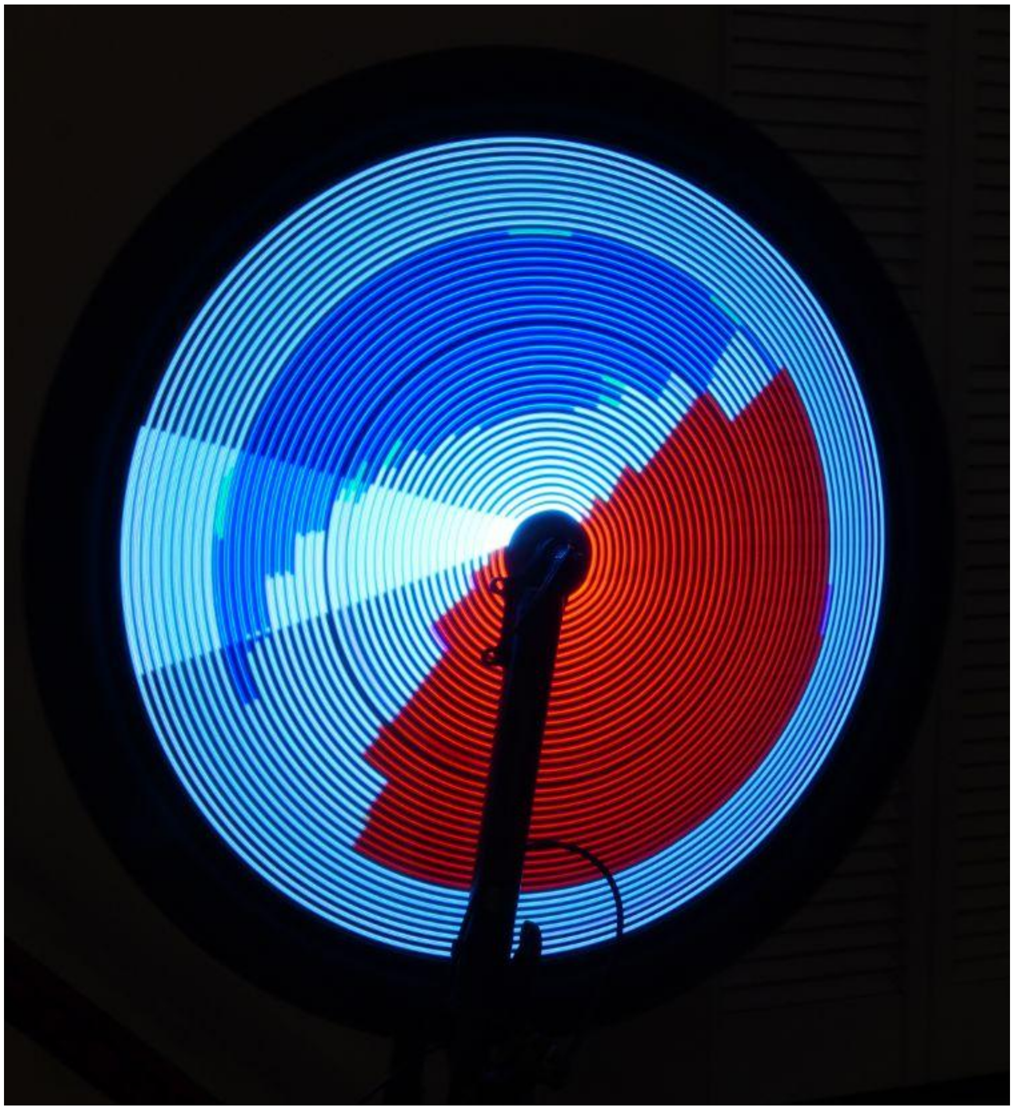
ECE Capstone Design Report

Team: Light Rider

Members: Akaash Beher, Jack Whittington
Course: ENEE408A
Instructor: William Hawkins
12/20/2022



Team Member: Akaash Beher

My two primary contributions were tailored towards building the codebase to take a 2D image and map the pixels to polar coordinates to be displayed on the bike wheel by lighting sectors of the wheel. The second contribution I had was developing the communication codebase between the microcontroller and a client which would be able to control the output of the LEDs. This was done using WiFi libraries to host a server on the microcontroller. From this, I spent time integrating the LED implementation Jack made with the WiFi database in order to have both systems work concurrently.

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment.

[illegible]

Akaash Beher (Dec 20, 2022 19:47 EST)

Team Member: Jack Whittington

Most of my time was spent focused on the implementation of the LEDs and achieving accurate timing for the LEDs. I programmed the LED system which takes an array of RGB values, outputs them to the LED strip, and changes what is output to the LEDs based on input from two sensors. I designed and 3D-printed our mounting device for the LEDs, microprocessor, sensors, batteries, and timing wheel. I also wrote some code in C to generate the pattern for our timing wheel so that it could be printed out on a sheet of paper. I spent a lot of time figuring out how many cycles of the LEDs we would be able to fit in a single revolution, tweaking the timing wheel design for reliability, and tuning the optical sensors.

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment.

[illegible]

Table of Contents

III. Executive Summary

The goal of the project was to mount one or more LED strips to the wheel of a bike and update them so that they would be able to display the speed of the bike in real time through calculations of wheel revolutions. We also wanted to create an app that would be able to turn the device on and off or change the color of the display.

Our plan was to have Jack focus on the display, timing of the LEDs and wheel construction while Akaash focus on getting the right information to the LEDs by converting images to polar coordinates, and developing the communication between the microcontroller and another device.

While brainstorming the implementation of the device, there were several facets of the design to be considered. These included what LEDs to be used in the design, the mechanisms to to trigger the updates of the LEDs, what wireless communication method was to be used, how to maintain a time-critical codebase implementation. These solutions to the decisions we made were based on several factors, such as cost, ease of implementation, familiarity of technology, and other variables.

During the development phases of the device mechanisms, several tests were done to ensure each system would be ready for prototyping. Two notable simulations were emulating revolutions for the infrared sensor and using on-board LEDs in place of the LED strips.

There were several stages to the prototype development, with each step centered around the fabrication of one aspect of the system. We began with the drivers for the LEDs, followed by the photosensor and then the hysteresis circuit, which was then integrated with the bike wheel and the timing disc. After the hardware components of the design were implemented, the remaining needed software was the program to map a 2D image to the wheel as well as integrate a communication protocol with the wheel display.

The validation and testing of the system was done concurrently with the prototype development, ensuring each piece of the system was able to work with each other. For the LEDs, a strip was taken and test code was written to light up one LED at a time for a strip until the entire strip was lit. Similar testing and implementation validation was done by unitizing aspects of the design and testing each part concurrently, then putting them together one at a time in order to ensure any errors could be traced to a single integration step.

IV. Main Body

A. Introduction

Both of us are cyclists, so when we were brainstorming ideas for our project, many of them involved bikes. Jack brought up that he had recently done a personal project involving individually addressable LED strips and the idea for Light Riders was born. We knew from the start that there would be some design challenges in trying to get the LEDs to update fast enough that we could display an image on a bike wheel, but we were up for the challenge.

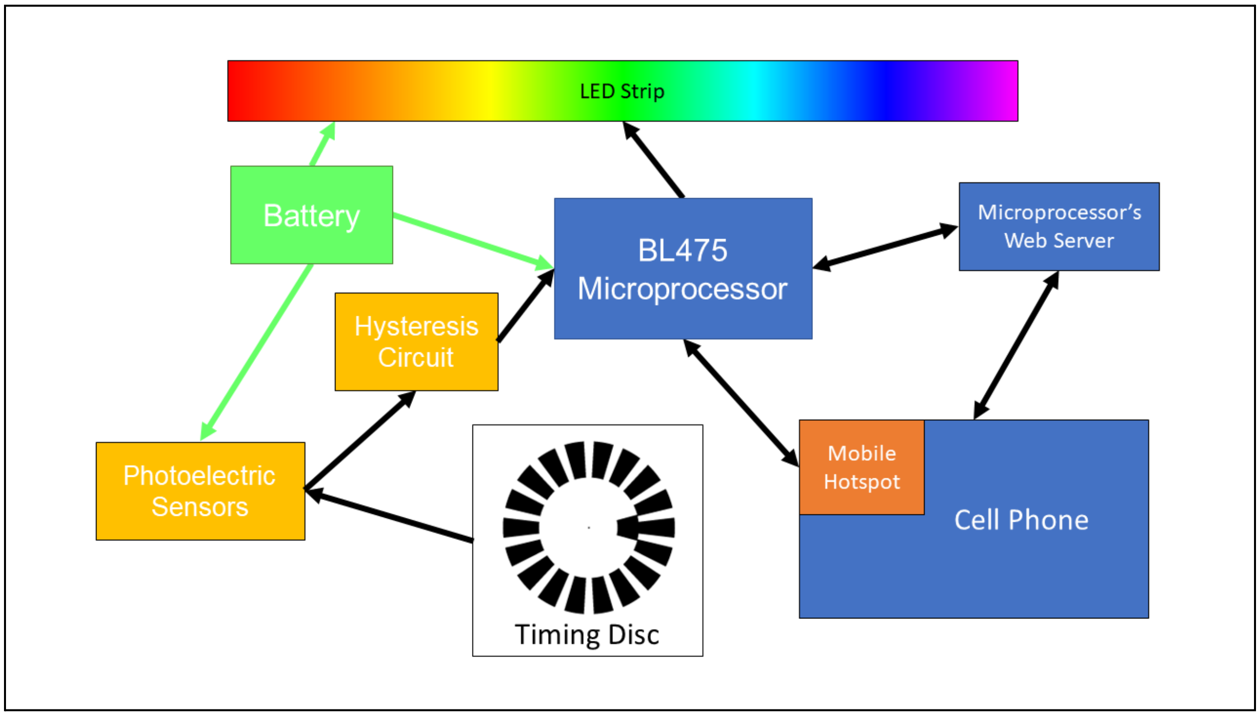
Our product would be marketed as an entertainment device, but also potentially a safety device for cyclists. Most bicycles come from a store with reflectors installed on the wheels specifically because they make the bike more visible at night [1]. Light Riders makes a bike much more visible without the need for an external light source such as a car’s headlights, and the moving lights that are changing color make it even harder to miss a Light Rider bike.

The Light Rider hardware system includes a 5V battery pack (two battery packs for the prototype), the timing disc, two photoelectric sensors, the microcontroller, all USB cables and other wires to connect everything, and all appropriate mounting hardware. On the software side, the micro can run independently, but can be controlled via a web page hosted on a cell phone’s hotspot feature.

When we were considering our design options for the LED update triggering system, we took a look to see if anyone else had made a similar product. We found a product made by LumenCycle that works in a very similar way to our project [2]. By carefully watching the marketing videos that they have on their website, we realized that they are using a gravity-based method to detect which direction is down on the wheel (this is important for orienting the image correctly). A downside of their approach was that the image appeared to be wobbling as the wheel turned, which is part of the reason why we decided to go with the photoelectric sensor-based design.

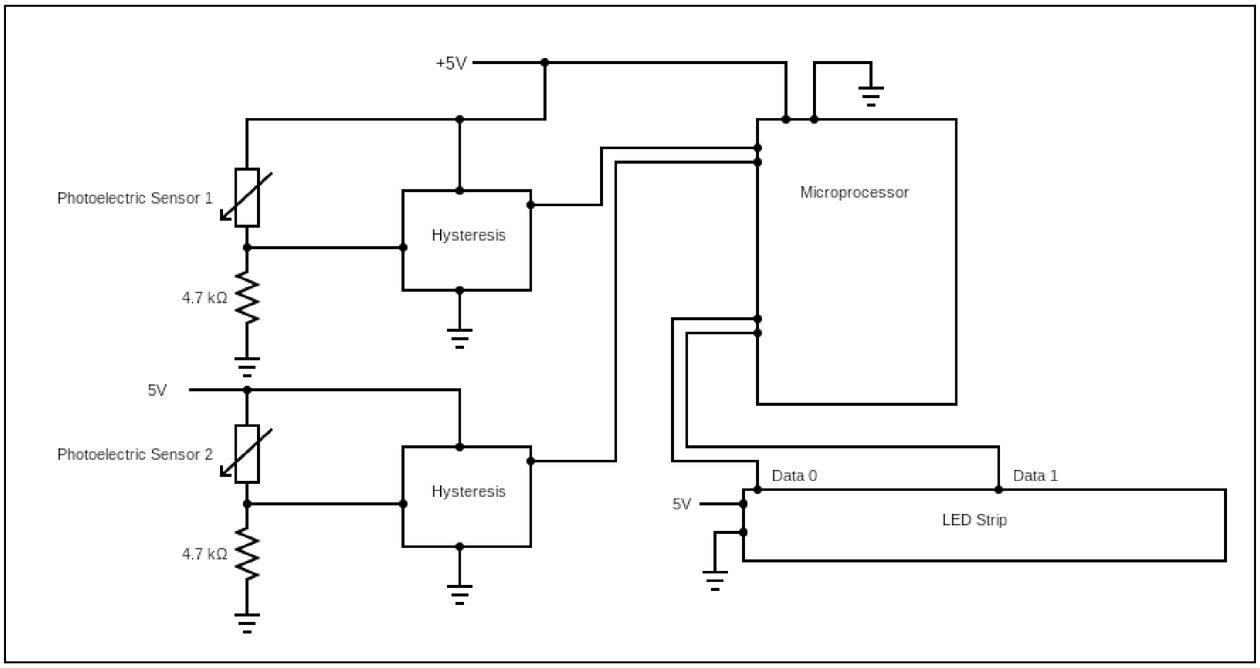
B. Goals and Design Overview

The original goal of the project was to create a system which would accomplish the task of drawing the wheel speed onto a bike wheel. The design specifications of the system included a developing a process for converting 2D image to polar coordinates, creating housing for micro and power supply to mount to wheel, developing an electronics schematic for any circuitry, displaying an image using spinning LEDs, monitor the speed of the biker, displaying that speed onto the wheel, and communicating with the system via Bluetooth. During the process of this project, the majority of these goals were accomplished in the way intended. Some of these goals were not required for development, but ended up being achieved due to the needs of the system, such as developing an electronics circuit for the system. With regards to our next goal, we wanted to display the user speed on the wheel, but we were short of that due to timing constraints. In addition, we switched from communicating over Bluetooth to using WiFi instead due to our familiarity with the latter, since integrating Bluetooth was not trivial.



See section F. Technical analysis for system and subsystems for further explanation on what each block in the diagram does.

Electric Circuit Schematic:



C. Realistic Constraints

- Time: project must be accomplishable within a single semester
- Limited to using an STM microprocessor
- Cost: We are not able to contribute more than $100 each
- Safety: Must not interfere with the steering/braking systems that exist on the bicycle
- Manufacturability: We have access to 3D printers, soldering stations, and breadboards, but were not able to get a custom PCB printed.

D. Engineering Standards

The first standard that we encountered in our project was when we were trying to get the LED strip to light up just one LED. We were able to find a datasheet for the WS2812 leds that we were using which showed us the very tight timing that was required to get the LED strip on [4]. We needed to be able to control our output pin within a tolerance of 150ns which, with an 80MHz system clock, would only be 12 clock cycles. This might have been doable by writing some very carefully timed assembly code, but we realized that it would make far more sense to use one of the on-chip peripherals which is available to us on our stm32L457xx microprocessor: the direct memory access or DMA peripheral. Using the DMA allows us to easily achieve the necessary precision on our output pin without taking up valuable CPU time.

Another standard that we encountered was when we went to start using the photoelectric sensor to trigger updates on the LEDs. We were able to find a datasheet for the sensor which was useful for determining which pins have what function on the sensor [4]. Using this sheet, we figured out an appropriate way to power the sensor, and how far away the target (the timing disc) needed to be.

We also ran into IP standards for TCP connections which meant that we had to abide by the standard protocols for connection creation and destruction.

E. Alternative Designs and Design Choices

When determining the method we would use to trigger updates to the LED strip, we considered several solutions as listed below.

Current Solution: Uses magnet and sensor (like common speedometers) to determine
Alternative 1: Uses rotary encoder mounted to the hub to determine rotational position
Alternative 2: Uses phone GPS location or accelerometer to determine rider speed, transmitted to micro via bluetooth.

From these solutions, we decided to create a list of criteria so we could quantitatively evaluate each system using a Pugh matrix. The resulting Pugh matrix is shown below.

Criteria	Optional Importance Weighting	Current Solution	Alternative 1	Alternative 2
Battery Life	5	1	1	1
Water/dust proof	5	1	-1	1
Impact on rider experience (vibration, noise, etc.)	6	1	0	1
Weight	4	1	0	1
Size	7	1	1	1
Expected lifespan (reliability)	2	1	-1	1
Safety	10	1	1	1
Accuracy of speed measurement	7	1	1	1
Accuracy of rotational position measurement (down direction detec	8	1	0	-1
Image resolution	8	1	1	1
Availability of Resources	8	1	1	1
Time to Implement	7	1	0	0
Ease to Implement	5	1	0	0
Cost to Implement	4	1	0	1
Total:		86	38	58

Our most important criteria was safety as the resulting user of the system may not be accustomed to the weight distribution of the wheel attached to the bike as well as ensuring safety with electronics and awareness when riding the bike. Several other notable criteria we had for the success of the system was accuracy or rotational position, image resolution, and availability of resources, as these were crucial to have a repeatable system that meets our intended design function. From the Pugh Matrix, the magnet and sensor was rated as the best solution for our design.

Our final implementation consisted of a design which has similar functionality to the selected design from the Pugh Matrix. The system to trigger updates was a mounted timing disc present on the outside of the wheel, and a set of two photoelectric sensors mounted to the inside of the wheel, emitting an infrared ray and receiving a signal regarding whether or not the surface is reflective or absorbent. This was done due to the challenge of mounting the hall effect sensor as well as precision from the system.

F. Technical analysis for system and subsystems

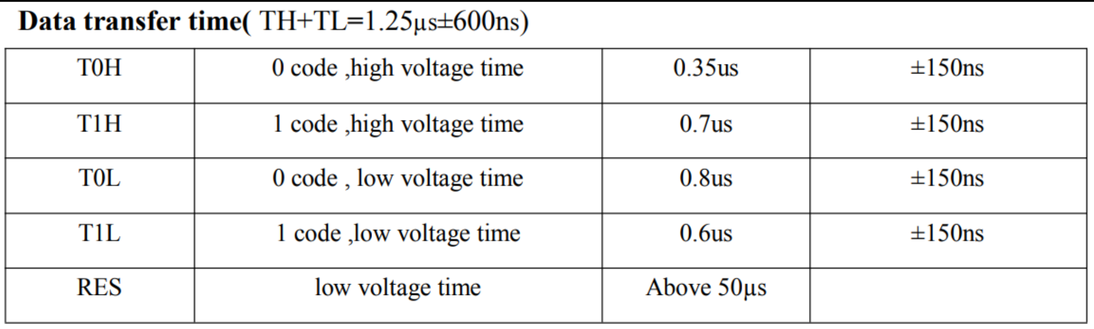
Design Systems

- Timing Disc
    - We found that the maximum number of segments we could fit on the timing disc before the photoelectric sensor was unable to distinguish the black and white stripes was 18 when we used a roughly 6 inch diameter disc.
    - We were able to get a more reliable signal from the sensor on the inner diameter by increasing the width of the black spot on the wheel.
    - The timing disc is printed out on a sheet of paper, taped to a 3D printed plastic disc, and then secured to the wheel hub.
    - The image that we print out is generated using a program which we made able to change the exact number of stripes, the thickness of the stripes, the output image resolution, and the inner and outer diameter of the stripes. This was useful when we were testing to see how many stripes we could fit on the wheel and still reliably detect them. Our final timing disc image is included in Appendix D
- Photoelectric Sensors
    - These sensors (2 of them) detect the black and white stripes on the timing disc as the sensors spin with the wheel.
    - Sensor 1 is mounted further away from the wheel hub and detects the 18 evenly-spaced stripes on the timing wheel. We set the GPIO trigger on the microprocessor to trigger on both the rising and falling edge of the signal which allows us to get 36 “slices” on the wheel.

      We noticed when we began triggering on both the rising and falling edge of the sensor, that the spacing between the two was not always very even. This was because the black stripes were too narrow on our timing disc, so instead of getting even 10 degree increments, the updates alternated between about 7 degrees and about 13 degrees. We were able

    to counteract this by simply printing a new timing disc with slightly thicker black stripes.
  - Sensor 2 is mounted closer to the wheel hub and it is only triggered once per revolution. This signal lets the microprocessor know when it should go back to the first update/slice in the image.
- Hysteresis Circuit
  - Originally, we had the sensor output simply go through a transistor, and into the digital input of the microprocessor. We found that this setup had a lot of noise when the sensor was on the edge of one of the black and white stripes. The simple hysteresis circuit made using two transistors adds the stability that allows the updates to be very consistent. Hysteresis circuits are more commonly made using op-amps, but we had transistors readily available, so we used them.
  - We added a separate, identical circuit for each of the two photoelectric sensors, and included a pull-down resistor on each sensor. We were able to change the resistance of the pull-down resistor to tune the voltage output of the sensor (our design worked best with a 5 kOhm resistance).
  - See Appendix F for schematic
- LED Strip
  - Each pixel on the LED strip requires 24 bits of data, 1 byte for each color in RGB. Each bit that is sent to the LEDs requires a precise time of the data pin being on and a precise time where the pin is off.

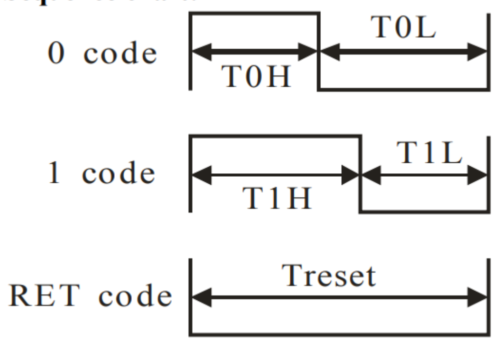
- Process for sending 1s and 0s to the LED strip from [3]:



Data transfer time(TH+TL=1.25µs±600ns)

T0H	0 code ,high voltage time	0.35us	±150ns
T1H	1 code ,high voltage time	0.7us	±150ns
T0L	0 code , low voltage time	0.8us	±150ns
T1L	1 code ,low voltage time	0.6us	±150ns
RES	low voltage time	Above 50µs	

Sequence chart:



- Once 24 bits have been sent for the first LED, we should immediately start sending the next 24 bits for the second LED, etc. until the end of the strip. At the end of the strip, we leave the signal high for the 50µs specified.

- Microprocessor: LED interface
  - We implemented the LED algorithm using the direct memory access (DMA) peripheral, a timer to trigger the DMA, the GPIO bit set/reset register (BSRR), and an array of uint32_t which we call dma_arr.

    Each uint32_t corresponds to 200ns of time (at 5MHz clock), so we need two 0s and four 1s to represent a 0 on the LEDs and three 0s and three 1s to represent a 1 (notice that multiplying these numbers by 200ns yields a result that is within the specification).

    This means 6 uint32_t in the array make up a single bit being sent to the LEDs. In total for our design with 40 LEDs, we needed an array of length: 40 LEDs x 24 bits/LED x 6 uint32_t/bit = 5760 uint32_ts. We also needed to add enough space at the end of the array for the Treset: 50µs/200ns = 250. We doubled this Treset to be safe, so our total array length is 5760 + 250x2 = 6260 uint32_ts. This takes up about 24KB of space on the microprocessor.

    We later realized that using 2 GPIO ports to send data to 2 led strips that were half the size could cut the array size in half, so in the final version of our code, this array only takes up about 13KB. This implementation had the added benefit that we could send data to the LED strips almost twice as fast.

Because we are using 2 ports, each word that the DMA reads will be a combination of 0x0001 (turn pin 0 on), 0x0100 (turn pin 0 off), 0x0002 (turn pin 1 on), and 0x0200 (turn pin 1 off).

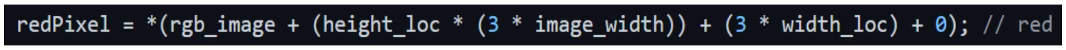
Along with the dma_arr, we need another array to store the RGB values for the pixels. We called this array colors_arr and it is slightly smaller than dma_arr because we just need 3 bytes/pixel (3 bytes/pixel x 36 segments x 40LEDs = about 4KB for one image. Note that any increase to the number of segments per revolution (essentially a resolution increase) would increase the size of colors_arr, but not dma_arr.

  - The majority of the CPU time for this program is spent updating the array that the DMA is looking at. The CPU is constantly checking to see if sensor 1 has changed state which triggers an update to the cycle number. Cycle number is used as an offset in the color_arr. The CPU then iterates over each bit in the cycle (for our setup this was 24 bits/LED x 40 LEDs = 960 bits) and programs the appropriate location in dma_arr.

  - The CPU is also constantly checking to see if sensor 2 has had a rising edge. When it has, the DMA will force the cycle number back to 1, restarting progress on the image. This means that any time the wheel passes the reset point and triggers sensor 2, the image will be re-oriented on the wheel. This is important for displaying images that need a distinct orientation which could be easily disturbed by turning the wheel backwards a small amount, and then continuing to turn it forwards.

- Image Conversion Software
  - A 2D jpg image is represented as a grid of pixels which are mapped in an x-y coordinate system, with each pixel consisting of an RGB value. However, the wheel does not have a synonymous x-y plane. Instead, it is able to send data to an LED which can be mapped to a radius distance from the hub as well as an angle relative to the positive x-axis. Therefore, software to convert a monitor's image representation to the bike wheels coordinate system is required.

    The goal of this process is to output a SECTORS*NUMBER_LEDS_PER_STRIP 1-D array so that we are able to load a single “row” of the array into the LED strip and go to the next “row” to send the LED values of the next sector. The first step of this process is loading in a square image’s pixel values, which was done using the stbi_load() function available through the <stb_XXX.h> series of libraries. From this, a 1-D array of pixels was created, where a pixel (x,y) could be located via the following calculation:



```
redPixel = *(rgb_image + (height_loc * (3 * image_width)) + (3 * width_loc) + 0); // red
```

Where rgb_image is the 1-D array, height_loc is the y value of the pixel, image_width is the width of the image, and width_loc is the x value of the pixel. From this, there are two more steps: mapping out the location of a pixel and then grabbing the pixel to output. Mapping the pixel involved adjusting the 2D image to account for the dead pixel space of the hub of the wheel. Next, the pixel data for a radii on an arc was mapped to the x-y cartesian plane, then centered around the middle of the image as opposed to the top left corner. The pixel was then passed into a color filtering algorithm to adjust the brightness as well as provide distinct color values instead of noise. This process resulted in a one dimension array of pixel structs, where each struct contains a uint8_t for red, green, and blue, of length 36 x 40 = 1440. This array was then pasted into the microcontroller code directly.

- Microprocessor: Web Server
    - For the project, a requirement was having communication where the microprocessor would have to interface with another device wirelessly. A previous implementation of a WiFi Web-Server was utilized to serve as the medium for wireless communication. A codebase was loaded onto the microcontroller which acted as a web server on a network. A phone’s hotspot is used to host a network where the microprocessor would be given credentials to access the network. From there, the microprocessor would host a web page, in which the LEDs could be turned on or off. A client would be able to go to the address of the webpage and alter its fields to send information to the microprocessor, which would then update the web page accordingly. After the WiFi web server was running, the microprocessor peripheral configuration settings were exported manually to the WiFi Server project. From there, the code to implement the LED strip lighting was integrated into the WiFi project and any conflicts or missing dependencies were resolved. From the completed merge, the LED strip was able to be controlled via a device connecting to the web page hosted from the microcontroller and sending data to the server.
- Power Distribution:

    - Our project is powered entirely by a “portable charger” 5V battery pack. We were able to get a set of 2 battery packs each with 15Ah at 5V. Based on our calculations in Appendix E, we expect a battery life of about 6 hours per battery with the current single strip of 40 LEDs
-

Redesign process:

Originally, we had decided that using a hall effect (magnetic field) sensor would be the best option for triggering updates on the LEDs, but we ended up switching to the photoelectric sensor instead. We made this decision for a few reasons. The biggest motivator for us was the ease of mounting. We decided it would be much easier to mount two photoelectric sensors and a single stationary disc, than having to mount many hall effect sensors or a ring of magnets. The other advantage that comes with the photoelectric sensor is the level of precision and flexibility that we could get. We are able to print out a black and white image on a sheet of paper with very good contrast, and it is very easy to simply print a new sheet with fewer stripes, wider stripes, etc.

One downside of this design, however, was that any dirt or grease that gets on the timing disc can cause issues with the sensors. This would be especially problematic in a real-world environment, so it may make more sense to go back to a hall effect sensor-based design for a full product.

G. Design Validation for System and Subsystems/Test Plan

Once we had the LED strip turned on and were able to set the color manually, we wanted to start to incorporate the photoelectric sensors into our design. Rather than jumping directly to using the sensors, we tested triggering updates to the LED strip using a button. We programmed the microcontroller to light up a higher index LED with each button press. Once we were successful while using a button, we moved on to adding in the photoelectric sensor.

One of the first integration tasks was to connect the photoelectric sensor circuit to the LED update triggering. At first we optimistically tried to connect the output of the sensor directly into a GPIO port. This method did not work at all because the sensor would never allow enough current through to produce enough voltage on the GPIO port to trigger a rising edge.

We realized that we needed to amplify the signal coming out of the sensor, so we added a transistor between the two. This change was effective in that we were now able to trigger updates on the LEDs. At this point we decided to mount the sensors on the wheel, hook up the timing wheel, and see if we could get consistent updates to the LEDs. The issue that arose was that each transition between black and white stripes would often trigger more than one update to the LEDs.

Normally we would simply try to debounce the input, but in our case, the wheel might stop at some point halfway between black and white which would cause rapid updates to the LEDs even when the wheel is stationary. The natural solution then, was to implement a hysteresis circuit. After lots of troubleshooting, we were able to get a working hysteresis circuit between the sensor and the microcontroller. This made the LED very reliable.

Table of Specifications:

Specification/Deliverable	Met/Delivered
Bill of Materials	Yes
Process for converting 2D image to polar coordinates	Yes
Housing for micro and power supply to mount to wheel	Yes
Electronics schematic	
Displaying image using spinning LEDs	Yes
Speed monitoring	No
Speed display	No
Bluetooth connectivity	No*

* We decided later in the project to use wifi as opposed to bluetooth, our wifi system, however, is not perfect as it did not integrate as well as we had hoped into the LED system.

H.Project planning and management

See Appendix G for the Gantt Chart that we used for this project.

This project was a learning experience in how difficult it can be to plan a project in advance. It is very difficult to say at the start of a project how much is going to be feasible, what new things may be learned, etc. We were not able to stay exactly on course with our Gantt Chart this semester and we believe this was a combination of not allocating sufficient time for debugging/troubleshooting and the fact that we were not able to commit as much time to the project as we had initially hoped.

I. Post mortem

- From Akaash’s Standpoint

The 2-D image to polar coordinate conversion subsystem went really well, as it was able to record the images onto the wheel arc as it was spinning, mapping the values correctly. The WiFi worked well too, being able send signals to the microprocessor with the LEDs following the command received. With regards to the system, I wish there was a better way to integrate the WiFi and LED codebases together. They are both time critical tasks relative to each other meaning that they interfere with their functionality. If the integration were to happen earlier on it would've been a good idea to play around with interrupts. Otherwise, the wheel was able to effectively update the LED image to display.

Regarding the technology of the project, it would've been nice to have another processor available to run both systems at once and they could communicate with each other serially. In addition, it would be nice to have a faster processor with larger memory so we could store more images. For resources, it would be nice if we had more time to put everything together at the end with the final date for submission being later.

- From Jack's Standpoint

  From the start of the project, I was most excited about getting a recognizable image displayed on the wheel. This is also where I spent most of my time this semester, so I was very happy that we were able to get to that point. I do wish we had spent more time on the bluetooth/wifi connectivity and started combining the systems earlier in the semester.

  That being said, I have several ideas on how to improve our design that I think would be very exciting to implement (see appendix J), and I do plan on continuing to work on this project after the end of the semester. I would also love to be able to get to the point of downloading an image on my phone and uploading wirelessly to the microcontroller, but app development is not something that I am very familiar with.

  Throughout the semester we have been fortunate enough to not have many resource constraints. For the most part, our most limiting factor was the amount of time that we were both able to put into the project (which is why I am excited to continue working on it over the coming winter break).

J. Salvage

- The biggest project goal that we did not fully meet was the bluetooth/wifi communication integration. We were able to get the WiFi and LED systems running at the same time, but not with the full update speed of the LEDs because too much time was spent blocking in the WiFi code. One possible solution to this would be to have a second microprocessor doing the communication over wifi or bluetooth, and letting the first one take care of updating the LEDs with them communicating with each other through UART or SPI.
- We were also not able to measure the speed of the wheel, but we are confident that it is very doable given more time. Especially considering the fact that we

already have an interrupt triggering with every revolution of the wheel. We would simply have to measure the time between interrupts using timers.

K. Recycle

The LED driver code could be used in any other project that needs WS2812 LEDs. Our latest version of the code also does not have any blocking segments, so it should be able to run some other program so long as that program is very light in load. The LED drivers could easily be packaged into a single function that is called in main within a while loop. Each call to the function could update the LEDs accordingly. The cartesian-polar coordinate code is also functioning if there is a need to map a 2-D image to a rotary LED set.

V. Conclusions

At the start of the project, we set out just to get a working prototype. The code for the LEDs was blocking until it received an update. Towards the end of the semester, we had to integrate the WiFi and the LED systems, which meant that we had to eliminate the blocking segments in both systems so that they could run simultaneously. We found out that even with the blocking segments removed, the WiFi system took too much CPU time for the LED system to function properly at the same time.

The biggest lesson that we learned was how to manage our time over the course of the project. We learned the importance of setting goals at the start of the project, and striving to meet those goals.

Overall,

VI. References

"Bike reflectors: What every cyclist should know," *Backroads*. [Online]. Available: https://www.backroads.com/pro-tips/biking/bike-reflectors-what-every-cyclist-should-know#:~:text=A%20bicycle%20reflector%20is%20a,visibility%20in%20dark%20riding%20conditions. [Accessed: 18-Dec-2022].

"Lumencycle," *LumenCycle*. [Online]. Available: https://lumencycle.com/. [Accessed: 18-Dec-2022].

"WS2812 Intelligent control LED integrated light source." [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf. [Accessed: 19-Dec-2022].

“Reflective optical sensor with transistor output - vishay.com.” [Online]. Available: https://www.vishay.com/docs/83760/tcrt5000.pdf. [Accessed: 19-Dec-2022].

VII. Appendices

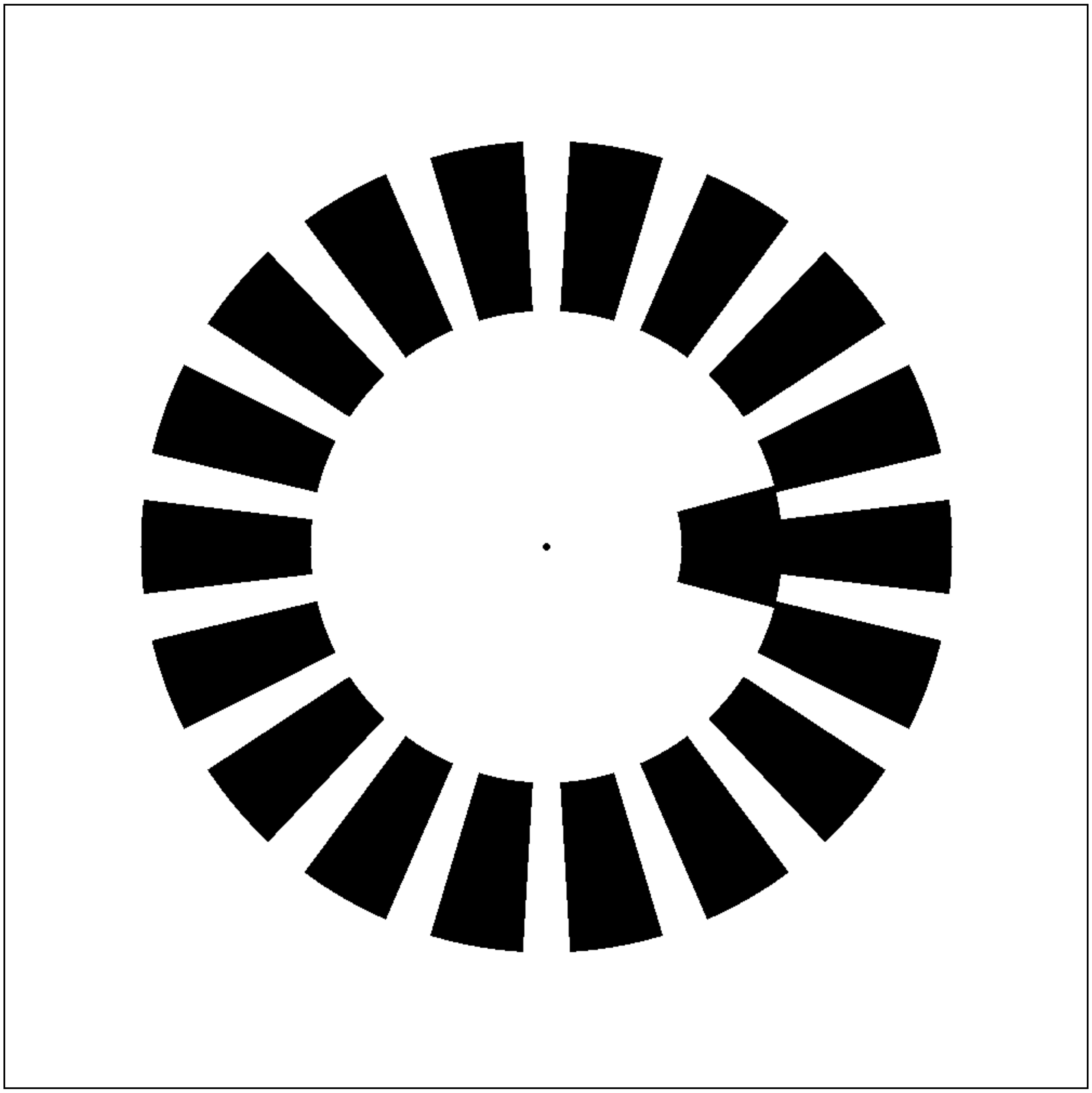
A. Appendix A: Our Github Repository
https://github.com/AkaashBeher/ENEE408ACapstone

B. Appendix B: The Image Processing Library we used
https://github.com/nothings/stb

C. Appendix C: The Adafruit Library used to obtain RGB color combinations
https://docs.circuitpython.org/projects/led-animation/en/latest/api.html#adafruit-led-animation-color

D. Appendix D: Our Timing Disc Image

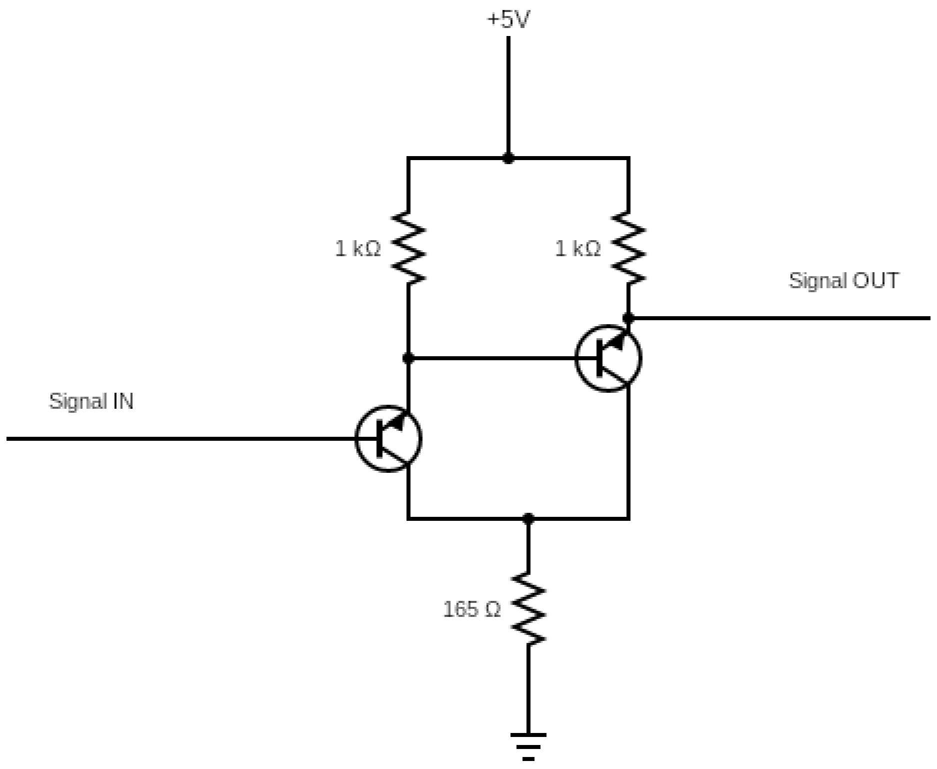
Note: Black stripes intentionally thicker than white stripes to make up for our sensor calibration



E. Appendix E

Each of our battery packs has an advertised capacity of 15,000mAh at 5V. According to the WS2812 datasheet, each LED will consume 60mA when at full brightness. We estimate that the microcontroller and sensors together will draw a maximum of around 150mA. If we were only powering the LEDs at full brightness and running the microcontroller and sensor array, we would expect 15,000mAh / (60mA/LED x 40LEDs + 150mA) = 5.8 hours. This is a very rough estimate which changes significantly depending on the brightness of the image that is being displayed.

F. Appendix F: Hysteresis Circuit



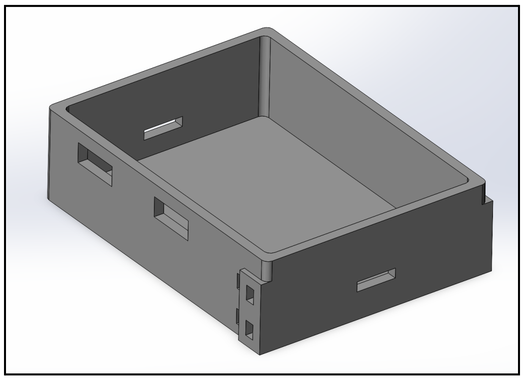
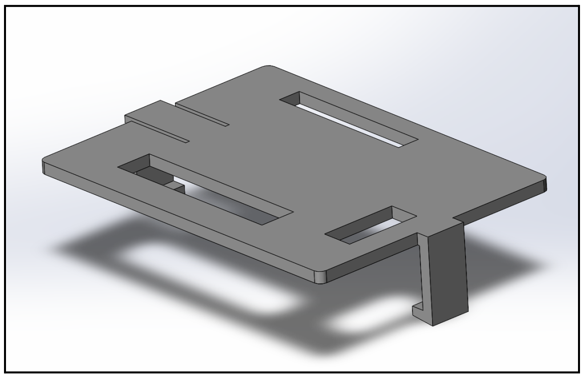
G. Appendix G: Gantt and Pert Charts

This link leads to the Gantt and Pert charts used to plan the timeline of the implementation of the project.

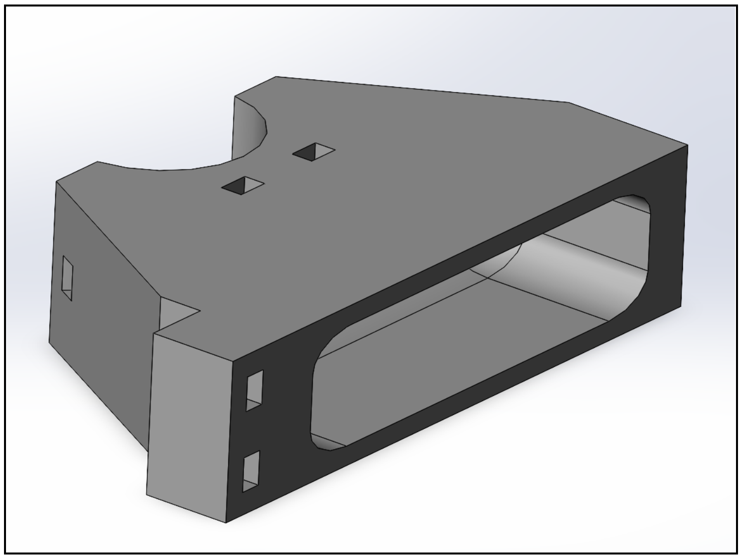
H. Appendix H: Bill of Materials

Item #	Part Name	Part Description	Unit Cost	Quantity	Total Cost
1	ADRESUNO WS2812B Individual Addressable LED Strip	LED Strip	$ 14.99	1	$ 14.99
2	B-L475E-IOT01A Discovery Kit	Development Board	$ 51.94	1	$ 51.94
3	LOVELEDI Portable-Charger-Power-Bank - 2 Pack 15000mAh	Battery Packs	$ 23.99	1	$ 23.99
4	2N3904 NPN General Purpose Transistor	Transistors	$ 0.06	4	$ 0.24
5	HiLetgo TCRT5000 Photoelectric Sensors	Photoelectric Sensors	$ 0.35	2	$ 0.70
6	Mounting Hardware	Dev Board Housing	$ 1.00	1	$ 1.00
7	Resistors	Resistors of varying values	$ 0.02	20	$ 0.40
				Total:	$ 93.26

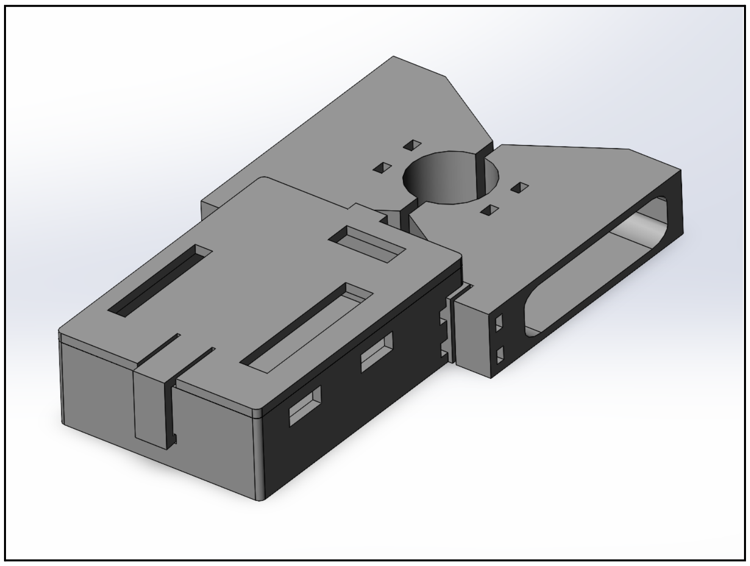
I. Appendix I: 3D Printed Mounting Hardware

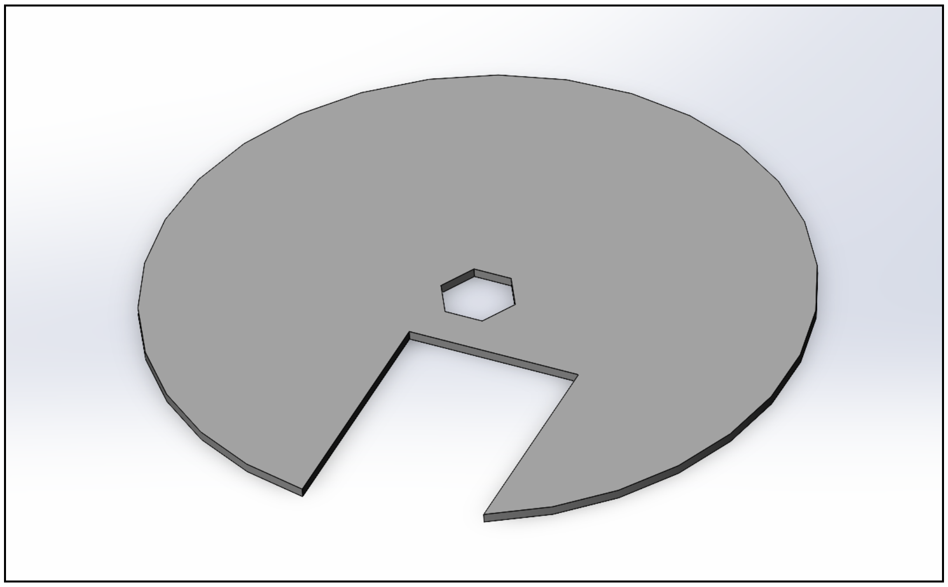
Microprocessor container: This case has holes for all necessary wire connections. The container and lid snap together. The container is held to the battery mounts using zip ties threaded through both parts.



(One half of the) Battery mount: The rechargeable battery slips into the cutout in the part and is secured with 2 zip ties. Because we had two batteries, we attempted to even the weight distribution on the wheel by mounting one on either side of the wheel hub.



Entire wheel hub assembly (spins with the wheel)



Timing Disc Plate: The timing disc was printed out and taped to this plastic disc for mounting. The notch is cut out to make space for the bicycle's front fork. The hexagon hole is a press fit with an existing nut on the wheel hub. Note that the timing disc does not spin with the wheel.

J. Appendix J: Jack's Ideas for Improvement

1. We were only able to achieve a resolution of 36 updates to the LED strip per revolution. I think that we could improve the resolution by using a timer to measure the time between updates, dividing this time by 2 or 3, and then triggering LED updates between the signal from the photoelectric sensor. This way we could potentially double, triple, or even quadruple the resolution of the image.
2. We also noticed that the color accuracy of the LEDs is not very good. Even giving the LEDs a brightness level of 1 makes them quite bright. Because of this, only images with a few, well-defined colors work well with the current system. I believe that if we had more time we could have implemented some kind of filter on the colors to make the colors more defined.