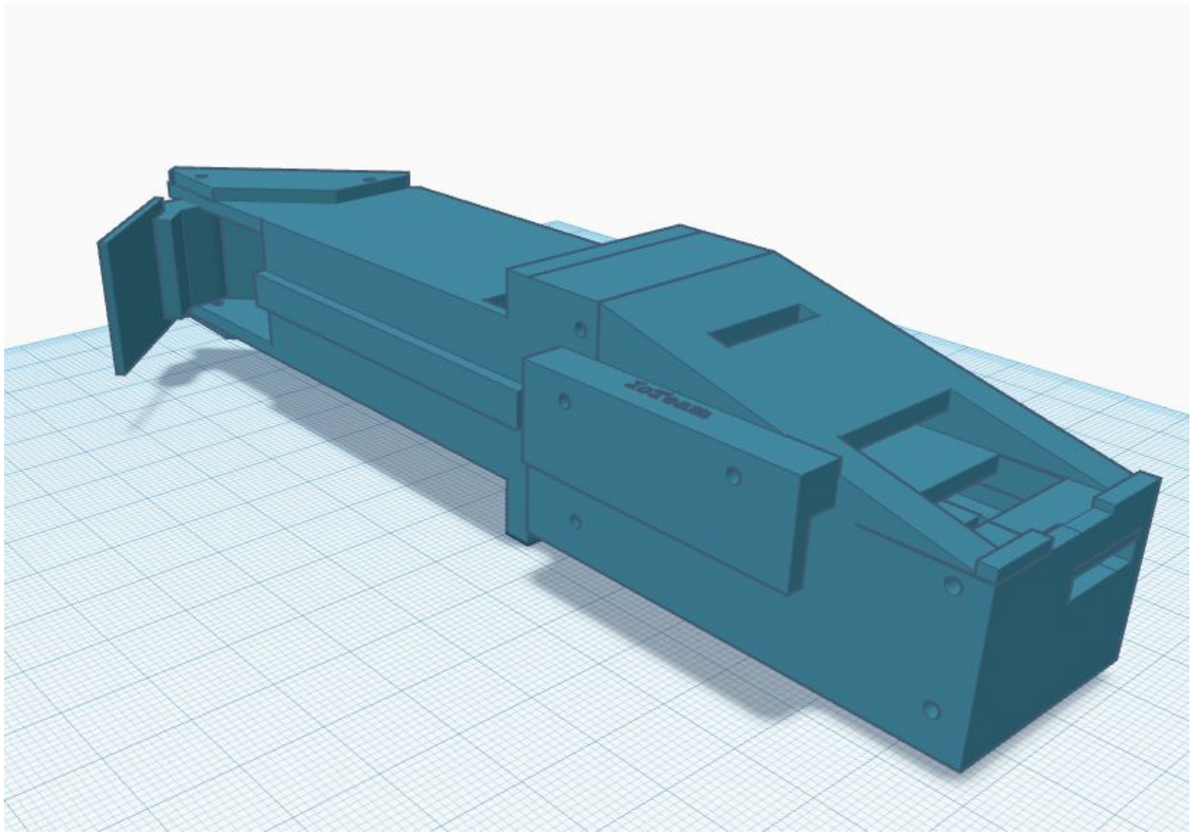
CS498 IoT Final Project Report “TheIoGlass”

Connor Burken (cburke8), Xiaohan Tian (xtian13)



Part 1: Motivation

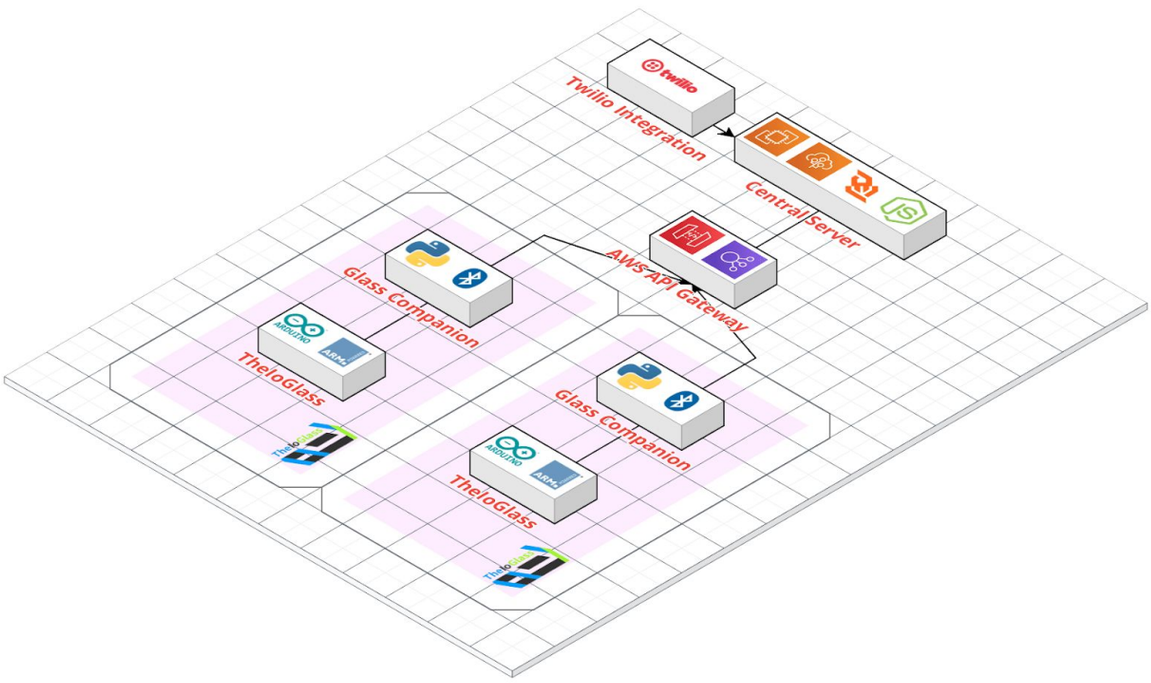
On April 15th, 2013, Google introduced a product originally developed by Google X named “Glass Explorer” known as “Google Glass” to the public, this product was released as a prototype on May 15th, 2014 for a limited period with a $1,500 price tag.
The “Glass Explorer” or “Google Glass” featured a dual core 1.2GHz Cortex A9 SoC chip “OMAP 4430” and a 630x360 display. It supports WIFI, Bluetooth and Micro-USB connection to the outer world.
As the name suggested, users can just simply put on the Google Glass as a “Glass” on their head and it will project a small screen on the user's left eye. Internally, Google Glass runs Android 4.4 “Ice Cream Sandwich” which provides massive various opportunities for the applications.
Subsequently, Google released the 2nd generation of the Google Glass prototype with upgraded memory and prescription frames support, but Google announced in 2015 that it will stop producing the Google Glass prototype. Although Google released another 2 versions of the Google Glass Enterprise Edition in 2017 and 2019, they are more focused on the Enterprise level and left the mainstream market.

We believe a glass-like HUD (heads up display) IoT device is extremely useful. For example, for a doctor, when he/she is performing some operations which are both hand occupied, a HUD will be really helpful. Another example would be for the people who are working in a dangerous environment or have their protection suit and gloves on, in scenarios like this, it is almost impossible to free their hands to reach any regular mobile device to check the screen to check numbers or read instant push messages. The augmented reality functionality of the HUD, can also be used for recreational purposes like laser tag “health” monitoring or in sports to monitor heart rate and speed while in the game.

We feel the main problem of the Google Glass is the price. For a device with a price tag of $1,500, not very many individual users are willing to make the purchase. Even for the enterprises, if they would like to conduct a massive deployment, the cost will be very considerate.

Our goal is to create a practical and budget digital glass-like HUD (heads up display) system, with various sensors, able to process and display information and graphic in the real time, and most importantly the cost should be **less than $100**.

Part 2. Technical Approach



(**Figure 1:** System Architecture)

The system generally includes 3 different components:

- The Central Server running on the cloud
- TheIoGlass End-Device, which includes:
    - The Glass itself
    - The Glass Companion App running on a device which has Internet Access
- 3rd Party Service Integration Components

The Central Server is running on AWS Elastic Beanstalk Service, right now it’s powered by two t2.micro size EC2 instances, and an API Gateway and a load balancer seat in front of them.
The Glass Companion is supposed to be an app running on any device (e.g. Computer, cell phone, etc.,) which has Internet connection. The Glass Companion will be in charge of communicating with both the end device and the central server. Due to time constraints, at this moment we only support running the Glass Companion on macOS.
The end device is a 3D printed glass-like wearable device featuring an ARM based SoC board and a 0.96” micro display which projects a virtual 2.62” screen in front of the user's eyes at around 25 cm distance.
Besides, our system also supports standard Webhooks which can be used to integrate 3rd party services. At this moment, we have integrated Twilio service into our system, users can receive push notifications via regular SMS messages.

Part 3. Implementation Details

3.1 The Central Server

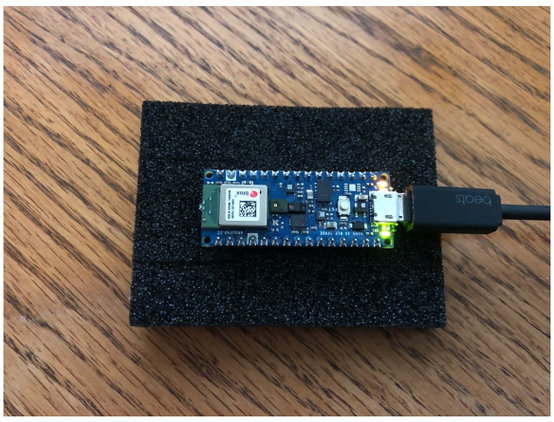
The central server is running on AWS Elastic Beanstalk Service, it’s been configured to use two t2.mirco sized AWS EC2 instances as a cluster with auto-scale up enabled in the load balancer. Additionally we have also leveraged AWS API Gateway to sit in front of the Beanstalk, allow extra security and additional management ability.
Currently the central server is coded using Node.js with Express.js framework with websocket support.

3.2 The Glass Companion

The Glass Companion App is designed to be a bridge between the end-device and the central server, like an agent. Theoretically any device with Bluetooth 4.0+ should be able to host the companion app. Due to the time constraint, for prototyping, we only implemented the Companion App for macOS (requires Bluetooth 4.0+ hardware support, any iMac/Macbook Pro after 2014 should be able to handle it).
The prototype version of the Glass Companion app is written using Python, when start, it will pair with the end device and then contact the central server to register it, and then establish a websocket tunnel with it.

3.3 TheloGlass

3.3.1 Hardware Specification



(**Figure 2:** An Arduino NANO 33 BLE Sense in prototyping, connecting to host via Micro-USB)

We are using an Arduino NANO 33 BLE Sense (ABX00031) as our main SoC. It features an Arm Cortex-M4 CPU clocked at 64 MHz, and it has a 1MB Flash Memory and 256KB SRAM. Besides, it has Bluetooth Low Energy (BLE) support, we use BLE to communicate with The Glass Companion app.

Besides Arduino NANO 33 BLE Sense includes various sensors on board including a built-in microphone. The power consumption is very ideal, the working voltage is 3.3V and current is less than 20mA. All those features are compacted within a 45mm × 18mm size. This makes us feel it is the right choice to power our device.

3.3.2 Optical System

The optical system consists of a 0.96” OLED display, a lens with 3 times magnification, a mirror, and a 35% reflective film for transparent display reflection. How it works, is that the lens is placed 62.5 mm from the screen which to the users eye makes the screen appear to be 25 - 30 cm away from the eye when in actuality it is only 6 - 8 cms away. The reason for that, is that the human eye can only focus on objects that are at least 20 - 25 cm away. This is similar to if you hold a book too close, the words become blurry.

Here are the equations that we used to calculate the lens distance from the display:

Image distance =

$$i = \frac{1}{\left(\frac{1}{foc\ len} - \frac{1}{obj}\right)}$$

Formula 1. Perceived Distance Calculation with Lens Placement

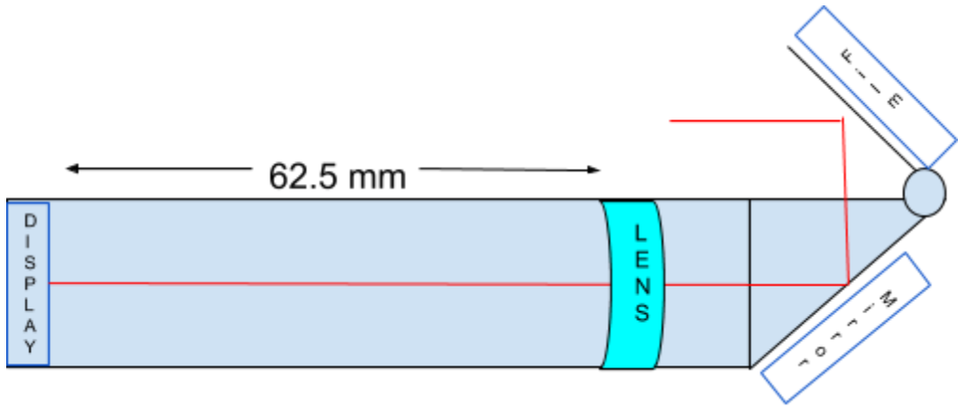
Where,

foc len - is the focal length of the lens

Obj - object distance

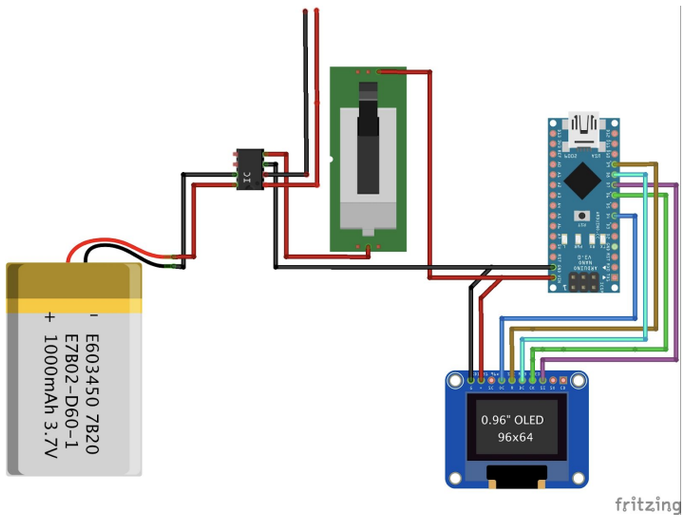
i - perceived distance

Our lens has a focal length of 83.3 mm, and our desired image distance is 250 mm. Solving for obj results in a perceived distance 62.5 mm. Please, see the following diagram to see how it works in our design:



(**Figure 3:** Optical Considerations in IoGlass Design)

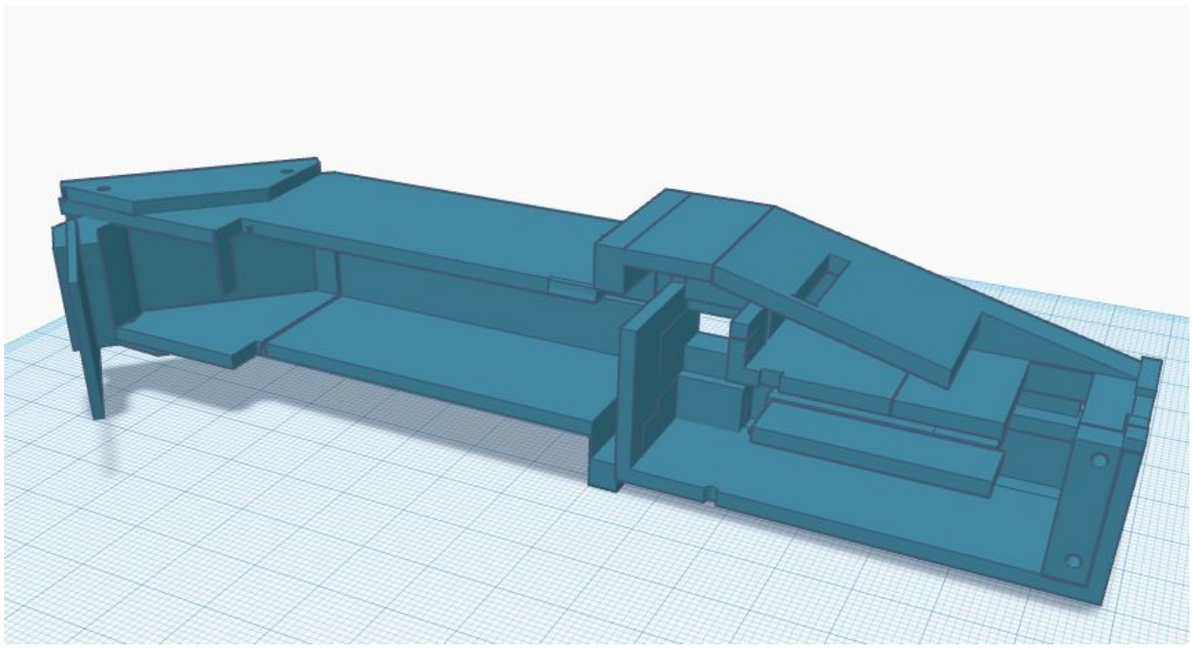
3.3.3 Circuit Design



(**Figure 4:** Circuit Design for Individual IoGlass)

For the circuit design, we needed to connect the OLED display to the SPI connection on the Arduino Nano. For some reason the SPI pins stopped working on our Arduino Nano so we switched to the software SPI implementation. In the diagram above, the SSD1331 only requires connections for Source Clock, MOSI (Master Out Slave In), DC, CS (Chip Select), and Reset. From a power perspective, the Arduino Nano is then connected to the switch for input power which is connected to the battery charging circuit. This allows us to disable power to the Arduino Nano while flashing new code onto the module, and save battery power while the Glass is not in use. The two stray wires on the diagram above are for connecting to a power supply for charging the battery, and the battery is connected to the VBAT-IN and GND pins on the charging circuit (labeled IC above).

3.3.4 Enclosure Design



(**Figure 5:** Cross-Sectional View of Enclosure Design for Individual IoGlass)

For the enclosure design we decided to use a 3D printer. We chose to use a 3D printer because of the flexibility it provides in terms of details. For this project we have very specific parts and not a ton of space to work with so space efficiency was key. Also, the PLA plastic the 3D printer uses is robust to protect the components. The 3D software tool that was used is TinkerCad, which is a fun and stream-lined way to design 3D objects.

The enclosure was split into 6 distinct parts, and screws were used to adhere the components together. In the figure, above the top shelf of the back housing is where the Arduino Nano is placed. The Arduino nano is placed there, because it conveniently allows the user access to the microcontrollers micro-usb port, and reset button. It also exposes some of the sensors built-in to the Nano BLE Sense 33 for future use. On the bottom the battery is placed, it is the heaviest component so lowering the center of gravity helps with the stability of the glass as it is hooked onto the user's glasses. On the second shelf the power circuit is placed, this area was chosen as a center of the enclosure, and allows for easy connection to the Nano and battery. Sandwiched between the optical shaft, and the back enclosure is where the OLED screen lives. This allows for a reduction of size of the optical shaft and only shows the “screen” part of the OLED pcb. On the end of the optical shaft the lens, mirror, and transparent film come together to display the image to the user.

There were a few challenges with the 3D printing, one was getting the files to be just right. It can be difficult to get things to align and fit as you design them too. Another issue was that we did not have the right length of screw to connect the lens shaft to the back enclosure, so we had to use electrical tape. Other than that it was a fun experience to prototype using a 3D printer.

3.3.5 Software Implementation

3.3.5.1 OLED Display

For the programming of the OLED display, the Adafruit SSD1331 library makes it easy. The library establishes a SPI connection with the screen, the library provides simple functions for writing text in various fonts and colors. It also provides functions for generating shapes and animations.

```
//SSD1331 Adafruit Lib Example Code
Adafruit_SSD1331 display = Adafruit_SSD1331(cs, dc, mosi, sclk, rst); // Init display
display.fillScreen(BLACK); // Clear screen, fill with a color
display.setCursor(0, 5); // Adjust the position to begin writing text
display.setTextColor(RED); //Set text color
display.setTextSize(1); //Set font size
display.println("Hello World!"); //display text
```

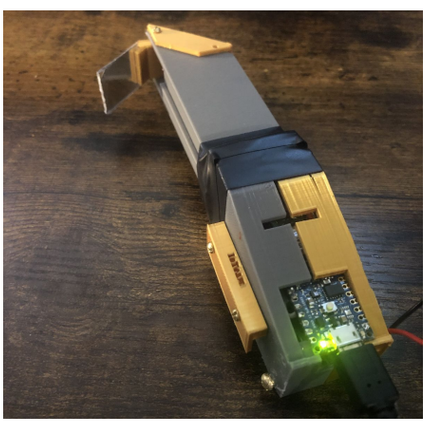
From a design perspective, the display portion of the code is on an idle screen that displays the temperature, while the bluetooth portion is polling for a new text message. Once a text message is received the user will see the message popup on their glass, the message will then disappear after 4-5 seconds. In the future, we would like to use a method of acknowledging incoming messages, either via a voice command using the Nano Sense's built-in microphone or monitoring for a head nod using the Nano Sense's built-in accelerometers.

3.4 3rd Party Service Integration

Since we support Webhook on our central server, we decided to expand our features by adopting 3rd party service. Right now we have integrated Twilio service, which allows external users to communicate with our end-device directly via regular SMS in their cell phone.

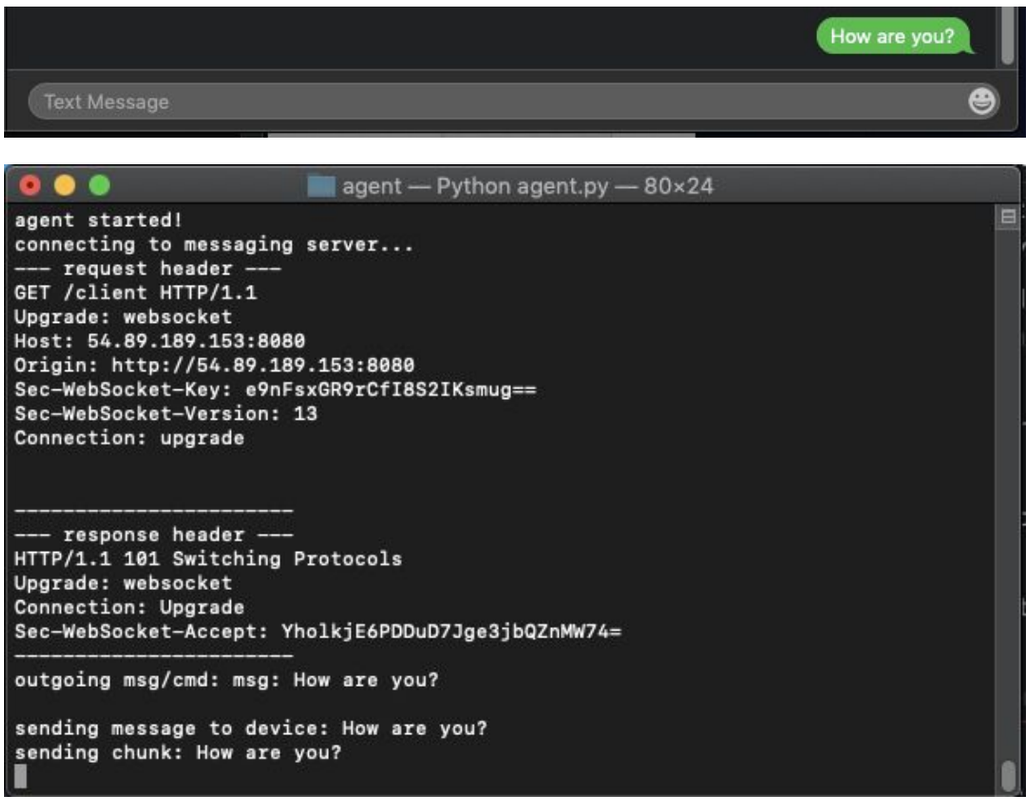
Part 4. Result

Here are the results of our final project. We were able to successfully 3D print and build the prototype as seen below:



(**Figure 6:** Final project prototype)

Here is a demonstration of our prototype in action:



(**Figure 7:** Sending a text message to the Twilio Account)

(**Figure 8:** Idle Screen -> Msg -> Return to Idle Screen on HUD)

We are very excited about the untapped potential of this project. We were able to build the whole prototype for under $100 which is 1/15th of the price of the original Glass. We plan to incorporate Tensorflow lite into the project, and use more of the onboard sensors on the Nano BLE sense to do more intricate operations in the future.

Credit:

Connor Burken (cburke8)

- Arduino NANO 33 BLE Sense Development
- TheIoGlass Hardware (SoC, Sensors, Micro-Display Module) & Software Development
- 3D Printed Glass Case Design & Implementation
- Glass Optical System Design & Implementation

Xiaohan Tian (xtian13)

- Arduino NANO 33 BLE Sense Development
- Central Server & Glass Companion App Development
- TheIoGlass Bluetooth Low Energy Module Communication
- Cardboard Glass Case Prototyping

Attribution / References:

- Heavily inspired by the Glass made developed by Alain Mauer
    - Influenced by his prototype and optical system
    - https://hackaday.io/project/12211-arduino-glasses-a-hmd-for-multimeter
- Optical formulas http://hyperphysics.phy-astr.gsu.edu/hbase/geoopt/image4.html
- Bluetooth LE:
    - https://rootsaid.com/arduino-ble-example/
    - https://github.com/adafruit/Adafruit_Blinka_bleio
    - https://github.com/adafruit/Adafruit_CircuitPython_BLE/
    - https://forum.arduino.cc/index.php?topic=432210.0
- AWS Elastic Beanstalk:
    - https://aws.amazon.com/elasticbeanstalk/
    - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-getting-started.html
- AWS Elastic Load Balancer
    - https://aws.amazon.com/elasticloadbalancing/?elb-whats-new.sort-by=item.additionalFields.postDateTime&elb-whats-new.sort-order=desc
- AWS API Gateway
    - https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-develop-routes.html
    - https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api.html