Jennifer Sevan

Valentine Scherer

CART 360 - Assignment “Then”

Dec. 7, 2018

Jennifer’s GitHub

GitHub Project

DOCUMENTATION

Concept Overview Reminder

Two umbrellas that connect over wifi drive an array lights based on pressure in the handle and also the distance between the two umbrellas. Both variables will drive different combinations of light patterns and color. Our aim was to allow people to relieve themselves from darkness and isolation that rainy weather elicits. And also, to find or be reminded of human connection with others in response to possibly harsh meteorological conditions that make people adopt a ‘to each their own’ attitude in city landscapes.

Ultimately future iterations of this design and applications of it could include elements of luminotherapy, that would aid especially people with seasonal affective disorder (SAD): The lights and intensity, might adapt to both the user needs (such as how much daylight has been present that particular day).

Electronic Components Used

For our project, we mainly used an FSR (Force Sensitive Resistor), two particle photon wifi development board (*Particle PHOTON Wi-Fi Kit with Comprehensive Development Tools and Free Cloud Access)*, a waterproof 144 LED RGB strip (1 meter long), and a FadeCandy. We had specific reasons for each of the materials. First in order to retrieve the location from the two points, and use that data to draw meaning, it made sense to use these particle boards as we could use them to fetch data (that would be sent to each other via the cloud - both micro controllers were subscribed to each other) using the Google Maps API. But also to use it as a fully functional microcontroller that would use that data to control the lights. We ended up deciding not to use the FadeCandy, as it didn’t make sense to use both the Photons and the FadeCandy. What we didn’t realize was that it is more commonly used to control much larger arrays of LEDS, such as matrices.

Setting up the photons, was a bigger process than we realized especially in terms of getting it to connect to the university entreprise Wi-Fi became a significant challenge. First we used the Particle tutorials to fetch geolocate the devices, which returns three consecutive values:

=> longitude, latitude, accuracy

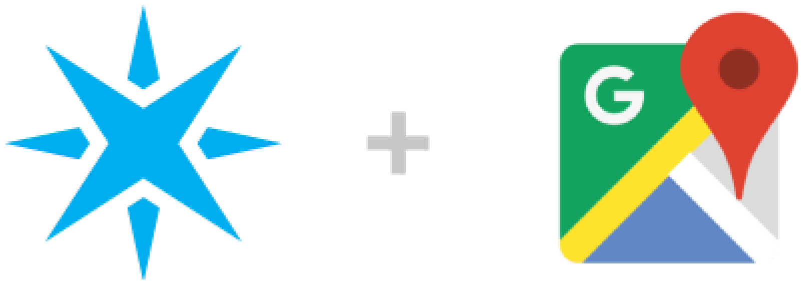
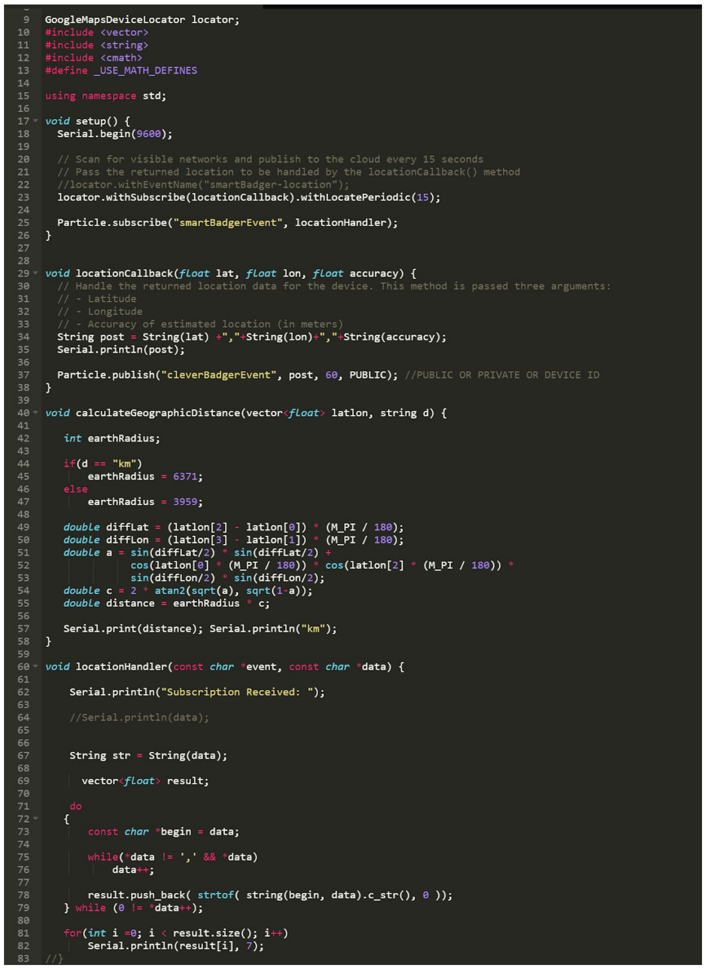
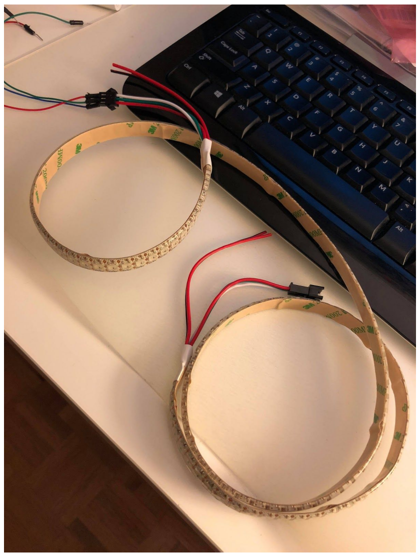
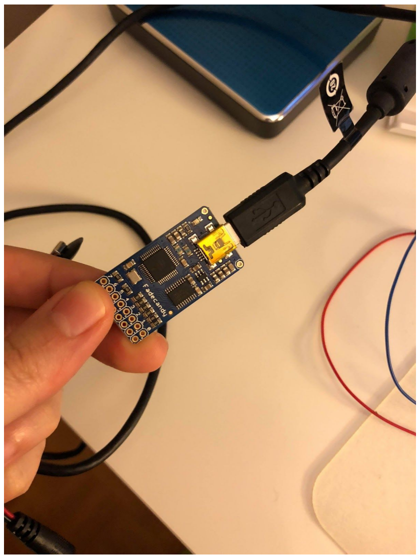


Fig A



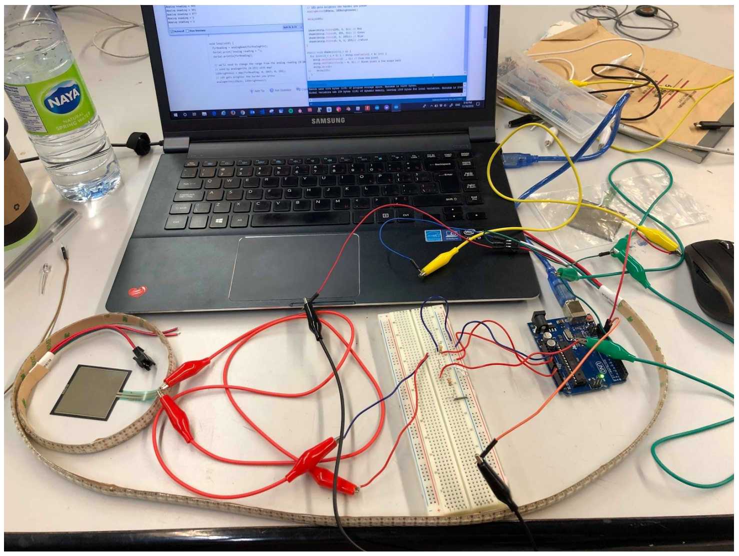
```
GoogleMapsDeviceLocator locator;
#include <vector>
#include <string>
#include <cmath>
#define _USE_MATH_DEFINES

using namespace std;

void setup() {
  Serial.begin(9600);

  // Scan for visible networks and publish to the cloud every 15 seconds
  // Pass the returned location to be handled by the locationCallback() method
  //locator.withEventName("smartBadger-location");
  locator.withSubscribe(locationCallback).withLocatePeriodic(15);

  Particle.subscribe("smartBadgerEvent", locationHandler);
}


void locationCallback(float lat, float lon, float accuracy) {
  // Handle the returned location data for the device. This method is passed three arguments:
  // - Latitude
  // - Longitude
  // - Accuracy of estimated location (in meters)
  String post = String(lat) +","+String(lon)+","+String(accuracy);
  Serial.println(post);

  Particle.publish("cleverBadgerEvent", post, 60, PUBLIC); //PUBLIC OR PRIVATE OR DEVICE ID
}

void calculateGeographicDistance(vector<float> latlon, string d) {

   int earthRadius;

   if(d == "km")
       earthRadius = 6371;
   else
       earthRadius = 3959;

   double diffLat = (latlon[2] - latlon[0]) * (M_PI / 180);
   double diffLon = (latlon[3] - latlon[1]) * (M_PI / 180);
   double a = sin(diffLat/2) * sin(diffLat/2) +
              cos(latlon[0] * (M_PI / 180)) * cos(latlon[2] * (M_PI / 180)) *
              sin(diffLon/2) * sin(diffLon/2);
   double c = 2 * atan2(sqrt(a), sqrt(1-a));
   double distance = earthRadius * c;

   Serial.print(distance); Serial.println("km");
}

void locationHandler(const char *event, const char *data) {

    Serial.println("Subscription Received: ");

    //Serial.println(data);


    String str = String(data);

      vector<float> result;

    do
   {
       const char *begin = data;

       while(*data != ',' && *data)
           data++;

       result.push_back( strtof( string(begin, data).c_str(), 0 ));
   } while (0 != *data++);

   for(int i =0; i < result.size(); i++)
       Serial.println(result[i], 7);
//}
```

We then proceeded to subscribe the devices to each other, the WEB IDE - and to send each other's JSON location objects, once "Subscription Received". As seen in the code (screenshot) the two devices are attributed their own names smartBadger and cleverBadger, and they are both subscribed to each other. Because of this they are able to retrieve the geolocating data that they are respectively publishing to cloud. The geolocation data was then used to calculate the distance between each other's respective location.

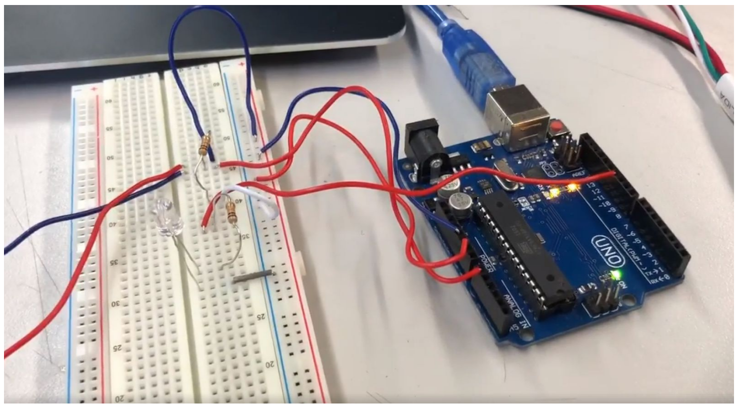



In the photo above, we were trying to make the LED strip work by testing out the library code Strandtest from Arduino. Unfortunately, the first ~10 LEDs weren't working properly. The code didn't work as it was meant to, despite looking over several times.

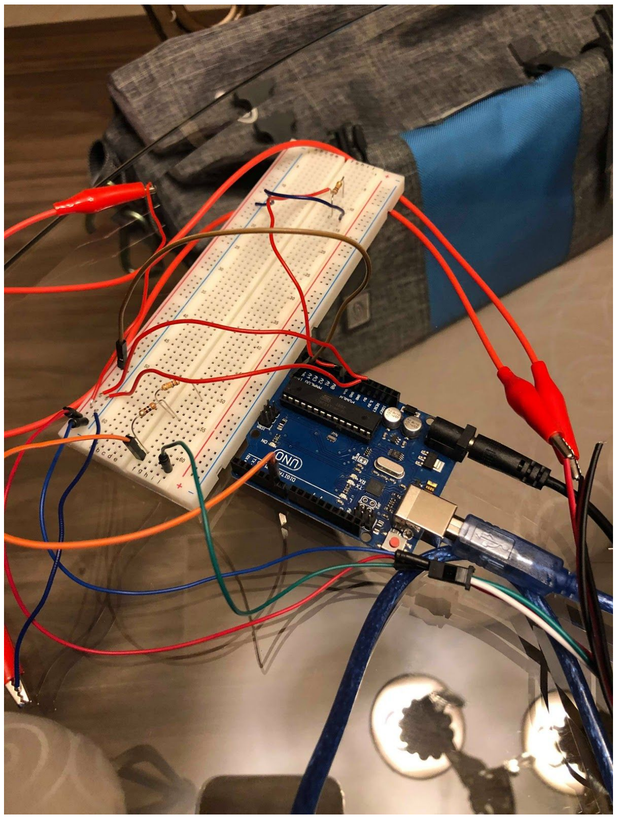
We attempting using the FSR to drive the the LED strip to test it out with the lights, even though we weren't sure as to why it wasn't working properly. It still didn't work.

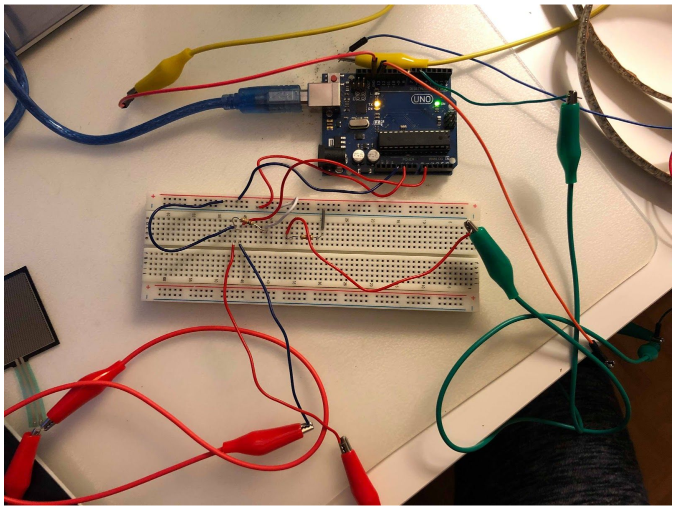


The FSR was able to receive pressure values but it wasn't cooperating with the LED strip. Initially, we thought maybe the code was wrong so we decided to deconstruct the process and take things step by step. In other words, we took everything apart and connected the FSR to a single LED light with 2 resistors to make sure the code works.



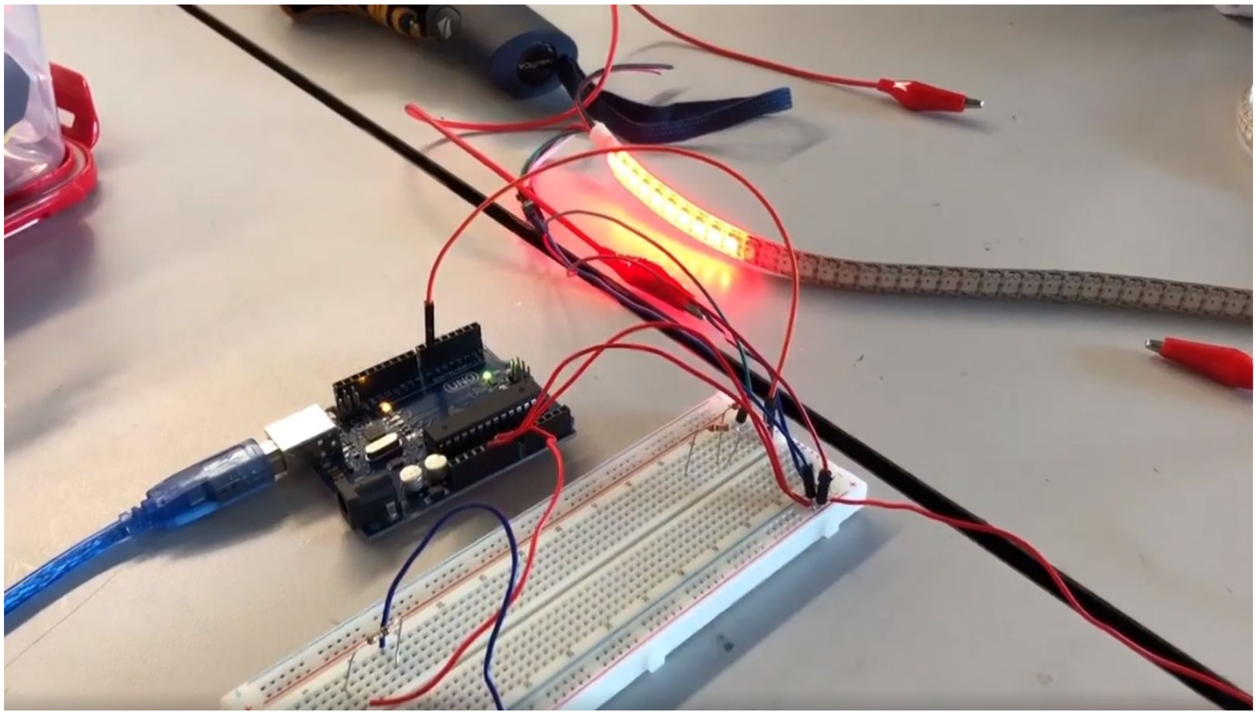
It worked! Next came connecting the LED strip again to understand what was happening. Upon further reflection, we realized we didn’t need to use the FadeCandy anymore since we would be using the photons for all the code anyways.

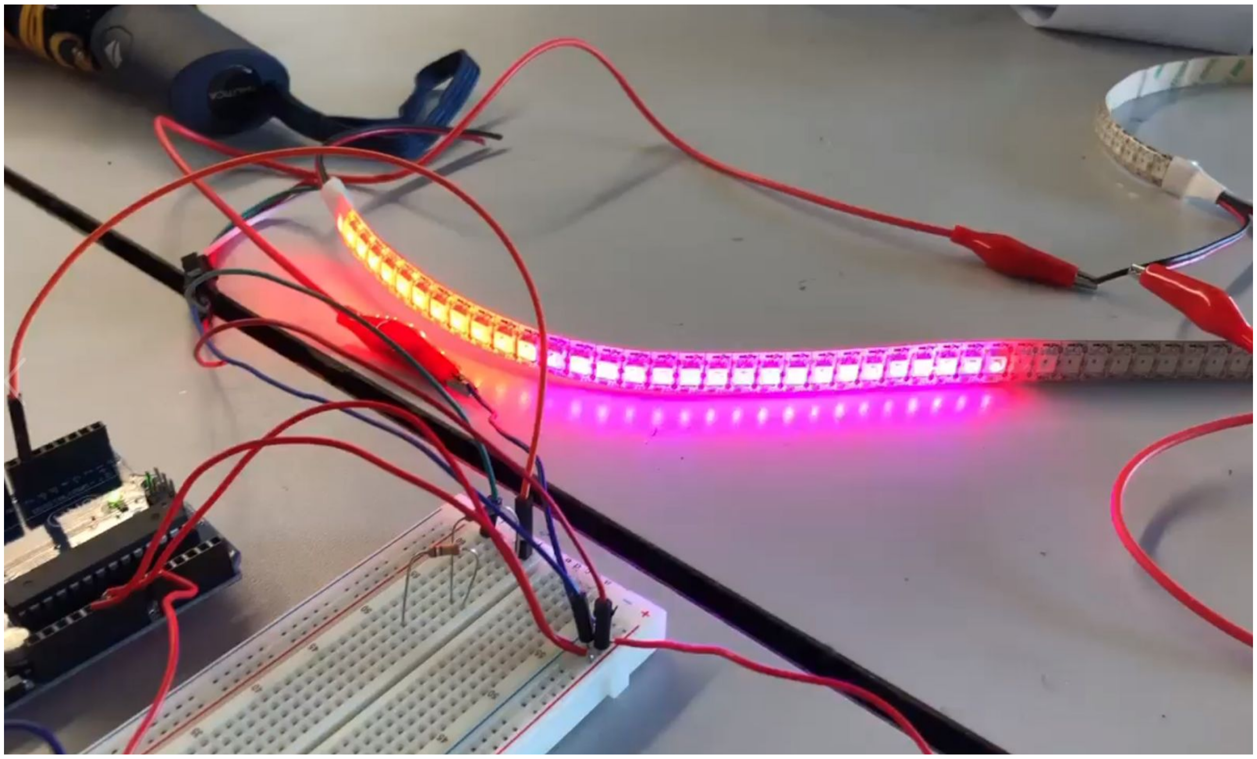




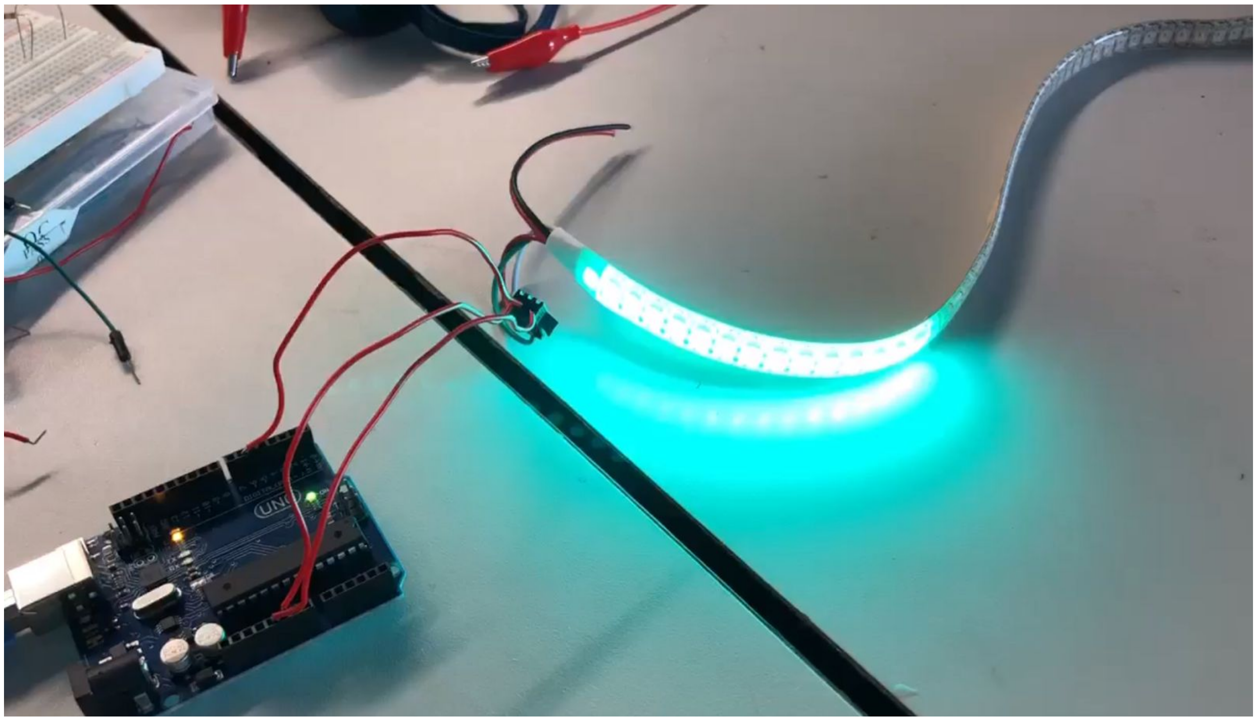
We then connected everything back to the breadboard.with a a couple of modifications.

The code we had makes the lights pulse slowly in a single color. Although, it was only working with the first ~10 LEDs.

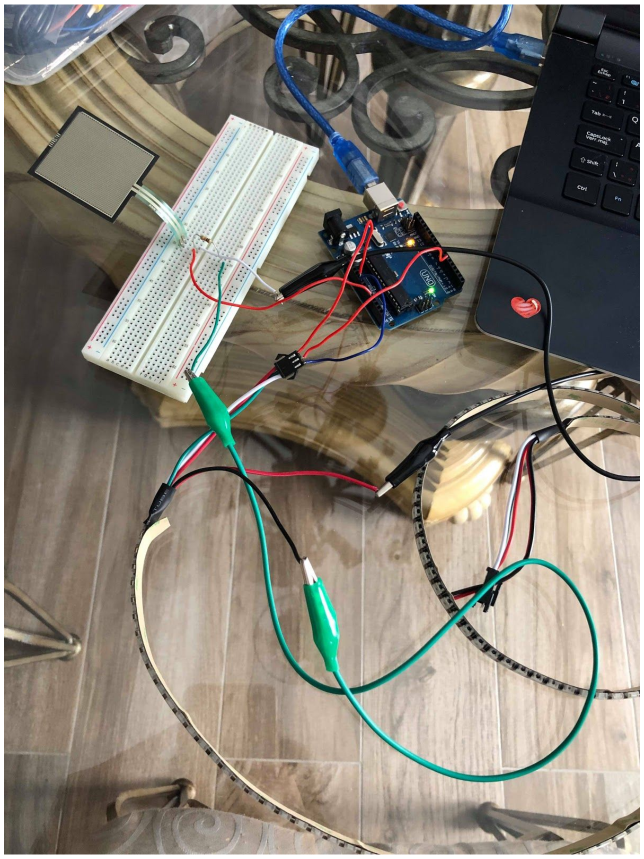




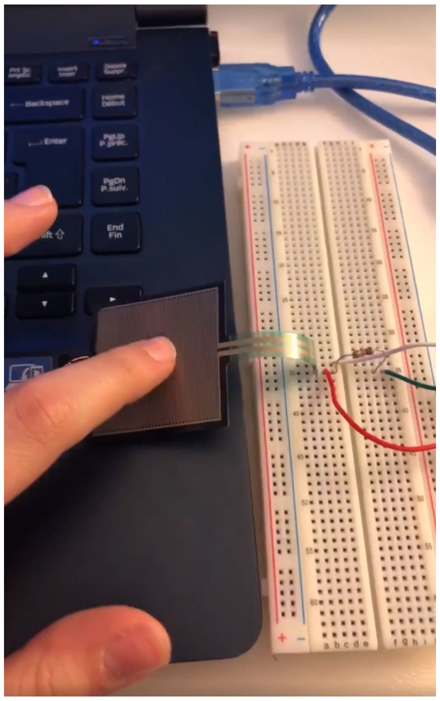
After a little help from a friend, we realized that using jumper wires was a bad idea because it was ruining the signal from the Arduino to the strip.

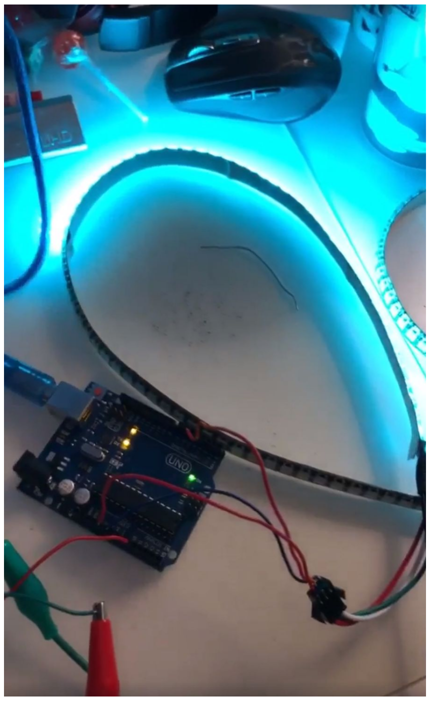


Things started to work a bit smoother. But the problem of the strip was still there. We tried to test the code with another strip provided to us in the lab to make sure our code was working properly. After we tested, we realized the code was perfect. Which means, the LED strip was defective. So we had to go and replace it with a new one from Abra.

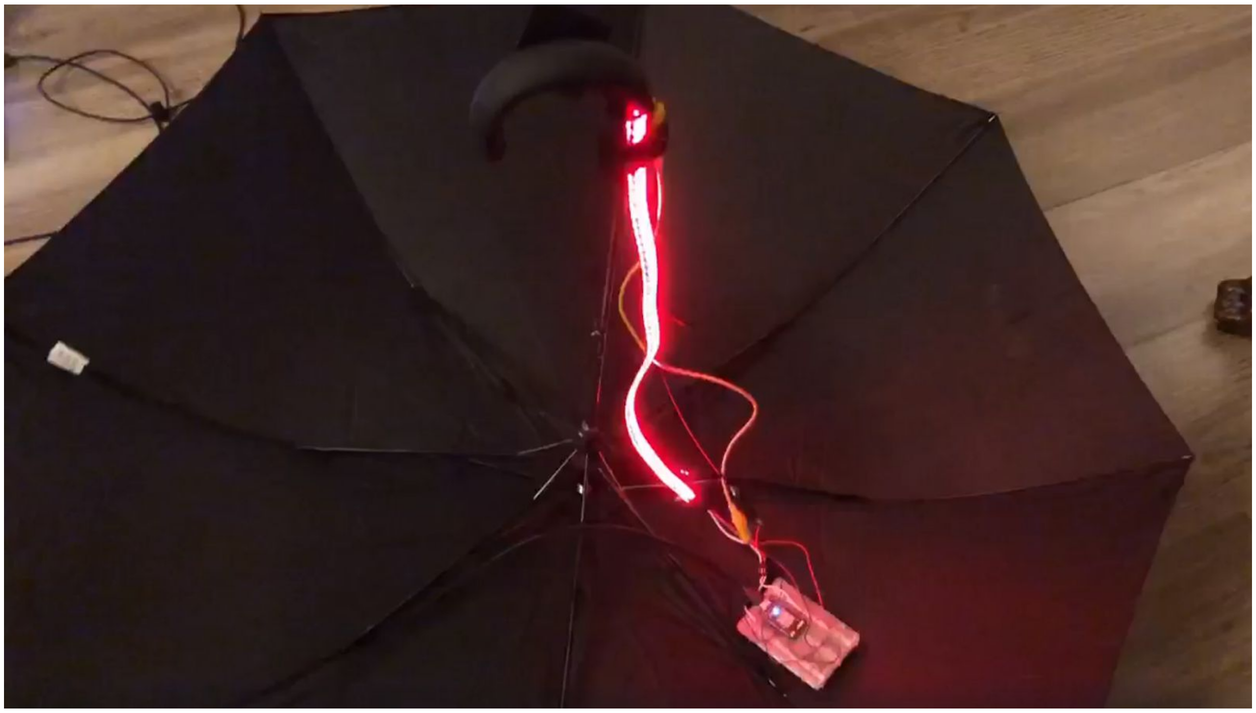


Once everything was setup, we tested out the code with the strip and the FSR. When the sensor is being pressed, depending on the force, the lights will light up accordingly.

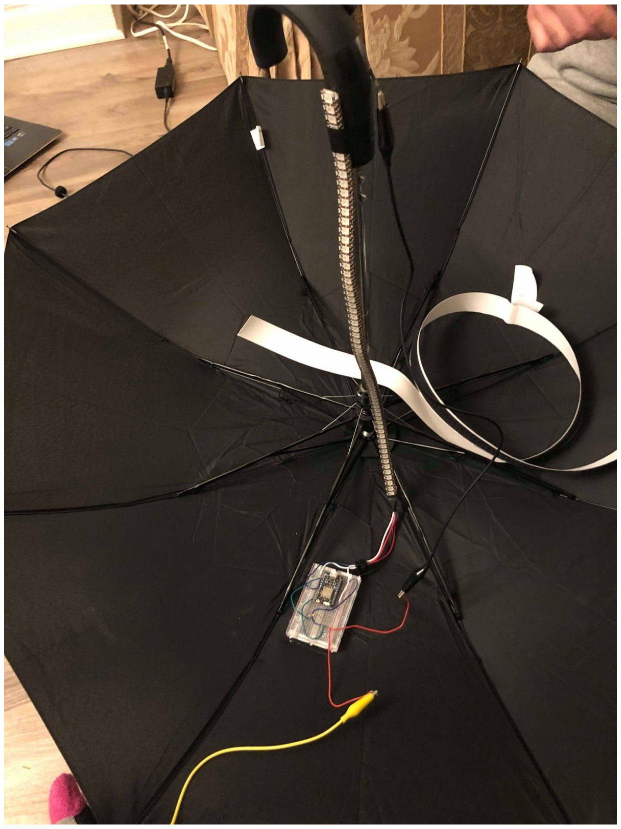




The LED strip and the FSR were working great. Now we wanted both photons to speak to one another when they get either too close or too far away. Both photons were working accordingly with the LEDs. The colors were changing and cooperating whenever they were being separated or getting closer. When they were too far away from each other, the FSR code gets online and that code starts to work.



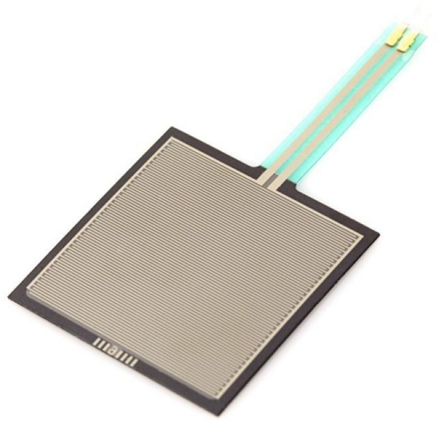
Once everything was working, we focussed on the design of the final version. Unfortunately, contrary to what we intended to make we had to place the LED strip along "stem" part of the umbrella. Whereas, we initially wanted to place several mini strips that created lines from the top, towards the bottom of the umbrella (and all around). This is because after testing out the other half of the strip, it seemed almost impossible to solder the strips together without making them malfunction out. The copper rings on the strips were too small, and close to the LEDs themselves, for them to be cut out let alone soldered. Doing so, caused the LEDS to have receive much less power, and seemed to be less responsive and more malfunctioning.



During our presentation, our photon burned due to last minute inversal of power and ground. Which means we have to get a new one to replace it in order to continue to complete the documentation.

Electronic Components Used - Documentation

Force Sensitive Resistors (FSR)



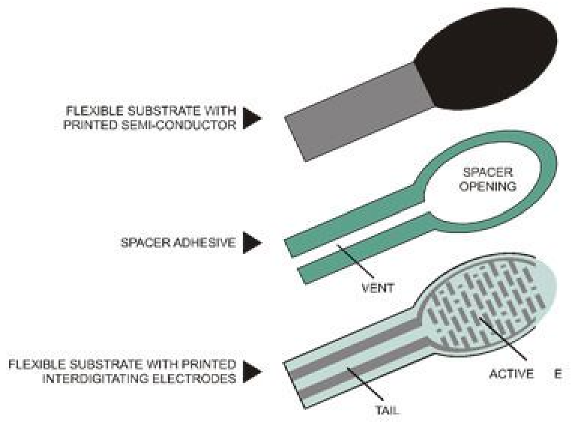
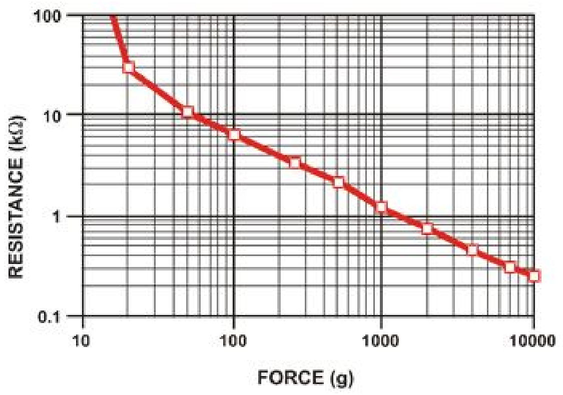

FLEXIBLE SUBSTRATE WITH PRINTED SEMI-CONDUCTOR
SPACER OPENING
SPACER ADHESIVE
VENT
FLEXIBLE SUBSTRATE WITH PRINTED INTERDIGITATING ELECTRODES
ACTIVE E
TAIL

Fig. B

Simply put, Force Sensitive Resistors, otherwise known as FSR's are "sensors that allow you to detect physical pressure, squeezing and weight" (Adafruit). They change their resistance (Ω) in response to on how much pressure is being applied to the outer flexible substrate that is, as seen in the diagram, semi-conductive. It is very advantageous to use these sensors as they aren't expensive, and the values are generally quite reliable, and can give a wide range of values. They can be replaced with Force Sensitive Capacitors that are more precise.

Generally, as applied 'force' increases, resistance decreases in response to it (see below diagram).



Ranges of Limitations:

From minimum to max

Resistance: 100 -- > 2000 Ω

Force: 0 -- > 10 kg/cm^2

Power: minimum 1mA (no max)

Temperature: -30°C to +70°C

Fig A.

Particle PHOTON

Wi-Fi Kit with Comprehensive Development Tools and Free Cloud Access

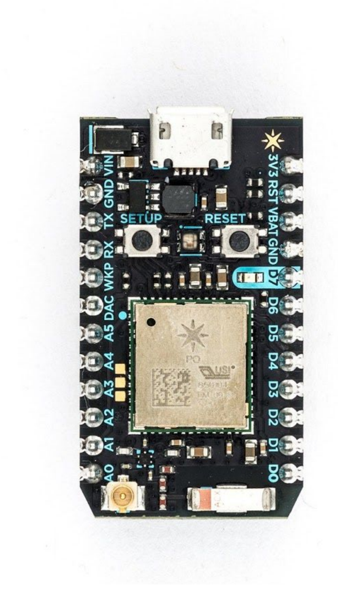
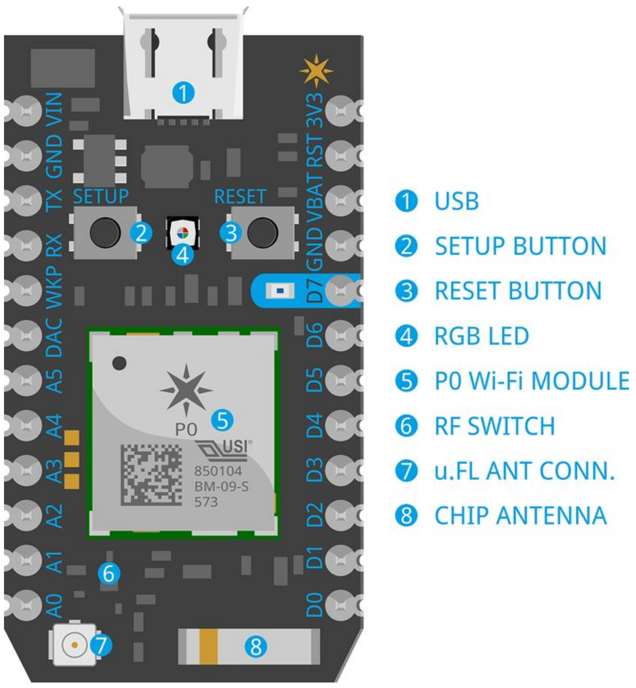


Fig. 2.

- This device is commonly used for making Wi-Fi connected projects.
- Uses IoT platform
- Has a Cypress Wi-Fi chip
- Devices connected to their own Cloud.
- Power through micro usb
- Will usually consume 80mA with 5V @ VIN.
- Datasheet recommends using, not overly long cables and to make sure they are high quality in order to prevent IR drops.
- Users are able to make changes in either internal or external mode of the photon device through a user API.
- Firmware can be written locally (IDE) or on the web.

Specifications:

24 pins total
STM32F205 120Mhz ARM Cortex M3
1MB flash
128KB RAM
802.11b/g/n Wi-Fi
RGB status LED



Main Pins:
VIN - input/output, input 3.6 to 5.5VDC. When powered (via microUSB) will output 4.8VDC voltage.
3V3 - output of on-board regulator. Connected to VDD (Drain supply) of Wi-Fi module. Maximum load is is 100mA when it is used as an output. If using VIN or microUSB power, the board will output 3.3VDC in voltage.
RST - Reset button, active low reset.
RX - Mainly for UART RX but alternatively PWM (pulse width modulation) or GPIO (general purpose input/output)
TX - Mainly for UART TX but alternatively PWM (pulse width modulation) or GPIO (general purpose input/output)

Fig 3.

LED RGB Strip

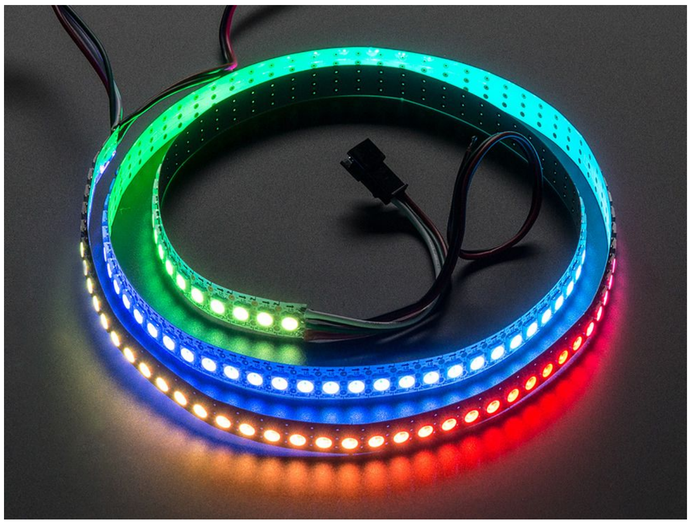


Fig 4.

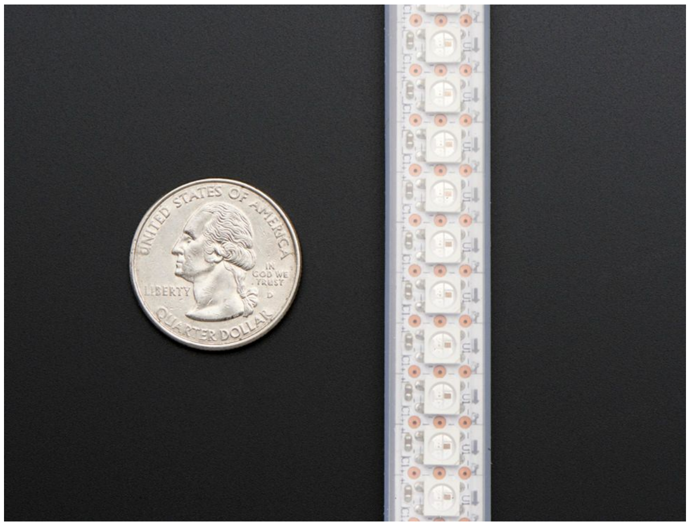


Fig 5.

LED-STRIP-15W-1M Addressable RGB strip 144 LEDs/m white with WS2812B 5050 SMD-Weatherproof 1 meter

Each individual LED is controllable (addressable) and has 24-bit color per pixel. PWM remains set, and therefore each LED remains addressed one "you stop talking to the strip" (Adafruit).

High power usage: 7 Amps @ 5V which is max rating, for all LEDs at white. (however, for regular RGB usage, will most likely use ⅓ or ½ of those particular values)
Recommended: 5V 10A power supply
One pin for input, one for output
Removable Waterproof outer-layer to protect LEDs
Strip made of PCB.

Specifications:
5V @ 60mA usage for each LED max
LED wavelengths: max 5V @ 60mA draw per LED
Uses JST SM connector (3 Pin)

References

Websites and Documents:

Ada, Lady, and Danny Nosonowitz. “Force Sensitive Resistor (FSR) + Datasheet.” *Adafruit*, Adafruit Learning System, learn.adafruit.com/force-sensitive-resistor-fsr.

“PHOTON DATASHEET (V016) Datasheet.” *Particle* , Particle, docs.particle.io/datasheets/wi-fi/photon-datasheet/.

Particle Retail. “Photon.” *Particle Retail*, store.particle.io/products/photon.

Adafruit Industries. “Adafruit NeoPixel Digital RGB LED Strip 144 LED - 1m White.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1507.

“Particle.” *Particle Datasheets Documentation | Photon Datasheet*, docs.particle.io/tutorials/integrations/google-maps/.

Images Used:

Fig. A. “Particle.” *Particle Datasheets Documentation | Photon Datasheet*, docs.particle.io/tutorials/integrations/google-maps/.
Fig B. Ada, Lady, and Danny Nosonowitz. “Force Sensitive Resistor (FSR) + Datasheet.” *Adafruit*, Adafruit Learning System, learn.adafruit.com/force-sensitive-resistor-fsr.
Fig 2. And Fig 3. Particle Retail. “Photon.” *Particle Retail*, store.particle.io/products/photon.
Fig 4. and Fig 5. Adafruit Industries. “Adafruit NeoPixel Digital RGB LED Strip 144 LED - 1m White.” *Adafruit Industries Blog RSS*, www.adafruit.com/product/1507.