Development for Design of Mechatronic Systems: Truer to the (Robot) MOBA Genre

Tony Qiu
Jacob M. Abel Summer 2019 Internship Report
Mechanical Engineering and Applied Mechanics
University of Pennsylvania

Abstract
Design of Mechatronic Systems (MEAM 510) introduces students to computer-controlled electromechanical systems, incorporating aspects of mechanical design, electronics and coding. Since the fall 2017 class, MEAM 510's final project has been based off a mix of Battle Bots and the Multiplayer Online Battle Arena (MOBA) game genre. There are a plethora of concepts that make up the MOBA genre; however, the first iteration of the project only included a few of the key elements such as basic robot to robot combat and attacking a nexus or base. The following year improved on the project by introducing a Top Hat system for each robot, automating the tracking of relevant game counting stats and providing visual indicators for the aforementioned. This summer was focused on adding more features to the final assignment to make the project more true to the MOBA genre while also exploring ways to improve existing components.

The following were explored and developed:

- Introducing 'fog of war' with OV7670 camera module
- Adding minimap feature through HTC Vive Base Station (Lighthouse)
- Experimenting with possible improvements to hit detection

OV7670 Camera

In the previous iterations of the robot MOBA final, students were able to see the entire playing field. In typical MOBAs, there exists a 'fog of war', which means each player has limited visibility to the direct surroundings of their robot. The goal was to mount cameras onto the robots and stream video over WiFi in real time; ultimately, the intention was that the students would be restricted to this first-person view to maneuver and orient their robots.

To accomplish this, the relatively cheap OV7670 camera module was used (Figure 1 below).

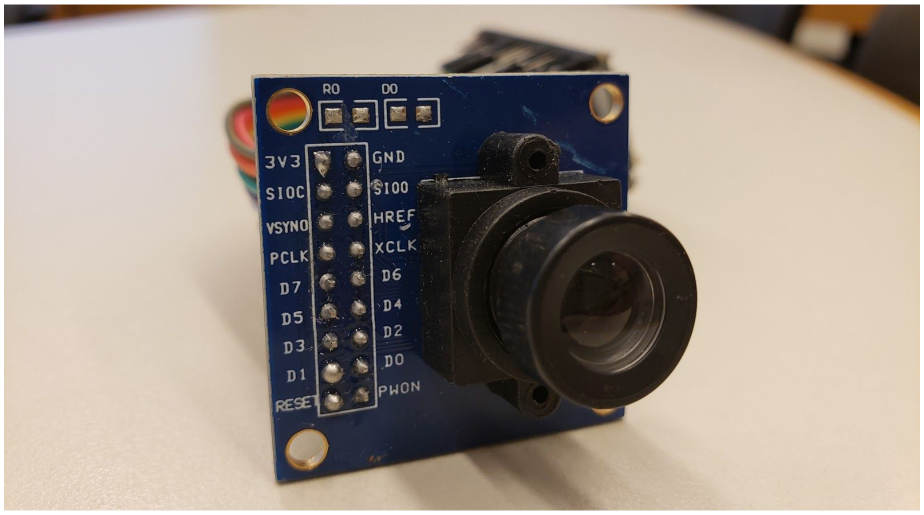


Figure 1: OV7670 camera module without FIFO

Researching previous development with the OV7670 and the ESP32, work done by Bitluni[1] and later, Mudassar Tamboli[2] were used. The method of projecting the OV7670 footage onto an oscilloscope was explored, and those results could be viewed below[3]; however, quality and fidelity of the produced image was poor.

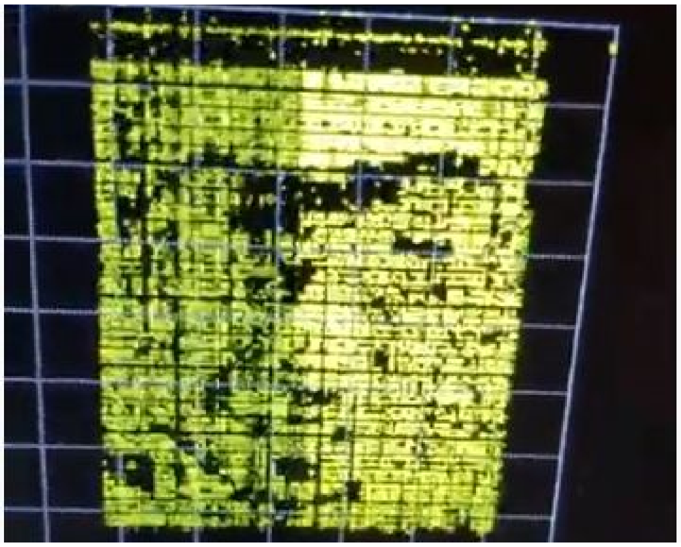
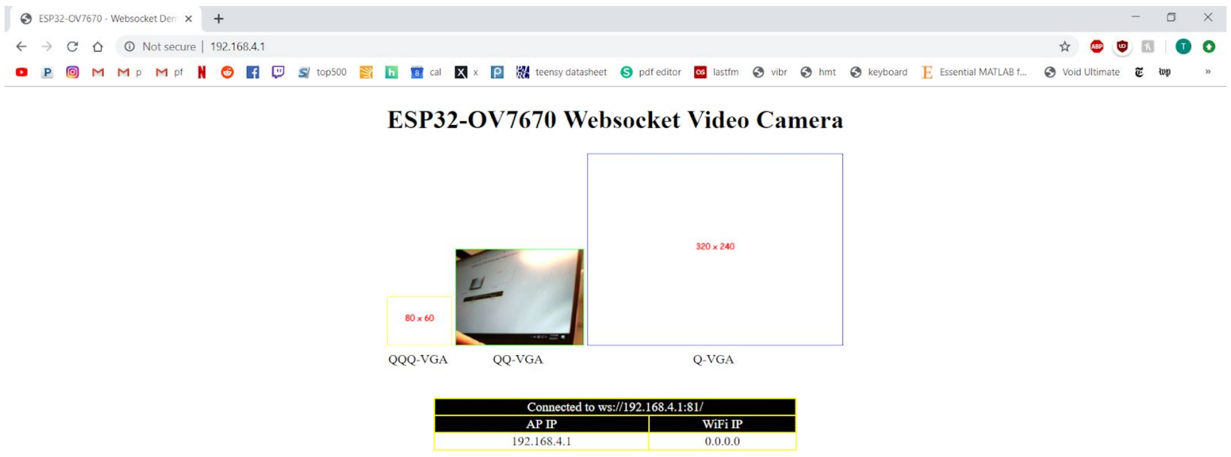


Figure 2: Oscilloscope-produced image, camera pointed at a face

An alternative method of broadcasting the OV7670 footage was developed by Tamboli: this approach used the ESP32 as a Websocket server, which could display video through an html web page. This was better than other researched methods since any device could access a web page through any browser, whereas, the other methods relied on an external video player through Python. The HTML website appears as Figure 3 below.



ESP32-OV7670 Websocket Video Camera

Connected to ws://192.168.4.1:81/	
AP IP	WiFi IP
192.168.4.1	0.0.0.0

Figure 3: HTML website for streaming

[1] Bitluni's work: https://bitluni.net/esp32-i2s-camera-ov7670
[2] Mudassar Tamboli's work: https://medium.com/@mudassar.tamboli/esp32-ov7670-websocket-video-camera-26c35aedcc64
[3] Oscilloscope demo: https://www.youtube.com/watch?v=PODnC_EyGJ0

A concern with WiFi video streaming was the additional latency that would arise due to the increased traffic of multiple ESP32s broadcasting simultaneously. During the performance of the final project, eight robots are expected to stream video at the same time. Using Tamboli's code, video can be streamed at Q-VGA (320x240p), QQ-VGA (160x120p) and QQQ-VGA (80x60p) resolutions. Expectedly, lower resolutions streamed with a lower latency. With only one camera broadcasting, it was observed that the latency was comparable between QQ-VGA and QQQ-VGA formats with a lag time of approximately 0.15s and 0.10s, respectively. Q-VGA streamed at noticeably slower rate in terms of frames per second and response time. Tests with multiple simultaneous streams were performed using the QQ-VGA resolution; with three simultaneous streams, the latency only increased up to 0.20s.

HTC Vive Base Station (Lighthouse)

Introducing fog of war would greatly reduce the amount of information a student pilot would receive. To alleviate this, a minimap feature would be added in conjunction with fog of war. This would provide a top view of the arena and the locations of the robots. To track each robot, the HTC Vive Base Station, sometimes referred to as a Lighthouse, was used (Figure 4 below).

Figure 4: HTC Vive Base Station

The Vive Lighthouse was originally designed as a part of the Vive VR suite. The Lighthouse is a passive device that works by continuously sweeping IR beams in two orthogonal axes. The Lighthouse emits an LED registration pulse before each IR sweep. By recording the time taken for the IR beam to be detected after a registration pulse, devices like VR controllers can determine its own location relative to the Lighthouse by means of triangulation. With additional diodes, orientation of the device can also be determined. In the figure below, the configuration of the light emitters in the Lighthouse can be seen.

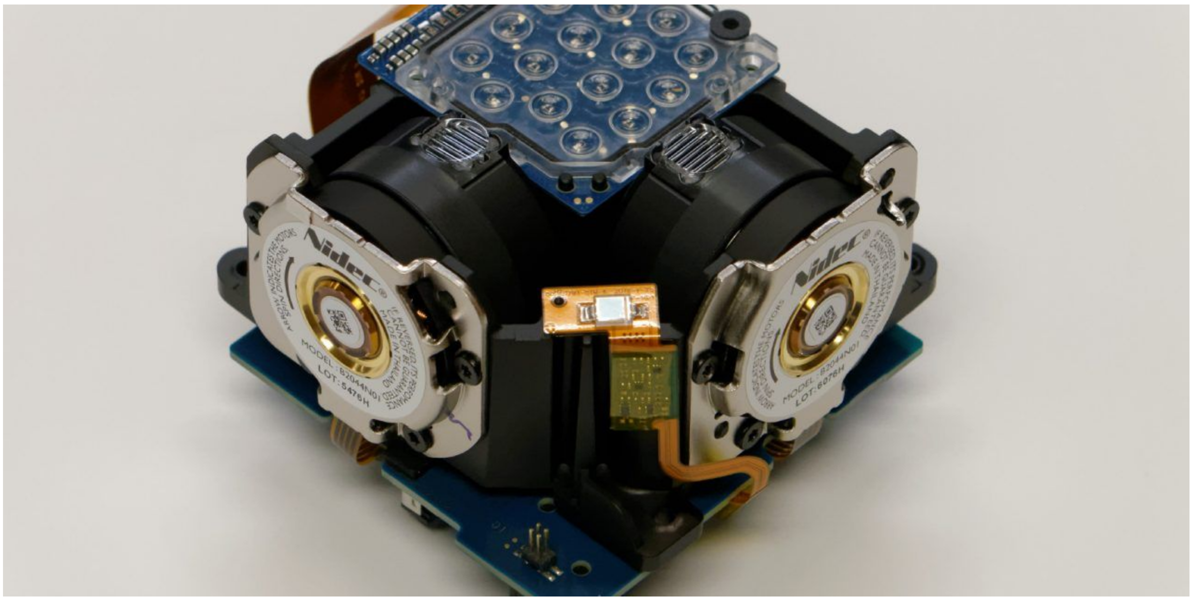


Figure 5: Dissected Lighthouse; array of LEDs for registration pulse (top), two Nidec motors house IR emitter

In terms of specifics of the Lighthouse, it emits a registration pulse from stationary LEDs at a 60Hz rate or approximately every 8.33ms. Following this, an IR beam sweeps by means of a motor at the same rate. The Lighthouse has three modes labelled A, B and C. In mode A, pulses and sweeps alternate every other. In mode B, a typical cycle behaves as the following:

1. Registration Pulse 1
2. Sweep in direction 1
3. Registration Pulse 2
4. Sweep in direction 2
5. Registration Pulse 1
6. Empty Sweep (no IR signal emitted)
7. Registration Pulse 2
8. Empty Sweep (no IR signal emitted)

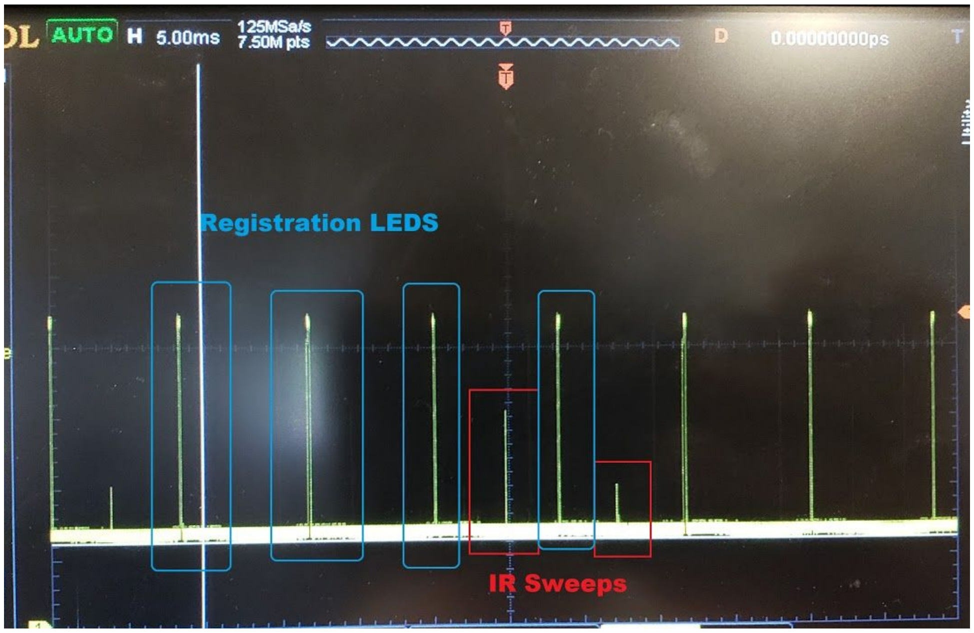


Figure 6: Mode B signal on oscilloscope

Mode C does not emit anything unless the Lighthouse is hooked up to another Lighthouse via a sync cable. In this configuration, one must be in mode B and the other in mode C. The pulses and sweeps for both Lighthouses should be synchronized, however, when Lighthouse B has an empty sweep, Lighthouse C will emit a real sweep.

Lighthouse Detection: Hardware

In the context of the MOBA project, each robot would need a photodiode to be tracked. The PD70-01C/TR7 photodiode[4] was used. The circuit diagram incorporating the photodiode with the ESP32 is shown in the figure below. The signal from the photodiode was amplified by the TLV272 op amp[5]. The resulting signal was then passed through a comparator[6] (TLC3702IP), which cleaned it up and converted it to a digital signal. Both DIP packages were chosen for their fast response time since the expected signals from the Lighthouse were also fast.

[4] Photodiode datasheet: http://www.everlight.com/file/ProductFile/PD70-01C-TR7.pdf
[5] Op amp datasheet: http://www.ti.com/lit/ds/symlink/tlv272.pdf
[6] Comparator datasheet: http://www.ti.com/lit/ds/symlink/tlc3702.pdf

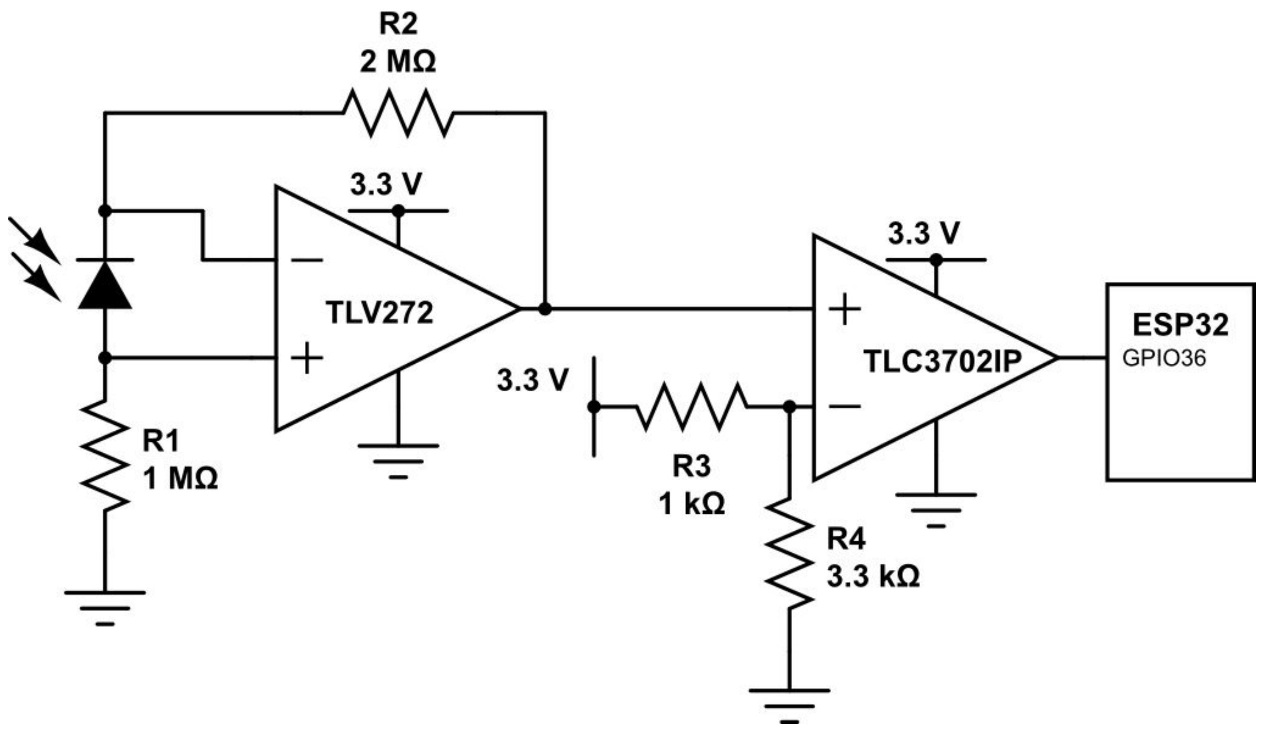


Figure 7: Photodiode circuit diagram

A few caveats with the photodiode were observed during testing: the photodiode was sensitive to light of any type. Since the photodiode was not exclusively sensitive to the light emitted by the Lighthouse, factors such as ambient light would greatly impact the signal. As a result, the amplification of the signal through the op amp needed to be tuned according to the location; the circuit configuration could then vary between Wu and Chen auditorium (the location of the final project performance) and the GM lab, for example. The final resistor values were found by adjusting potentiometers until the signal was usable. Since total light intensity onto the photodiode altered the signal, the stage lights in Wu and Chen auditorium needed to be turned off to allow for a readable signal.

An issue with reading the signal from the Lighthouse was differentiating the registration pulses from the IR sweeps. The method used was measuring the period of each type of signal. Through testing, the registration pulse and IR sweep periods differed in magnitude. It was found that the lengths of these periods could vary slightly based on distance and angle from the Lighthouse; the registration pulse was roughly 150µs, and the IR sweep was about 30µs. This was measured on the oscilloscope by capturing a still of the signal and moving the built in cursors as seen below.

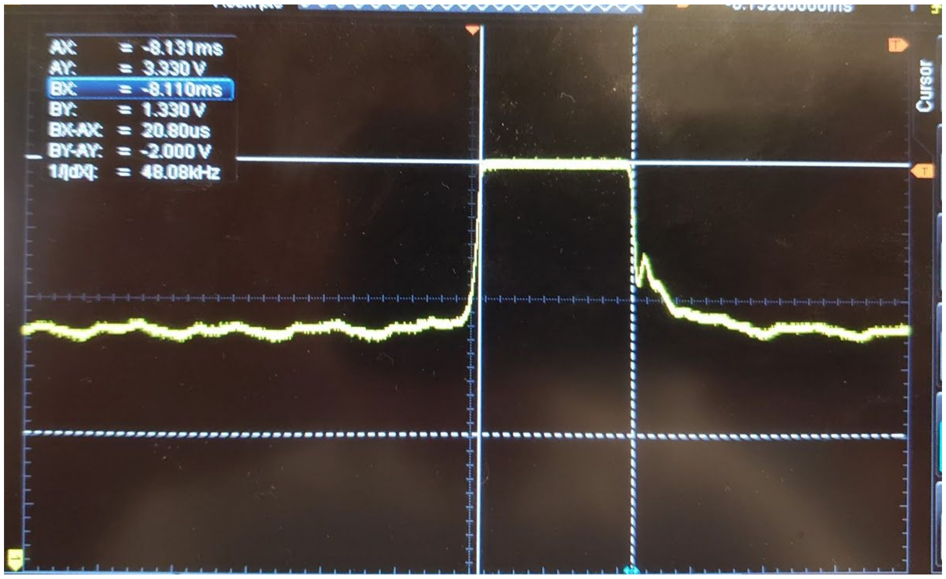


Figure 8: Measuring period on oscilloscope; note cursors A and B, BX-AX is the period length

Reasons for sorting the two pulse types (LED and IR) by pulse width instead of amplitude were the sensitivity of the photodiode signal and limitations of the ESP32. Since total light intensity on the photodiode affected the overall amplitude of the signal, factors like simply walking past the sensor could shift the signal, making the configuration require a more ideal and controlled environment. Additionally, light intensity varies with distance, so when the robot moves to and away from the Lighthouse, the signal amplitude will also shift. As for limitations by the microcontroller, measuring the amplitude would require using the analogRead() function, which could not run quickly enough to respond to the rate at which the Lighthouse operates.

Lighthouse Detection: Software

Since the Lighthouse is a passive device, the robot must be responsible for detecting the Lighthouse signals and calculating its own position. Each type of pulse from the Lighthouse occur at 60Hz; together, they effectively form a 120Hz signal. Interrupts are used to detect and measure the pulses and pulse widths; interrupts respond quickly enough to capture the microsecond-long periods.

Although IR sweeps and registration LEDs have distinct periods, the x and y axes sweeps are identical. To differentiate between recording an x or y coordinate, the Lighthouse must be set to mode B. Recall that in this setting, the latter half of a cycle does not emit IR light. This means that before the first IR sweep of a cycle, there are three consecutive registration LED pulses without an IR sweep. In the code, a counter is kept of consecutive LED pulses to determine sweep axis.

The code for detecting the Lighthouse and returning a position uses an events and services model to ensure that position can be constantly updated in conjunction with the other expected robot tasks. Similar to the way existing VR controllers track position, the code records the time between IR sweeps and registration LEDs. Using that time value and basic trigonometry, a corresponding x or y coordinate can be returned.

Hit Detection

The 2018 iteration of the MOBA final introduced a whole new system for tracking hits from robots to other robots or nexi. Unfortunately, there were a few technical difficulties which resulted in this system not consistently working. The footage from the final livestream was reviewed. Across the entire final, a total of two robots were defeated in combat and two nexus hits were successfully recorded. About 10-15% of hits were actually registered. This is a lower number than desired and results in slower than intended gameplay.

In addition to simplifying the way hits are registered on the server side, an alternative physical sensor was explored. In the previous iteration, each Top Hat includes a button which when hit, registers a hit. This can be difficult to consistently hit; generally, the robots were equipped with hammer arms that would need to line up before swinging. An initial idea as an alternative to the button was some sort of vertical rod that detects displacement from any direction. Upon further research, whisker switches appeared to fulfill this description, pictured below.

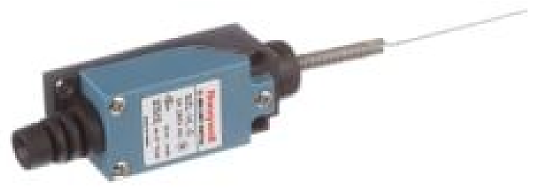


Figure 9: Whisker switch

It should be easier to actuate the whisker switch since a sweeping arm does not need to align with anything and just has to be in range. Through testing, a continuous micro servo (sg90) was capable of actuating the whisker switch. With this information, almost 50 whisker switches were purchased with hopes of improving the ease of robot combat.

Conclusion

This work proves that individually, the proposed features can perform their function. Within the context of the final project performance conditions, these additions are expected to work. The following steps would be the coupled implementation of all components with a fully functional MOBA robot. It is planned to incorporate the OV7670 and minimap detection to the existing Top Hat system so that it can be a part of the fall 2019 iteration of MOBA.