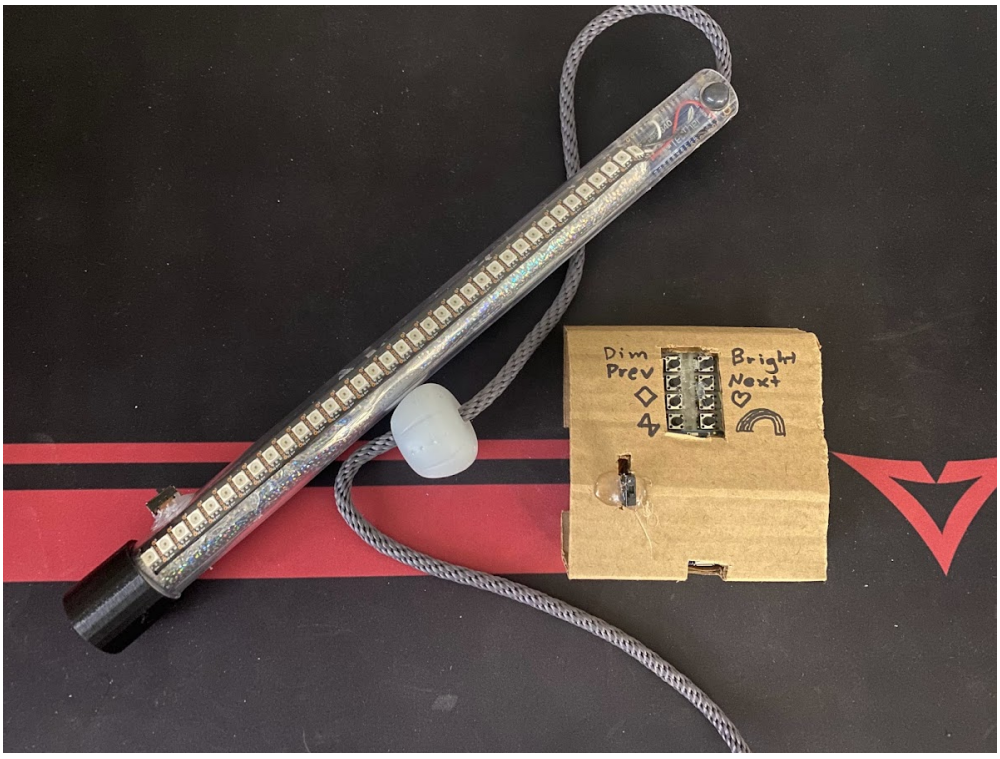
Tonlé Bloomer
EN.525.743.8VL.FA22
Embedded Systems Development Lab

SpinDance : Audience Interacting Poi

Final Design Report - December 12th 2022

Overview

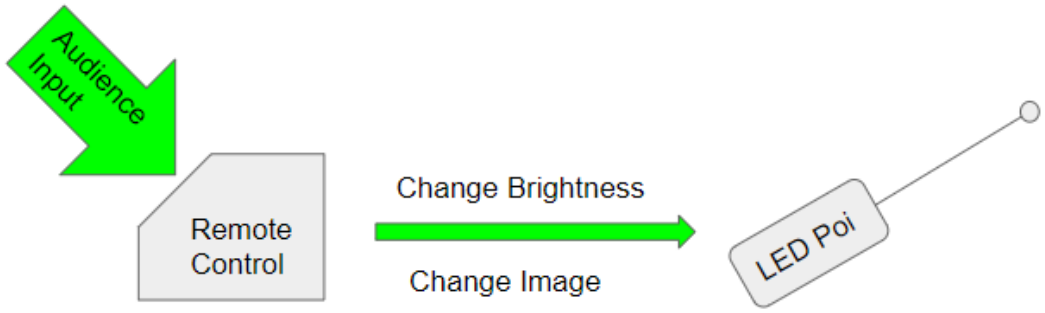
SpinDance consists of 2 devices: 1 LED poi and a user-friendly remote control. Poi are dancing implements, typically consisting of a handle, string or chain, and a tethered weight that can be spun in a variety of patterns. Most people may be familiar with them from fire dancing in Hawaii, but LED versions also exist. This project integrates two poi that use LED strips to display visual persistence images as they are spun with a remote control that allows the audience to change the image and brightness without ever stopping the performance. An image of both final constructions are given below.



Spindance final construction.

Project Description

The LED Poi will display persistence images stored in flash memory. There are a total of 54 patterns stored in flash memory. On initial power on the poi default to the first image in the sequence (Adafruit logo), and when the audience selects a different pattern on the control box the poi change to that image. The poi mechanical design and software starting point were heavily informed by reference (1), which is an Adafruit learning tutorial that details building similar poi that communicate via an infrared receiver. In this project instead of an infrared remote I constructed a Bluetooth Low Energy remote to communicate to the poi. Both the remote control and the LED poi run off of lithium polymer batteries, include battery charge management, and recharge via microUSB. The remote control software was built on the Central BLEUART Adafruit example in reference (2).



This system is entirely self-contained (does not fit into a larger system). The two components can be carried anywhere together and have no external dependencies.

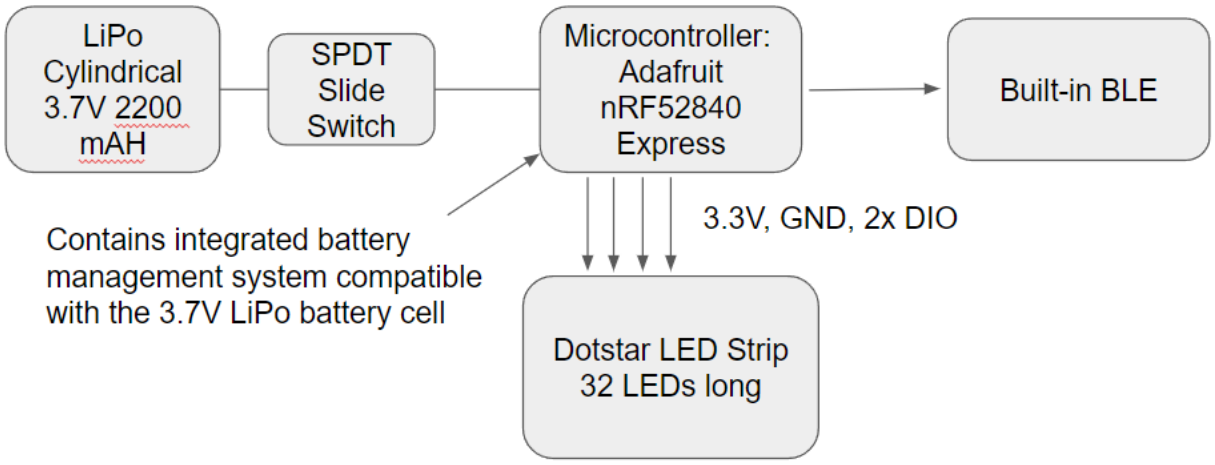
Functional Description

I include separate functional descriptions for each component.

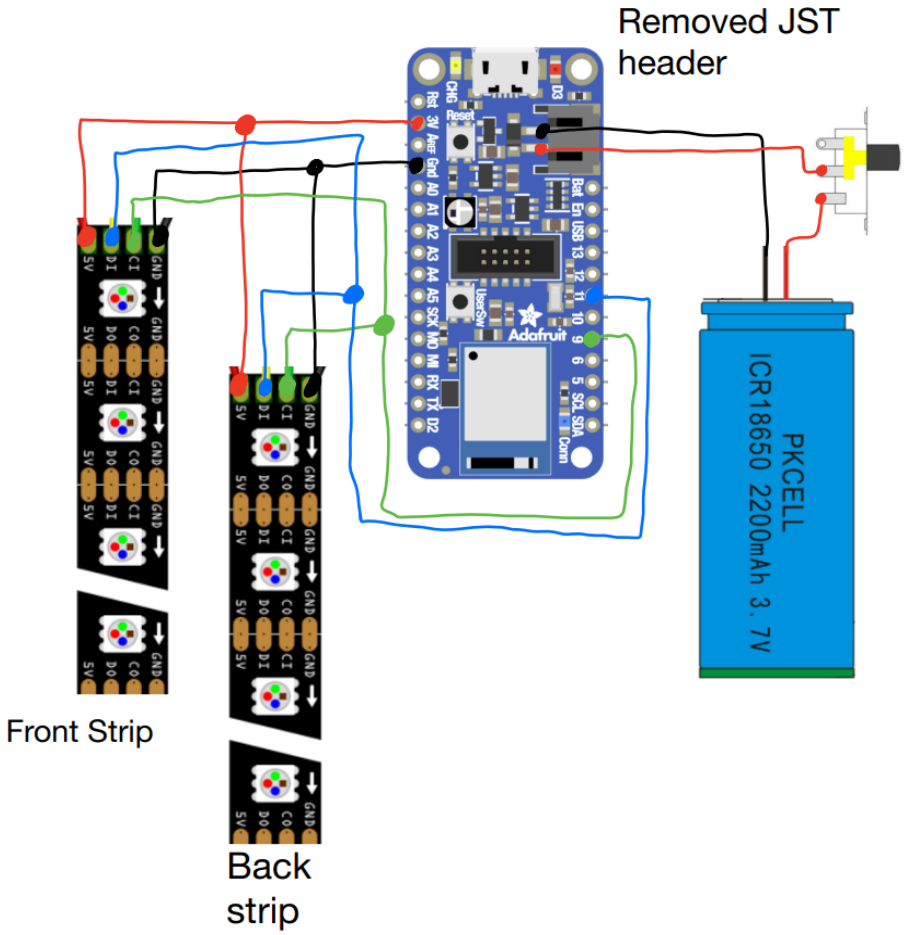
1x Poi:

The brains of the poi is the Adafruit Feather nRF52840 Express (3). This microcontroller includes a 32-bit ARM Cortex-M4F with 1 MB of flash memory and 256 KB of SRAM. It includes a 500 mA 3.3V regulator and an integrated LiPo charger that is compatible with the 3.7V 2200 mAH cylindrical battery that the poi will be powered on. The nRF52840 Express also includes built-in Bluetooth Low Energy. A simple SPDT slide switch will turn the component on and isolate the battery when off. The Feather will drive the Dotstar LED strip which requires power, gnd, data, and clock lines on separate digital pins. While the Dotstar LEDs run on 5V logic, they operate perfectly fine at 3.3V in this application.

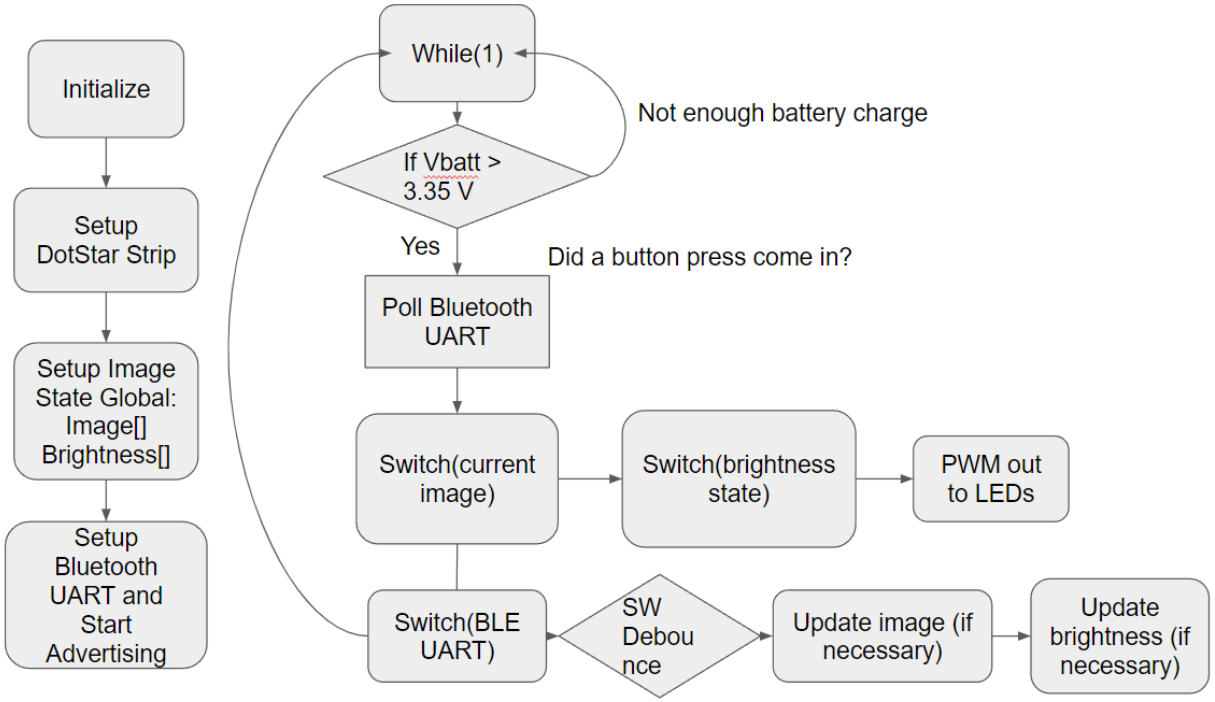
Flowchart for the poi, showing all interfaces are compatible, is below:



Wiring schematic:



The flow diagram for the poi software is below:



To modify the behavior of the poi (or add additional features in the future), the first software block of note is line 228:

```
  uint8_t len = readPacket(&bleuart, 1);

  if (len > 0){ //If I've received a packet

    if (packetbuffer[1] == 'B') { //If packet type is button press

      button1 = packetbuffer[2] - '0';

      button2 = packetbuffer[3] - '0';

    }

  }
```

This breaks down the UART string received over BLE into two buttons that determine switch state logic in lines 309 onwards:

```
switch(button1) {

        case 1:

          switch(button2) {
```

```
            case 0:
              autoCycle = !autoCycle;
              break;
            case 1:
              if(bLevel < (sizeof(brightness) - 1))
              strip.setBrightness(brightness[++bLevel]);
              break;
            case 2:
              if(bLevel)
                strip.setBrightness(brightness[--bLevel]);
              break;
```

These switch statements modify the poi behavior based on the UART string received. For example, if the UART string received is !B15, then the 1 will be extracted out as button 1 and the 5 will be extracted out as button 2. In my switch statement logic this button combination sets the current image to image 30:

```
            case 5:
              imageNumber = 30;
              imageInit();
              break;
```

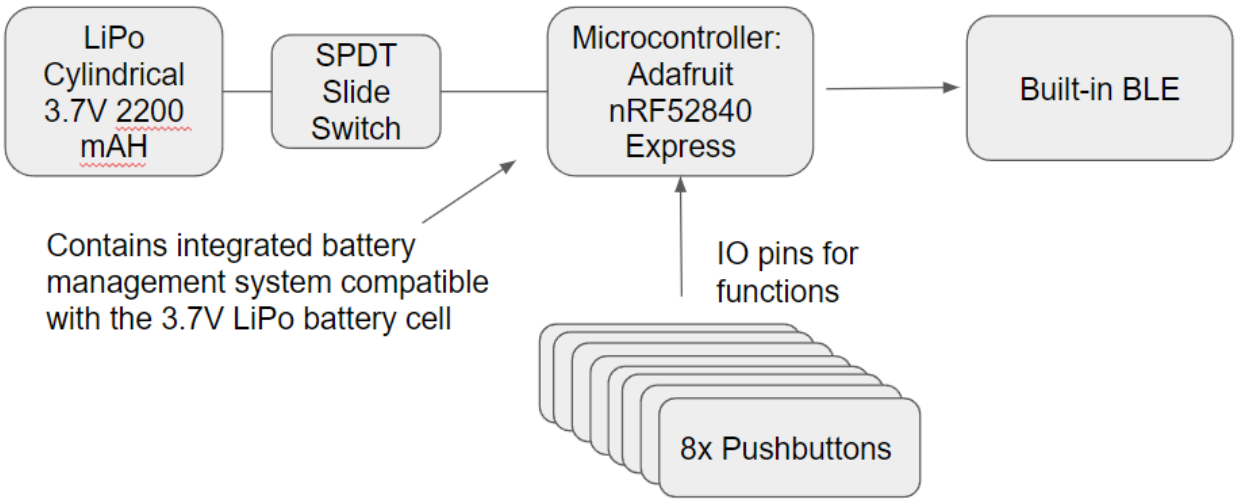
In this way an almost unlimited number of behaviors could be devised for the poi based on UART strings (which in turn can be devised or set based on the remote control code).

Remote Control:

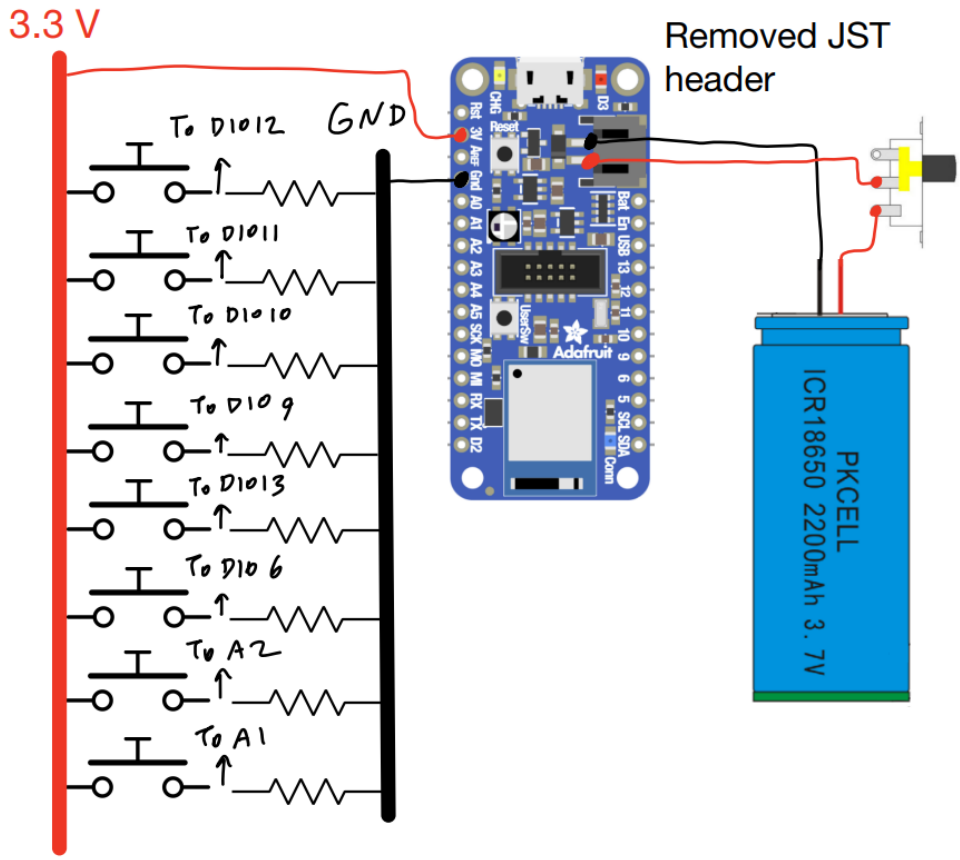
The remote control also uses an Adafruit nRF52840 microcontroller. The control box includes a LiPo 3.7V 2200mAh battery and built-in battery charge electronics. There are 8 separate push buttons with labels corresponding to 8 behaviors. The behaviors are detailed below:

Remote Control Functions
Increase Brightness
Decrease Brightness
Previous Image
Next Image
Go to Image 16 (Rainbow diamond pattern)
Go to Image 30 (Heart pattern)
Go to Image 41 (Diamond spade pattern)
Go to Image 50 (Simple Rainbow pattern)

Whenever the push buttons change state (software debounce), then the Feather sends a Bluetooth transmit for either a new brightness or an image. Flow diagram is given below, demonstrating all interfaces are compatible.

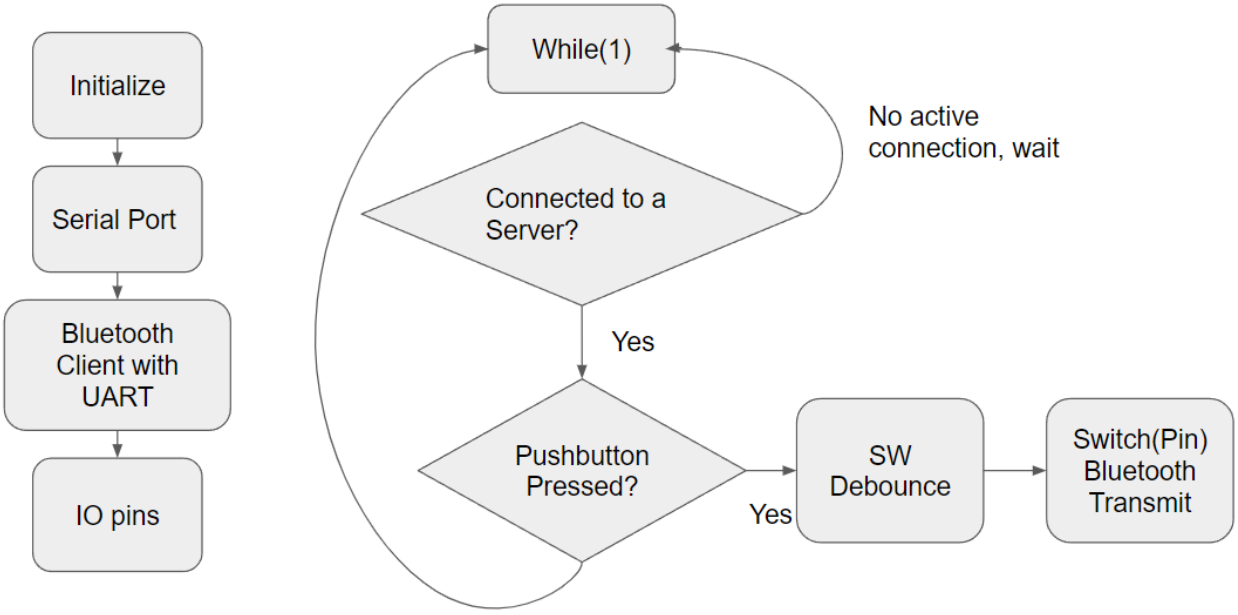


Wiring schematic:



The resistor value across all the resistors is 7.5kohms.

The flow diagram for the remote control software is given below:



To modify the behavior of the remote control (add features, change behavior) the code most important to pay attention to begins on line 219. Here the pins are first debounced, then read. The UART string out to the server (the poi) is determined based on the if/else logic that begins in line 220. Sending a command such as “!B16” asks the poi to perform the button 1 function 1 and button 2 function 6 behavior as described in the Poi component description. To modify the pins and debounce behavior simply change the global variables beginning on line 25.

The primary code of interest is included below:

```
215   if ( clientUart.discovered() )
216   {
217     // Discovered means in working state
218     char sendstr[4+1] = {0};
219     if((millis() - lastDebounceTime) > debounceDelay){
220       if(digitalRead(dimmer) == HIGH){
221         strcpy(sendstr, "!B12");
222         flag = 1;
223       }else if(digitalRead(brighter) == HIGH){
224         strcpy(sendstr, "!B11");
225         flag = 1;
226       }else if(digitalRead(lastImg) == HIGH){ ···
229       }else if(digitalRead(nextImg) == HIGH){ ···
232       }else if(digitalRead(pres5) == HIGH){ ···
235       }else if(digitalRead(pres10) == HIGH){ ···
238       }else if(digitalRead(pres16) == HIGH){ ···
241       }else if(digitalRead(pres25) == HIGH){ ···
244       }//end button poll
245       if(flag == 1){
246         clientUart.print( sendstr );
247         Serial.println(sendstr);
248       }
249       flag = 0;
250       lastDebounceTime = millis();
251     }//close button debounce if
```

By changing the remote control software in tandem with the poi software, an almost limitless number of behaviors could be implemented.

Interface Descriptions

The project interfaces are relatively simple, in that there’s no internet connectivity and the interfaces are standardized. Dotstar LEDs are designed to operate on a 5V clock and data bus, but will operate at 3.3V as in this design. Dotstar LEDs don’t have any special pin or timing

requirements, and have an active Arduino support library. Adafruit Feathers are able to establish client/server pairing over BLE. The 3.7V 2200 mAH battery are sufficient to drive the LEDs and the Bluetooth Low Energy radio long enough for many songs in a demonstration (light dancing). The battery management circuits in the Adafruit Feather are safe and appropriate for the cylindrical 3.7V battery I selected. There are also sufficient IO on the microcontrollers.

Image Conversion

While not part of the constructed system, the image conversion process is also important to include in this design document. The tutorial in reference (1) is heavily drawn on for the base software for the poi and poi design, and it includes an essential Python program for converting BMP or GIF images into a graphics.h file that the Arduino IDE compiles with the main code onto the Adafruit Feather board. See the git repository here: https://github.com/adafruit/Adafruit_Learning_System_Guides/tree/main/Kinetic_POV and look for the convert.py program under the convert directory. What this program does is essentially breaks the .bmp or .gif images into column-by-column arrays of RGB values for each pixel and stores and maps them in the graphics.h file. Since the poi are 32 LEDs long, an image height of 32 pixels can be fully displayed by the poi. Since this Python code needs to modify the header graphics.h file and then the code needs to be compiled and flashed to the Adafruit Feather poi board, I could not identify any way to add new images in real time.

Materials

Bill of Materials + Tools:

Name	Link	Quantity
Adafruit Feather nRF52840 Express	https://www.adafruit.com/product/4062	2
Poi Handle and Leash	https://flowtoys.com/knob-leash-smithy-pair	1
Battery	https://www.adafruit.com/product/1781	2
Poi End Cap	https://www.thingiverse.com/thing:1901778	1
SPDT Switch	https://www.adafruit.com/product/805	2
LED Strips	https://www.adafruit.com/product/2241	1 meter

Plastic Tubing	Specifications: 14” x 1” (inner diameter > ⅞”)	1
Wooden Dowel	Specifications: 8" x 5/8"	1
Holographic Vinyl	https://www.amazon.com/Holographic-Glossy-Rainbow-Silver-Adhesive/dp/B07XZ41DTL/ref=sr_1_5?keywords=holographic%2Bvinyl&qid=1664835408&qu=eyJxc2MiOiI2LjU4IiwicXNhIjoiNi4yOSIsInFzcCI6IjUuODcifQ%3D%3D&s=arts-crafts&sr=1-5&th=1	1
Resistors (7.5k - 10k)	Consumables	8
Breadboard/Protoboard	Consumables	1
Pushbutton switches	Consumables	8
Cardboard	Consumables	As Needed
Leaded Solder	Consumables	As Needed
Solder Iron	Tools	
Dremel	Tools	
Hot Glue + Glue Gun	Tools	As Needed
Hand Drill	Tools	
Hand Files	Tools	

Consumables and tools were available and used at my workplace’s makerspace. Solder stations, test equipment, and saws/dremel for cutting the wooden dowel and plastic tubing were also available there. Hot glue is the primary adhesive for the construction.

Development Plan and Schedule

The original schedule vs the work completed is compared below.

Date	Baseline Schedule	Actual Accomplished
October 10th	Components in-hand, tested for functionality (with up to 10/12th for all components)	There were a number of components outstanding, but all software was installed and LEDs were tested with Bluetooth

October 17th	Prototype poi (LEDs with MCU, no housing)	Incomplete prototype built. BLE demonstrated with Bluefruit phone app
October 24th	Prototype remote control (MCU with pushbuttons, no housing)	Had enormous issues assembling the poi with full mechanical housing.
October 31st	Have all components communicating over Bluetooth	Prototyped remote control with Arduino UNO and BLE UART add-on (ended up switching to Feather with built-in BLE later)
November 7th	Modify the image persistence software to operate via Bluetooth control following the flow diagrams	No Progress - Family Emergency
November 14th	Modify image persistence software to work with Bluetooth control	Fully constructed one LED poi (finally!). Control implemented via phone
November 21st	Thanksgiving Vacation (margin)	Attempted constructing second poi, but after several hours decided to descope second poi from project (with instructor concurrence)
November 28th	Documentation cleanup + demonstrating filming, video editing	Prototyped remote control with Adafruit Feather (replacing UNO and BLE UART add-on). Implemented code to establish client-server pairing between poi and remote. Sent commands via serial port to control poi
December 5th	Final Demonstration	Constructed remote control with pushbuttons and battery (no housing yet). Full functional prototype
December 12th		Added housing to remote control. Filmed demonstration video (in case of any issues in live demo)

Risk

There were two primary risks, although they were mitigated to the maximum extent possible at the beginning of the project. Attempting to integrate different open source repositories and projects always carries the risk that they may require much heavier modification than anticipated, costing schedule. My mitigation for this was to identify projects from “trusted” authors (mainly Adafruit learn). Also, the Bluetooth HC-05 module I originally baselined may have been too power hungry. The mitigation was that I ordered Adafruit Bluetooth LE SPI helpers in parallel in case I needed to pursue BLE (which I did, but through the Adafruit Feather rather than the LE SPI helpers).

Assembly Instructions

Development Environment:

I used the Arduino IDE because Adafruit provides a huge number of Arduino libraries. If not already installed the IDE GUI can be easily installed here: https://docs.arduino.cc/software/ide-v2

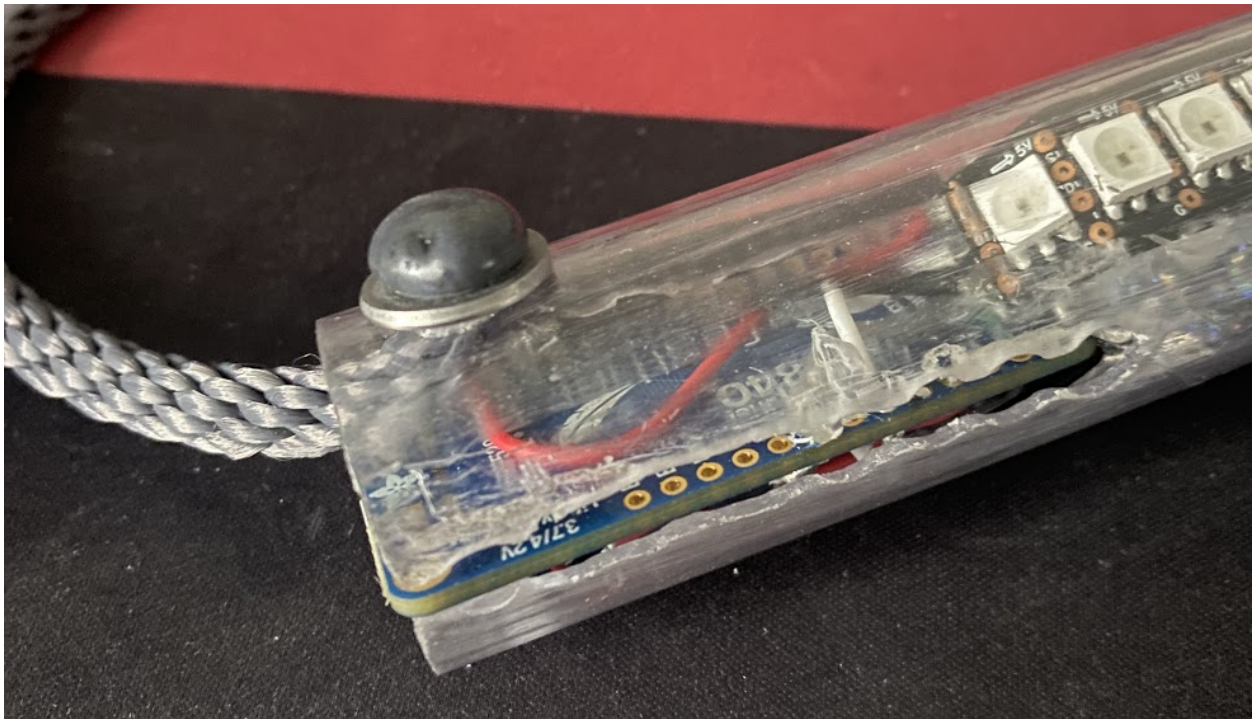
The libraries for the Adafruit Feather and Dotstar strips can be installed following the instructions in reference (3) and reference (4). Also pay attention in reference (3) to instructions on updating the bootloader.

All of the code I wrote for this project is available on my public github here: https://github.com/tonleb97/VisualPoi

Simply clone this repository. Under /supernova_poi/ the supernova_poi.ino file needs to be programmed to the poi's MCU. Under /ControlBox/ the central_bleuart.ino file needs to be programmed to the remote control's MCU.

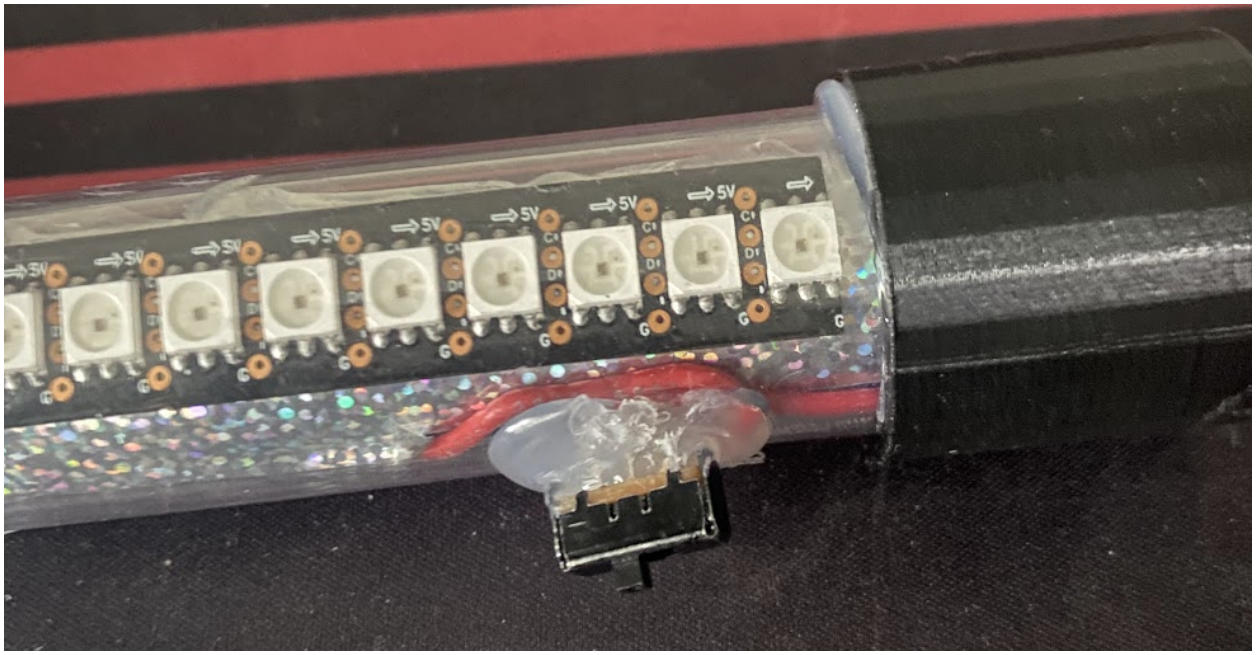
Wiring/mechanical assembly:

Although I won't provide step-by-step instructions for assembling the poi, much of the mechanical design is detailed in reference (1). Be sure to have the wiring diagram in the component description section completed **without the SPDT battery switch** before inserting the assembly into the poi. Additional lesson learned: the LED strip + MCU assembly did NOT fit in the plastic tubing I purchased, and I would HIGHLY recommend purchasing a tube with a larger inside diameter (>1⅛"). To complete my assembly I used a dremel to cut down the entire length of plastic tubing so I could pry it open to insert the electronics assembly. I also cut a window at the top of the tube to seat the Adafruit Feather:



This image also illustrates how the Poi handles are attached to the tube through a single drill hole.

Instead of placing the SPDT slide switches in the 3D printed end caps I drilled a hole in the tube and installed the switches externally. Before adding the switch I had two loose wires, one from the battery and one from the battery connector on the Adafruit Feather that I made available in that hole of the tube.



I also CANNOT emphasize enough how important **leaded solder** was in constructing the poi and remote components. Lead-free solder does not flow easily, and broke and burned some of my Dotstar strips and microcontrollers. It was extremely difficult to get a good electrical connection with lead free solder as well, which caused multiple solder cycles and unnecessary reheating (and potential damage).

To assemble the remote I simply wired the pins to the pushbuttons and resistors on the breadboard with the Adafruit Feather installed, and hot glued the LiPo remote to the side of the breadboard. To finish the housing for the remote control I placed the breadboard + Feather + battery + wires assembly in a cardboard housing:

Performance

Although there are no quantitative measurements to evaluate the performance of the project, I believe the demonstration speaks for itself. Have fun!

Differences from PDR -> CDR -> Final Report

There were a number of things I changed throughout the lifecycle of this project. At PDR I originally envisioned a touchscreen on the remote control that would allow an audience member to draw an image and have it display on the poi's persistence of vision display. That was too ambitious for this semester, and I removed it from the scope, although I'd like to pursue it in the future. At CDR I baselined two poi, using the SAMD51 Thing Plus microcontroller, and an Arduino UNO for the remote control using normal Bluetooth. By the time of the final report I constructed one poi (not two), switched the microcontroller to the Adafruit Feather, and switched from Bluetooth to Bluetooth Low Energy. At CDR I also had baselined a rotary knob for the brightness control, but I realized it made more sense to have a button interface the same as all the other functions. A rotary knob would have added complexity to the user while adding trivial technical complexity and no additional features.

References

(1) https://learn.adafruit.com/supernova-poi/ : primary reference for poi construction and code starting point

(2) https://learn.adafruit.com/bluefruit-nrf52-feather-learning-guide/central-bleuart : remote control starting point

(3) https://www.adafruit.com/product/4062 : microcontroller, Adafruit nRF52840 Express

(4) https://learn.adafruit.com/adafruit-dotstar-leds/power-and-connections : info on LED hookup

(5) https://www.hyperionhoop.com/shop/mtsp.php?paID=18 : freely available Poi LED patterns

(6) https://docs.arduino.cc/built-in-examples/digital/Debounce : SW debounce