HUDWare

Sanghoon Lee, EE, David Meschisen, EE, Jori M. Platt, EE, Minwo Wang, EE

***Abstract*—HUDWare is an augmented reality (AR) heads-up-display (HUD) device that connects a skier to their mobile device. HUDWare uses wink detection to allow the user to manipulate the device to view data like messages or the weather within their ski goggles distraction-free. It is an attachment that connects to the skier's existing goggles to both display data and receive input from the user. Information generation is handled by the mobile phone application which takes in the control signals from the wink detector and outputs text for data values.**

I. Introduction

BEING alone with nature is one of the great things about skiing. The downside to this solitude is isolation. If a friend or family member is trying to reach the skier, the skier will remain blissfully unaware. Even if they were to know that a message has been sent, they would need to then go through the trouble of reading the message in the middle of the run.

Skiers and snowboarders commonly embrace the cold and blistering weather to tackle difficult slopes and terrain. When they head out, they take their cell phones. This could be for safety, for music, or for the many features and applications that now can add to the skiing experience. Applications for mobile devices can now track a skier's speed, distance traveled, and location. Unfortunately, all this information is locked away until the skier returns to base. Live statistics are unachievable due to the inaccessibility of mobile devices while skiing. In order to change a song, a skier will have to remove their glove in the freezing temperature, dig through their coat to find their phone, and then attempt to navigate the menu with shivering fingers, before reversing the process and placing the phone back into their coat pocket. If someone is trying to reach them with their phone, it is unlikely that they will even notice. Calls and messages ring uselessly against the vibrations and noise of the slopes. If the skier is lucky enough to notice their phone, they must stop their run, quickly remove their gloves, and find their phones to respond, again exposing their fingers in unpleasantly cold temperatures.

The successful demonstration of HUDWare is a stepping stone for AR-based technologies to become more realizable in the future. It aims to set an example that such products can be inexpensively fabricated and still maintain robust performance in tough environments. In a world where smartphones are becoming more prevalent among many people, there is a need for such a device to seamlessly integrate within a person's daily schedule to keep up with notifications and events. Essentially, HUDWare shows that there exists another medium for smartphones to interact with a user in a passive manner. The user will not have to divest much attention to the phone while still be able to control it using simple gestures.

There are already devices that can send a display image based on information from a mobile device, or preset settings. These devices however are only one directional, information is sent to the user, but the user cannot change the type of information sent. Heavily interactive displays for certain AR devices are cumbersome and difficult to integrate into an active setting like skiing. Similar systems include Google Glass and the Microsoft HoloLens [1]. Both systems fail to be practical for a skiing HUD due to their price and impractical nature. Google Glass uses a touch bar and voice commands while the Microsoft HoloLens requires hand gestures, both which would be difficult to manage with gloves on, let alone while skiing. Vocal commands and hand gestures are inconvenient when the user's voice is muffled by a scarf and their hands by gloves. In addition to being impractical to use, AR devices are expensive and do not lend themselves to the rigor of athletic activities. To be integrated into the skiing, or any active and quick paced environment, they need to be hands-free, durable, and relatively inexpensive.

At its core, our goal for a HUD for skiing would need to be able to take information from the user quickly and in real time, use that information to control the device and the information, and remain unobtrusive throughout the whole process. The device will need to display relevant and important information, have a reasonable battery life, and be easily integrated into the skier's current gear. More specifically, this device must not add significantly more weight to the user's helmet (of about 100g, which is considerably lightweight). It also must withstand typical skiing conditions, which has a temperature range of about -20 to 30℃. The battery must provide enough power to run about 4 hours, which will allow it to be feasible for at least a half days' worth of skiing. Sending control signals at a minimum of 2 frames per second (FPS) allows for smooth instantaneous updating of information on the goggles. Most importantly, for the sake of the user's safety, the AR projection must not impede the user's sight. Additional information regarding requirements for our device is provided in Table 1. By combining these features into a single device, we will be able to enhance the skiing experience as well as advance practical AR technology.

S. Lee from Lexington, MA (email: sandlee@umass.edu)
D. Meschisen from Acton, MA (email: dmeschisen@umass.edu)
J. M. Platt from Chesterfield, NH (email: jplatt@umass.edu)
M. Wang from Amherst, MA (email: mwang@umass.edu)

Table 1
GENERAL REQUIREMENTS

Requirement	Specification
Lightweight attachment	Less than 100g
Operable in reasonable temperatures	Temperature range -20 to 30℃
Reasonable battery life	4 hours
Quick feedback	Control signals at 2 FPS minimum
Unobtrusive	Unobstructed view

II. Design

A. Overview

HUDWare will bridge the gap between the skier and the phone with a two-part system that displays the information from the mobile device and user interface that allows the skier to manage the phone. The display system begins at the phone application which collects kinematic data, messages alerts, phone calls, and music information. This information is transmitted through an open source Programmable Interface Controller (PIC) microcontroller board (IOIO, pronounced "yo-yo") to a microcontroller (MCU) which drives an Organic Light Emitting Diode (OLED) display. This display is reflected off a transparent material to give the illusion of a heads-up-display in the skier's field of vision. The user can then control the interface by winking either the left or the right eye. This is picked up by a camera located in the center of the device above the field of view of the skier. The microprocessor then processes the image to determine if the user closed only one eye. The result of this analysis is used to generate control signals, which are then sent back to the mobile device.

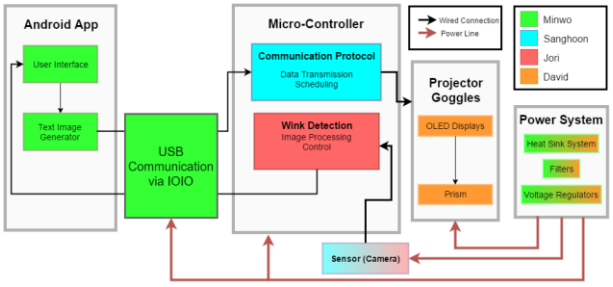

Figure 1. Block Diagram of system components

Figure 1 displays the different subsystems that comprise HUDWare. Each component is color coded to the engineer designing the subsystem. The Android application takes in control signals and feeds text data to the IOIO device. The IOIO handles communication between the microcontroller and the mobile application via a wired USB connection. The MCU handles both driving the projector display and analyzing the images captured by the camera. The wink detector is an image processing algorithm that takes the images from the camera, determines if one eye is closed, and sends corresponding control signals back to the IOIO. The projector display uses an OLED screen and a prism-like structure to create an image of relevant information appear in the skier's field of view. The power system feeds into both the MCU and the IOIO. By extension of being powered by the MCU, this power system will also handle the demands of both the projector display and the camera. Finally, all the components on the right of the diagram (MCU, camera, projector, and power system) will be housed either in or around the goggle attachment (not shown).

B. Wink Detector

The user control system is a wink detector system. This works in conjunction with the camera subsystem to detect if the user is closing one and only one eye as well as which eye is closed. The camera takes a picture of both eyes which is fed to the MCU. The MCU then uses an image processing technique called a Hough Circle Transform. This detects any circles in an image which would correspond to an iris or pupil. Each circle detected has defined center points and radii. If the circle center coordinates places it on the left side of the image, this would correspond to a left eye open, or a right wink. Likewise, a circle on the right side indicates a left wink. When only one circle is detected, the wink detector will turn on a corresponding output pin which is sent as a control signal to the IOIO device.

Alternative designs for this system were considered. Initially, we planned to use a sample image of a closed eye and run a similarity metric using a calibration algorithm; this however was computationally heavy and not accurate. An eye tracking system was also considered. While this would give us more control over the device, concerns rose over the computational costs and thus the approach was abandoned. Using our current system, we would be able to detect if both eyes are closed and we could use that as an additional user input, but that system runs the risk of detecting normal blinks instead of control winks. If we used two frames to ensure there were no accidental blinks, then the system would force the user to keep their eyes closed for longer which would quickly become dangerous. Given all these concerns, we decided to pursue only a wink detector algorithm using the Hough Circle Transform.

A Hough Circle Transform is a computer vision technique that was taught in ECE 597IP Image Processing at the University of Massachusetts. In principle, the parametric equations $x=a+R\cos(\theta)$ and $y=b+R\sin(\theta)$ describe a circle [2]. The transform works by describing each circle in terms of variables a and b instead of x and y and letting x and y become constants. To detect a circle in an image, an edge detector is used first, in this case we used a canny edge detector. The algorithm then cycles through the image and increments the corresponding location in the a and b coordinate map each time it meets an edge in an x and y location. The peaks of this coordinate map correspond to the x and y locations of the centers of the circles with radius R. By repeating this process, we can search for a range of radius values. By altering the sensitivity of the edge detector and the threshold for the peaks in the a and b coordinate map, the algorithm can be calibrated to accurately detect pupils and irises. In addition, the wink detector can be set to only accept one circle within a selected pixel distance. Other calibration parameters include the minimum and maximum radii allowed, and the amount of blurring before the edge detector. Blurring allows the algorithm to reduce noise at

the cost of computational expense and loss of detail.

Table 2
WINK DETECTOR REQUIREMENTS AND SPECIFICATIONS

Requirement	Specification
Accurate	Reads 90% of the frames correctly
Quick	Reads each frame in 0.25s or less
Responsive	Minimum 2 FPS
Connects to the IOIO	Controls 2 binary output pins for control signals

Table 2 demonstrates the individual requirements for which this subsystem is responsible. This system will be the control interface for the user, so to maximize ease of use, the system will need to be quick and accurate. The image processing portion of the interface will need to run in less than a quarter of a second to allow for the time it takes for the camera to capture an image. Currently, the system runs in around 0.143 seconds and identifies an image correctly roughly 85% of the time. This was calculated by running a parallel Python performance script that allowed us to measure certain quantities like time elapsed and amount of data used. The accuracy was determined by taking a series of pictures of a configuration of eyes (both open, one closed, both closed) and comparing the results when sent through the wink detection algorithm. The algorithm detects irises in the image and sends the corresponding output signals. The accuracy will change as we move the camera to the prototype attachment and fix the calibration. Once a final calibration setting is reached, basic statistical null hypothesis testing will occur to ensure that the system is reaching the desired 90% success rate.

C. Camera

The camera subsystem will function as the input to the wink detector. It will be mounted within the goggles above the user's field of view. In this location, it will capture both user's eyes and send images at a fixed rate to the microcontroller. Requirements for this subsystem include capturing both eyes, and capturing an image within a quarter of a second. The last requirement is necessary to achieve the two frames per second requirement of the user interface specified in the wink detector block.

The camera chosen is a Raspberry Pi Camera Board v2. This has a resolution of 8 megapixels, a light weight of 3.4g with the cable, and a small profile of 25mm x 23mm x 19mm [3]. The camera contains a high-resolution Sony IMX219 Image sensor [4]. This is a high-speed sensor that has a lens shading correction function and two exposure controls to support Binning Multiplexed Exposure HDR (BME HDR). The lens shading correction suppresses optical unevenness. The BME HDR is an image processing technique that synthesizes both dark and bright aspects in the photo between short and long exposure images. Using these two different exposures, the camera can minimize the effects of high contrast. Table 3 below shows the relationship between image size and frame rate as provided by the datasheet.

Table 3
RELATIONSHIP BETWEEN IMAGE SIZE AND MAX FRAME RATE

Image Size	Frame Rate	Notes
3280 x 2464	30 fps	
1640 x 1232	120 fps	
1408 x 792	180 fps	High-sensitivity mode
1280 x 720	198 fps	High-sensitivity mode
960 x 540	240 fps	High-sensitivity mode

Due to the proximity of the camera to the user's face, we are using a fisheye lens to widen the aperture to approximately 180 degrees so that we will be able to fully capture both eyes. Figures 2 and 3 demonstrate the effectiveness of the lens. Each picture was taken with the camera at the distance it will need to be to fit within the goggle attachment. The fisheye lens allows us to easily capture both eyes and fulfill our requirement.

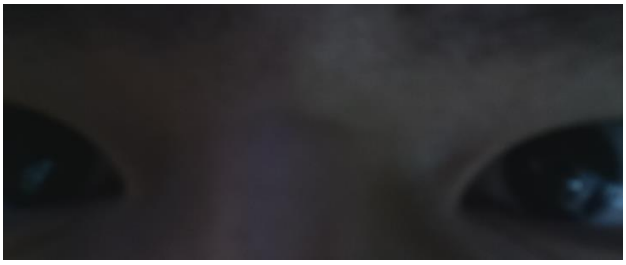

Figure 2. Captured image without the fisheye lens

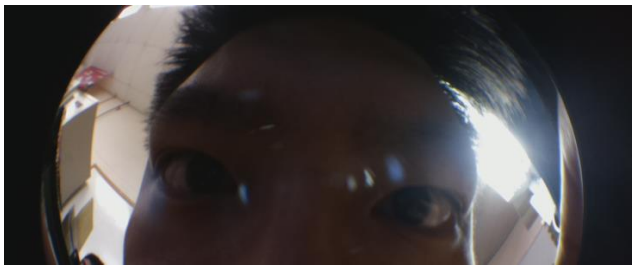

Figure 3. Captured image with the fisheye lens

In order to demonstrate the functionality of the camera, we have written and run test Python scripts that allow us to capture photos and save them on the memory of the microcontroller. One of these test codes included a timer to keep track of the time elapsed between image capture and image retrieval. Currently, the system stores the image in memory which increases the time detected by the code, 0.65s. We will eventually move the image directly to the wink detector without storing it, so this time is expected to be significantly reduced. If this fails to cause the subsystem to reach the requirement, we can reduce the size of the image to increase the speed.

D. Microcontroller

The microcontroller will serve as the central hub between all the other subsystems. The MCU drives the OLED for the display, runs the wink detector, drives the camera, and communicates to the mobile device via the IOIO. This device receives serial input from the IOIO and images from the camera. In return, it outputs binary I/O pin values to the IOIO and drives the display. The MCU will need to implement a scheduling protocol to manage the different tasks through multi-threading.

We are using a Raspberry Pi Zero. This is a small device, only

65mm x 30mm x 5mm. This allows it to be easily used for a wearable device as it is not bulky or heavy (9g). The Zero has a BCM2835 processor with a 1 GHz ARM1176 core. It has 512 MB of RAM which will be enough to run both the wink detector and the OLED display code [].

We are currently using a Raspberry Pi B+ to prototype. This is a slower device with its 700 MHz Low Power ARM1176JZFS processor, but because of its additional USB ports and HDMI port, it is easier to quickly prototype on other components []. Since both devices have the same Linux OS, we do not anticipate too many troubles shifting code from one MCU to the other.

Table 4
B+ PERFORMANCE STATISTICS

Performance Statistic	Value
Power Consumption	.5-.7W
Memory Usage (1 image)	9%-10% Max
Memory Usage (5 images)	12%-13% Max
Memory Usage (Drive OLED)	2.5% Max
Running Temperature	39℃

As shown in Table 4 above, the B+ runs on low power consumption, while utilizing less than a fifth of the memory to run processes. More importantly, the B+ does not heat up considerably, maintaining about 39°C while running. This will be a crucial aspect in our prototype design, since it is not desirable to have a very hot component located in proximity to the user. Again, these performance statistics will be remeasured when the Zero is implemented, since the Zero contains a different processor than the B+.

To test the communication aspect of the microcontroller, first a serial data communication test was performed between the laptop terminal and the B+. Text data was sent in both directions between the two devices, demonstrating the capability to communicate with the IOIO serially. The ultimate goal will be to transmit through Serial Peripheral Interface (SPI), since it is able to provide higher data rates. Since the MCU will need to drive both the OLED and the wink detector, some form of scheduling will need to be implemented. A potential solution for this would be to add a second Zero. Another solution to this issue would be to implement multi-threading processes; this will be further explored before making a definitive switch to this idea.

Table 5
MICROCONTROLLER REQUIREMENTS

Requirement	Specification
Maintain communication between application and OLED	
Drive Pi Camera	
Support Wink Detector and OLED	
Does not overheat	Temperature remains less than 80℃
Works within reasonable temperature	Operating range: -20℃ to 30℃
Low power consumption	<1W
Memory constrained	<50% of RAM
Size	<80mm x 80mm x 10mm

The requirements for this subsystem are shown above in Table 5. As shown in the test results, the temperature requirements, memory constraints, and power consumptions have all been met with the B+. Additional testing will need to be performed once the subsystem is switched to the Zero.

E. Mobile Phone Application

The mobile phone application collects data from the phone and sends information through the MCU to display. It will receive input commands from the IOIO to adjust the information it is displaying or to control the phone. We used Android Studio to write the phone application for Android devices. The application is currently designed for the Huaiwei Honor 5x smartphone, which uses Google Android 5.1 OS [5]. The requirements of this subsystem are shown in Table 6.

Table 6
PHONE APPLICATION REQUIREMENTS

Requirement
User friendly interface
Gathers information from phone sensors
Sends messages and phone call data
Allows IOIO to manipulate the phone
Able to manipulate music

The phone application can currently turn music on and off from I/O pin input, displays information from the phone sensors, and is accessible to users as seen in Figure 4.

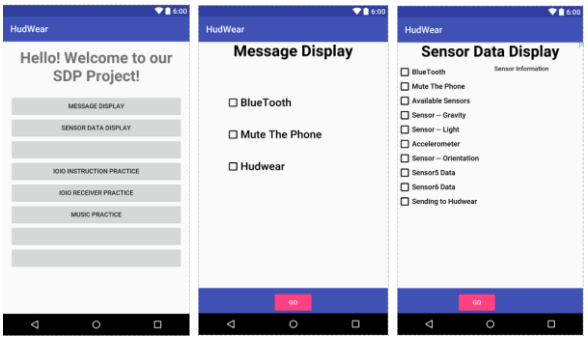

Figure 4. Main menu layout (left), Message display layout (center), Sensor display layout (right)

The layouts for the application have already been completed. We have implemented music, statistical data extraction, and part of the message display functionalities.

The music function allows the user to listen to songs and change the song sequence via I/O pins. First, we needed to add “MusicService” in “AndroidManifest.xml” which guarantees the application permissions it needs to do system level methods. We imported .mp3 files into the resource directory, allowing us to play music within the application. In the Java class, we created a media player to recall those songs. This media player can be controlled through I/O pins.

The sensor layout seen above shows the phone’s sensor sets, which we are using to display information about our surroundings. Using these sensors, we are able to detect light intensity, gravity, acceleration, and orientation. With GPS data

and calculations, we expect to be able to determine speed as well.

The message display function is still ongoing. We have an Android API to help the application show messages when they arrive. Currently however, our phone does not have service, so testing the system right now is difficult.

Figure 5 below shows the test circuit used for the application. This test circuit includes the IOIO device which will be described in more detail later. An external control circuit was used to simulate an I/O pin turning on and off. Using this system, we could control the phone via external switches.

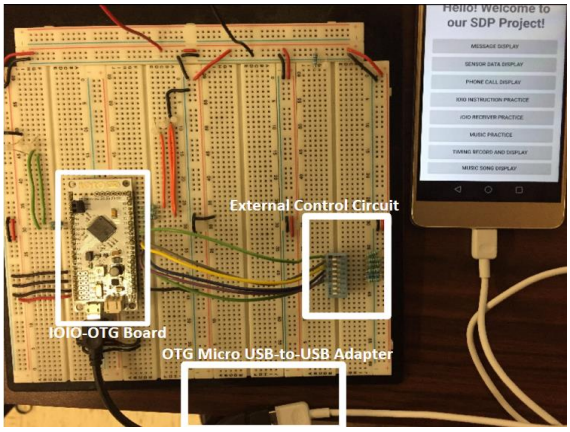

Figure 5. Phone Application, IOIO-OTG Board, and External Control Circuit

Like the MCU, this subsystem will also need to implement some protocol to handle taking and sending information. We are planning to use a multi-threading framework to re-write the phone application. This will allow such a complex software system the ability to be stable for scheduling, synchronization, and throwing exceptions.

F. IOIO-OTG

The IOIO-OTG board connects the mobile phone to the MCU. Using a wired USB connection, this device allows us to communicate at a fast rate. Usually with a mobile device, the phone is default set to slave in the master/slave paradigm. This device allows us to swap back and forth to fit the needs of the project. This subsystem needs to provide stable serial data transmission, read GPIO and analog inputs, and write GPIO outputs.

The Android IOIO-OTG board is a development board that uses a PIC microcontroller [6]. It provides two pairs of serial data communication, 46 GPIO pins, analog input, SPI, and I2C interfaces to handle external hardware. This provides enough functionality to handle all the requirements needed from this subsystem.

Table 6
IOIO PROTOCOLS AND SPEED

Protocol	Throughput	One-way Latency	Stability
Open Accessory	600 kB/S	1 ms	High
Android Debugging Bridge	300 kB/S	4 ms	Low

Table 6 shows the different speeds for the two protocols available on the IOIO. Clearly, the Android Open Accessory (AOA) protocol is more attractive. In addition to the speed and stability, with AOA we do not need to set the phone in USB debugging mode as we do for the Android Debugging Bridge (ADB). ADB is commonly used for downloading Android code from the desktop. We have implemented the IOIO transmitter by importing the IOIO API and the Utility library. The transmitter can control each GPIO pin and set them to either high or low from the phone application. It sends serial data at common baud rates, e.g. 9600, 19200, 38400, and 115200.

While input data was tested in the method described for the phone application, a test setup was created to transmit binary GPIO and serial byte data. Using an Atmega32 MCU, we sent serial data from the IOIO. The Atmega32 showed this information turning a series of LEDs on or off. This testing board is shown below in figure 6.

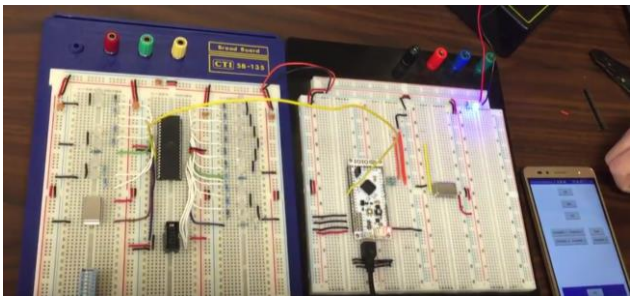

Figure 6. Test board for serial data transfer from IOIO

Alternative approaches to the IOIO device were considered for communicating between the application and the microcontroller. These options are shown in Table 7 with their respective bandwidth, range, and complexity. Bluetooth offers the lowest bandwidth but is straightforward to implement. Wi-Fi has the highest bandwidth, but would be difficult to implement due to protocols. Both Wi-Fi's and Bluetooth's data transmission can be unstable for dynamic situations like skiing. Since wireless connectivity was not necessary, the IOIO device proved to be the best option due to its stability and low complexity.

Table 7
COMMUNICATION LINK DESIGN ALTERNATIVES

Communication	Bandwidth	Range	Complexity
Bluetooth (v4.0)	800 kB/s	30 ft	Pairing Devices
Wi-Fi (802.11n)	11 MB/s	300 ft	Accessing Internet
IOIO-OTG	5 MB/s	Wired	Configuring Board

G. Display

The display for this device will be generated by an organic light emitting diode (OLED) display that reflects off a transparent material. An OLED works by having a series of thin organic films between two conductors. By applying current, light is emitted from the device [7]. The device we are using is driven by a SSD1351 chip. To provide necessary setup and functionality for the display, we are using a library from Github, py-guagette_master_2 [8]. Using this library, we can display

.png images as well as text and ASCII characters.

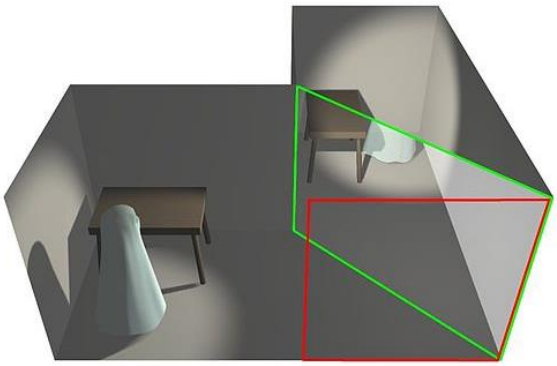

Figure 7. Pepper's Ghost Diagram

The AR display works on a principle called Pepper's Ghost shown in Figure 7 above [9]. This effect uses two light sources, in our case the natural light of the environment and the light from the OLED [10]. By placing a film in the green plane, some of the light from the OLED will appear to be directly in front of the user. This allows us to hide the OLED on the side of the goggles outside of the user's field of view and still create a transparent display. For this subsystem, we are using a sheet of LEXAN clear acrylic. The optimal angle for this sheet, or angle where the user sees total reflection, is 42.2° from the red plane [11].

Table 8
DISPLAY REQUIREMENTS

Requirement	Specification
Focused and Legible Display	
Brightness Minimum	120 nits
Target Range	200-300 nits
Operable at reasonable temperature	-20℃ – 30℃
Low power consumption	<1W

The requirements for this subsystem are shown above in Table 8. This display needs to be bright enough to be legible while skiing in the middle of the day and needs to be able to work under cold temperatures. Current measurements of the system's brightness are not yet recorded, though we will be using a light meter to ensure that the subsystem meets the specification. The display is not currently finalized and is only set up to provide a general idea as to the practicality of this setup.

A few different design alternatives were considered for this display. Texas Instruments has a Pico projector chipset that would have fit the requirements of the projector well, but the chipset itself was difficult to assemble due to their format as ball grid arrays. A transparent OLED was also considered, as it would provide the light required without needing a reflective film, but these proved impossible to procure. Given the complexity and ease of manipulation, an opaque OLED was chosen for the display.

H. Power System

The power system will support the HUDWare device for its outdoor setting. This system will need to supply power to the IOIO and MCU, and by extension, the camera and OLED. This subsystem will supply enough power and energy to run the device for four hours. Table 9 shows the power drain for each system.

Table 9
POWER BUDGET

Device	Power
MCU	0.5 - 0.7 W (Worst Case 1W)
OLED	1 mW (Worst Case 2 mW)
IOIO	2.5W
Total	3W (Worst Case 4W)

Using the above table and basic arithmetic, we can determine that we need 16 watt hours for this device. Given that one AA battery provides 3.1 watt hours [12], and that we will need to configure the batteries to provide the optimal voltage, we calculate that we would need at most six lithium ion AA batteries. A phone battery will most likely provide the power we need, but we do not currently have data for it. Most of the work for the power system is slated for the second semester.

III. PROJECT MANAGEMENT

Table 10
MDR GOALS

Goal	Specification	Achieved
Projector Display Assembled	Prototype Prism Constructed	Yes
	OLED Display Setup	Yes
Application Layout	Partial Functionality	Yes
Wink Detector	Wink Detection Algorithm	Yes
	Moved into microcontroller	Yes
MCU communication setup	Serial to MCU	Yes
	Serial from MCU	Yes

Table 10 lists the goals our team aimed to accomplish by the midway design review presentation date. Each subsystem had separate objectives for the MDR presentation. The projector display needed to have a basic system in place so that an image was visible to the user. This was not intended to be a final configuration so much as a proof of concept. To this end, a makeshift prototype was created that showed the OLED display image across the field of view of the user. Further work in this subsystem includes optimizing reflection, finalizing OLED placement, and converting text data into a display that can be shown to the user. While there is plenty of work to be done in this area, the display has been created to the point specified at the preliminary design review.

The mobile phone application subsystem needed to have partial functionality by the MDR date. At the time of the presentation, it would take in sensor data from the phone itself and display the information in the application user interface. The application was also able to control the phone in several different ways, such as muting the mobile device and playing different songs. More work will be done to improve upon the range of capabilities. Also worth including in this section is the work done with the IOIO device. While this was not specified in

the MDR deliverables, work in the IOIO allowed for communication to and from the phone application. This allows us to further the goal of communicating to the microcontroller unit. Further work in the IOIO subsystem would include connecting the device to the MCU and improving the communication speed.

There were two milestones in the wink detector subsystem. The first was a functional algorithm that would detect if one eye was closed in a picture of the user's face. The second portion of this subsystem was moving it from a computer into the MCU's system. Currently, both milestones have been reached and there is a functional wink detector inside of the MCU. In order to complete these tasks, the wink detector code needed to be built first on a computer. For ease of use, this was done in MATLAB. An equivalent program was developed in Python so that the microcontroller would able to run the code. Finally, this had to be imported into the microcontroller which involved calibration and importing the correct libraries. Like the mobile phone application and the IOIO device, the camera sensor subsystem falls closely in line with the functionality of the wink detector. Although no deliverables were specified, the camera is currently able to take a picture of both eyes from a distance that would fit comfortably within the goggles. Further work in both subsystems would include calibrating the wink detector, improving the speed of the whole algorithm, testing the system to ensure requirements have been met, and permanently fixing the camera to the prototype attachment.

While there are several components to the microcontroller subsystem, for the MDR presentation date, we specified that we would have communication to and from the MCU. As promised, we sent serial data from the microcontroller to a desktop computer and from the computer to the microcontroller. Future work on the MCU will allow it to communicate directly to the IOIO device, implement a scheduling protocol to alternate between wink detection and image display, move to the Raspberry Pi Zero from the B+, and finding a location for the MCU on the prototype attachment.

Two subsystems that were not addressed above include the prototype attachment and the power system. Quick designs were created to examine the possibilities of the attachment, but little work was done to this extent. Likewise, work done in the power system has been limited. A power budget was determined, along with analysis of how much power and energy the system will need, but most of the work is yet to be completed. Both subsystems were intentionally left until after MDR since the priority for the team was to get the more complicated individual parts completed. Further work will include designing a three-dimensional model of the goggle attachment, researching, and testing a power source, and combining the subsystems into the attachment.

The team is working together well. For the first semester, most of the work was done with individual subsystems, requiring each member to work independently. While the bulk of the work was done individually, when a team member had difficulty achieving a goal, other team members stepped in to assist. Most commonly, this duty fell to Lee, whose subsystem impacted everyone's, putting him in the unique position of being involved slightly in every subsystem. The team has been able to communicate with each other through both a group messaging platform as well as weekly meetings. The group meets once a week as a team to work on current action items, and again later in the week to discuss progress and goals with our faculty advisor. This allowed us to achieve our goals for the end of the semester, and hopefully our CDR goals as well.

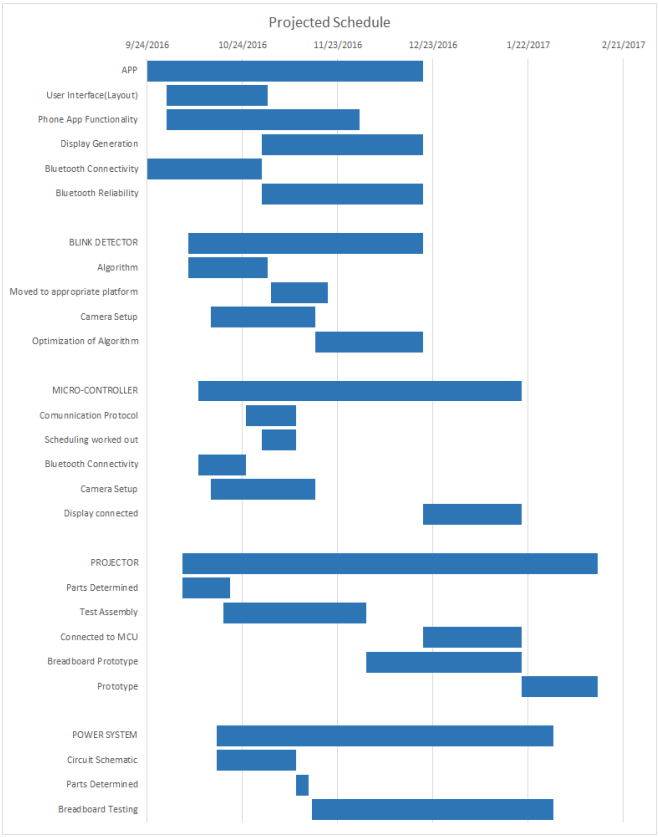

Figure 8: Schedule for the first semester

Figure 8 demonstrates the goals that we have set for the fall semester. This chart was created early in the project and understandably represents somewhat different goals than what we decided to pursue. For instance, the original design includes Bluetooth connectivity, a feature we decided to scrap in favor of a wired USB Connection. Some of the work we anticipated being able to achieve was not accomplished, such as scheduling on the microcontroller and any work on the power system. These were conscious decisions aided by the advice of our evaluators to focus our attention on more important priorities.

IV. Conclusion

HUDWare's current state is where we would like it to be as we end the Fall 2016 semester. We have almost every subsystem completed to a point where we will be able to combine them to form a complete system. Obviously, subsystems such as the prototype and power system will need to be created in this upcoming semester, but the main focus of this time period will

be combining the individual contributions of each team member.

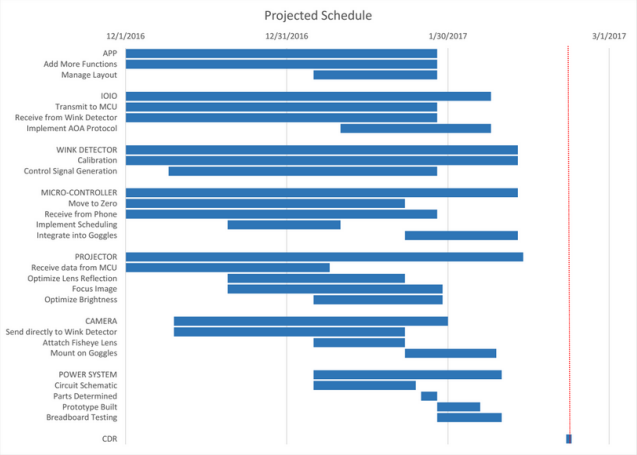

Figure 9: Revised schedule for second semester

Figure 9 is the revised schedule for this upcoming semester. All the work is scheduled to be completed well in advance to the comprehensive design review date. This allows us buffer time to test systems as well as giving us the ability to fix the system if something goes wrong. Almost all the work involves improving or combining subsystems. This clearly represents our goals for the upcoming semester. We would like to improve the subsystems we have, and combine them together to create a workable prototype. We are anticipating problems in combining the subsystems, but by working together we will be able to overcome any difficulty.

ACKNOWLEDGMENT

We would like to acknowledge the assistance of several members of the University of Massachusetts faculty. Professor Parente assisted us in the direction of the wink detector algorithm. Professor Holcomb aided with the microcontroller section. Professor Wolf provided much needed direction for the project as a whole given his experience in working with another Senior Design Project group that had similar ambitions as we have. In addition to these professors, we would like to thank John D'Errico of Eastman Chemical who provided us with a sample laminate for the projector medium.

REFERENCES

[1] "Microsoft HoloLens: Mixed Reality Blends Holograms with the Real World," Feb. 29, 2016. [Online]. Available: https://www.youtube.com/watch?v=Ic_M6WoRZ7k. [Accessed: Dec. 13, 2016].

[2] H. Rhody, "Lecture 10: Hough Circle Transform," Rochester Institute of Technology, 2005.

[3] "Raspberry Pi Camera Board v2 – 8 Megapixels," *adafruit.com*, [Online]. Availabe: https://www.adafruit.com/products/3099. [Accessed: Dec.13, 2016].

[4] "IMX219PQ," *sony-semicon.co*, [Online]. Available: http://www.sony-semicon.co.jp/products_en/new_pro/april_2014/imx219_e.html. [Accessed: Dec. 12, 2016].

[5] A. Dobie, "Honor 5X Specs," *androidcentral.com*, Jan. 6, 2016. [Online]. Available: http://www.androidcentral.com/honor-5x-specs. [Accessed: Dec. 12, 2016].

[6] Ytai. "IOIO Documentation," *github.com*, Oct. 14, 2016 [Online]. Available: https://github.com/ytai/ioio/wiki. [Accessed: Dec. 12, 2016].

[7] "OLED Introduction and Basic OLED Information," *oled-info.com*, [Online]. Available: http://www.oled-info.com/introduction. [Accessed: Dec. 8, 2016].

[8] Guyc, "Py-Gaugette," *github.com*, Nov. 5, 2016. [Online]. Available: https://github.com/guyc/py-gaugette. [Accessed: Dec. 8, 2016].

[9] "Pepper's Ghost," *wikipedia.com*, [Online]. Available: https://en.wikipedia.org/wiki/Pepper's_ghost. [Accessed: Nov. 12, 2016]

[10] B. Costa, "Explaining the Pepper's Ghost Illusion with Ray Optics," Jan. 11, 2016. *comsol.com*, [Online]. Available: https://www.comsol.com/blogs/explaining-the-peppers-ghost-illusion-with-ray-optics/. [Accessed: Nov 20, 2016]

[11] "Optical & Transmission Characteristics," *plexiglas.com*, 2000. [Online]. Available: http://www.plexiglas.com/export/sites/plexiglas/.content/medias/downloads/sheet-docs/plexiglas-optical-and-transmission-characteristics.pdf. [Accessed: Dec. 13, 2016].

[12] "Energy Tables," *allaboutbatteries.com*, [Online]. Available: http://www.allaboutbatteries.com/Energy-tables.html. [Accessed: Nov. 12, 2016].