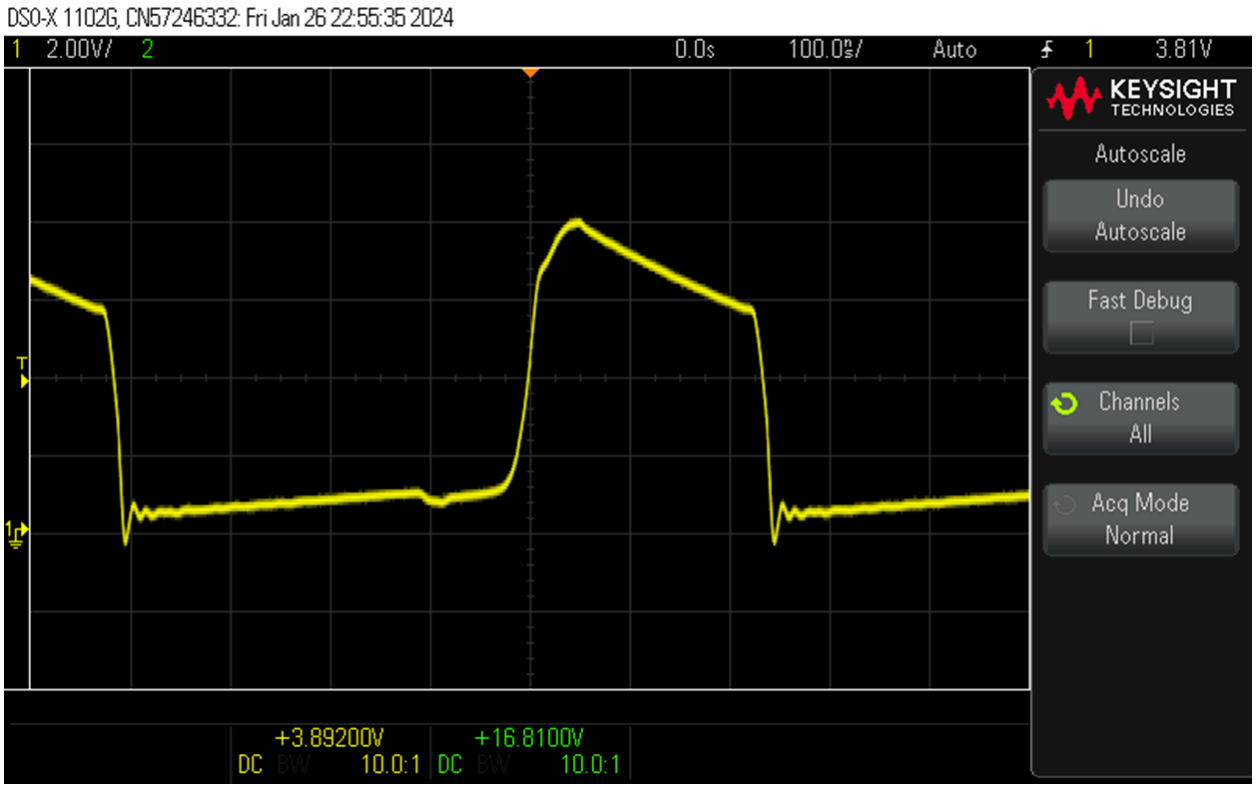


Figure 4: Voltage at V3 (See Figure 1)

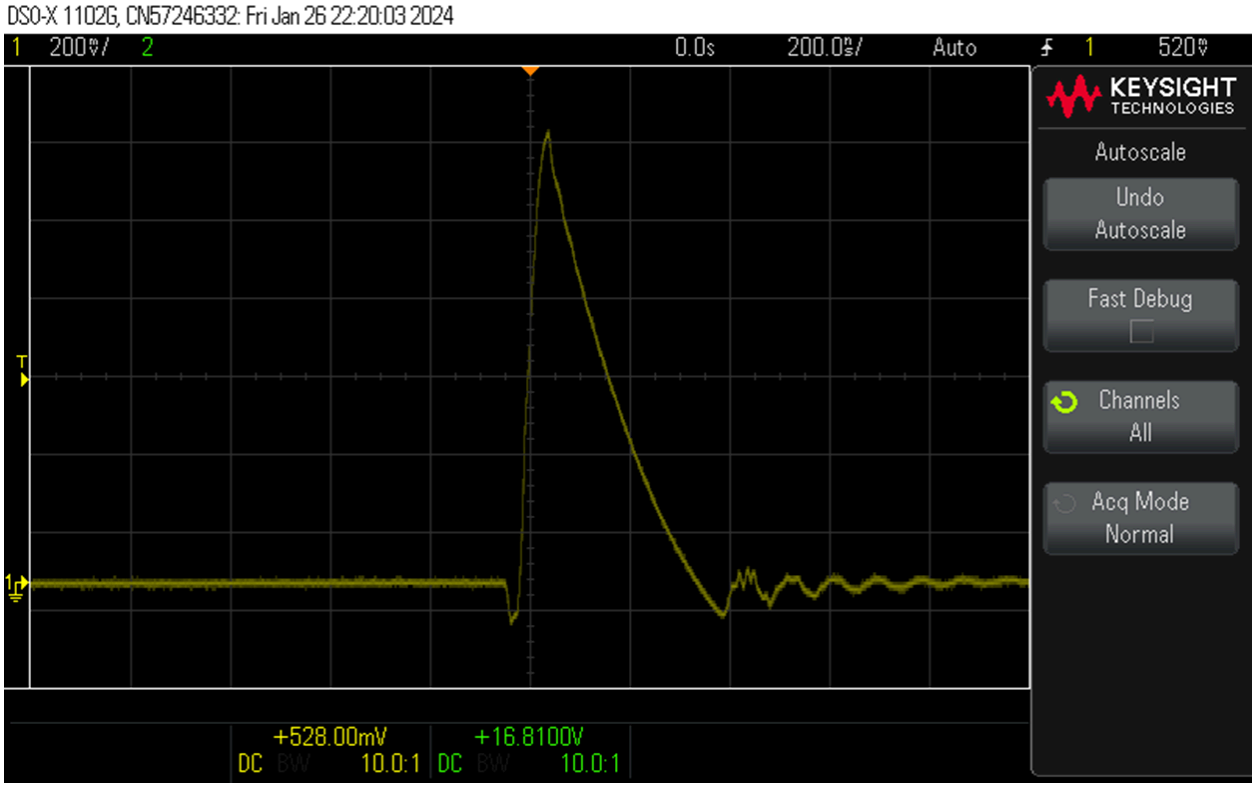


Figure 5: Voltage at V4 (See Figure 1)

Next, we worked on integrating the Arduino Metro Mini to control the circuit. We programmed the MCU to provide a simple on-off function for the flashlight and connected it to the circuit. Once that was working, we designed the PCB schematic and layout, optimizing for space while also keeping it practical to hold and use. While we were waiting to receive the PCB, we designed a new code for the Arduino that included bright, dim, SOS, and low-power modes. Then, once we had our PCB, we soldered the components to the board and were left with a functioning prototype SOS flashlight. Finally, we optimized the code for timings on the SOS signal and low-power mode, then measured data to calculate the expected lifespan in SOS and low-power modes.

Key Materials

The components expressly required by this project include two AAA batteries, two LEDs, and a push button. An Arduino Metro Mini was used to control the circuit.

Component	Cost
AAA Batteries (2)	$4.88 (Amazon)
White LEDs (2)	$1.16 (DigiKey)
Push Button	$0.10 (DigiKey)
Arduino Metro Mini	$14.95 (DigiKey)

The full Bill of Materials can be found in the Appendix.

Cost Analysis

This project proves to be relatively inexpensive, although the prototype is much more costly to design than an individual model of a mass-produced version would be, as buying components in bulk is much more cost-effective. The total cost for our prototype was $28.46, as calculated using the full Bill of Materials, excluding shipping costs and fees.

Schedule

Initially, we created a Gantt Chart to have a timeline of when certain tasks would be accomplished. We were able to follow this timeline effectively, never straying too far from a task's scheduled time of completion. At the time of writing this report, we still have to complete a few remaining loose ends and final calculations (ex. comment code, complete a battery lifetime assessment, etc.). We expect to get these completed by the final presentation.

Below is our initial Gantt Chart, followed by our updated Gantt Chart.

Figure 1. Shows our initial Gantt Chart.

SOS Headlamp	1/26/2024	2/2/2024	2/9/2024	2/16/2024	2/23/2024	3/1/2024	3/8/2024
Building circuit							
Integrating Arduino							
PCB design							
Ordering PCB							
Creating code							
Soldering PCB							
Testing code							
Finalizations							
Final presentation / report							

Figure 2. Shows our updated Gantt Chart.

SOS Headlamp	1/26/2024	2/2/2024	2/9/2024	2/16/2024	2/23/2024	3/1/2024	3/8/2024
Building circuit							
Integrating Arduino							
PCB design							
Ordering PCB							
Creating code							
Soldering PCB							
Testing code							
Finalizations							
Final presentation / report							

Results

Buck-Boost Circuit Design

For our circuit design, we followed the example of a basic buck-boost circuit given during the lecture period. The schematic for this circuit is shown in Figure 1. We built this circuit on a breadboard using a function generator as our power source, creating a prototype circuit to be built upon. Then, we integrated an Arduino Metro Mini to control the circuit instead of the function generator.

Programming/Integrating MCU

To code our Microcontroller Unit (MCU), we based our foundation on the example code provided by Dr. Gallagher. Then, we began working on implementing the SOS function and the low-power mode function. We decided how we wanted the additional code to be implemented, wrote pseudo code to visualize the flow of functionality, and then implemented the code.

PCB Schematic and Layout

When designing our PCB, we began with setting up the schematic based on the physical circuit we built. The majority of this portion was simply transferring the design into a digital format. We had to measure the components to ensure we had the correct footprints so that when it came to designing the layout, all our dimensions were correct. We also added a screw terminal to provide a way to connect the AAA batteries to the Arduino for power. The schematic can be found in the Appendix.

After we had set up our schematic, we designed the PCB layout. Our goal in this was to minimize the space used while maintaining a usable form factor. Once we had our layout, we organized the ratsnest, connected the components, and did our copper pours, with one layer being a 3.3V net and the other layer being a ground net. The layout can be found in the Appendix.

Assembling PCB

Once we received our PCB designs from OSHPARK, we collected our materials from our prototype circuit built on a breadboard and transferred them onto the PCB. We soldered the components into place and were left with a complete circuit.

Low-Power Mode

For low-power mode, we utilized the example code provided by Dr. Gallagher. We first studied how the code functioned and how and when low-power mode would be active in the circuit. We set our interrupt pin as the push button and made it so the flashlight would be sleeping/idling/in low-power mode when in the “Off” state (state 1). This state is reached any time the flashlight is not in the bright, dim, or SOS settings.

Validation

With a fully functioning flashlight with bright, dim, SOS, and low-power modes, the last task was to take measurements of power consumption. To do this, we found the current draw for each of the modes of the flashlight. In the bright mode, about 65 milliamps are drawn. In the dim mode, it decreases to roughly 9.7 milliamps drawn. For the low-power mode, we were able to get the current draw down to 0.72 milliamps. The SOS mode fluctuates between the bright mode current draw and an off-setting current draw, where the LEDs are off, but low power mode is not active. In this setting, the flashlight draws about 9 milliamps.

Lifetime Assessment

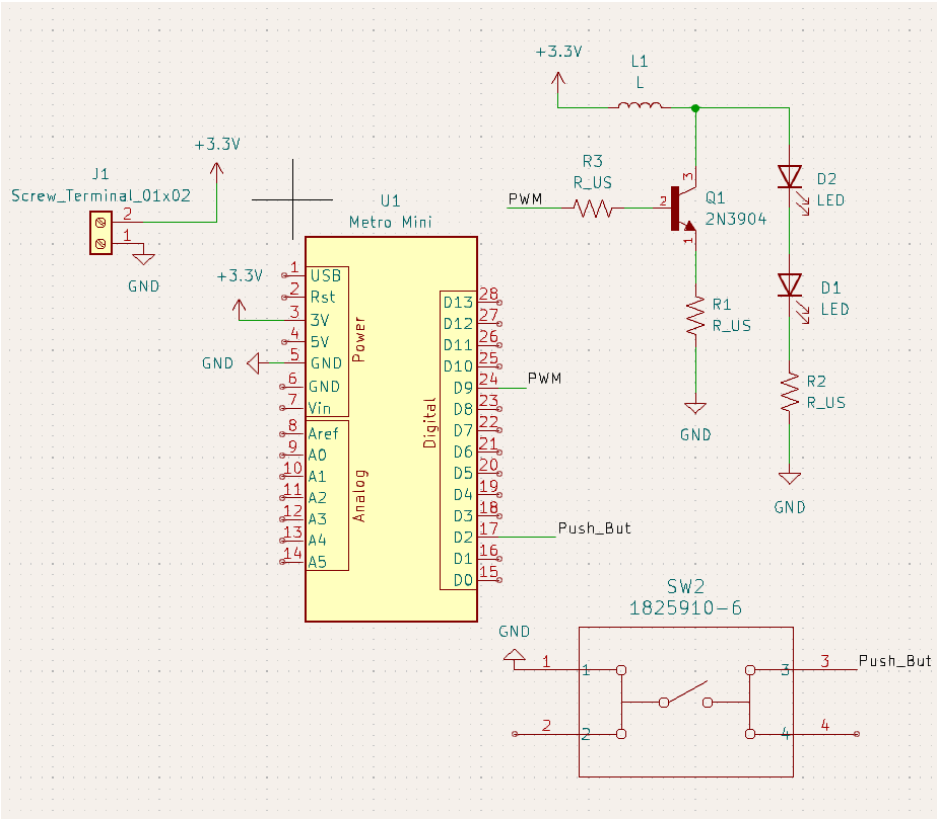
According to Microbattery, One Lithium AAA battery has a capacity of roughly 600 milliamp-hours (mah), so with two AAA batteries, we get a capacity of 1200 mah. Using this, we can figure out how long the flashlight can last off of two AAA batteries in low-power mode and in SOS mode. In low-power mode, by dividing the AAA batteries' capacity by the current draw, we can find that the flashlight can last 1667 hours or about 70 days. We can use the same method to find the lifespan of the bright and dim modes, resulting in 18 hours and 124 hours, respectively. To find how long the flashlight can last in SOS mode, we need to find the proportion of its time that is spent in the bright setting, and the portion spent in the off setting. Based on the source code found in the Appendix, we can calculate that a full cycle takes about 3.05 seconds and that 0.75 seconds of the full time is spent in the bright mode, leaving 2.3 seconds in the off setting. Using the draw in these two modes, we can find that the average current draw in SOS mode is roughly 22.6 milliamps. With this average current draw, we can find a total SOS lifespan of 53 hours and 5 minutes.

Appendix

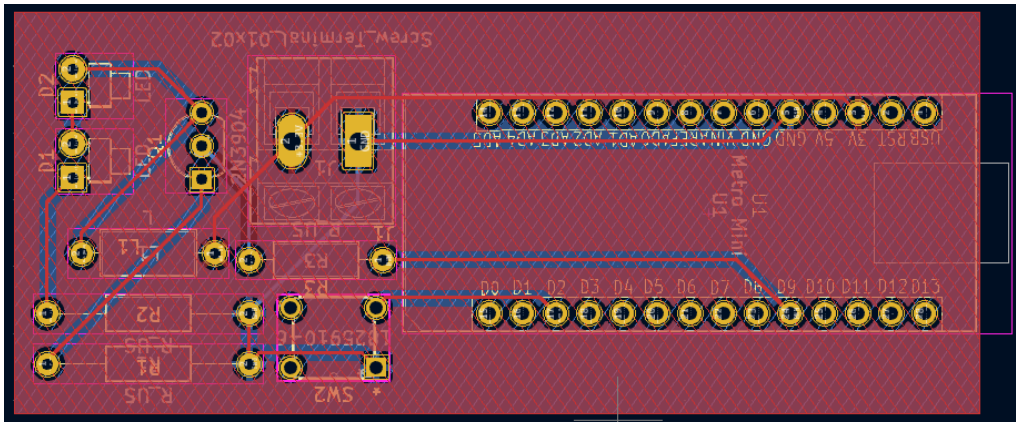
Full Bill of Materials

Component	Cost
AAA Batteries (2)	$4.88 (Amazon)
White LEDs (2)	$1.16 (DigiKey)
Push Button	$0.10 (DigiKey)
Arduino Metro Mini	$14.95 (DigiKey)
Resistors (3)	~$0.30 (DigiKey)
Transistor	$0.35 (DigiKey)
Inductor	$0.27 (DigiKey)
Screw Terminal	$0.70 (DigiKey)
PCB	$5.75 (OSHPARK)

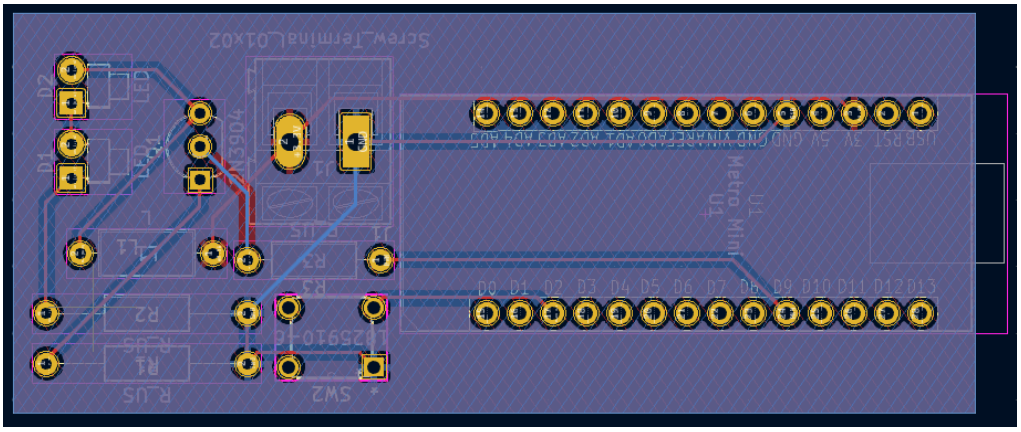
Full Schematics and Layout



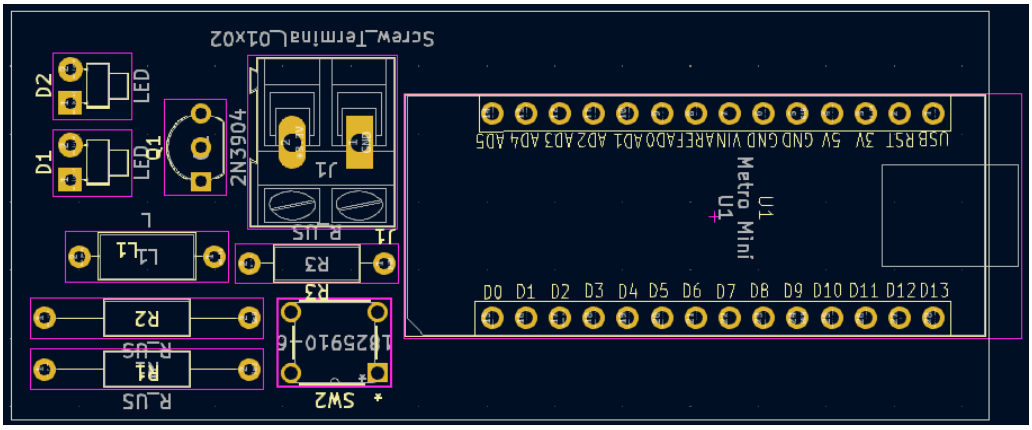
Schematic



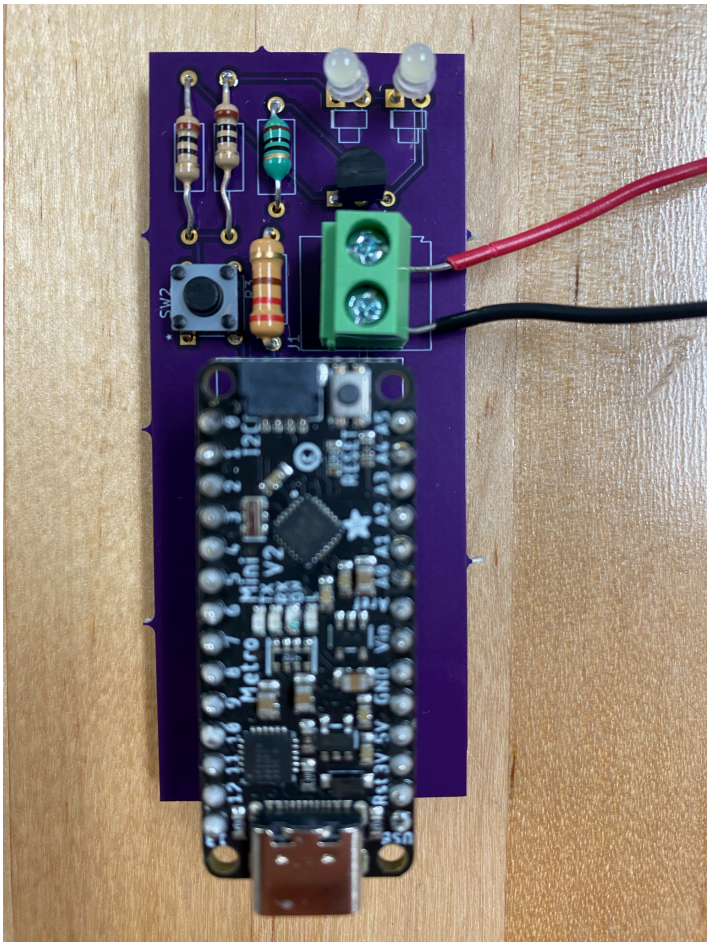
3.3V Net



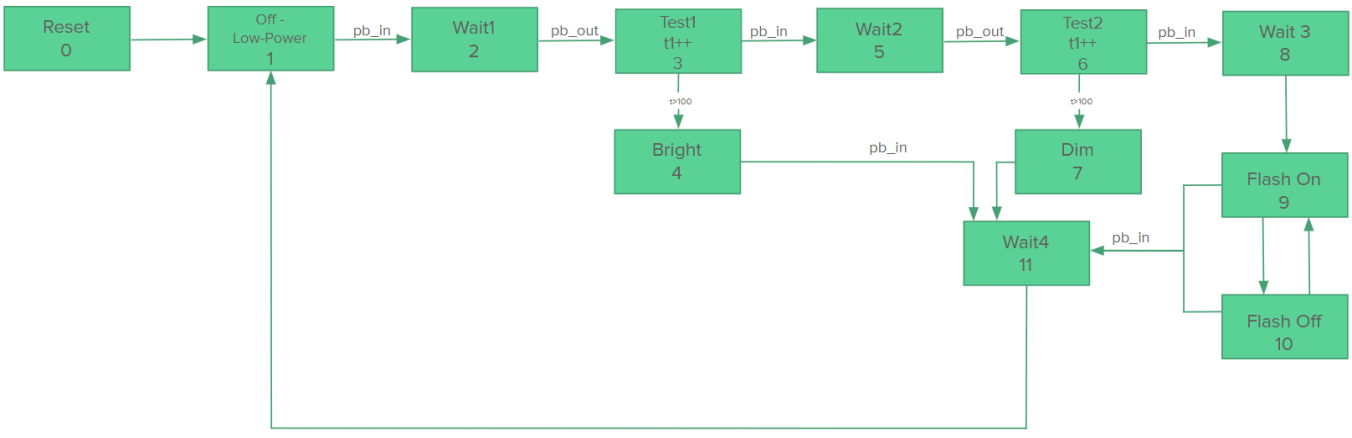
Ground Net



No Net Selected



Final Prototype



State Machine Diagram

Source Code

```
/*
SOS Flashlight
Created by Spencer Hart and John Poirier
Collaborated with Dr. Matthew Gallagher

This program is intended to control the buck booster flashlight. It will generate a
pwm signal that can be used to pulse the buck booster circuit. A single input push button
shall control what state the flashlight is in. This program can be

turned on:  one fast button push    >>>bright
            two fast pushes         >>>dim
            three fast pushes       >>>SOS
turned off: one button push

*****************************************************************
Functions:
main loop simply reads the pin and calls the FSM

void blinkBright(void)  maximizes the light our circuit can generate from the leds
                        by charging up the inductor to its maximum energy and discharging as
                        fast as possible. Number of loops optimized to require about 5ms for
                        one execution of this function.

void blinkDim(void)     maximizes the light our circuit can generate in each pulse but has a lower
                        frequency to make the leds appear dim. Number of loops optimized to require about 5ms for
                        one execution of this function.

void FSM1(void) our FSM that controls the flashlight

                        states        name
                        case 0: //    reset
                        case 1: //    Off
                        case 2: //    Wait1
                        case 3: //    Test1
                        case 4: //    bright
                        case 5: //    Wait2
                        case 6: //    test2
                        case 7: //    dim
                        case 8: //    wait3
                        case 9: //    SOS on
                        case 10://    SOS off
                        case 11: //   wait4

*********************************************************************
Input Output
*********************************************************************


Use the two three pin headers on which to attach your board.
You should supply your own power on your board and share only
signal and ground
*********************************************************************
Variables
*********************************************************************
global buttonState records whether the button is in or not and is read from the FSM to determine
whether the state should be exited.
```

```
 */

/*************************************************************/
/**************Defines Here************************/
#include <myTimer.h>    // Make sure myLibs is installed

#include <LowPower.h> //LowPower library

#define  PW  20
#define PER 50   //  Maximum 32000


#define PBPIN 2
#define LEDPIN 9

#define highTime1 25
#define lowTime1 25
#define lowTime2 200

const int wakeUpPin = 2; //declaring push button as wake-up pin

int SOS_flag = 1; //creating a flag for SOS state-tracking
int SOS_counter = 0; //creating a counter for SOS pattern
int dot_ON_time = 50; //setting on-time for dots
int dot_OFF_time = 50; //setting off-time for dots
int dash_ON_time = 150; //setting on-time for dashes
int dash_OFF_time = 50; //setting off-time for dots
int dot_counter = 0; //creating a counter to keep track of dot cycles

/*************************************************************/
/******* Prototype Function declarations ******/
//*************************************************************/

void FSM1(int);    //main state machine


/*************************************************************/
/**************Global Variables Here************************/
/*************************************************************/




void setup() {
  // initialize the LED pin as an output:
  //pinMode(LEDPIN, OUTPUT);
  //initialize pb as interrupt pin
  pinMode(wakeUpPin, INPUT);
  // initialize the pushbutton pin as an input:
  pinMode(PBPIN, INPUT_PULLUP);
  initPWM(PER);    // Pin 9
   Serial.begin(9600);
}

void loop(){
   static int buttonState=0;   // variable for reading the pushbutton status
   buttonState=digitalRead(PBPIN);

   FSM1(buttonState);       //set to a new state depending on the state_opt
   variables
   delay(5);

  }

```

```
122
123
124 /*****************************************************************************/
125 /*****************   FSM1(void   ***********************/
126 /*****************************************************************************/
127 /*  This is the main state machine*/
128
129 void FSM1(int buttonState)
130 {
131   static int state=0;
132   static int t1=0;
133       Serial.println(state);
134       switch( state ){   //Switch specifies which variable controls the "case"
135               case 0: //   reset
136               {
137                /*    Do what the state does    */
138                 pwmPW_PER(0,PER);    // off
139                /*    Criteria to leave the state    */
140                state=1;
141                }
142
143              break;
144
145              case 1: //    Off
146               {
147                /*    Do what the state does    */
148                 pwmPW_PER(0,PER);    // off
149                 //sleep here, wakeup on interrupt
150
151                 attachInterrupt(0, wakeUp, LOW);
152
153                 // Enter power down state with ADC and BOD module disabled.
154                 // Wake up when wake up pin is low.
155
156
157                 LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
158
159                 // Disable external pin interrupt on wake up pin.
160                 detachInterrupt(0);
161
162                /*    Criteria to leave the state    */
163               if(buttonState==0){
164                 state=2;
165                 t1=0;
166               }
167
168               }
169             break;
170
171            case 2: //    Wait1
172               {
173                /*    Do what the state does    */
174
175                 pwmPW_PER(PW,PER);    // bright
176                 t1++;
177
178                /*    Criteria to leave the state    */
179               if(buttonState==1){
180                 state=3;
181                 }
182
183            }
184             break;
```

```
185         case 3: //   Test1
186             {
187               /*    Do what the state does    */
188               pwmPW_PER(PW,PER);
189               t1++;
190
191               /*    Criteria to leave the state    */
192               if(buttonState==0){
193                 state=5;
194                 t1=0;
195                 }
196               else if(t1>=100){   //0.5s
197                 state=4;
198                 }
199
200             }
201              break;
202
203         case 4: //    bright
204             {
205               /*    Do what the state does    */
206               pwmPW_PER(PW,PER);
207
208               /*    Criteria to leave the state    */
209               if(buttonState==0){
210                 state=11;
211               }
212
213             }
214              break;
215
216         case 5: //    Wait2
217             {
218               /*    Do what the state does    */
219               pwmPW_PER(PW,PER);
220               t1++ ;
221
222               /*    Criteria to leave the state    */
223               if(buttonState==1){
224                 state=6;
225               }
226             }
227              break;
228
229       case 6: //    test2
230             {
231               /*    Do what the state does    */
232               pwmPW_PER(PW,PER);
233               t1++;
234
235               /*    Criteria to leave the state    */
236               if(buttonState==0){
237                 state=8;
238               }
239               else if(t1>100){
240                 state=7;
241               }
242             }
243              break;
```

```
244         case 7: //    dim
245                 {
246                 /*    Do what the state does    */
247                 pwmPW_PER(PW,PER*20);
248
249                 /*    Criteria to leave the state    */
250                 if(buttonState==0){
251                   state=11;
252                 }
253               }
254               break;
255         case 8: //    Wait3
256                 {
257                 /*    Do what the state does    */
258
259                 pwmPW_PER(PW,PER*20);
260                 /*    Criteria to leave the state    */
261                 if(buttonState==1){
262                   state=9;
263                   t1=0;
264                 }
265               }
266               break;
267
268
269
270
271         case 9: //    SOS on
272                 {
273                 /*    Do what the state does    */
274                 pwmPW_PER(PW,PER);
275                 t1++;
276                 /*    Criteria to leave the state    */
277                 if(buttonState==0){
278                   state=11;
279                 }
280
281                 //Determines how long the leds will be bright in the SOS sequence
                    for the dots
282                 else if(SOS_flag==1){
283                   if(t1>dot_ON_time){
284                     t1=0;
285                     state=10;
286                   }
287                 }
288
289                 //Determines how long the leds will be bright in the SOS sequence
                    for the dashes
290                 else if(SOS_flag==-1){
291                   if(t1>dash_ON_time){
292                     t1=0;
293                     state=10;
294                   }
295                 }
296               }
297               break;
```

```
299     case 10: //    SOS off
300                 {
301                  /*    Do what the state does    */
302                  pwmPW_PER(0,PER);
303                  t1++;
304                  /*    Criteria to leave the state    */
305                 if(buttonState==0){
306                   state=11;
307                 }
308
309                 //Determines how long the leds will be off in the SOS sequence
                    for the dots
310                 else if(SOS_flag==1){
311                   if(t1>dot_OFF_time){
312                     t1=0;
313                     SOS_counter++;
314                     if(SOS_counter==3){
315                       //This will mark the end of one SOS cycle, restarting the
                          cycle
316                       if(dot_counter==1){
317                         SOS_counter = 0;
318                         dot_counter = 0;
319                         delay(1000);
320                       }
321                       else{
322                         //flip the flag so the dots become dahses
323                         SOS_flag = -SOS_flag;
324                         SOS_counter = 0;
325                         //keep track of dot cycles to know when one full SOS
                            cycle is complete
326                         dot_counter++;
327                         delay(500);
328                       }
329
330                     }
331                     state=9;
332                   }
333                 }
334                 //Determines how long the leds will be off in the SOS sequence
                    for the dashes
335                 else if(SOS_flag==-1){
336                   if(t1>dash_OFF_time){
337                     t1=0;
338                     SOS_counter++;
339                     if(SOS_counter==3){
340                       //flip the flag so the dashes become dots
341                       SOS_flag = -SOS_flag;
342                       SOS_counter = 0;
343                       delay(500);
344                     }
345                     state=9;
346                   }
347                 }
348               }
349               break;
```

```
351         case 11: //    Wait4
352                 {
353                  /*    Do what the state does    */
354
355                  pwmPW_PER(PW,PER);
356                  /*    Criteria to leave the state    */
357                  if(buttonState==1){
358                   state=1;
359                  }
360               }
361                break;
362
363              default:
364                break;
365           }
366       }
367
368
369       void wakeUp()
370   {
371     // Just a handler for the pin interrupt.
372   }
373
```

Sources

Dr. Matthew Gallager, https://brightspace.uvm.edu/d2l/home/63830

Microbattery, “AAA Battery”, https://www.microbattery.com/alkaline/aaa.html