Cook Engineering Design Center

Final Design Review

Submitted in partial fulfillment of the requirements for

ENGS 90: Engineering Design Methodology and Project Initiation

Drone Object Sensing with Light Angle Sensor

March 2, 2018

Sponsored by

Analog Devices, Inc.

Project Team #6

Hamza Alsarhan, Martin Anguita, Lotanna Ezenwa,

Ping-Jung Liu, Prescott McLaughlin, Arun Reddy

Faculty Adviser

Dr. Minh Q. Phan

THAYER SCHOOL OF ENGINEERING AT DARTMOUTH

Executive Summary

Analog Devices, Inc. (ADI) has developed a new optical sensor capable of measuring the three-dimensional angle to an infrared (IR) light source. ADI seeks to showcase the capabilities of their novel “Light Angle Sensor” (LAS) in a robotics application in order to appeal to their target market. Therefore, ADI has requested that the sensor be integrated within the navigation system of a quadcopter drone, allowing it to follow an object.

ADI currently has evaluation boards that demo the sensor in a limited capacity due to the synchronous implementation of the sensor and light source. For this reason, the current evaluation boards would not be applicable in the inherently asynchronous drone-based demonstration. Our team was tasked with addressing the company’s need for an engaging demonstration platform, as well as the asynchronous system capabilities needed to implement it.

Over the course of ENGS 89 and 90, our team has methodically worked towards creating a demonstration in which a quadcopter can follow a moving infrared light source in three-dimensional space. Our work in ENGS 89 largely focused on creating the electronics that allow the LAS to measure a 3-D position of the light source. Then, in ENGS 90, we progressed towards using the system in a drone object-following scenario. This second phase of the project was primarily focused on obtaining a reliable distance measurement from the LAS, through the creation of an intensity model, as well as designing the control algorithms that would maneuver the drone with respect to the light source.

After comparing our final product with the objectives we set out to achieve, the team believes that the project has been a success. The demonstration we have created showcases the capabilities of the sensor in an exciting manner. The quadcopter safely and reliably follows a moving light source in three-dimensions.

Upon sharing our achievement with the sponsor, they expressed enthusiasm and approval of our final deliverables. In summary, we believe that our product will provide an effective promotional platform for ADI’s exciting new technology.

Table of Contents

List of Abbreviations & Acronyms

AAF	Anti-Aliasing Filter
ADC	Analog-to-Digital Converter
ADI	Analog Devices, Inc.
AFE	Analog Front End
BPF	Band Pass Filter
BW	Bandwidth
DAQ	Data Acquisition System
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FC	Flight Controller
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
HF	High Frequency
IIR	Infinite Impulse Response
IR	Infrared
LAS	Light Angle Sensor
LED	Light Emitting Diode
MCU	Microcontroller Unit
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
Op-Amp	Operational Amplifier
PCB	Printed Circuit Board
PD	Photodiode
PPM	Pulse Position Modulation

PWM	Pulse Width Modulation
SBUS	Serial BUS
SMD	Surface-mount Device
SNR	Signal-to-Noise Ratio
SOA	State-of-the-Art
SPS	Samples per Second
TIA	Transimpedance Amplifier

1. Overview

Analog Devices, Inc. is an industry leader in the design, manufacture, and marketing of a wide array of integrated circuits (ICs). Within its line of optical sensing products, the company has developed a novel device called a “light angle sensor” (LAS) capable of discerning the X and Y coordinates of an infrared light source, as described through Figure 1.

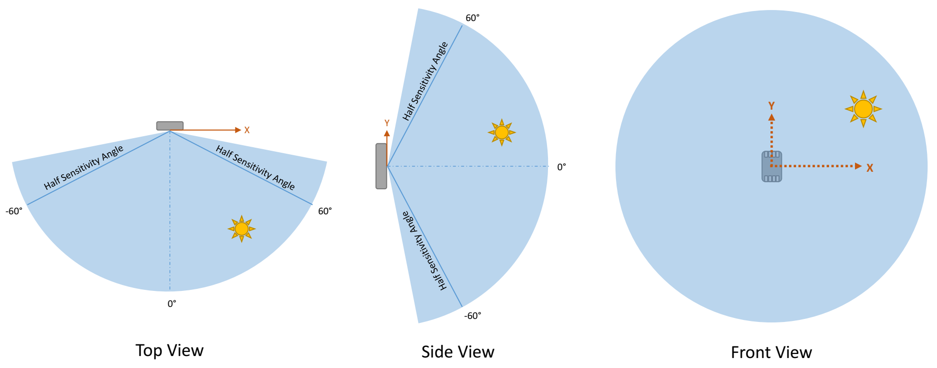
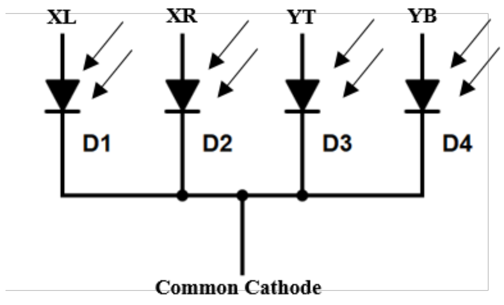


Figure 1. Illustration of LAS functionality and field of view. The sensor can detect the X and Y coordinates of an IR light source in its 2-D field-of-view. It has a peak sensitivity to light in the 860 nm IR band.

The LAS is a completely passive component consisting of four photodiodes, configured such that their relative responses can be used to calculate the X and Y coordinate of a light source in its field of view. Figure 2 offers an illustration of this functionality and the X, Y calculations.



$$x = \frac{x_R - X_L}{x_R + x_L}$$

$$y = \frac{y_T - y_B}{y_T + y_B}$$

Figure 2. Illustration of the 4 PDs and the use of their photocurrents to calculate X and Y

ADI is in the preliminary phases of bringing their LAS to the marketplace. At this stage, the company is searching for a demonstration that will showcase the sensor and serve as its promotional platform as they launch this new technology. The first step they have taken is to implement the LAS in a system that senses the location of an object through an on-board, synchronous light source. What they are now leaning towards is an asynchronous application in which the LAS is implemented using an off-board, asynchronous light source—particularly targeted towards the robotics industry. This type of implementation has important implications for their sales as it opens up a much larger list of potential applications of the LAS.

1.1 Problem Statement

As ADI has considered robotics applications of their LAS, they have further narrowed their interest in a drone object-following platform due to its current popularity in the technology sector. Given this, ADI needs their LAS to be used asynchronously on a quadcopter so viewers can easily grasp that the LAS is detecting a 3-D spatial position and using this measurement to assume complete control of the drone.

1.2 Objectives & Requirements

Our high level objective is to satisfy the broader goal of producing this demonstration of the LAS in a drone-based application. To begin approaching this project, it is important to outline the attributes of a successful demonstration, which appear in Table 1. In this, we must consider where this demonstration will be used, who will view it, and who will operate it.

Table 1. High Level Objectives and Requirements

Objective	Requirements	Metric
Safety	Manual User Override	Y/N
	Embedded Fail-Safes	Y/N
	Avoid Eye Damage	IR Output < Safe Limit
Ease of Use	Sponsor Can Operate	Short Orientation Time
	Streamlined Package	Self-Contained
Clarity of Sensor Functionality	Sensor Assumes Total Control of Drone	Transparent to Viewer
	3-D Functionality Conveyed	Transparent to Viewer
Captivation	Viewer Engaged by Demo	Viewer Surveying

1.3 Deliverables

We aim to deliver a safe and functioning demonstration that showcases the operability of the LAS in a drone-based object-following application

- Within this high level product, the following sub-systems are required and are to be delivered:
  - A printed circuit board (PCB) that incorporates the light angle sensor, analog front-end circuitry and digital signal processing necessary to measure the 3-D position of an asynchronous light source
  - A PCB that implements an LED driving circuit which will serve as the IR light source in the demonstration
  - Appropriate documentation of hardware and software implementation details

2. Methodology of Approach

The development work involved in creating this demonstration can be described chronologically in two phases. The first phase (Phase 1) focused on enabling the LAS to measure the 3-D location of a light source, while the second phase (Phase 2) centered on using this location measurement for object tracking in a quadcopter.

During Phase 1, we determined that measurement of 3-D location would hinge upon the physical creation of two subsystems connected by an asynchronous optical link: (1) an infrared light source (the object to be tracked), and (2) the sensor system (to be mounted on the quadcopter). Thus, Phase 1 largely consisted of electrical hardware development of these two subsystems. In order to inform their design, the team identified key functions of the light source as well as the sensor system. These can be seen in Table 2 below.

Table 2. Key functions of the two electrical subsystems

Electrical Sub-System	Key Functions
Infrared Light Source	Produce an optical signal that can be easily recognized by the LAS
Sensor System	Obtain useful analog signals out of the LAS
	Perform processing to obtain X, Y and distance measurements
	Execute control algorithms to produce motor actions for light source tracking
	Include ability to regain manual control of drone

The two subsystems in Phase 1 created the platform for asynchronous 3-D position tracking, allowing Phase 2 to focus on object-following. In this phase, system parameters, such as LED current and amplifier gains, were chosen to enable tracking at distances upward of 1 meter. The remainder of Phase 2 involved modeling the photodiode responses to create an accurate distance measurement, as well as develop the control algorithms that allow the drone to maintain a 3-D position with respect to the light source.

Requirement specifications of each subsystem, as well as the overall demonstration, can be found in Table 3.

Table 3. Requirement specifications for each subsystem, as well as overall demonstration

	Requirement Specification
Infrared Light Source	Distinct from ambient lighting
	Accurate waveform frequency and duty cycle
	Frequency low enough to be sampled by MCU ADC ($< f_{sample}/10$)
	Output power sufficient for high SNR at operating distances
Sensor System	Reliably detect spectral content at LED frequency
	Provide adequate BW for control algorithm
	Discern spatial changes within 1.5 m distance
	Function using 5 V or 3.3 V onboard power
	Maximize SNR
Overall Demonstration	Achieves closed-loop stability in < 5 seconds
	Flight time > 2 minutes per battery

2.1 Design Development

In this section, a more detailed discussion of both phases in the design process is presented.

2.1.1 Phase 1: Electrical Subsystem Construction

Infrared Light Source Subsystem

Since the infrared light source serves as a beacon for the quadcopter during flight, it must have sufficient power and viewing angle to operate in this dynamic application. The team identified SFH 4716AS, a high-power LED with 150° angle and 860nm wavelength, as the optimal light source for this application. An important consideration of the light source is the uniformity of its intensity across irradiance angle. Since the sensor system has no information about the light source's irradiance angle, any non-uniformities would affect our intensity-based distance measurement (discussed later in this document). Figure 2 shows total photodiode response data collected while rotating the light source (0° corresponds with the LED being pointed directly at the sensor). We can see that the intensity is rather uniform at moderate angles, and it falls sharply at extreme angles. This is favorable, as the sensor either sees a reliable optical signal, or it sees virtually no signal—removing any ambiguity in its measurements.

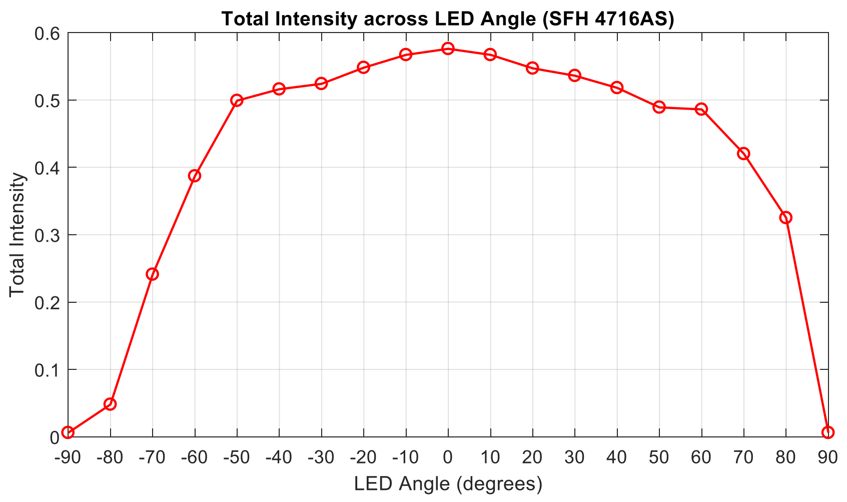


Figure 2. Total intensity across the irradiance angle of the SFH 4716AS LED.

The optical signal from the light source must be easily distinguishable from ambient lighting to maximize SNR. For this reason, the LED current was modulated at a frequency of 500 Hz, so as to avoid interference of 60 Hz and 120 Hz content in indoor lighting. Appendix A discusses the choice of a square waveform over a sinusoidal waveform for optical signal modulation. The waveform was generated using a low-cost microcontroller to ensure consistency in frequency and duty cycle.

The infrared light source was eventually implemented on a stand-alone PCB, which incorporates the microcontroller, MOSFET driving circuit, and the LED. Power is provided by a portable power bank via Micro-USB. Current through the LED can be modified by adjusting the value of the current-limiting resistor, R1, seen in Figure 3.

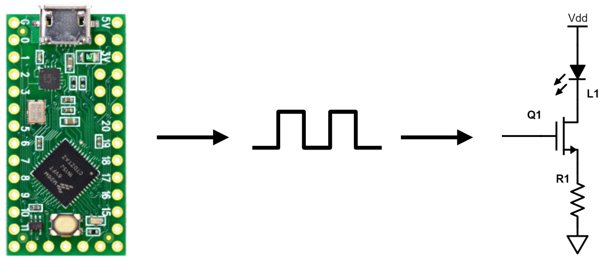


Figure 3. Infrared light source concept diagram

Sensor Subsystem

Design of the sensor subsystem was driven by the four main functions previously presented in Table 2. The first function of the sensor system (obtaining useful analog signals from the LAS) necessitates analog front end (AFE) circuitry, a discussion of which can be found in Appendix C. The team determined that the second, third, and fourth functions of the sensor system could be fulfilled with the use of a dedicated microcontroller. A discussion of microcontroller selection can be found in Appendix D, ending in the choice of the Teensy 3.5 development board. This microcontroller would sample and process the AFE outputs, as well as execute the control algorithms that would maneuver the quadcopter. We decided that autonomous drone control by the LAS would best be implemented by interfacing with the flight controller in the same manner as the hand-held remote control receiver. This method would avoid interference with the lower-level stability controls running on the flight controller, and allow us to mimic the receiver's PWM outputs (which can be easily studied and have an intuitive effect on overall drone motion). In addition, this approach would allow a switching feature between manual and autonomous mode to be implemented in firmware. The general idea of this approach is illustrated in Figure 4.

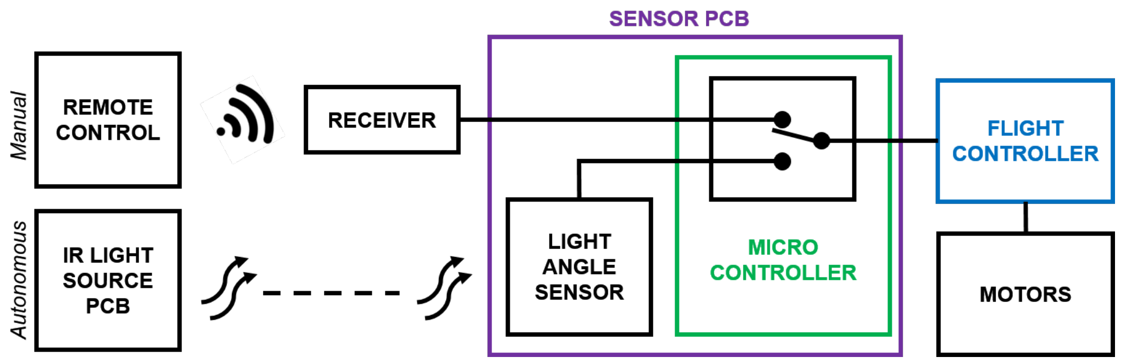


Figure 4. High-level approach to integrate sensor system with quadcopter. Switching between Manual and Autonomous mode implemented within microcontroller

Within this general approach, significant development work was involved in designing the sensor PCB (Appendix E) as well as the digital processing in the microcontroller (Appendix F & B). The functionality of the sensor PCB and its interface with the quadcopter flight controller is depicted in Figure 5. Specific blocks, such as intensity correction, linearization, and control algorithms will be discussed with Phase 2.

After the infrared light source and sensor system PCBs were designed, fabricated, and populated, Phase 2 of development began.

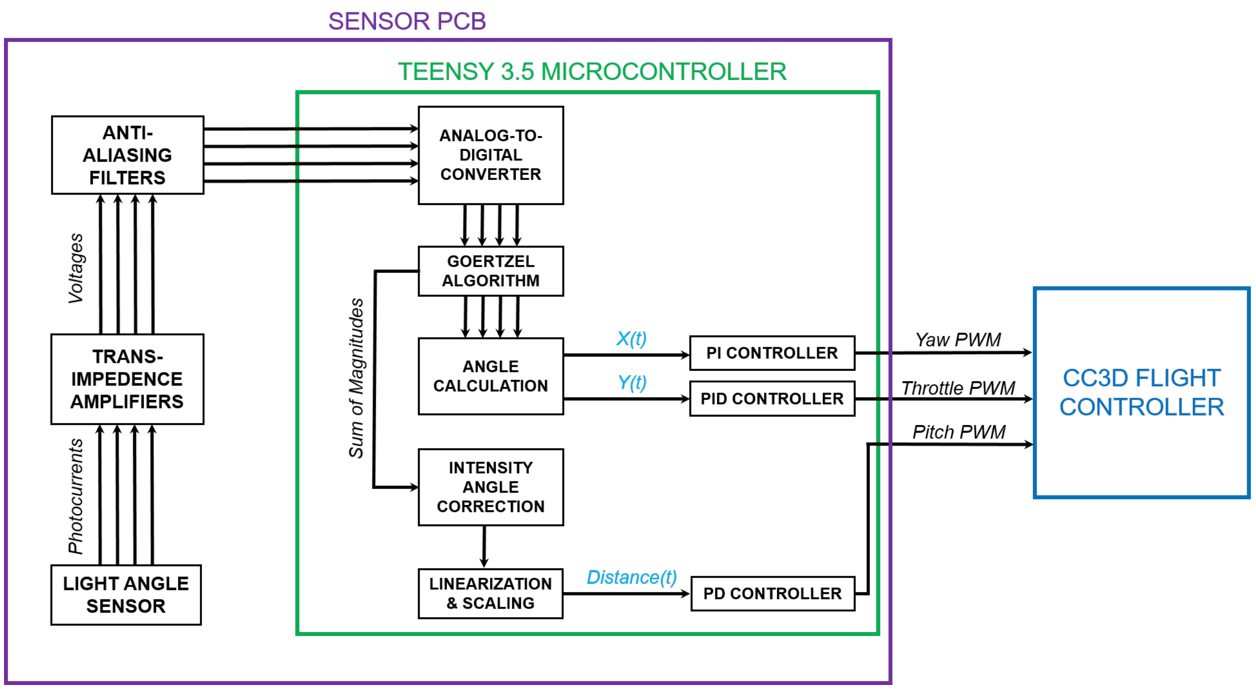


Figure 5. Functional block diagram of sensor PCB, which incorporates AFE and Teensy 3.5 microcontroller. The system interfaces with the flight controller in the same manner as the remote control receiver. In manual mode, the sensor system reproduces receiver outputs—acting as a normal remote-controlled quadcopter

2.1.2 Phase 2: Drone Object-Following Implementation

Distance Measurement

ADI's light angle sensor allows simple calculation of X and Y spatial coordinates from the relative response magnitudes of the four photodiodes. However, the LAS does not provide direct measurement of the third spatial coordinate needed for 3-D tracking—the distance to the light source. Two methods of distance measurement, one using triangulation and the other using the intensity of light, are analyzed in Appendix G. Triangulation was deemed unfeasible, since it would require a physical separation of sensors that is unachievable on the frame of the drone. As such, the team moved forward with an intensity-based approach.

At first glance, summing the response of the four photodiodes (which we will refer to as 'total intensity') may seem like an effective metric for distance. However, this metric was found to be highly angle-dependent. Angle dependence is problematic since the light source might be perceived as being far away (in which case the drone would respond by moving forward), when in reality the light source is close to the drone, but at an angle. Figure 6 shows this angle dependence from collected data, as well as the polynomial fit performed to correct for it.

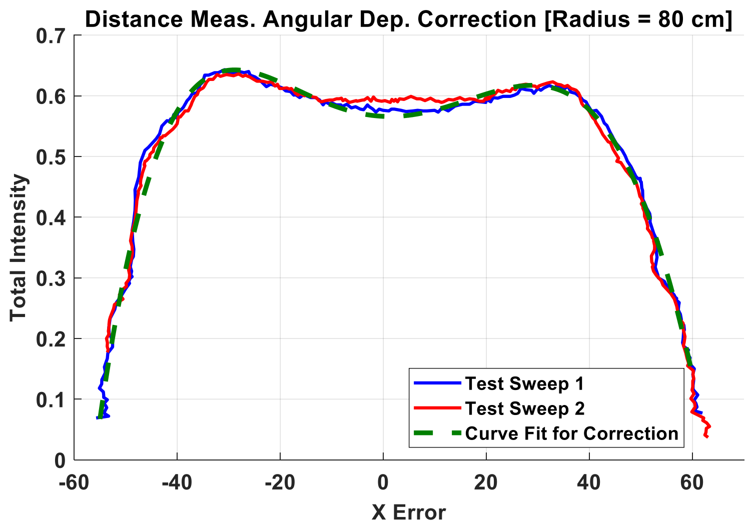


Figure 6. Plotted data and polynomial fit for angular sweep across X error at fixed radius of 80cm. A very similar process was performed for Y error dependence correction

It is clear from Figure 6 that in addition to being a function of the actual light intensity (which we will call I_{REAL}), total intensity is also a function of the X and Y error. Which can be described by Equation (1):

$$I_{TOTAL} = I_{REAL} * f(X, Y) \tag{1}$$

Since our quantity of interest is the real intensity, we can rearrange Equation (1) into Equation (2):

$$I_{REAL} = \frac{I_{TOTAL}}{f(X, Y)} \tag{2}$$

Now, if we can identify the function f(X,Y), we can create an angle-independent intensity metric. A 6th and 8th order polynomial fit was performed for X and Y dependence respectively. Dividing the total intensity metric by these polynomials evaluated at the current measured X and Y values gives the real intensity. Figure 7 shows real intensity data collected by sweeping X angle at a fixed 80cm radius. The flatness of the plot implies that angular dependence has been removed. Appendix H contains a 3-D plot showing the same result for X and Y dependence simultaneously.

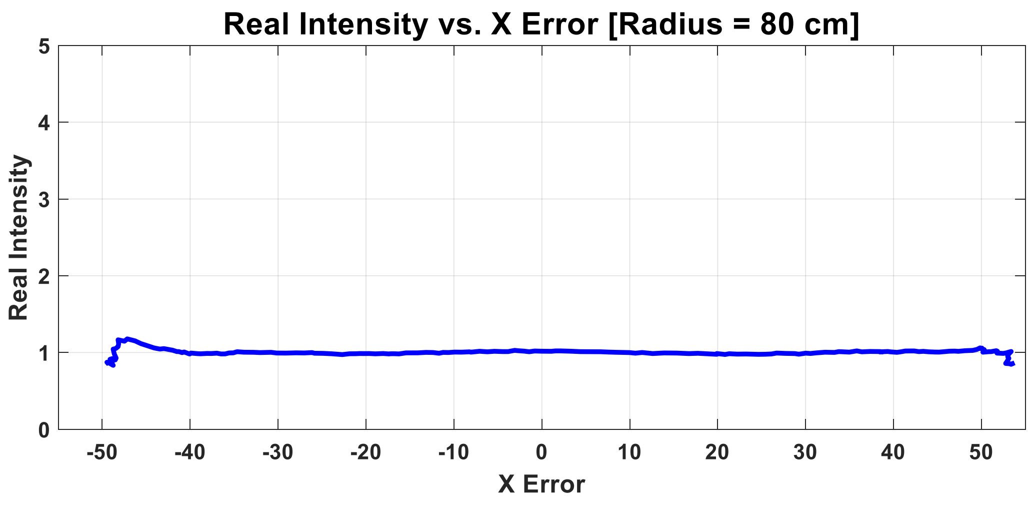


Figure 7. Real intensity as measured across a sweep of X error at a fixed radius of 80cm. Real intensity is measured as roughly 1 since polynomial fit data was also collected at 80cm radius. Flatness implies angle independence.

Now that angular dependence has been removed from the total intensity metric, it must be linearized and scaled to arrive at an accurate measurement of distance. A nonlinear distance metric would result in asymmetric flight behavior on either side of the distance set-point. Since light intensity follows an inverse-square law over distance, the relationship between our new intensity metric and distance can be described as follows, where k and b are constants that will help match the distance metric to centimeter values:

$$I_{REAL} = \frac{k}{(distance[cm] + b)^2}$$

After collecting I_{REAL} values across physical distances from the sensor, the data was used to solve for k and b. As seen in Figure 8, this process creates a fairly accurate model of I_{REAL} over distance. By inverting the model, we can use the measured I_{REAL} value to solve for distance in centimeters. Figure 9 shows the distance measurement against the physical distance. The measurement was deemed sufficient for our purposes, as it changes monotonically and linearly about our chosen operating point of 80cm.

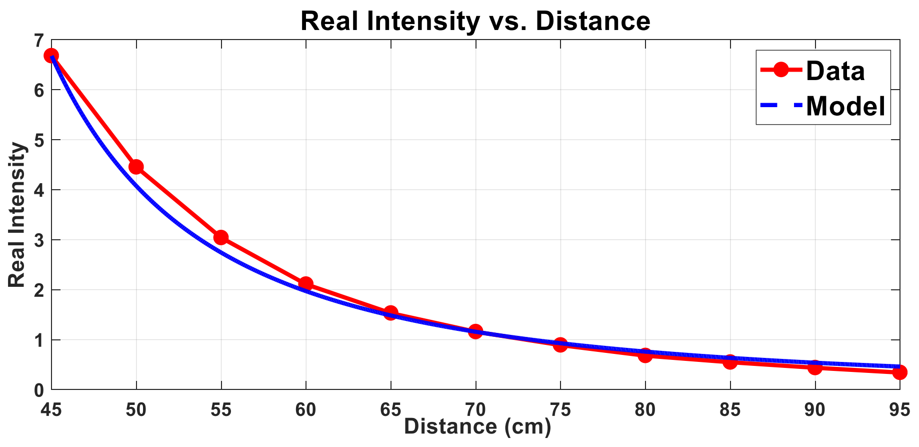


Figure 8. Real intensity data and model across distance

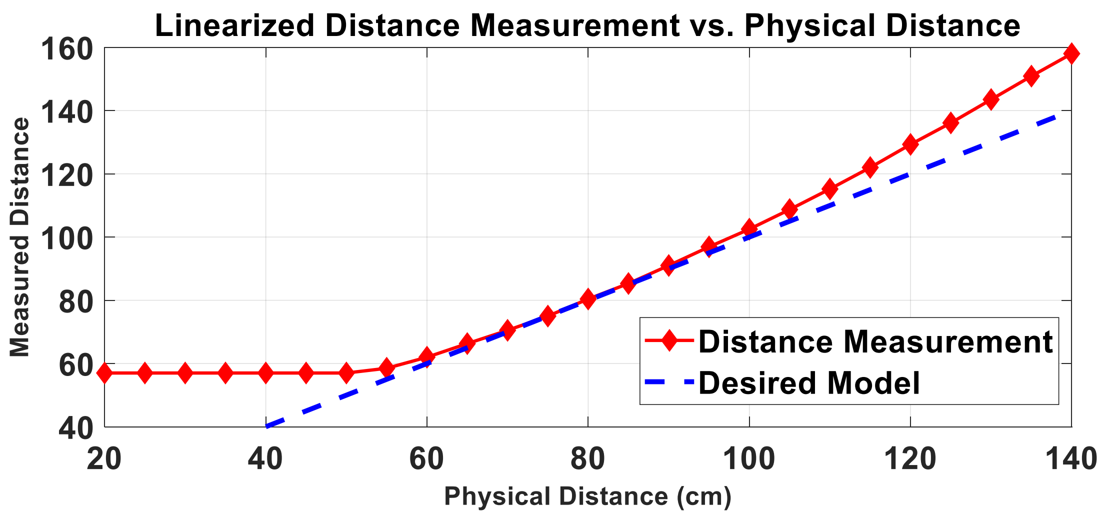


Figure 9. Linearized distance measurement vs. physical distance. Distances under 50cm saturate the AFE and are treated as the same distance

Now that the sensor system has the capability to accurately measure the distance to the light source, the control algorithms can be developed.

Control Algorithm Development

To this point, we have given the quadcopter the ability to measure the 3-D error to its desired position set-point. The remaining task is to maneuver the quadcopter for smooth convergence towards the set-point.

The remote control for the drone has four commands: Roll, Pitch, Yaw, and Throttle. However, we determined that the autonomous flight system would only need to employ three of the four to correct for the position error, since both Yaw and Roll can correct for X error (one rotationally and one translationally). Yaw was selected over roll because it would create a more dynamic demonstration, by enabling the drone to turn with the object it is following.

Since the sensor system outputs are intended to mimic receiver PWM signals, the team began by studying the receiver outputs during manual flight. For this, we needed a system capable of monitoring these values in real-time. The team developed a Bluetooth data acquisition system (DAQ), which enabled logging of flight commands, LAS measurements, and other values on the microcontroller for debugging purposes. An example of our monitoring capabilities can be seen in Appendix I.

Once an understanding of the PWM values (and their effect on the quadcopter's motion) was achieved, we set bounds on the minimum and maximum control efforts that the autonomous system could output to the flight controller. This helped ensure safe flight testing of each control algorithm.

A PID-like feedback control algorithm was chosen for its simplicity, effectiveness, and the fact that it can be tuned empirically. Since the team lacked an accurate mathematical model of the system being controlled, an empirical tuning method called the Ziegler-Nichols method was used to determine the P, I and D gains for optimal transient responses [22]. A system model for the quadcopter is difficult to obtain since significant low-level compensation occurs within the flight controller.

Separate controllers were designed and tested independently for each of the three flight commands, while retaining manual control of all other flight commands. System behavior was observed visually and through X and Y error data sent through the DAQ system. This iterative approach yielded the results in Table 4 below.

Table 4. Controller Gains

Flight Command	Action	Controller Type	Final Gains	Reasoning
Throttle	Up/Down	PID	P = 0.82	-
			I = 5E-3	Bias point changes across battery life
			D = -12	Momentum causes overshoot
Yaw	Rotation	PI	P = 2.1	-
			I = 0.01	Steady state error observed with P only controller
			D = 0	No rotational momentum; no overshoot
Pitch	Front/Back	PD	P = 1.5	-
			I = 0	Distance error is tolerable and unnoticeable
			D = -50	Overshoot towards user is dangerous, need damping

3. Deliverables

When comparing our final product with the objectives set out to accomplish in section 1.2, it is evident that the demonstration produced for our sponsor has been successful. A close inspection of our product against our requirements shows that our drone system has exceeded our expectations. In the following section, the performance results, along with results of our design methodology, are presented in order to provide a detailed understanding of the components involved in the demonstration.

3.1 Drone Demonstration

The high level deliverable for this project was to produce a safe and functional demonstration of the LAS autonomously controlling the flight of a quadcopter. Our final product includes a drone outfitted with this capability, as well as an IR light source mounted to a wand that serves as the object to follow. Photos of these deliverables can be found in Appendix M.

In terms of the lower-level components that comprise this demonstration, we have succeeded in constructing the sensor system PCB, the infrared light source PCB, and will compile a user manual in addition to the software and documentation to hand over to ADI.

After sharing a video of our product demonstration with the sponsor, they expressed their enthusiasm and satisfaction with the results of the project.

3.2 Satisfaction of Requirements

We compared the final product to our overarching requirements, which outline the attributes of a successful demonstration.

Safety

- Manual Override: This is implemented through the use of a mono-stable switch on the remote, allowing the demonstration operator to regain full control by simply releasing the switch.

- Fail-Safes: These have been embedded in the software so that in the occurrence that SNR is diminished, or the drone exits its operational volume (12.4 m^3), the microcontroller will output safe control signals (causing drone to hover) and avoid reacting to noisy readings by the LAS. In addition, the control efforts available to the autonomous system are limited to conservative values.
- Eye safety: This is accounted for by ensuring that the power output of the LED is within the safe range for the human eye. Calculations can be found in Appendix J.
- Low battery alarm: When the battery falls under a pre-set threshold (10.2 V), a buzzer indicates that the drone is unsafe to fly

Ease of Use

- Sponsor Can Operate: This remains to be validated, however the current demonstration requires that the demonstrator have a rudimentary ability to take-off and land the drone. The orientation for this will include a lesson about the effective control volume of the sensor and how to pilot the drone into this window.
- Streamlined Package: Each component is fully integrated to form a product consisting of a functional object-following drone and an external light source mounted on a wand for the following to be demonstrated. Each component simply requires the user to plug in the associated power sources.

Clarity of Sensor Functionality

- Sensor Assumes Total Control of the Drone: This is made clear in the way that the user can simply enter the control volume, flip the autonomous control switch, and take their fingers off the joysticks. From that point forward, the LAS provides the control inputs for flight.
- 3-D Functionality Conveyed: The drone follows in all 3 spatial dimensions as it uses its detection of X, Y, and distance errors to control yaw, throttle, and pitch, respectively.

Captivation

- Viewer Engaged by Demo: This was validated through our survey, which includes a video of the drone following a light source. Viewers are then asked: “on a scale of 1 to 10 how amusing do you find this demonstration of the sensor?”

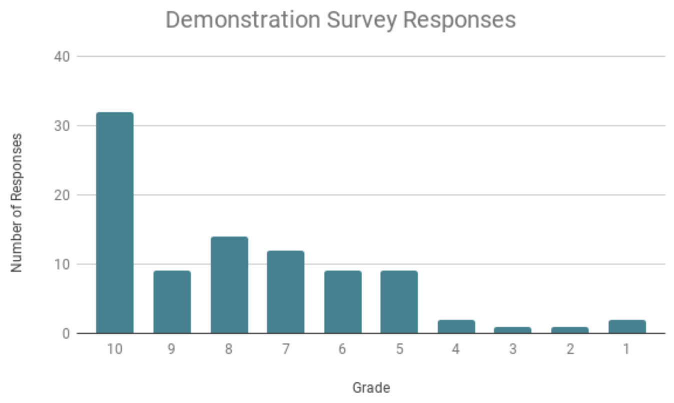


Figure 10. Survey results to evaluate the entertainment factor of the demonstration.

The survey received 94 responses, of which 35% rated it with a 10 and 74% responded with a rating of 7 or higher.

3.3 Requirement Specification Testing

We also evaluated the project against the more detailed requirement specifications shown in Table 5, which outline the technical necessities involved in producing our high-level deliverable.

Table 5. Revisiting Requirement Specifications to validate sub-components and overall product.

	Requirement Specification	Satisfied
Infrared Light Source	Distinct from ambient lighting	
	Accurate waveform frequency and duty cycle	
	Frequency low enough to be sampled by MCU ADC ($< f_{sample}/10$)	
	Output power sufficient for high SNR at operating distances	
Sensor System	Reliably detect spectral content at LED frequency	
	Provide adequate BW for control algorithm	
	Discern spatial changes within 1.5 m distance	
	Function using 5 V or 3.3 V onboard power	
	Maximize SNR	
Overall Demonstration	Achieves closed-loop stability in < 5 seconds	
	Flight time > 2 minutes per battery	

Infrared Light Source

We modulated the LED current at a frequency of 500 Hz and 50% duty cycle, which has proved to be easily distinguishable from ambient lighting. The chosen frequency allowed us to sample each of the four PD signals at a rate of 10 kSPS in the sensor-side microcontroller. Our design allowed for adjusting the power output of the LED through a current limiting resistor. The final LED driving current was chosen to be a 0.8A peak (400 mA average) square wave and allowed for our control radius to range from 0 to 2 meters. The resulting control volume was found to be sufficient for easy operation.

Sensor System

The sensor system we created successfully identifies the 500Hz content on each of the four photodiodes (details in Appendix B), and it uses the relative magnitudes to calculate the X and Y angles. The signal is sampled at a rate sufficient to produce a 34 Hz update rate on these measurements, which has proven to maintain stability in our control algorithm. The sensor PCB is designed to operate using the 5 V available on the FC and receiver. We maximized SNR through the previously mentioned TIA gains and LED output power, as well as by choosing low noise components.

Overall Demonstration

Our overall demonstration was evaluated in terms of its ability to convey the sensor's capabilities as well as its closed-loop stability and flight time. We used our Bluetooth data acquisition system to obtain transient response data for each of the three flight commands in autonomous mode. We created "step inputs" to the systems by rapidly shifting the position of the light source. The transient response plots in Figures 12, 13, and 14 on the following page provide an indication of settling time. They also illustrate how spatial error from the LAS generates the control effort associated with each of the three algorithms.

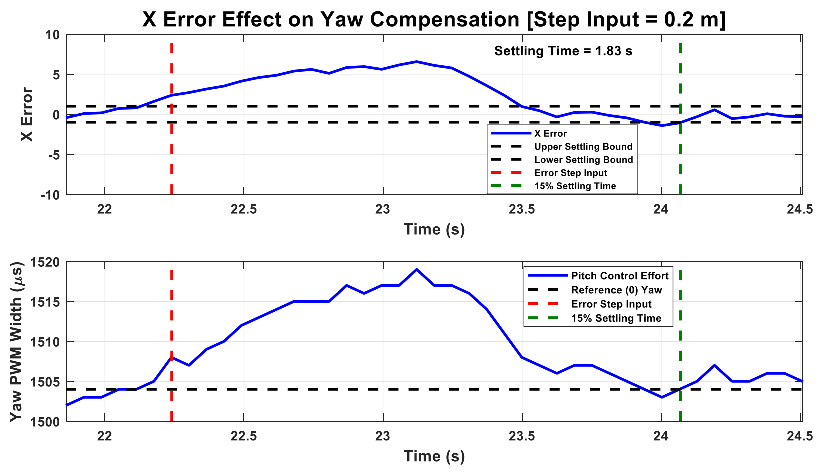


Figure 11. Transient Response of Yaw compensation for step input of 0.2 m X error offset. For the purposes of the demo a 15% settling boundary was chosen and tests yielded a 1.83 second settling time.

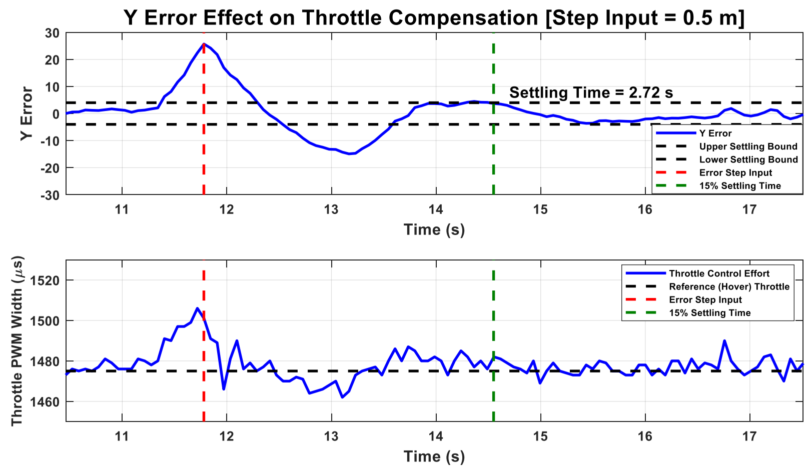


Figure 12. Transient Response of Throttle compensation for step input of 0.5 m Y error offset. For the purposes of the demo a 15% settling boundary was chosen and tests yielded a 2.72 second settling time.

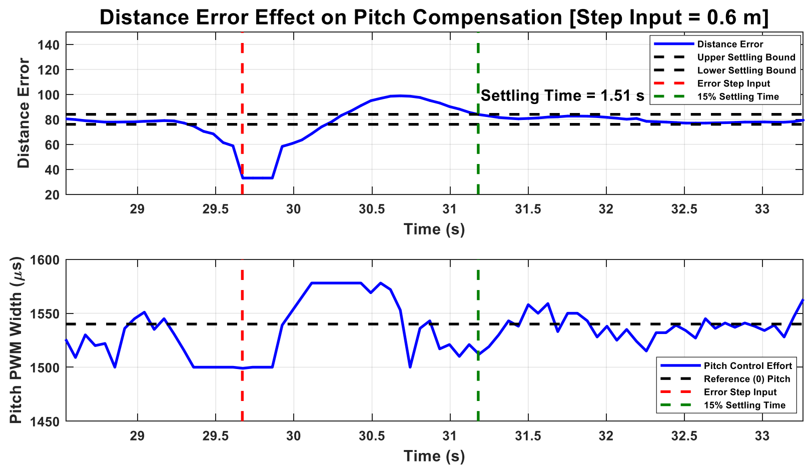


Figure 13. Transient Response of Pitch compensation for step input of 0.6 m distance error offset. For the purposes of the demo a 15% settling boundary was chosen and tests yielded a 1.51 second settling time.

The final requirement specification we evaluated was the flight time available for the demonstration. The drone is configured to fly with either a 2200 mAh or 3300 mAh battery and thus has two possible demonstration durations shown in Table 6.

Table 6. Average flight duration for batteries. Testing included 5 autonomous flights with each battery size until the low voltage alarm indicated insufficient charge for further safe flight.

Battery	Average Flight Duration
2200 mAh	9 minutes 36 seconds
3300 mAh	12 minutes 11 seconds

3.4 Novelty

Object tracking and following is not a novelty in the drone market; however, the means by which tracking is implemented can vary. CMOS cameras are the predominant modality for object-tracking on drones by using the video feed to calculate position errors. Still, image processing algorithms are complex, difficult to implement and can make mistakes. Some other sensors used for navigating drones are infrared and ultrasonic ranging sensors, whose processing algorithms are much simpler due to their analog nature, yet lack the ability to fully track an object. The LAS, however, offers the best of both technologies by providing the tracking abilities of a CMOS camera, and the simplicity of an analog sensor. In this way, the LAS proves to be an improvement on current SOA found in the research phase of the project, making this an innovative technique for drone object following.

4. Economic and Cost Analysis

After several conversations with our sponsor, it became clear that ADI does not intend on marketing the full drone system as a consumer product. Instead, the company's main objective for this demonstration is to boost the sales of their LAS. For this reason, this market analysis focuses on the potential LAS market and how the cost of the drone demonstrations compares to the potential revenues resulting from it.

ADI's ultimate goal is to showcase this platform at trade shows, where clients from various industries can appreciate the sensor's versatility. Moreover, ADI's Marketing Manager reaffirmed that although it is difficult to show correlation between the sales before and after a demonstration, there is a direct relationship between the quality of the demonstration and the increase in sales. Difficulties in quantifying this correlation stem from the fact that most demonstrations are function specific as per the request of a client and are usually produced as a collaboration between ADI and the client's engineering teams.

4.1 Market Analysis

Our sponsor intends to target the automobile and robotics markets. Market research shows that consumer electronics is the broadest market that could employ ADI's LAS, which would therefore be the total addressable market in the U.S. (TAM). Estimated TAM revenue was 224 billion USD in 2016 [21]. Our served addressable market (SAM) consists of the sensor market in the US, with an estimated revenue of 55.4 Billion USD [17]. The markets foreseen utilizing the sensor, our target market (TM), are the sensor markets of the automobile and robotics markets, which are at 6.27 billion USD and 510 million USD, respectively. Assuming that optical sensors make up 10% of our 6.78 billion USD TM and a conservative 5% market share, the total net revenue for ADI will be around 33.9 million USD.

4.2 Cost

The costs accrued to build the full drone system can be split up into 2 main sections; electronics and drone platform. The total cost of the electrical components (SMDs) used in our drone system is \$21.31, as can be seen in Appendix K. As for the PCBs, it costs \$47.25 to produce both the LED and sensor PCBs. The drone and the transmitter cost \$112.5 and \$129, respectively. Finally, the LAS used in the demonstration priced at \$2.71/sensor when sold in a batch of 1000. Therefore, the total cost for building the drone demonstration adds up to be \$312.77.

5. Future Works

Moving forward with this project, there are several ways to enhance both the safety and performance of the demonstration. Firstly, propeller guards can be added to reduce the harm inflicted in the case of collision with people or surroundings. Moreover, an indicator light can be added in order to visually announce when the drone is flying autonomously. On the performance front, implementing frequency modulation into the IR LED could be used to trigger either a lading or takeoff pattern. In this way, a pilot can tell the drone to either land or take off autonomously by changing the frequency of the LED with the push of a button.

In addition to safety, a more reaching extension is to build a swarm of drones that could follow each other's light sources. In such a swarm, one drone could be leading others; lowering the need for multiple skilled pilots. To do so, we would need to extend control effort limits for increased speed and gain of the controller.

6. Conclusion

This project has yielded a product that exceeds the expectations of both the team members and project sponsor. The team is proud to have created a streamlined and effective product.

The demonstration achieves its goal of implementing the LAS within a robust control system to assume full control of the drone and showcase the powerful capabilities of the sensor. We have achieved all of this while considering the environment this demonstration will be used in, as well as the safety concerns involved. Our team believes that the drone can be handed off in its present state to be safely utilized at trade shows, so long as it is equipped with propeller guards. It performs reliably in untethered autonomous flight, with the only unresolved factor being the orientation required to teach our sponsor to take-off and land the drone manually.

We are incredibly appreciative to have had such a helpful and knowledgeable sponsor that was as committed as we were to bringing this finished product to fruition. We would like to thank Tyler Ray and Dvij Bajpai for their continued feedback and support. Our appreciation is also greatly extended to Professor Phan for his instrumental assistance in developing our control algorithm and Professor Halter for helping us throughout every step of this project.

References

[1] Achtelik, Markus, Tianguang Zhang, Kolja Kuhnlenz, and Martin Buss. "Visual Tracking and Control of a Quadcopter Using a Stereo Camera System and Inertial Sensors." *IEEE Xplore*, 18 Sept. 2009, doi:10.1109/ICMA.2009.5246421. Accessed 9 Nov. 2017.

[2] Algabri, Mohammed, Hassan Mathkour, MOhammed A. Mekhtiche, Mohammed A. Bencherif, and Mansour Alsulaiman. "Wireless Vision-Based Fuzzy Controllers for Moving Object Tracking Using a Quadcopter." *International journal of Distributed Networks*, vol. 13, no. 4, 27 Apr. 2017. Accessed 9 Nov. 2017.

[3] *Analog Devices*, 3 Oct. 2017, https://wiki.analog.com/resources/eval/user-guides/eval-adicup360. Accessed 9 Nov. 2017.

[4] Bartak, Roman, and Adam Vyskovsky. "Any Object Tracking and Following by a Flying Drone." *IEEE Xplore*, 10 Mar. 2016, doi:10.1109/MICAI.2015.12. Accessed 9 Nov. 2017.

[5] Evans, Brian L. "Goertzel Algorithm.", University of California, Berkeley, https://ptolemy.eecs.berkeley.edu/papers/96/dtmf_ict/www/node3.html. Accessed 9 Nov. 2017.

[6] International Commision on Non-Ionizing Radiation Protection. *Guidelines on limits of exposure to incoherent visible and infrared radiation*, International Commision on Non-Ionizing Radiation Protection. , Health Physics, 2013, pp. 74-96, www.icnirp.org/cms/upload/publications/ICNIRPVisible_Infrared2013.pdf. Accessed 9 Nov. 2017

[7] "EXCALIBUR LOW-NOISE HIGH-SPEED JFET-INPUT OPERATIONAL AMPLIFIERS." *www.ti.com*, Texas Instruments, Inc., Dec. 2009, www.ti.com/lit/ds/symlink/tle2072.pdf. Accessed 9 Nov. 2017.

[8] "Forecasts." *Driverless car market watch*, 26 Oct. 2017, www.driverless-future.com/?page_id=384. Accessed 9 Nov. 2017.

3

[9] Joshi, Divya. "Commercial Unmanned Aerial Vehicle (UAV) Market Analysis – Industry trends, companies and what you should know." *TECH INSIDER*, 8 Aug. 2017, www.businessinsider.com/commercial-uav-market-analysis-2017-8. Accessed 9 Nov. 2017.

[10] Kendall, Alex G., Nishaad N. Salvapantula, and Karl A. Stol. "On-Board Object Tracking Control of a Quadcopter with Monocular Vision." *IEEE Xplore*, 26 June 2014, doi:10.1109/ICUAS.2014.6842280. Accessed 9 Nov. 2017.

[11] "Kinetis K64F Sub-Family Data Sheet." *PJRC Electronics*, 8 June 2015, https://www.pjrc.com/teensy/K64P144M120SF5.pdf. Accessed 9 Nov. 2017.

[12] Mondragon, Ivan F., Pascual Campy, Miguel A. Olivares-Mendez, and Carol Martinez. "3D Object Following Based on Visual Information for Unmanned Aerial Vehicles." *IEEE Xplore*, 28 Nov. 2011, doi: 10.1109/LARC.2011.6086794. Accessed 9 Nov. 2017.

[13] Odame, Kofi M. "Compensating the Transimpedance Amplifier." Thayer School of Engineering at Dartmouth, Hanover, NH.

[14] "OSLON Black Series (850 nm) - 150° Version 1.3 SFH 4716AS." *Viewpoint Systems*, OSRAM Opto Semiconductors GmbH , 1 Feb. 2012, www.osram-os.com/Graphics/XPic7/00204353_0.pdf. Accessed 29 Jan. 2016.

[15] OSRAM. *Eye Safety IREDs used in lamp applications*, OSRAM. , OSRAM, 2010, www.osram-os.com/Graphics/XPic2/00052113_0.pdf/Application%20Note%20Eye%20Safety.pdf. Accessed 9 Nov. 2017.

[16] Pestana, Jesus, Jose L. Sanchez-Lopez, Srikanth Saripalli, and Pascual Campoy. "Computer Vision Based General Object Following for GPS-denied Multirotor Unmanned Aerial Vehicles." *IEEE Xplore*, 21 July 2014, doi:10.1109/ACC.2014.6858831. Accessed 9 Nov. 2017.

[17] "Sensors - Demand and Sales Forecasts, Market Share, Market Size, Market Leaders." , The Freedonia Group, Inc, Oct. 2012, https://www.freedoniagroup.com/Sensors.html. Accessed 9 Nov. 2017.

[18] *STMicroelectronics*, 2016, www.st.com/en/evaluation-tools/nucleo-f401re.html. Accessed 9 Nov. 2017.

[19] "Using Microcontrollers in Digital Signal Processing Applications." *www.silabs.com*, Silicon Laboratories, Inc., https://www.silabs.com/documents/public/application-notes/an219.pdf. Accessed 10 Nov. 2017.

[20] *Viewpoint Systems*, https://www.viewpointusa.com/industrial-embedded/when-is-an-fpga-worth-it-and-when-is-it-not-when-developing-an-industrial-embedded-system/. Accessed 9 Nov. 2017.

[21] Wholesale revenue consumer electronics (CE) shipments in the U.S. from 2009 to 2017 (in billions of U.S. dollars) https://www.statista.com/statistics/272115/revenue-growth-ce-industry/

[22] Ziegler, J G., and N B. Nichols. "Optimum Settings for Automatic Controllers." 1942.

Appendix A: Light Source Modulation Scheme

The overarching premise for modulating the light signal stems from the need to distinguish the IR light signal from any ambient IR light. Non-IR light is of minimal concern due to the tight wavelength response of the photodiode around 850nm. By modulating the light at a particular frequency, the signal received by the sensor can be analyzed at a particular frequency to reveal our signal of interest.

The first step in this was to choose a frequency that fit the parameters associated with avoiding ambient noise as well as having the capability of being sampled by the microcontroller ADC. With this being the case we want to avoid frequencies like those present at 60 and 120 Hz in building electricity and also be able to be sampled well above the Nyquist Limit (>>2*fsource). We used these parameters to arrive at a source frequency of 500 Hz.

We proceeded to make a choice in signal shape between sin and square wave. The sin wave was initially appealing due to its distinguishing features of amplitude and phase however upon looking into the use of a square wave we found that it has many of the same attractive features. Sine waves are nice due to their spectral purity, but we confirmed that the majority of energy in a square wave is in the fundamental harmonic. When filtering for the given frequency the square wave can be discerned due to this quality and the sources features can be confirmed for tracking.

The square wave was chosen and implemented with a 555 timer initially, however we looked into a digital option (Teensy LC) for modulating the signal as well. We proceeded to test the two sources against each other and implement our chosen filter to determine how much of the fundamental energy was leaking out of our chosen 500 Hz bin. As shown in Figure A1, the digital solution has a greater magnitude at 500 Hz, exhibiting greater spectral purity and thus improved SNR.

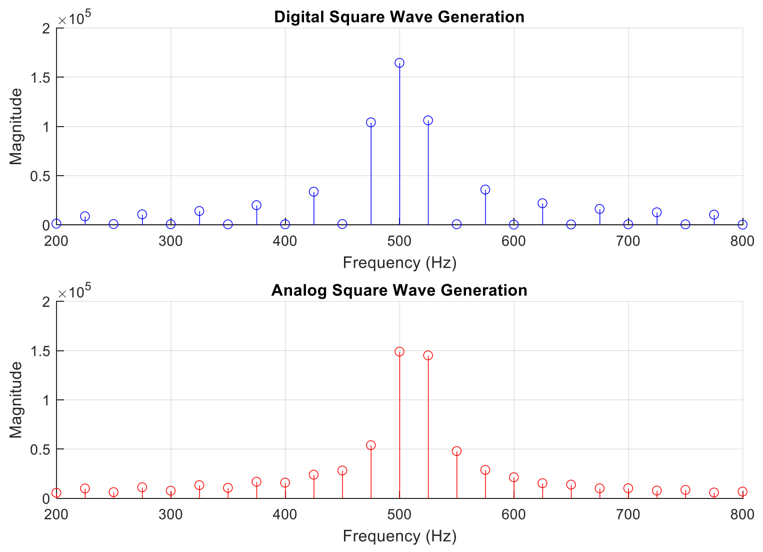


Figure A1. Detected Spectra of Analog and Digitally Generated Square Waves

An added feature of the digital solution using the Teensy LC returns to our overarching requirement that the system be replicable for future use by ADI. The removal of analog components allows for a more streamlined approach that increases performance across a range of our specifications. Analog components add a layer of complexity in achieving a consistent and precise solution and with this, the digital modulation approach emerges as the winner.

Appendix B: Modulated Signal Detection

Once the modulated IR signal from the light source reaches the sensor system, it must be digitally demodulated to perform the spatial measurements. In our case, this demodulation process is rather simple and amounts to recognizing the magnitude of the 500 Hz content on each of the four photodiode channels. We compared a few demodulation methods and chose the best alternative.

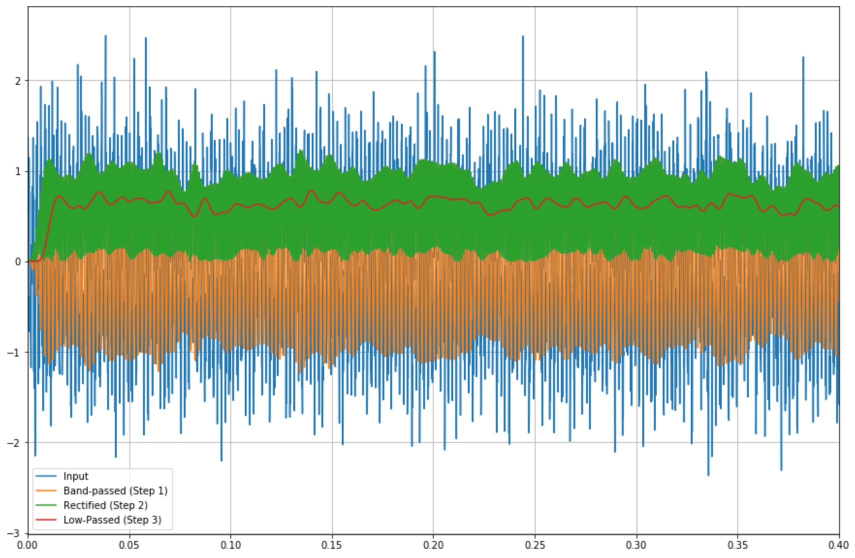


Figure B1. 3-stage demodulation scheme using BPF, rectifier, LPF

The first option considered is most intuitive in the time-domain. It essentially consists of applying a tight band-pass filter around the modulation frequency, rectifying the result, and determining the envelope of the rectified signal with a low-pass filter. Figure B1 shows a simulation of how this method would demodulate a varying 500Hz signal.

While this process seems simple, the tight band-pass filter and final low-pass filter are computationally taxing since they require hundreds of multiplications with digital filter taps. Since these computations will be performed on a processing unit that is also performing other functions, it is in our best interest to minimize computation time and maximize bandwidth.

Another option is to perform an FFT and look for the magnitude of the modulation frequency. For our application, computing only a few FFT bins would be sufficient, since the rest of the spectral content is of little interest to us. When computing individual terms of a DFT, a DSP technique called the Goertzel algorithm is computationally advantageous over the FFT [5]. When computing M points of the DFT from N data points, the complexity of the two methods are:

$$FFT: O(KNlogN)$$
$$Goertzel\ Algorithm: O(KNM)$$

As long as the number of DFT bins needed is less than log(N), which is true in our case, the Goertzel algorithm proves less computationally complex than the FFT. For this reason, the Goertzel algorithm was chosen as the demodulation method.

Appendix C: Analog Front End Design

The ADI LAS consists of four photodiodes. A photodiode is an optoelectronic device that outputs a photocurrent proportional to the intensity of incident light. Since "current" signals are cumbersome to process, our first task is to convert this current into a voltage with a circuit called a transimpedance amplifier.

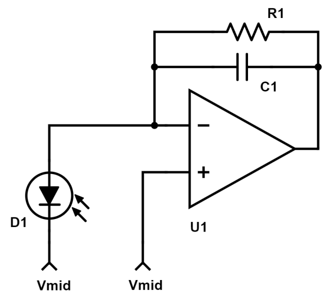


Figure C*1.* Compensated Transimpedance Amplifier Schematic

A transimpedance amplifier is implemented with an operational amplifier (op amp) in negative feedback. We selected an op amp (TLE2074) based on its extremely low input current noise rating (2.8 fA/√Hz) [7], which is critical in sensitive TIA applications. Below is a diagram of an op amp in negative feedback.

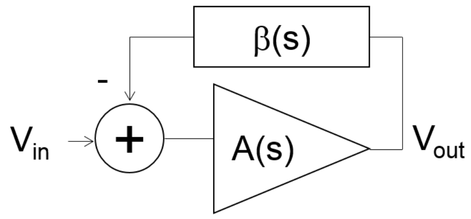


Figure C*2.* Operational Amplifier in Negative Feedback

We can see from the transfer function below that the system will become unstable (attempt to output an arbitrarily large voltage) when $A(s)\beta(s)$ = -1.

$$\frac{Vout(s)}{Vin(s)} = \frac{A(s)}{1 + A(s)\beta(s)} \tag{C1}$$

Thinking about this quantity in the complex plane, we can describe the unstable condition as $|A(s)\beta(s)|$ = 1 and $\angle A(s)\beta(s)$ = -180 degrees. Our goal is to prevent this condition from arising, namely by ensuring that the phase angle stays above -180 degrees when the quantity is at unity magnitude. This can be visualized in a frequency response with a quantity called the phase margin, which is the difference in the phase response from -180 degrees when the magnitude response reaches 0dB (unity gain).

We can avoid the occurrence of this condition by adding a zero to the system in the form of a feedback capacitor. After adding the feedback capacitor, the transfer function becomes:

(C2)

$$A(s)\beta(s) = \frac{A_0(1 + sRC_F)}{(1 + s\tau)(1 + sR(C_F + C_p))} \tag{7}$$

Where A_0 is the open-loop gain of the op amp, τ is a parameter that can be calculated from the open-loop gain and gain-bandwidth product, C_F and C_p are the feedback capacitance and parasitic capacitance of the photodiode, and R is the feedback resistance.

We also need to ensure that we have sufficient gain of our photocurrent and that our frequency of interest (modulation frequency) is not being attenuated. The relationship between the photocurrent and TIA output voltage is given by the following:

$$\frac{V_{out}}{I_{light}} = \frac{R}{(1 + sRC_F)} \tag{C3}$$

Since the photocurrent generated by each photodiode is on the order of micro amps, we would need to apply a gain on the order of 1 million to bring the signal into the volt range. For this reason, a 2MΩ feedback resistor and a 15pF feedback capacitor was chosen. To verify that this design would satisfy our objective of stability, sufficient gain, and a passband that extends beyond our modulation frequency, both equation C2 and C3 were simulated in MATLAB. Plots can be viewed in figures C3 and C4.

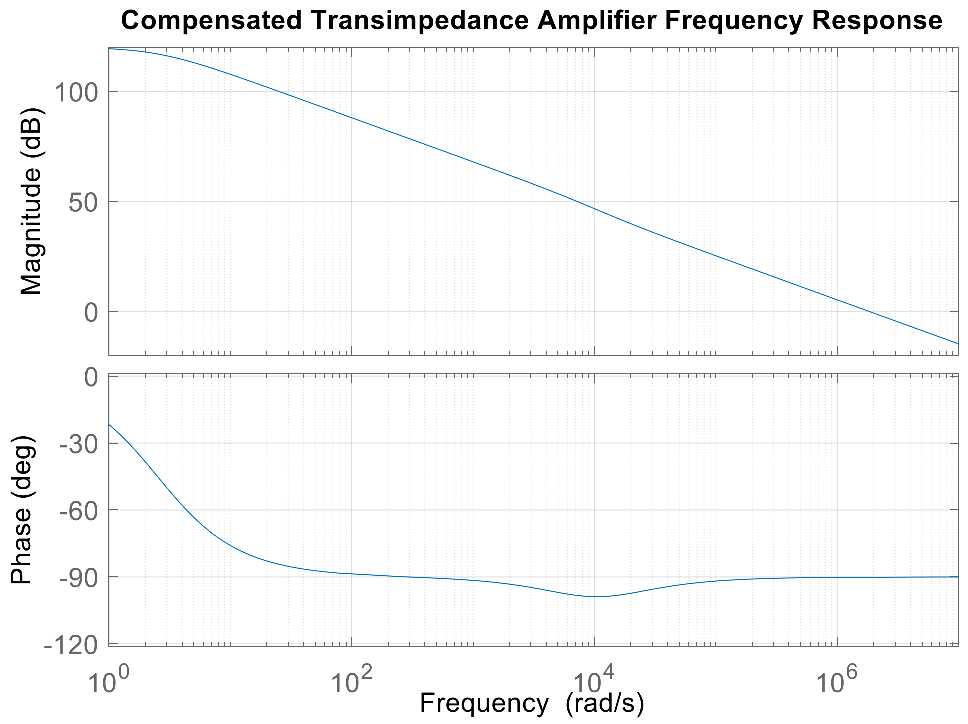


Figure C*3*. Compensated TIA Frequency Response

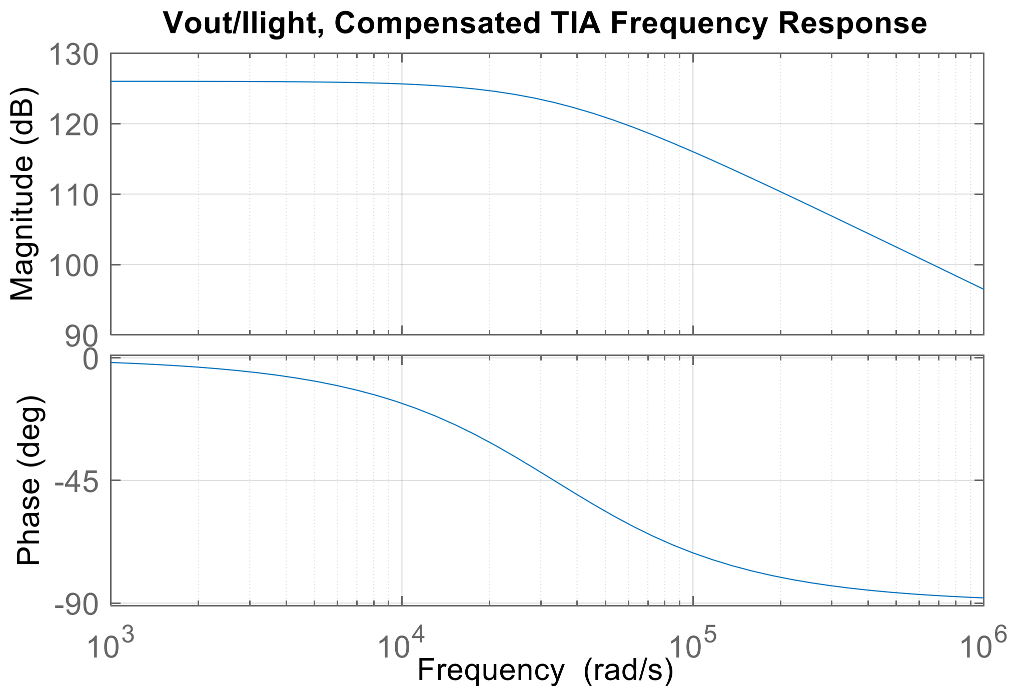


Figure C*4*. Vout/Ilight TIA Frequency Response

The major takeaways are that with our chosen values of feedback resistor and capacitor, we achieve stability (89.8 degree phase margin), appropriate passband location, and sufficient passband gain (126dB).

Now that a voltage proportional to the photodiode current response has been obtained, the signal must be sampled and processed. However, one crucial piece of analog circuitry must be incorporated prior to sampling—the anti-aliasing filter (AAF). To understand the need for the AAF, we can think of the sampling process as multiplying our time-domain signal by a comb function (a series of Dirac delta functions). Now thinking in the frequency domain, the spectrum of the sampled signal can be represented as the Fourier transform of the time-domain signal convolved with the Fourier transform of the Comb function (as a result of the convolution theorem). Since the Fourier transform of a comb function is another comb function, the resulting spectrum is a periodic replication of the original signal's spectrum. If the original signal has significant spectral content above half the sampling frequency, spectral overlap (also called aliasing) will occur. To avoid this, an anti-aliasing low-pass filter was implemented with a cutoff frequency close to our signal of interest. Since our sampling rate is several times larger than our modulated light frequency, a first-order passive filter was deemed sufficient, since such a filter with cutoff at about 500Hz would have -20dB attenuation at 5000Hz (half of the 10 kHz sampling rate). Going forward, our team will experiment with a higher-order AAF and compare performance. The AAF currently being implemented is shown in C5.

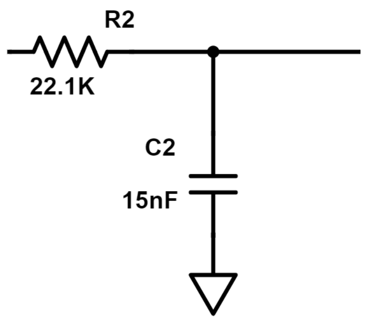


Figure C5. Schematic representation of AAF

These designs were replicated identically for each of the four photodiode channels.

Appendix D: Processing Unit Selection

A major decision within our solution space is the choice of processing unit that will sample the analog signals and perform the computation necessary for obtaining the three spatial coordinates of the light source. The first choice that must be made is that between an FPGA and a microcontroller. The predominant consideration dictating this choice is our objective to showcase the simplicity of the LAS in that its outputs are four analog signals, rather than a multitude of pixels in the case of a CMOS camera. In order to drive home the point that ADI's sensor requires a simpler interfacing system than the SOA, the choice of microcontroller over FPGA is beneficial. This is because programming an FPGA is generally a more complicated process than writing sequential code for a microcontroller. While not key requirements in creating our demonstration, cost and power consumption are typically key considerations for developers, both of which also point to the choice of a microcontroller. Lastly, an FPGA, since processing is parallelized, is more difficult to debug, which would slow our development process [20].

After deciding to use a microcontroller over FPGA for our data processing, a particular model of MCU still needs to be chosen. The sponsor had suggested two models: the STM Nucleo-64 development board as well as the ADICUP-360 development platform. Table D1 shows some relevant specifications of these two boards, as well as those of a third option: the Teensy 3.5.

Table D1. Processing MCU Comparison

	Teensy 3.5	**STM Nucelo-64**	**ADICUP360**
Physical Dimensions	6.2cm X 1.8cm	8.3cm X 7.0cm	10.2cm X 5.4cm
Clock Speed	120 MHz	82 MHz	16 MHz
ADC Sample Rate	4600 kSPS	2400 kSPS	3.9 kSPS
# ADCs	2	1	2
ADC Resolution	16 bits	12 bits	24 bits
Arduino Compatible	Yes	No	No
Price	$24.25	$12.74	$45.00

[3, 8, 11]

Having chosen an IR modulation frequency of 500Hz, the system we implement must be capable of sampling the four analog signals at a rate high enough to capture our frequency of interest. We would like to sample the signal at several times its highest contained frequency for improved performance. The ADICUP360 platform can be ruled out just from this requirement. Even if both ADCs are used in tandem, the effective sampling rate that can be obtained is 6.8 kSPS. Split among the four channels, this implies we can sample each channel at a maximum of 1.7 kSPS, which is just over three times the highest frequency of interest. Particularly in a prototyping stage, it is in our best interest to choose a microcontroller that will not come close to limiting overall system performance. If, for some reason, we decide to increase our modulation frequency, the ADICUP360 platform would likely not be able to demodulate our signal.

The Teensy 3.5 was originally considered due to familiarly by some of the team members. The Teensy platform was ultimately chosen for two main reasons: compatibility with the Arduino programming language and its small form factor. Arduino is a C-based embedded programming language which provides useful high-level functionality for frequently used processes. A microprocessor that is compatible with Arduino will speed up development of our prototype considerably. However, the level of abstraction provided by some Arduino functions come at a cost—speed. We decided that this would not be a major issue because the Teensy 3.5 boasts a 120MHz clock, partially mitigating the loss of speed in some Arduino functions. In addition, the Arduino functions can be overwritten in C or ARM Assembly for optimization if needed.

The Teensy's small form factor is another reason it would perform well in our application. With an area of only about 11 cm^2, the Teensy 3.5 development board could feasibly be placed directly on a small PCB that incorporates the entire sensor system. For these reasons, the Teensy 3.5 is our choice of processing unit in this project.

Appendix E: PCB Design

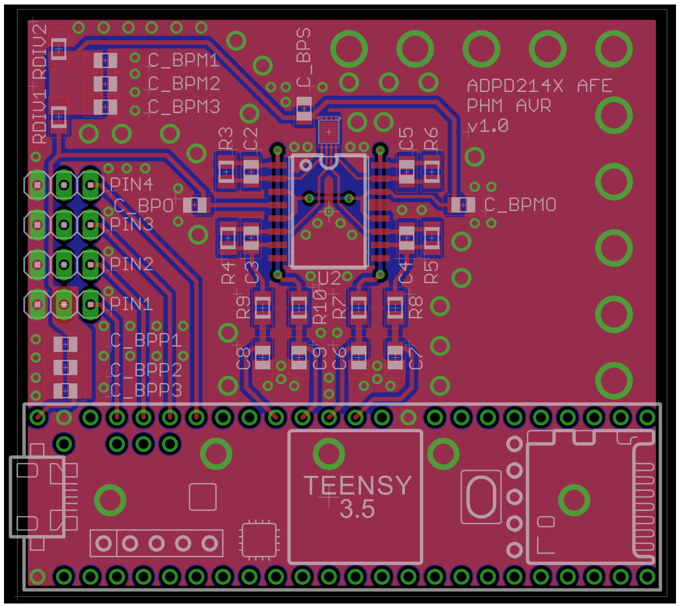


Figure E1. First Revision of Printed Circuit Board Layout for Sensor AFE & MCU

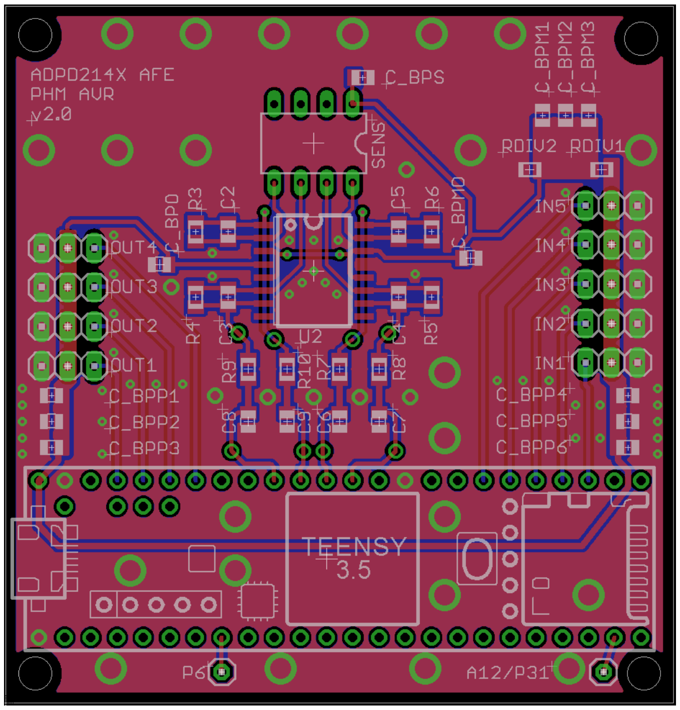


Figure E2. Final Revision of PCB Layout for Sensor AFE & MCU

There are many considerations to be made when laying out the Sensor PCB as its functionality can be affected by a multitude of factors and parasitic effects. The aspects considered have included:

Inductance Loops: As signals complete their loops to their respective components they create a magnetic field based on the parasitic inductance associated with loop length. This can create noise on the board that can interfere with other signals and decrease SNR. This is ameliorated by reducing loop lengths, shielding sensitive signals, and using bypass capacitors.

Signal Coupling: Signals can couple with one another through magnetic fields and with this it is important to note which signals are sensitive to noise and take steps to limit it. The way that this has been limited is by noting that our 4 channel signals from the sensor are sensitive all the way to their output so they have been separated by a ground signal between each to cause magnetic fields to be limited to the signals interaction with this ground.

High frequency noise: Due to the fact that the MCU must take in the sensor signals after the AAF's it has been placed on the same board to limit the distance these signal must travel. With this, the HF functions of the MCU produce noise that can also interfere with the signals. To deal with this we have placed ground planes on the top and bottom that prevent this noise from coupling with our 4 channel signals.

Reference Consistency: Each signal is passing through various components and at the same time its magnitude is being referenced to the ground that particular component utilizes. It is worth acknowledging that grounds meaning can vary throughout locations on the board so efforts must be made to keep signal referencing consistent. This has been approached by placing bypass capacitors for components as well as their ground references as close together as possible.

Future steps for PCB work will include validating this design's functionality and making any alterations that may be required in a future version as well as designing a PCB for the LED driving circuit.

We also designed, fabricated, and populated an LED PCB for our external light source such that our Teensy LC microcontroller could drive a square wave through our LED and the current could be controlled through a current-limiting resistor.

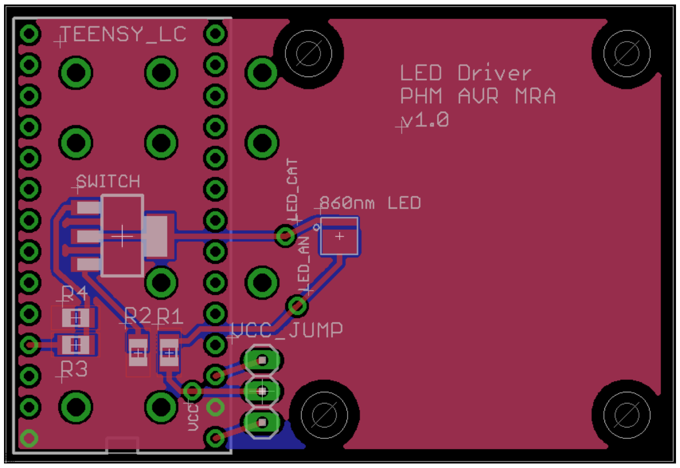


Figure E3. Final LED PCB used to drive our square wave signal through the LED.

Appendix F: Teensy Code

```
#include <TimerOne.h>
#include <math.h>

/*This code runs on the Teensy 3.5 microcontroller. It is designed to sample the analog
signals corresponding to the four LAS photodiodes, detect the magnitude of 500Hz content,
calculate X, Y, distance, and use these to control the quadcopter via PWM outputs to the
flight controller.

Authors: Arun Reddy and Prescott McLaughlin

#define modFreq  500 //LED modulation frequency
#define Fsample  10000 //per channel sampling rate (Hz)

#define XR      2
#define XL      1
#define YB      3
#define YT      2

#define XR_index 0
#define XL_index 1
#define YB_index 2
#define YT_index 2

//PWM input pins
#define ROLL 36
#define PITCH 37
#define YAW 39
#define THROTTLE 38
#define SWITCH 35

//PWM output pins
#define ROLL_OUT 23
#define PITCH_OUT 22
#define YAW_OUT 20
#define THROTTLE_OUT 21

//PWM output resolution
#define PWM_RESOLUTION 16

//Flight mode switch parameters
#define SWITCH_THRESHOLD_LOW 1000
#define SWITCH_THRESHOLD_HIGH 1500

//Light intensity threshold (SNR deemed insufficient after this value)
#define DISTANCE_THRESHOLD 300

//Maximum error magnitude
#define E_MAX 70.0

//Throttle control parameters
#define MAX_SAFE_THROTTLE 1520
#define MIN_SAFE_THROTTLE 1415
int throttleBias = 1450;

//Yaw control parameters
#define MAX_SAFE_YAW 1580
#define MIN_SAFE_YAW 1435
#define ZERO_YAW 1507

//Pitch control parameters
#define MAX_SAFE_PITCH 1578
#define MIN_SAFE_PITCH 1500
#define ZERO_PITCH 1540
```

```
#define DISTANCE_SETPOINT 80 //following distance in (cm)

int calculatedPitch = ZERO_PITCH;

boolean autoFlightMode = false; //0 is user mode, 1 is sensor nav mode
boolean transitionFlag = false;

//Data Processing Variables
const float targetFreq = 500;

const int samplesPerPeriod = Fsample / modFreq;
const int NPeriods = 10; //number of periods to capture in one sampling burst
const int N = NPeriods*samplesPerPeriod;  //how many samples to take before processing them

const int PWM_FREQ = 68; //output PWM frequency (Hz)
const float outputPWMperiod = 1.0/PWM_FREQ;
const int fullScalePWM = pow(2, 16);

long dataBuffer[4][200]; //buffer for storing data, 4 channels with 200 samples each CHANGE TO SIZE N

int dataBufferIndex = 0;

float sine;
float cosine;
float Q1;
float Q2;
float coeff;

float XRmag = 0;
float XLmag = 0;
float YBmag = 0;
float YTmag = 0;
float X = 0;
float Y = 0;
int intX = 0;
int intY = 0;
float totalIntensity = 0;

//Data Correction Variables
float correctionCoeffsX[] = {-8.0386e-12, 4.6137e-10, -8.3266e-8, -4.2996e-7, 1.5913e-4, -3.7879e-4,
0.7167};
float correctionCoeffsY[] = { 3.2423e-14, -2.8621e-13, -1.65e-10, 1.6724e-9, 9.4394e-8, -2.6331e-6,
7.0754e-5, 6.9864e-4, 1.0998 };
float correctionFactorX = 0;
float correctedIntensityX = 0;
float correctionFactorY = 0;
float correctedIntensityXY = 0;

//Data Linearization Variables
float K = 2122.36;
float b = -27.17;
float distance = 0;

//PWM Variables
int pulseWidthRoll = 0;
int pulseWidthPitch = 0;
int pulseWidthYaw = 0;
int pulseWidthThrottle = 0;
int pulseWidthSwitch = 0;

int outputRollVal = 0;
int outputPitchVal = 0;
int outputYawVal = 0;
int outputThrottleVal = 0;

char txString[10];

//Throttle PID variables
float pGainT = 0.82;
float pTermT = 0;

float dGainT= -12;
float dTermT = 0.0;
float prevY = 0.0;

float iGainT = 0.005;
float iTermT = 0;
float iStateT = 0;

int calculatedThrottle = throttleBias;

//Yaw PI variables
float pGainY = 2.1;
float pTermY = 0;

float iGainY = 0.01;
float iTermY = 0;
float iStateY = 0;

//Pitch PD variables
float distError = 0;
float pGainP = 1.5;
float pTermP = 0;

float dGainP = -50;
float dTermP = 0;
float prevDist = 0.0;

//Since photodiode response decreases past the saturation point, we assign all intensities
//beyond a certain value "SATURATION_INTENSITY" to the same distance "SATURATION_DISTANCE"
#define SATURATION_INTENSITY 6.6
#define SATURATION_DISTANCE 33


void setup() {

        pinMode(XR, INPUT);
        pinMode(XL, INPUT);
        pinMode(YB, INPUT);
        pinMode(YT, INPUT);

        pinMode(ROLL, INPUT);
        pinMode(PITCH, INPUT);
        pinMode(YAW, INPUT);
        pinMode(THROTTLE, INPUT);
        pinMode(SWITCH, INPUT);

        pinMode(ROLL_OUT, OUTPUT);
        pinMode(PITCH_OUT, OUTPUT);
        pinMode(YAW_OUT, OUTPUT);
        pinMode(THROTTLE_OUT, OUTPUT);


        pinMode(6, OUTPUT); //timing analysis pin

        //ADC settings
        analogReadResolution(16);
        analogReference(0);

        analogWriteFrequency(ROLL_OUT, PWM_FREQ);
        analogWriteFrequency(PITCH_OUT, PWM_FREQ);
        analogWriteFrequency(YAW_OUT, PWM_FREQ);
        analogWriteFrequency(THROTTLE_OUT, PWM_FREQ);
        analogWriteResolution(PWM_RESOLUTION);

        Timer1.initialize(1e6 / Fsample);
        Timer1.attachInterrupt(samplingISR);
        initGoertzel(targetFreq);
        Serial.begin(9600);
        Serial3.begin(115200);
}

//Interrupt Service Routine responsible for filling data buffers at a precise sampling interval
void samplingISR() {

        if (dataBufferIndex < N) {

                dataBuffer[XR_index][dataBufferIndex] = analogRead(XR);
                dataBuffer[XL_index][dataBufferIndex] = analogRead(XL);
                dataBuffer[YB_index][dataBufferIndex] = analogRead(YB);
                dataBuffer[YT_index][dataBufferIndex] = analogRead(YT);

                dataBufferIndex++;
        }
}

//Resets variables in the Goertzel Algorithm
void resetGoertzel() {
        Q1 = 0;
        Q2 = 0;
}

//Initializes Goertzel variables to detect our target frequency
void initGoertzel(float targetFreq) {
        int k;
        float floatN = (float)N;
        float w;

        k = (int)(0.5 + ((floatN * targetFreq) / Fsample));
        w = (2.0*PI*k) / floatN;
        sine = sin(w);
        cosine = cos(w);
        coeff = 2.0 * cosine;

        resetGoertzel();
}

//Performs the Goertzel Algorithm
float performGoertzel(uint8_t channel_index) {

        Q1 = 0;
        Q2 = 0;

        int sum = 0;
        int mean = 0;
        for (int i = 0; i<N; i++) {
                sum = sum + dataBuffer[channel_index][i];
        }

        mean = (int)(sum / N);

        for (int i = 0; i < N; i++) {
                float Q0;
                Q0 = (coeff * Q1) - Q2 + (float)dataBuffer[channel_index][i] - mean;
                Q2 = Q1;
                Q1 = Q0;
        }

        return (Q1*Q1 + Q2*Q2 - (Q1*Q2*coeff));
}

//Calculates X, Y and distance
void processData() {

        XRmag = performGoertzel(XL_index);
        XLmag = performGoertzel(XR_index);
        YBmag = performGoertzel(YB_index);
        YTmag = performGoertzel(YT_index);

        totalIntensity = (XRmag  + XLmag  + YBmag  + YTmag) / 1e12;

        X = 100*((XRmag - XLmag) / (XRmag + XLmag));
        Y = 100*((YTmag - YBmag) / (YTmag + YBmag));

        correctIntensityX();
        correctIntensityY();
        convertIntensitytoDistance();

        dataBufferIndex = 0;
}

//Corrects X dependence in total intensity metric
void correctIntensityX() {

        correctionFactorX =
                correctionCoeffsX[0] * pow(X, 6) + correctionCoeffsX[1] * pow(X, 5) +
                correctionCoeffsX[2] * pow(X, 4) + correctionCoeffsX[3] * pow(X, 3) +
                correctionCoeffsX[4] * pow(X, 2) + correctionCoeffsX[5] * X + correctionCoeffsX[6];

        correctedIntensityX = totalIntensity / correctionFactorX;
}

//Corrects Y dependence in X corrected intensity metric
void correctIntensityY() {

        correctionFactorY =
                correctionCoeffsY[0] * pow(X, 8) + correctionCoeffsY[1] * pow(X, 7) +
                correctionCoeffsY[2] * pow(X, 6) + correctionCoeffsY[3] * pow(X, 5) +
                correctionCoeffsY[4] * pow(X, 4) + correctionCoeffsY[5] * pow(X, 3) +
                correctionCoeffsY[6] * pow(X, 2) + correctionCoeffsY[7] * X + correctionCoeffsY[8];


        correctedIntensityXY = correctedIntensityX / correctionFactorY;

}

//Linearizes corrected intensity metric and scales to (cm) measurement of distance
void convertIntensitytoDistance(){
        if (correctedIntensityXY > SATURATION_INTENSITY)
                distance = SATURATION_DISTANCE;
        else {
                distance = (-2 * correctedIntensityXY  * b + sqrt(pow(2 * correctedIntensityXY * b, 2) -
                        4 * correctedIntensityXY * (correctedIntensityXY * pow(b, 2) - K)))
                        / (2 * correctedIntensityXY);
                distance = distance + 12.4; //correction for changed gain
        }
}

//Sends values to data acquisition unit (DAQ) for real-time debugging and analysis
void sendXYtoDAQ() {

        sprintf(txString, "<%.2f,%.2f,%.1f>", X, Y, distance);
        //sprintf(txString, "<%.1f>", totalIntensity);
        Serial3.print(txString);
}

//Reads the mode switch signal from remote receiver to determine Auto/Manual mode transitions
void readModeSwitch() {
        pulseWidthSwitch = pulseIn(SWITCH, HIGH);
        if (pulseWidthSwitch > SWITCH_THRESHOLD_LOW && pulseWidthSwitch < SWITCH_THRESHOLD_HIGH) {
                if (!autoFlightMode) {
                        transitionFlag = true;
                        resetIntegralTerms();
                }
                autoFlightMode = true;
        }
        else
                autoFlightMode = false;
}

//Converts a pulse width in microseconds to an argument for PWM output function
int convertWidthtoPWM(int pulseWidth) {
        return (fullScalePWM) * (pulseWidth / 1e6) / outputPWMperiod;
}

//Outputs calculated throttle value to appropriate pin, and limits to safe values
void outputThrottle(int pulseWidth) {
        if (pulseWidth > MAX_SAFE_THROTTLE)
                analogWrite(THROTTLE_OUT, convertWidthtoPWM(MAX_SAFE_THROTTLE));
        else if (pulseWidth < MIN_SAFE_THROTTLE)
                analogWrite(THROTTLE_OUT, convertWidthtoPWM(MIN_SAFE_THROTTLE));
        else
                analogWrite(THROTTLE_OUT, convertWidthtoPWM(pulseWidth));
}

//Outputs calculated yaw value to appropriate pin, and limits to safe values
void outputYaw(int pulseWidth) {

        if (pulseWidth > MAX_SAFE_YAW)
                analogWrite(YAW_OUT, convertWidthtoPWM(MAX_SAFE_YAW));
        else if (pulseWidth < MIN_SAFE_YAW)
                analogWrite(YAW_OUT, convertWidthtoPWM(MIN_SAFE_YAW));
        else
                analogWrite(YAW_OUT, convertWidthtoPWM(pulseWidth));
}

//Outputs calculated pitch value to appropriate pin, and limits to safe values
void outputPitch(int pulseWidth) {

        if (pulseWidth > MAX_SAFE_PITCH)
                analogWrite(PITCH_OUT, convertWidthtoPWM(MAX_SAFE_PITCH));
        else if (pulseWidth < MIN_SAFE_PITCH)
                analogWrite(PITCH_OUT, convertWidthtoPWM(MIN_SAFE_PITCH));
        else
                analogWrite(PITCH_OUT, convertWidthtoPWM(pulseWidth));
}

//Performs the passthrough of remote commands in manual mode (reads and reproduces PWM)
void passthroughPWM() {

        pulseWidthThrottle = pulseIn(THROTTLE, HIGH);
        pulseWidthPitch = pulseIn(PITCH, HIGH);
        //pulseWidthRoll = pulseIn(ROLL, HIGH);
        pulseWidthYaw = pulseIn(YAW, HIGH);

        outputThrottleVal = convertWidthtoPWM(pulseWidthThrottle);
        outputPitchVal = convertWidthtoPWM(pulseWidthPitch);
        //outputRollVal = convertWidthtoPWM(pulseWidthRoll);
        outputYawVal = convertWidthtoPWM(pulseWidthYaw);

        analogWrite(THROTTLE_OUT, outputThrottleVal);
        analogWrite(PITCH_OUT, outputPitchVal);
        //analogWrite(ROLL_OUT, outputRollVal);
        analogWrite(YAW_OUT, outputYawVal);
}

//Resets the integral terms of PID controllers, so that value is refreshed upon re-entry to auto mode
void resetIntegralTerms() {
        iTermT = 0;
        iTermY = 0;
}

//Throttle control algorithm (up/down motion)
void throttleControl() {
	if (transitionFlag) {
		throttleBias = pulseWidthThrottle;
		transitionFlag = false;
	}
	else {

		pTermT = pGainT * Y;

		dTermT = dGainT * (prevY - Y);
		prevY = Y;

		iStateT += Y;
		iTermT = iGainT * iStateT;

		if (iTermT > (MAX_SAFE_THROTTLE - throttleBias))
			iStateT = (MAX_SAFE_THROTTLE - throttleBias) / iGainT;
		else if (iTermT < (MIN_SAFE_THROTTLE - throttleBias))
			iStateT = (MIN_SAFE_THROTTLE - throttleBias) / iGainT;

		calculatedThrottle = throttleBias + (pTermT + dTermT + iTermT);

		outputThrottle(calculatedThrottle);
	}
}

//Yaw control algorithm (angle adjustment)
void yawControl(){

	pTermY = pGainY * X;

	iStateY += X;
	iTermY = iGainY * iStateY;

	if (iTermY > (MAX_SAFE_YAW - ZERO_YAW))
		iStateY = (MAX_SAFE_YAW - ZERO_YAW) / iGainY;
	else if (iTermY < (MIN_SAFE_YAW - ZERO_YAW))
		iStateY = (MIN_SAFE_YAW - ZERO_YAW) / iGainY;

	outputYaw(ZERO_YAW + (pTermY + iTermY));
}

//Pitch control algorithm (forward/backward motion)
void pitchControl() {

	distError = distance - DISTANCE_SETPOINT;

	pTermP = pGainP * distError;

	dTermP = dGainP * (prevDist - distance);
	prevDist = distance;

	/*if (distance > 80)
		calculatedPitch = MAX_SAFE_PITCH;
	else if (distance < 60)
		calculatedPitch = MIN_SAFE_PITCH;
	else
		calculatedPitch = ZERO_PITCH;*/

	calculatedPitch = (ZERO_PITCH + (pTermP + dTermP));
	outputPitch(calculatedPitch);
}

//Calculates aboslute value of a number
float absoluteVal(float val) {
	if (val > 0.0)
		return val;
	else
		return -val;
```

```
}

//Outputs the safest values to each of the commands
void hover() {
        outputThrottle(throttleBias); //output hover throttle
        outputYaw(ZERO_YAW);
        outputPitch(ZERO_PITCH);
}

void loop() {

        if (autoFlightMode) {
                interrupts();
                if (dataBufferIndex > (N - 1)) {
                        noInterrupts();
                        digitalWrite(6, HIGH); //timing analysis pin
                        processData();
                        digitalWrite(6, LOW); //timing analysis pin
                        sendXYtoDAQ();
                        if (distance < DISTANCE_THRESHOLD) {
                                pitchControl();
                                yawControl();
                                throttleControl();
                        }
                        else
                                hover();
                        readModeSwitch();
                        interrupts();
                }
        }

        else { //manual flight mode
                noInterrupts();
                sendXYtoDAQ();
                passthroughPWM();
                readModeSwitch();
        }
}
```

Appendix G: Triangulation vs. Intensity

Our ultimate goal in this project, which is to enable a quadcopter to track and follow a light source in three-dimensional space, requires that our sensing apparatus be able to accurately measure the distance to the source. The sensor does not provide a direct way of measuring the third spatial coordinate (distance). We have identified two potential methods by which to determine the third spatial dimension of the light source with respect to the sensor. The first relies on the intensity of the IR signal from the light source and using an accurate model of intensity over distance to determine the distance of the IR source relative to the sensor. The second uses two sensors placed a fixed distance apart to triangulate the Z location of the source using the X and Y angles provided by the sensors 2-D functionality.

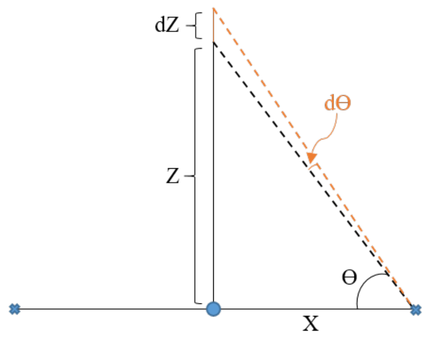


Figure G1. Triangulation Calculation Diagram

Triangulation is the approach currently utilized by ADI in their demo and with this, we began by evaluating its feasibility for a longer range solution. We did this by realistically choosing a maximum sensor separation of 10 cm that could be implemented on a PCB, with X in Figure G1 representing 5 cm. The next step was to sweep Z from 0 to 2 meters and evaluate how the angle of the object to the sensor is affected (Figure G2 top). From this it became apparent that at longer range the triangulation method should be evaluated by the metric dΘ/dZ to determine the degree of angular resolution needed in our operating range.

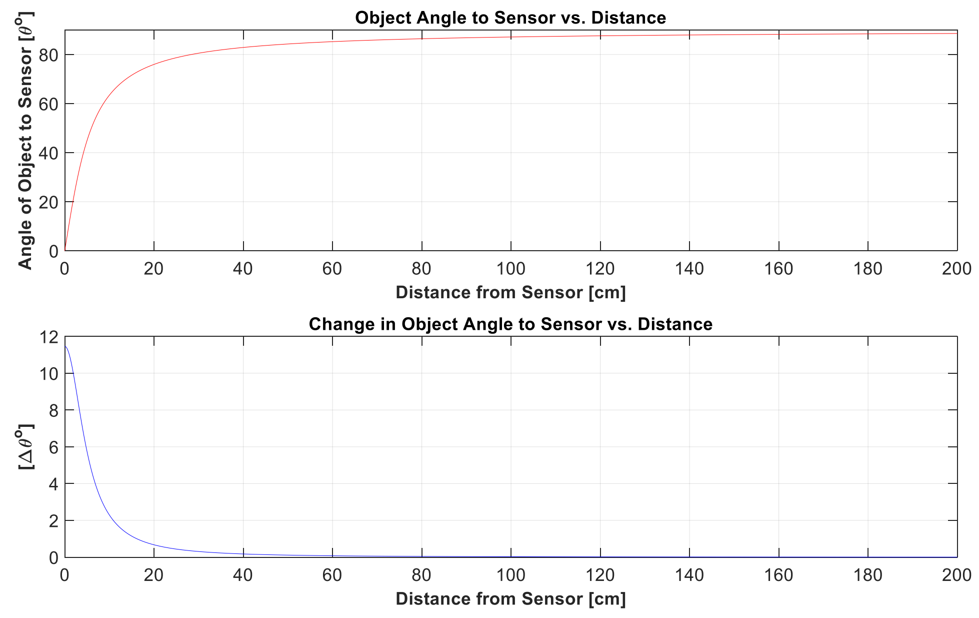


Figure G2. Angular dependencies a function of distance

Using the angular resolution approach, Figure G2 bottom shows the change in angle as distance is swept out to 200 cm. This plot illustrates the quick falloff that occurs in angular change as distance increases given the sensors fixed separation. This suggests that discerning angular changes will be difficult at the distances we will be operating in as there are very small angle deltas for very large spatial gaps in distance.

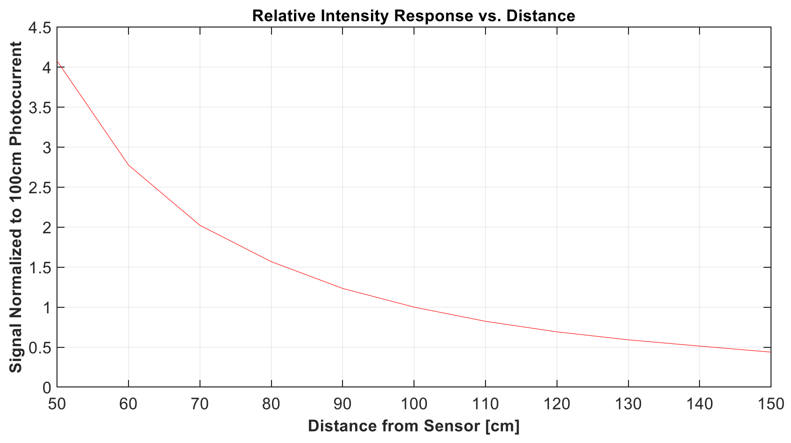
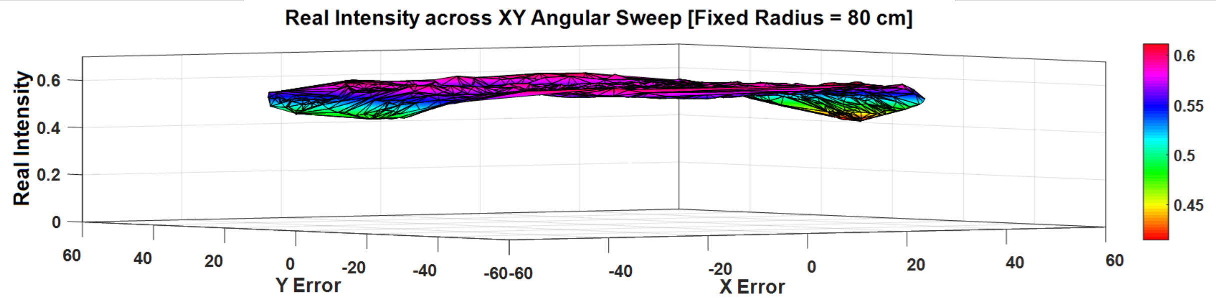


Figure G3. Sensor Photocurrent response as a function of distance (Z)

Our sponsors at ADI followed this by carrying out a relative sensor photocurrent characterization as a function of distance. The plot in Figure G3 indicates promise for this method as it is clear that the signal is definitively capturing distance changes out to 150 cm, and more.

Appendix H: Removal of X and Y Angular Dependence



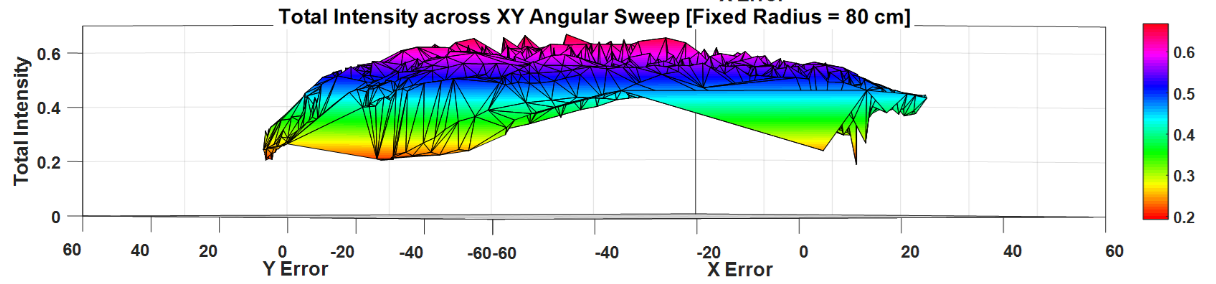


Figure H1. Real intensity: corrected (top), and total intensity: uncorrected (bottom)

Appendix I: DAQ Plots

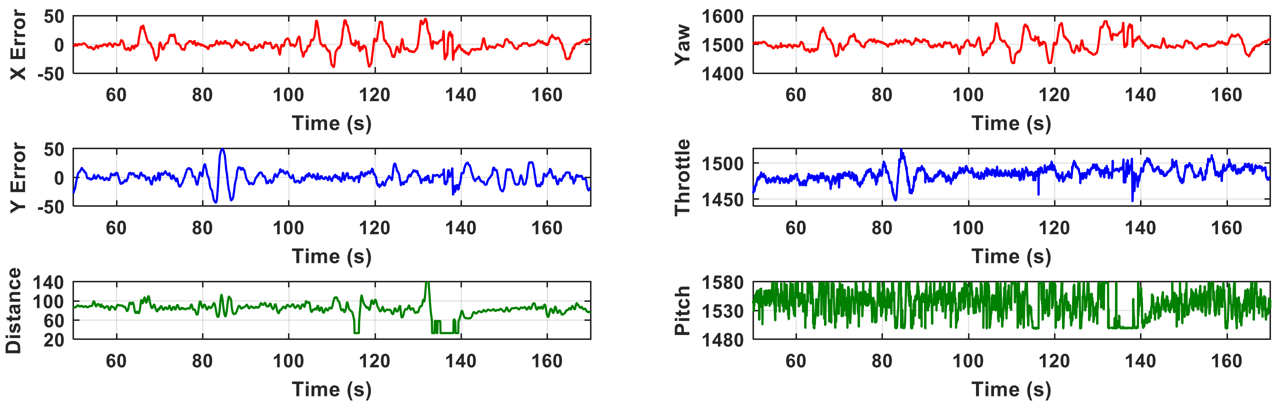


Figure I1. Representative plot of 3-D error measurements as well as their correlated control signals

Appendix J: Eye Safety Concerns

Research into the hazards of using an infrared light source as our tracking beacon was conducted in order to assess the limits of exposure to the light source. A report describing the calculations of these limits has been published by the International Commission on Non-Ionizing Radiation Protection (ICNIRP), in order to model general exposure scenarios to IR-A radiation (λ = 780-1400 nm) [6] . The manufacturer of the infrared emitting diode (IRED) being issued a guideline based on the ICNIRP's report, and both were used in conjunction to determine the health risk posed by the IRED. [15]

Damage to a person's eye can occur in either the retina or the cornea, and must therefore be analyzed separately. In order to determine the maximum allowed irradiance E_{IR} ($\mathrm{E}_{e,\max}$), the following was used:

$$E_{IR} = \sum_{\lambda=780}^{3000} E_\lambda \cdot \Delta\lambda \leq \frac{18000}{t^{0.75}} \quad [W \cdot m^{-2}] \tag{J1}$$

Where E_λ is the spectral irradiance in W/m^2/nm, $\Delta\lambda$ given in nm, and t is in seconds. Irradiance $\mathrm{E}_{e,\max}$ can be calculated using the radiant intensity I_e that found in the datasheet for the IRED, and the distance d using the inverse square law:

$$\mathrm{E}_e = \frac{\mathrm{I}_e}{d^2} \quad [W \cdot m^{-2}] \tag{J2}$$

Where d is in meters and I_e in W/sr.

Calculations for the radiance exposure limits for the iris were done with the following equation:

$$L_{IR} = \sum_{\lambda=780}^{1400} L_\lambda \cdot R(\lambda) \leq \frac{6000}{\alpha} \quad [W \cdot m^{-2} \cdot sr^{-1}]\ (t > 10s) \tag{J3}$$

Where L_λ is the spectral radiance in W/m^2/nm/sr, α the angular subtence of the source in radians, and $R(\lambda) = 10^{(700-\lambda)/500}$ being the burn hazard weighting function. A very accurate approximation for L_{IR}, however cab be obtained from the following equation in order to use datasheet values:

$$L_{IR} \approx \frac{\mathrm{I}_e \cdot R(\lambda)}{((l+w)/2)^2} \quad [W \cdot m^{-2} \cdot sr^{-1}] \tag{J4}$$

Where l and w are the length and width of the light source in meters.

Once all values were calculated, considering a forward current of 1 amp through the IRED and an exposure duration of over 1000 seconds, it is evident that there is no inherent danger in the implementation of this light source. In fact, the risk posed by the IRED under these conditions, even when viewing from a distance of 20 cm, which is the closest recommended measurement for realistic calculations, our case falls into the standardized exempt group (no hazard) established by OSRAM and the ICNIRP. The results of these calculations are depicted in figures J1 and J2, along with the resulting radiance from the IRED: 1.91E4 W/m/sr.

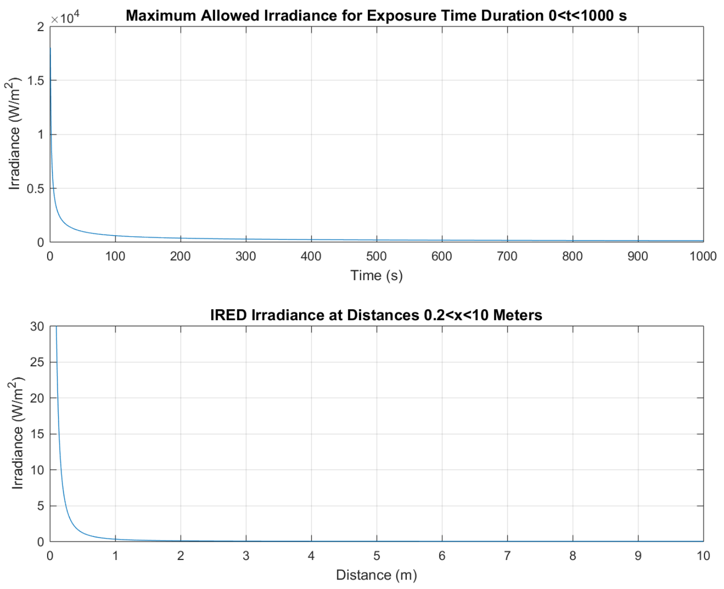


Figure J1. Comparison plots between the maximum allowed irradiance for a given time exposure, and the equivalent irradiance given by our infrared emitting diode, at the given distance for an exposure of 1000 seconds.

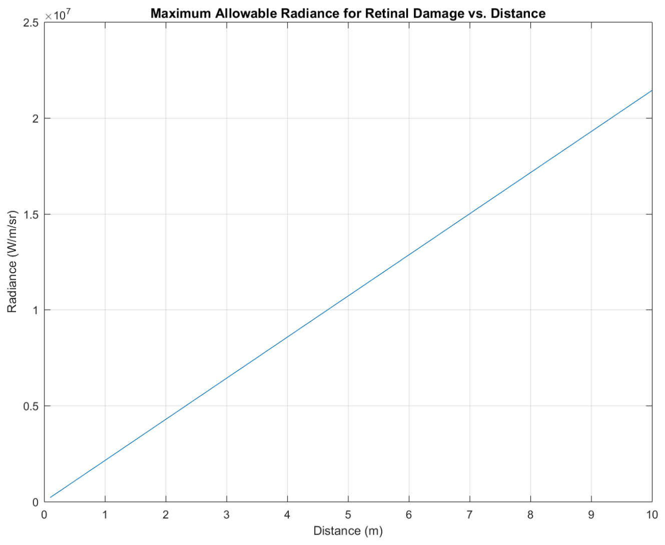


Figure J2. Maximum allowable radiance as a function of distance in W/m/sr. When analyzing the Radiance from our infrared light source (1.91E4 W/m/sr), it is clear that it is well below the danger threshold.

Appendix K: PCB Bill of Materials

Table K1. Sensor PCB Bill of Materials

Bill of Materials: Sensor PCB					
Designator	**Part #**	**Value**	**Footprint**	**Quantity**	**Price**
C_BPM1, C_BPP1	C0805X475J9RACAUTO	4.7 uF	0805	2	$0.79
C_BPM2, C_BPP2, C_BPO, C_BPS	0805YC105JAT2A	1 uF	0805	4	$0.16
C2, C3, C4, C5	600F150FT250XT	15 pF	0805	4	$1.92
C6, C7, C8, C9	CL21B153JBANNNC	15 nF	0805	4	$0.15
IN1-5, OUT1-4	PRPC040SAAN-RC	PIN	.100"	1	$0.66
RDIV1	ESR10EZPJ154	150 kΩ	0805	1	$0.10
RDIV2	ERJ-P06J274V	270 kΩ	0805	1	$0.12
R3, R4, R5, R6	RMCF0805FT22MO	22 MΩ	0805	4	$0.29
R7, R8. R9, R10	RNCP0805FTD22K1	22 kΩ	0805	4	$0.10
SENS	ADPD214X Brekaout	-	DIL-08	1	-
TEENSY 3.5	DEV-14056	-	-	1	$31.25
U2	TLE2074CDWR	-	SOIC-16	1	$5.25
TOTAL PRICE					**$40.79**

Table K2. LED PCB Bill of Materials

Bill of Materials: LED PCB					
Designator	**Part #**	**Value**	**Footprint**	**Quantity**	**Price**
860nm LED	SFH 4716AS	22 kΩ	3SMD	1	$4.96
JUMPER	QPC02SXGN-RC	-	.100"	1	$0.10
R1, R3	RC0805JR-070RL	0 Ω	0805	2	$0.10
R2	ERJ-6DQFR56V	560 mΩ	0805	1	$0.35
SWITCH	IRLL024ZTRPBF	MOSFET	SOT-223	1	$0.76
TEENSY LC	DE-13305	-	-	1	$14.38
VCC_JUMP	PRPC040SAAN-RC	PIN	.100"	1	$0.66
TOTAL PRICE					**$20.65**

Appendix L: Expense Report

Table L1. Total Expenditure Report

ENGS 89/90 Purchasing Summary			
Item(s)	**Vendor**	**Purchaser**	**Cost**
LEDs and breakout boards	Digi-Key	Arun	$42.87
High current BJTs	Digi-Key	Arun	$10.55
Parallax Controller	Parallax	Martin	$178.96
props/cables	Hobby King	Martin	$37.86
Teensy MCU and landing gear	Amazon	Arun	$43.45
Op amps	Digi-Key	Arun	$27.01
mini quadcoper	Banggood	Arun	$13.01
XT60 power converters	Banggood	Arun	$25.48
props/landing gear	HobbyKing	Martin	$20.91
mini drone batteries	amazon	Arun	$16.99
2 batteries	getFPV.com	Martin	$69.98
parallax cover	Parallax	martin	$19.85
teensy LC	Amazon	Arun	$16.93
DYS 320 Quadcopter	HobbyKing	Prescott	$151.47
Passive Components	Digi-Key	Prescott	$63.97
Parallax Controller	Parallax	Martin	-$178.96
ARRIS 250mm quad	Amazon	Martin	$266.99
Extra Teensy LC and 3.5	Amazon	Arun	$48.80
Soldering Stuff	Amazon	Arun	$41.52
Female headers + passives	Digi-Key	Arun	$23.05
Batteries and PPM encoder	Amazon	Arun	$20.98
PCB components	Digi-Key	Arun	$69.82
Sensor PCB Rev 2.0	OSH Park	Prescott	$99.19
Sensor PCB Rev 1.0	OSH Park	Arun & Prescott	$59.20
LED Driver PCB Rev 1.0	OSH Park	Prescott	$51.40
extra LEDs	Digi-Key	Arun	$32.40
passives + standoffs	Digi-Key	Arun	$59.64
Total Expenditures			$1,333.32

Appendix M: Deliverables

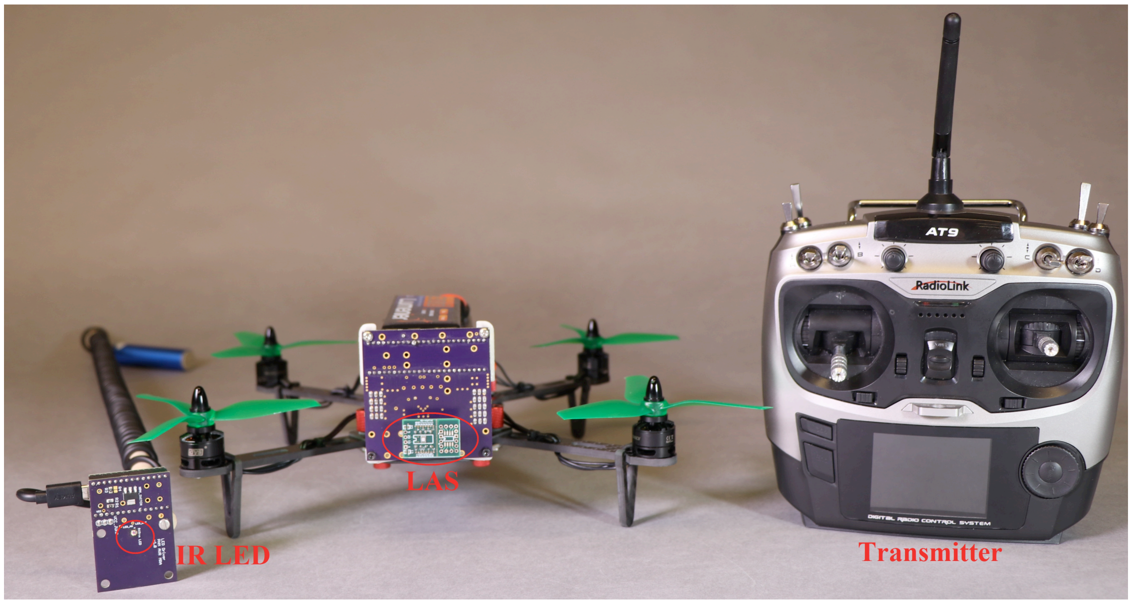


Figure M1. Deliverables from left to right: IR LED on PCB, drone with LAS on PCB, and handheld controller.

Appendix N: System Topology DSM

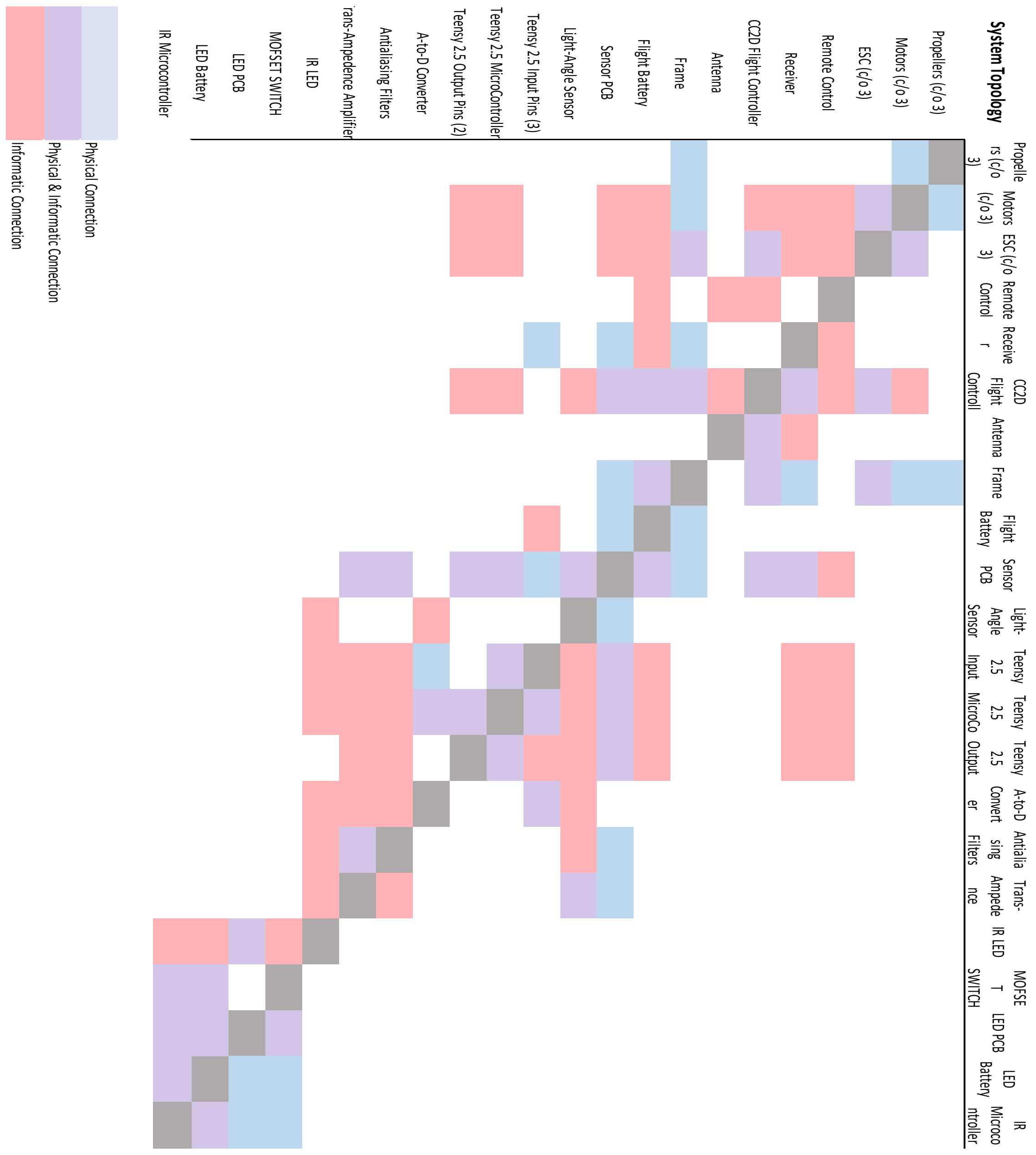


Figure N1. System Topology DSM

Appendix O: Technology-Interface Risk DSM

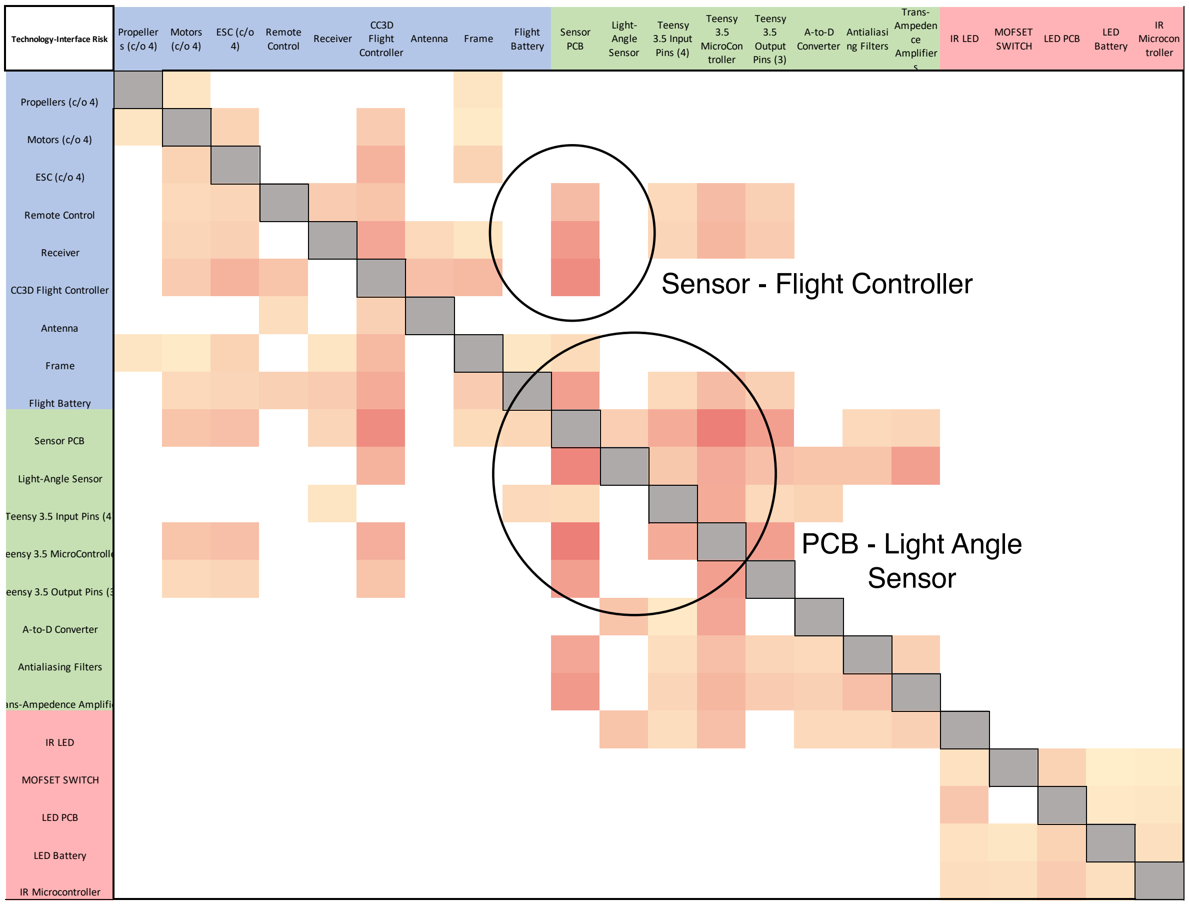


Figure O1. A design structure matrix quantifying the risk-assessment criterion on each technological interface of the system. First the system topology is overlaid, and a rank (1-9) was assigned to each form element. Then, information interfaces were scaled up. This produces a visible area for inspecting system complexity.