Deployable Prototype Documentation
Intelligent Headlights

Alisha Rosario, Juan Chico, Konstantin Komarchuk

Table of Contents

..

List of Tables

List of Figures

Executive Summary

The objective of the Final Prototype documentation is to document the whole year of work that lead up to the fully working Intelligent Headlights system. This documentation acts as a record of all the steps that were gone through to make the Intelligent Headlights possible. The Societal Problem that resulted in the Prototype being designed was Nighttime Driving. Since 3.5 times more fatal accidents happen in the Nighttime and the amount of nighttime accidents have yet to decrease despite the decrease in daytime accidents it was apparent this was a critical issue to be solved. To decrease the amount of accidents at night the Intelligent Headlights give light to the user where needed depending on the Speed and angle the car is at. If the driver is on the freeway then the light will become narrow and if the driver is driving in residential the light will become wider. The lights will move based on the drivers turn angle as well, so as the car turns the headlights will turn. Lastly, the headlights will adjust themselves up or down when the driver is going up or down an incline or over a speed bump to ensure there is always a well-lit path ahead of them. All of these conditions were outlined in our Design Contract for making the product. In addition to that, as stated by Federal Codes, the low beams and high beams, along with the turn signal are user controlled. For added safety Daytime Running Lights are in the Intelligent Headlight to increase visibility in the daytime. They are only on when the low beams, high beams, or Turn Signal are off. It is important to follow the regulations we are aware of so the product can be marketable. The Budget is also another thing we considered for the Intelligent Headlight while making it, so we could be competitive with Automotive Companies prices. The idea was to make the intelligent Headlights affordable for the majority of people and implementable in any car. Our budget range before we started buying items was between $500-$700. At the end of Spring 2017 Semester our budget fell a little about the middle of the price range. The total included parts we did not use because of alternative solutions that worked better in implementing our product. Excluding the items that were not used for our Final product it would have brought the total spending about $65 dollars below $500. The tasks for the Intelligent Headlights were broken into different components and divided amongst the team members based on their skills. The Daytime Running Lights, Turn Signal, and LED's were given to one member. The turning conditions, Angle and speed measurements were given to another member. Lastly, the structure and movement of the LED's were given to the last member of the group. Through working together we were able to mitigate problems with inaccuracy of measurements, functioning of the LED, and spacing problems in the headlight case. Currently, the final prototype is finished and abides by the Design Contracts requirements. In terms of marketability, there are still some tests that should be done on the headlights that we did not have the resources to do. There are also little things that can be improved on to make the product better quality. For the tests, all the components except the sensor underwent temperature test that tested the components below -18 degrees Celsius and above 43 degrees Celsius. The components passed the test and functioned without a problem afterwards. The components were then run continuously for 14 hours straight. They still functioned fine after the test was done on them. The sensor we used was excluded from this and instead was tested for accuracy of data measurement at different intervals under the same condition. The results showed that there was a small error that would not be a cause for concern in our product. The other test done on the sensor was a change in voltage test to see if running different voltages through the sensor while it was collecting data would affect its accuracy. This resulted in a small error that was negligible for our product. In conclusion, throughout the year we have performed tests on our components, solved our problems, and stayed within budget to make Intelligent Headlights possible. Without dividing our tasks appropriately we may have not been able to finish the prototype and help solve the Societal Problem.

Intelligent Headlights

Alisha Rosario, Jun Chico, Konstantin Komarchuk
Department of Electrical and Electronic Engineering
California State University, Sacramento

***Abstract*— Driving conditions are remarkably different in the nighttime. vision is reduced and it can be more difficult to see vulnerable road users such as pedestrians, cyclist, and motorcyclist. As a result of decreased visibility, the distance a driver can see is shorten. In addition, hazards can often seem to appear out of nowhere. To reduce the danger of driving at night due to poor lighting conditions, my team and I have been working on an Intelligent Headlight System. This system adapts to the many driving conditions by providing the driver light where its needed it the most. To provide the driver light where its needed the most, our system supports the following features. In case of either a left or right turn the intelligent headlight system focuses the light to left and right side respectably. While going on an uphill or downhill, the system would then focus the light downward or upward depending on the incline of the vehicle. In addition, at low and high speeds the headlight system widens and narrows the light respectively. Although this type of system is currently being implemented in newer vehicles, our focus is to develop an alternative system with similar safety features at a fraction of the cost.**

Keywords—Intelligent Headlights, Safety, Nighttime Visibility

I. INTRODUCTION

For a successful Deployable Prototype multiple considerations need to be accounted for. To start Intelligent Headlight came about from a need. The need was defined by the societal problem of Nighttime Driving. In the Intelligent Headlights documentation, it is important that we go over the whole year and all the main topics that we had to document and plan in order to have a successful prototype. The documentation is documenting the final product of the Intelligent headlight and the process it took to get have a working deployable prototype.

II. SOCIETAL PROBLEM

In the world of modern technology, some aspects of our lives tend to be more technologically advanced, while others stay neglected. The society has created a fast-paced lifestyle that requires the average user to rely more and more on technology. With time, technology has advanced to assist daily lives, providing comfort as well as provide safety assistance.

In the automotive world, we see multiple advances in the art of passive and active safety, such as airbags, seat belts, and crush zones. As time goes by, some features of the automotive world remain unchanged, or at least not significantly changed.

For Nighttime Drivers there was a significant need for safety improvement. Nighttime Drivers have vision impairment due to Nighttime myopia which affects any driver regardless of age. Nighttime myopia gives the driver less visibility and makes it harder for driver to distinguish objects. This can cause danger for both the pedestrian and drivers. Nighttime myopia also affects how the driver perceives depths. This could affect their judgement of distances or how far objects are from them. The combination of depth and object perception affects other drivers, pedestrians, and the driver own safety, making it more likely an accident will occur.

Drivers at night base most of their decisions on what they can actually see. The percentage of fatal accidents that occur at night is 40%. Though Daytime accidents have decreased over time, Nighttime accidents remain the same. Nighttime Driving fatalities occur 3.5 times more than Daytime Driving fatalities.

Driver still show a level of confidence and low difficulty for driving as much as they do in the daytime. This confidence causes driver not to anticipate or plan ahead to avoid lower contrast/difficulty to see hazards or object. Reduced visibility at night along with other variables can make nighttime driving even more dangerous. Recreational drivers, consisting of more young people than old that drive at night are less attentive to their surroundings. This is assuming they are not distracted by other tasks or are not fatigued at night.

Fatigue decreases response times and is a factor in accidents at night. Fatigue is a factor that is uncontrollable and depends on the amount of sleep the driver has had as well as the time of night they are driving. Inexperienced drivers who drive at night may also experience slower response times regardless of fatigue.

Lastly. Drivers rely on their peripheral vision to interpret things off to the side and to know where to look next. Peripheral vision is less accurate than looking straight at an object and sometimes objects are missed because only objects of a certain size are detected. This is even truer when driving in the darkness. The Nighttime Crash Frequency Reduction Association believes the lighting at night has a connection to visual performance.

To improve the visibility of the road, we created a smart headlight system, which would change the way the road is illuminated according to the way the vehicle moved. This system would provide lighting where it is necessary, so that the driver can see more, while using much less energy than any incandescent light provides.

Proper lighting of the road is a critical part of any driver on the road today. With speeds ranging from 25 to 85 mph, proper illumination of the road ensures that the driver can adequately respond to different hazards and dangers.

With all the dangers on the road, and the speeds of the cars, average drivers needed something that will be significantly cheaper, smarter, and efficient. This would significantly improve safety and visibility of the road ahead. We decided that using high intensity LEDs would be a useful and efficient way to provide bright light that can be easily directed and would cost a fraction of an HID or even an incandescent bulb. By using several LEDs on moving platforms, we could direct the light where it was necessary, and the driver would see much more in the direction they are turning than any other light on the market.

The idea was to make the system be affordable for anyone, and using several affordable components, it was made possible to do so. There are currently several systems that are similar to ours, and are manufactured by the brand that makes the cars, so we are one of the first ones to offer an aftermarket system like that.

One more problem we wanted to address was that not every car manufacturer offers a smart headlight system. So, in order to help as many people as we could, we had to make our system universal and independent of the make and the model of the car. This was achieved by making the entire system standalone and independent of the vehicle, while requiring minimal modifications to allow implementation on multiple makes and models of cars.

The idea behind our project was that we can create a system that is affordable, efficient, universal and important of all, provide sufficient lighting for the driver to see ahead to be able to make proper decisions at the modern speeds on the road.

III. Design Idea Contract

The Design idea contract from Fall 2016 to Spring 2017 has been improved from what it originally started from. The design idea contract was made to address visibility of drivers at night as discussed in the section above. In Fall 2016 our original design idea was to achieve solving the societal problem through means that were beyond our scope. Through a Heads Up Display, information such as speed and traffic conditions would be displayed on the windshield. The originally design idea also required a thermal system to detect the pedestrians and other hazards at night and alert the driver to the dangers. After discussing these ideas with the instructor our Design Idea was modified to be more reasonable with our time constraint and less distracting to the driver.

The Modified Design Idea Contract required that Intelligent Headlights were made. These intelligent headlights were required to provide High beams and Lows beams through user control, which is represented through the use of a pushbutton. The Light Emitting Diodes (LED's), components that provide light to the user and act as the headlights had to be controlled by servos to spread light accordingly. The way the LED's were moved was based on turn angle of the driver's car. These conditions allowed the visibility of the user to be increased through the use of servos and sensors.

For the Spring 2017, Modifications were made to the Design idea contract. To further improve our design, additional conditions for spreading the light as the driver needed were made. The conditions were whether the car was moving up or down an incline or speed bump and what the speed of the car is. The Headlights would adjust downward if the driver is driving up an incline or angles upward when going over a speedbump. The headlights will adjust upward if the driver is going down an incline or angles downward when going over a speedbump. This allows the driver to be able to see in front of them at all times. For the speed, if the driver is driving in residential the headlights will adjust outward, to widen the visibility of the user. If the driver is driving on the freeway, the lights will adjust inward to narrow the visibility of the user. Two of the LED's will always remain stationary to make sure there is never a blind spot when the servos adjust to all conditions from both Fall 2016 and Spring 2017.

Lastly, For Spring 2017, Daytime Running Lights (DRL's), lights that are always on in the daytime to increase driver visibility, were added. A turn signal was also added to the contract that is user controlled and represented by a pushbutton. The LED's were reduced from 6 LED's to 5 LED' which stays within the agreed upon LED requirement, 5-7 LED's.

IV. Budget

The team spent a little over $630 total on all items included and not included in the project. All the money came out of the pocket of the team members. For different reasons, some of the items included in Table 1.1 below were purchased but not used for the project. Some items had to be replaced due to faults, while others were used to redesign certain components in the project. The items that were not used are the last 8 items on the list of the spending report shown below. The price for all the materials actually used in making the product comes out to $431.14. Estimated spending was at $500-$700 when the project was planned, meaning we stayed within our desired spending forecasts if we include all the items regardless if they were used..

TABLE I.
Project Spending [1]

Project Spending			
Item	*QTY*	*Price*	*Total*
OBDII Reader	1	$ 9.79	$ 9.79
Black PLA	1	$ 20.00	$ 20.00
Sunsbell LED Strip Light	1	$ 10.99	$ 10.99
Raspberry Pi 3	1	$ 39.99	$ 39.99
Raspberry Pi US power	1	$ 7.99	$ 7.99

supply			
SparkFun LSM6DS3 IMU	1	$ 19.95	$ 19.95
10 pcs Aluminum Heatsink	1	$ 5.98	$ 5.98
SanDisk 32GB microSDHC	1	$ 11.70	$ 11.70
DROK LN298N	1	$ 6.99	$ 6.99
IMU BNO055	1	$ 34.95	$ 34.95
LEDs	5	$ 6.99	$ 34.95
DRL	1	$ 29.95	$ 29.95
Push Button	2	$ 4.00	$ 8.00
Arduino Uno	1	$ 15.00	$ 15.00
Voltage regulator	1	$ 3.00	$ 3.00
LED Driver	1	$ 3.39	$ 3.39
Servo Shield	1	$ 15.08	$ 15.08
Servo shield header kit	1	$ 5.95	$ 5.95
Servos	12	$ 3.29	$ 39.48
Fiberglass	1	$ 18.04	$ 18.04
Headlights (pair)	1	$ 75.00	$ 75.00
Light lens (Plano-convex)	1	$ 5.00	$ 5.00
Tubing	1	$ 9.97	$ 9.97
LED Driver	5	$ 3.39	$ 16.95
LEDs	11	$ 6.99	$ 76.89
Relay Shield	1	$ 27.95	$ 27.95
Arduino Mega	1	$ 45.95	$ 45.95
RGB Strip	1	$ 11.99	$ 11.99
MOSFETs	4	$ 3.00	$ 12.00
5 pcs 4 wire micro stepper motor	1	$ 8.99	$ 8.99
Voltage Regulator	1	$ 3.00	$ 3.00
			Total: $634.48

V. Project Milestones

The project was completed in several phases. At the end of Fall 2016 semester we were to provide a laboratory prototype of the project, and in Spring 2017, a deployable prototype. The components of the headlight were to be completed at different time within the nine-month period of the project.

First, a headlight was to be purchased and disassembled. At the same time, some of the team began working on programming the Raspberry PI while the other part of the team was writing the code for operating the LEDs. By the middle of November 2016, the basic LED unit was created, as seen in Figure 1. Servos and LEDs were attached to the LED unit, and the entire LED unit assembly was put together to be placed into the headlight housing for the presentation.

During the assembly, Raspberry PI and the IMU sensor were connected and programmed to collect data from a moving vehicle. Once data was collected, it was then analyzed to determine proper thresholds for operating the servos. Towards the end of Fall 2016 semester, the LED assembly, the Raspberry PI and the servos were programmed and operating in time for the project demonstration in December 2016.

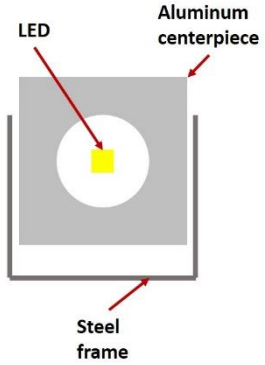


Figure 1: LED Unit [2]

Beginning with the new semester, the team had to slightly modify the project to include an OBDII reader, as well as replace some faulty parts. Some of the team members began working on the DRLs and the turn signal, which was done using one LED strip. At the same time, a 3D model was being created to print out the filler panel to give the light some shape.

By the end of March 2017, the 3D printing idea was abandoned, and a fiberglass filler panel was created and painted. By the middle of the Spring 2017 semester, the headlight had all the components working, although not incorporated together. By mid-April, the entire assembly of the headlight was put together and working, with only a few minor programming things to solve and fine tune.

By mid-May, the project was to be completely working. For the presentation, we collected data that is to be run through the Raspberry PI and sent to the headlight to respond. Another important milestone was to create a simple two minute video to showcase our project to the visitors of the presentation of the system.

VI. Work Breakdown Structure and Task Assignment

The work Breakdown structure can be seen at the end of the document. [39] It had to be separate because it was a pdf file and including it in the report would have made it hard to read.

VII. Risk Assessment and Mitigation

Through the two semesters our team had to deal with a few risks when creating an intelligent headlight system. As a result, we had to mitigate the risk by replacing the part or using something else completely.

During the first semester, we had to mitigate the functioning of the LED's, modify our LED unit design, and account for IMU inaccuracy. Two factors that needed to be considered when our team discussed about the functionality of the LEDs lights include Voltage regulation and the heat of the overall system. Two of the major risk associated with the LEDs were that the LEDs will either not light at all or will burn out while testing the system, if not handled properly. To mitigate this problem, we had to make sure that we use resistors for each LED used that will limit the amount of current each LED receives so that they don't overheat and burned out.

One of the greatest risks we had to take during the first semester was the use of small stepper motors. The motors were about 8 mm wide by about 10 mm long. Due to size constraints of our system, we couldn't go any larger than that. The Figure below shows what our original system looked like with stepper motors.

We did not know if the motors would hold, and to avoid the risk of trying and failing, we decided to redesign the assembly to accommodate the issue of torque and structural strength. We had to redesign the motor and centerpiece configuration, which of course took more time. We had to redesign our Tilt and Pan system to include two servers per LED and reduce the number of LEDS we would use from 7 to 5.

Another Risk our team went through in the first semester of Senior Project was that the Inertial Measurement Unit (IMU) being used to gather information about the vehicle was not accurate because of the large amount of noise interference we were getting from the IMU. The noise level made it very difficult to accurately determine whether the vehicle was in motion or not, even when the vehicle was not moving at all. We had considered many different alternatives from using two IMUs and taking the average number of the two, to finding an algorithm that would reduce the noise level to a minimum. The problem our team had from using either one of them was that it would have taken weeks or months trying to reduce the noise level coming from either solution. Thankfully, another group in Senior Project was dealing with the same problem and was able to find a solution by buying a more expensive IMU that had its own filtering system. The filtering system reduced the amount of noise level to almost zero. As a result, we mitigated our own problem by using the same IMU the other Senior Project group used to solve theirs.

In between the first and second semester we had a risk that our 3D design was not going to properly fit our headlight system because of the uneven shapes inside of the system. Our original mitigation to this problem was that we were going to have to file each of the 3D printed parts in order to fit in our design. As soon as the first 3D parts finished printing we knew that it would be almost impossible to file each of those parts in the different shapes that the system had. After given it some thought, Konstantin brought up the idea of molding the inside of the headlight using fiber glass. Using fiber glass would allow us to mold the inside of the headlight even with its different uneven shapes. The procedure took about a week to complete and allowed us to continue the process of attaching the remaining components in the system.

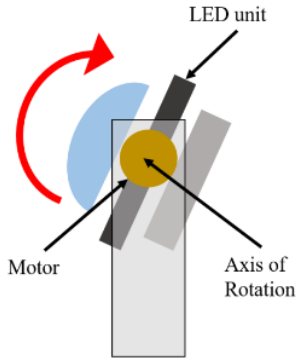


Figure 2: Tilt and Pan System [3]

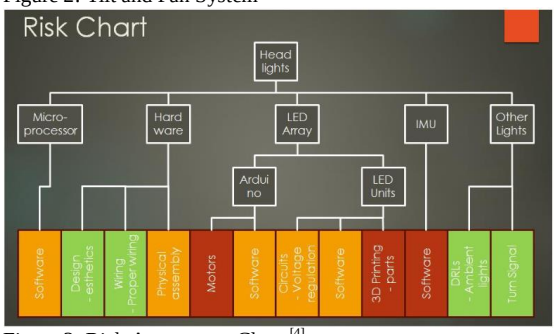


Figure 3: Risk Assessment Chart [4]

The figure above shows the risk assessment from the above mention risks. The green headings represent a low impact in our system, a yellow heading represents a medium impact in our system and a red heading represents a high impact in our system.

Towards the end of the semester our team also had a risk in which the speed of the vehicle at any given time was not being produce by the IMU. Even though we were using the IMU that had its own filtering to eliminate the noise, it was not enough to reduce the errors we were getting when trying to calculate the speed of the vehicle. In order to calculate the speed of the vehicle we were taking the acceleration and integrating it. Every time we integrated the acceleration we were also adding a small amount of error into the equation that allows us to calculate the speed of the vehicle. As a result, all of those small errors accumulated over time and made the speed of the vehicle inaccurate. To mitigate this problem in a short period of time, we decided to use an additional component to the system. Using an OBDII reader allowed us to get the speed of the vehicle at any given time because the On-Board Diagnostics (OBDII) reader connected to the on board diagnostics system of the vehicle that gave the speed from the vehicles own sensors.

VIII. Design documentation

The design philosophy of our intelligent headlight system is about offering the driver the same safety features that are being implemented in today's newer vehicle at a fraction of the price. In addition, our design would be able to support a wide variety of vehicles, not just vehicles produced by one manufacture. It all starts with the latest LED technologies that are brighter than incandescent light bulbs in today's vehicles. Using five LEDs per headlight allows the system to focus the light were the driver needs it the most. In order to focus the light where it is needed the most, each headlight uses a set of 3 pan/tilt servo brackets. These brackets are placed towards the left side in the left headlight, and towards the right side of the right headlight. The placement allows the light to in either up or down/left or right direction. The two remaining LEDs are stationary on the right side of the left headlight and on the left side of the right headlight. This prevents a blind spot when a turn is made.

In order to control the intensity of the LEDS, as well as, the turn signal of the system the system utilizes an Arduino that will control the high beams, low beams, and turn signal based on user input. For the user input 3 different buttons are used. One button turns the headlights on and activates the low beam, while another button activates the high beams when pressed. The third button activates the turn signal when pressed. The turn signal was achieved using an LED RGB (Red, Greed, Blue) Strip, which is a strip that changes color according to which colors are being sent to it through programming. The turn signal is an amber color that is a chasing light. A chasing light is achieved by turning each LED on within the strip one at a time consecutively at a fast speed. An additional Arduino is used to control the servos to move the LED's where they need to go. The directions they can move are left, right, up, and down. These are based on the signals sent to the Arduino by the Raspberry Pi. The intelligent headlight uses a Raspberry Pi to communicate with the second Arduino. The Raspberry Pi is the brain of the system. It is constantly analyzing data from the attached sensors and senses when the vehicle is turning, speeding, and going up or down a hill. Based on any of those scenarios the Raspberry Pi will then send a different signal to the second Arduino and based on that signal, the second Arduino would then proceed to move the brackets into the desired position. This is in order to focus the light were the driver needs it that most.

Finally, the system uses two different sensors that consistently send out data to the Raspberry Pi about the current position of the vehicle. The sensors include an IMU which allows the Raspberry Pi to sense when the vehicle is turning in either direction and also when the vehicle is going up or down a hill. In addition, the system also uses an OBDII reader that constantly reports the speed of the vehicle to the Raspberry Pi. The speed of the vehicle obtained through the OBD II Reader allows us to narrow the light if the user is driving on the freeway and widen the light if the driver is in a residential area or at a slower speed. These additional conditions allow for users to be more aware of pedestrians in

Figure 4: Intelligent Headlight [5]

the case they are in residential and more aware of what's ahead of them if they are on the freeway. Using all of these components working independent, our system is able to focus the light where the driver needs it the most. The figure at the top of the page shows what makes the intelligent headlight system, intelligent.

IX. Deployable prototype status

At end of the Spring 2017 semester, the deployable prototype was to be completed by our team. The project is completely done in terms of the goals we set in the beginning of the project. It has all the required components of a headlight, the LEDs, the turn signal, and the DRL. The light is directed using a lens, and only three user inputs are used.

With all the regulations of the automotive market, it is difficult to judge where to place the readiness of the project to be installed on a vehicle. Headlights are a critical component, so much more testing needs to be done before calling it ready for sale to the general public. Production of such a system will likely require use of different servos and different microprocessors.

The project works like it is supposed to, but there are a few minor details that need to be taken care of before installing the light unto the car. Some extra wiring needs to be created for powering all the electrical components in non-ideal environment of driving. The project works on ideal power conditions, without accounting for significant voltage changes.

The project is complete in terms of the goals we set: we are using angular velocity to turn LED units inside the headlight to direct light. We're also using an OBDII reader to determine the speed of the vehicle and make decisions based on the data collected – one major component of the project that we needed to incorporate. The other major components include the turn signal, which operates just fine, even with some pleasant visual effects. All servos work precisely and point the light where asked by the Raspberry PI. Data is collected seamlessly; communication works and all electronic components work like a clock.

The project has not been installed on the vehicle, so the project did not have real life testing. All the simulations have been run, but in general the project needs significantly more testing than we would offer in the timeframe we had. The project should adhere to multiple federal (USDOT) and state regulations that determine what headlight should comply and criteria that it should meet. With the multiple DOT regulations, there should be a significant amount of research done to make sure we meet all of them, which would likely require possible help from lawyers or consultants. At the time of completion, there is no guarantee that it will be completely according to regulations. We did our research, and as much as we could find, we attempted to adhere to all regulations. We adhere to the Federal codes on the Headlights to the best of our knowledge. The high beams, low beams, and turn signals are controlled by user input. The DRL's are only on when the low beams or high beams are not on, and the headlights are a consistent brightness for all the LED's when they are in high beam as well as low beam. That was just to name a few of the Federal codes we followed to the best of our knowledge.

In summary, the project is complete, but it is not ready to be installed on any vehicle. There are multiple tests that need to be conducted on the light, such as vibrations, dust, water and all possible weather conditions. The programs may also have bugs, which were could not be determined using laboratory conditions. All the requirements of the project proposition were met, so the project is complete, but there are many more steps to take towards actual deployment of the system unto the market.

X. Deployable Prototype Marketability Forecast

There is still a much-needed effort to refine the deployable prototype to achieve a marketable device based on our research in the market review. However, because this prototype was done by college students most of the refine work is due to the lack of tools and money needed to make this a better marketable product.

With that in mind foremost our team would have to research whether or not The Insurance Institute for Highway Safety (IIHS) finished developing a system for evaluating headlight performance which was due early 2017. This system will require vehicles to receive a 'good' rating to be considered a Top Safety Pick Plus, the IIHS's top safety rating. In addition, the ratings will also look at a new type of lights such as LEDs and HIDs, along with other technologies like auto-dimming high beams. This is to insurer that we are meeting the safety standards of our LED system. Within the intelligent headlight system there are couple of changes in hardware that would also need to be change in order to achieve a marketable device. This includes a redesign of the servo bracket that is durable and is able to withstand higher temperatures. Using both the Arduino and Raspberry Pi is a cheap alternating option for our current processing needs, but I would suggest replacing it with something that has more processing power. Whether or not both Arduino and Raspberry Pi is used, a casing for the processor of the system would have to be build that would protect the processor from external forces. The IMU used by the system would also have to be placed in a shield case to protect it against the same external forces that can damage the processor of the system.

Instead of Using an OBDII reader, using another sensor to detect the speed of the vehicle would be a better option, for example, a GPS sensor. Having the OBDII reader plugged-in the onboard diagnostic port means that, any one fixing the vehicle would have to unplug it in order to use their own tools to find out what is wrong with the vehicle. In order for our system to achieve a marketable device, changes to the software would also have to be made. One of the major code fine tuning that would have to be made in both Arduino and Raspberry Pi is the action of which the system should take if one of those two stopped working. If that is the case we would want to make sure that the servos are set to their default position which is to look onward until repairs to the system are made. This is to ensure that the lights are not facing different directions when either of the Arduino or Raspberry Pi malfunctions.

The communication between the Raspberry Pi and the OBDII reader would need tuning in case there is an interruption between them. Finally, and most importantly security changes to the Raspberry Pi would have to be made. This is to ensure that no one is able to connect to the Raspberry Pi via Bluetooth or through its Wireless network.

XI. Conclusion

The Intelligent Headlights came about due to the need for Nighttime Driving accidents to decrease. Without this need, our product wouldn't have been something we could have pursued for our project. When designing a product there are a multitude of things to be aware of. Budget, Risks, Laws, and marketability are all important things that were considered through our making of the intelligent headlights. We were able to split up the tasks needed to make intelligent headlights possible based on our individual skills. Teamwork is essential to making a product that works correctly.

As a team we were able to mitigate the problems we encountered while staying in our budget range, which allowed us to reach our milestones. The end result of the year was having a functional prototype that would help increase the driver's visibility through the controlling of light based on driving conditions.

Acknowledgment

Juan, Konstantin, and Alisha thank the Sacramento State Mechanical shop that helped build the LED units. Without their help a vital component used to bring the project idea together would not have been made.

Juan, Konstantin, and Alisha thanks Amy and Team 2 for sharing information on the current IMU we are using in our project. Without her recommendation the project would have been time consuming. Through her recommendation, we were

able to acquire an IMU that does not produce noise even when stationary.

References

[1] Komarchuck, K. “LED Unit.” Senior Design, Fall 2017.
[2] Komarchuck, K. “Project Spending” Senior Design, Spring 2017.
[3] Komarchuck, K. “Title and Pan System.” Senior Design, Fall 2017.
[4] Komarchuck, K. “Risk Chart.” Senior Design, Fall 2016.

[5] Komarchuck, K. Rosario,A. Chico, J. “Intelligent Headlight.” Senior Design, Spring 2017.
[6] Komarchuck, K. “locations of Screws and Bolts on Bumbper.” Senior Design, Spring 2016.
[7] Komarchuck, K. “Approximate location of a box on upper ti bare.” Senior Design, Spring 2016.
[8] Komarchuck, K. “Connect USB into Appropriate Spot.” Senior Design, Spring 2016.
[9] Komarchuck, K. “Bolt Locations.” Senior Design, Spring 2016.
[10] Komarchuck, K. “Bumper Fasteners.” Senior Design, Spring 2016.
[11] Komarchuck, K. “OBD II Slot Location” Senior Design, Spring 2016.
[12] Komarchuck, K. “Hardware Block Diagram” Senior Design, Spring 2016.
[13] Rosario, A. “LED Driver Diagram” Senior Design, Spring 2016.
[14]. P. Burgess. "Wiring: LEDs/LED Strips." learn.adafruit.com. November 2015. [Online]. Available: https://learn.adafruit.com/digital-led-strip/wiring. [Accessed: March 12, 2017].
[15] "Cree Xlamp XM-L2 High Power LEDs" ledsupply.com. [Online]. Available: http://www.ledsupply.com/leds/cree-xlamp-xm-l2-leds. [Accessed: March 12, 2017]

[16]. "T-Pro MG90S 9G Metal Gear Servo 1.8kg / 13.4g / 0.10sec." hobbyparts.com. [Online]. Available: http://www.hobbypartz.com/servo-mg90s.html. [Accessed: March 12, 2017].
[17] K. Townsend. "Overview: Sensors" learn.adafruit.com. November 2015. [Online]. Available: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview. [Accessed: March 12, 2017]
[18] Chico,J. “Raspberry Pi and Arduino Communication.” Senior Design, Spring 2016
[19] Chico,J. “Arduino and Servo/Turn Signal/High Beam Control” Senior Design, Spring 2016
[20] Komarchuck, K. “LED unit U shaped steel frame” Senior Design, Spring 2016.
[21] Komarchuck, K. “Centerpiece that holds the cover with lens and the LED” Senior Design, Spring 2016.
[22] Komarchuck, K. “Cover with lens, lens goes into hole in front. Back side is hollow” Senior Design, Spring 2016.
[23] Komarchuck, K. “Plate for LED array. Aluminum, holds all LED units together” Senior Design, Spring 2016.
[24] Komarchuck, K. “Plate for LED array. Aluminum, holds all LED units together” Senior Design, Spring 2016.
[25] Komarchuck, K. “Filler panel with LED unit assembly partially installed” Senior Design, Spring 2016.
[26] J.D. Bullough, N.P. Skinner, T.T. Plummer. “Adaptive Driving Beam Headlights: Visibility, Glare and Measurement Considerations” A Transportation Lighting Alliance Report. TLA 2016-01. June 2016. [Online]. Available: http://www.lrc.rpi.edu/programs/transportation/TLA/pdf/TLA-2016-01.pdf. [Accessed: January 27, 2017].
[27] B.L. Hills. “Vision, Visibility, and perception in driving” Transport and Road Research Laboratory.Vol.9 pp.183-216. November 1979. [Online]. Available: http://journals.sagepub.com/doi/abs/10.1068/p090183?id=p090183. [Accessed: January 27, 2017].
[28] D.A.Owens and M.Sivak. “The Role of Reduced Visibility in Nighttime Road Fatalities” University of Michigan Transportation Research Institute. Report No.UMTRI-93-33. November 1993. [Online]. Available: https://deepblue.lib.umich.edu/handle/2027.42/49541 [Accessed: January 27, 2017].
[29] K.S.Opiela, C.K. Anderson, and G. Schertz. “Driving after Dark” Federal Highway Administration. Vol.66. No.4. January/February 2003. [Online]. Available: https://www.fhwa.dot.gov/publications/publicroads/03jan/05.cfm. [Accessed: January 27, 2017].
[30] M.Green. “Seeing Pedestrians at Night” Marc Green PhD Human Factors. 2013. [Online]. Available: http://www.visualexpert.com/Resources/pedestrian.html. [Accessed: January 27, 2017].
[31] “Night Driving” Law Offices of Michael Pines, APC. 1998-2015. [Online]. Available: https://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/nighttime-driving/. [Accessed: January 27, 2017].
[32] J.Worland. “How Daylight Saving Time Can Be Dangerous” TIME. October 2014. [Online]. Available: http://time.com/3549442/daylight-saving-time-traffic-deaths/. [Accessed: January 27, 2017].
[33] N.Greenfieldboyce “Most Nighttime Crashes With Teen Drivers Happen Before Midnight” KQED Public Media. August 2016. [Online]. Available: http://www.npr.org/sections/health-shots/2016/08/03/488521545/most-nighttime-crashes-with-teen-drivers-happen-before-midnight. [Accessed: January 27, 2017].
[34] “The Most Dangerous Time to Drive: As we ‘Fall Back’ to Shorter Days, Take Extra Care on the Road” National Safety Council. 2017. [Online]. Available: http://www.nsc.org/learn/safety-knowledge/Pages/news-and-resources-driving-at-night.aspx. [Accessed: January 27, 2017].
[35] “Tutorial – Arduino and the TLC5940 PWM LED Driver IC ” TronixStuff, 2013. [Online]. Available: http://tronixstuff.com/2013/10/21/tutorial-arduino-tlc5940-led-driver-ic/ [Accessed: 25-Oct-2016]

[36] M, Day. “LED Driver – Paralleled Outputs: Provide High-Current Outputs” Texas Instruments, 2006. [Online]. Available: http://www.ti.com/lit/an/slva253/slva253.pdf [Accessed: 3-Nov-2016]
[37] Texas Instruments. “TLC5940 16-Channel LED Driver With DOT Correction and Grayscale PWM Control” Texas Instruments, 2014-2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/tlc5940.pdf [Accessed: 25-Oct-2016]
[38] Bosch Sensortec. “BNO055: Intelligent-axis absolute orientation Sensor” [online]. Avalialable: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf [Accessed: Spring 2016]
[39]Chico, J. “Gannt Chart”. Senior Design, Spring 2016.

GLOSSARY

LED- An acronym for Light Emitting Diode. LED’s give off light when they have voltage going through them.
DRL- An acronym for Daylight Running Lights. DRL’s are lights in the headlights of cars that are on in the daytime to give the drivers some extra visibility.
IMU- An acronym for Inertial Measurement Unit. The IMU take measurements of different circumstances. For the purpose of the headlights it will be measuring the acceleration and angles of the car.
RGB LED Strip: A light that changes colors using the 3 primary colors. It can turn red, blue, or green. By mixing the colors in a certain combination it can turn various colors. RGB stands for Red Green Blue while LED stands for Light Emitting Diode. There are multiple LED’s on one strip that can be individually controlled to emit a certain color through programming.
Servo: A device that turns to a specified angle based on programming.
OBDII Reader: An On-board diagnostics tool that allows access to the car data, including speed.

APPENDIX A: USER MANUAL

Read installation procedures carefully before installing any components. Make sure package includes all items listed below. Use proper tools for installation. Refer to a professional if you have no prior experience in disassembly and assembly of a vehicle. Use caution when handling electronic parts, to avoid causing any cosmetic or physical damage.

Tools required:

- 10 mm socket and ratchet
- Phillips #2 screwdriver
- Zip-ties
- Flat-head screwdriver or clip removal tool
- Drill with Phillips #2 screwdriver bit (Recommended)

Items included in package:

- 2 x Headlights (one left, one right)
- 2 x USB wires
- 1 x USB Bluetooth adaptor
- 1 x OBDII Bluetooth reader
- 1 x Box with processing computer
- 1 x Power cable for computer

Step 1.

Remove front bumper by using 10 mm socket to remove bolts and screws. Store on a stand or safe place to avoid scratching the paint. Refer to image below for bolt and screw locations.

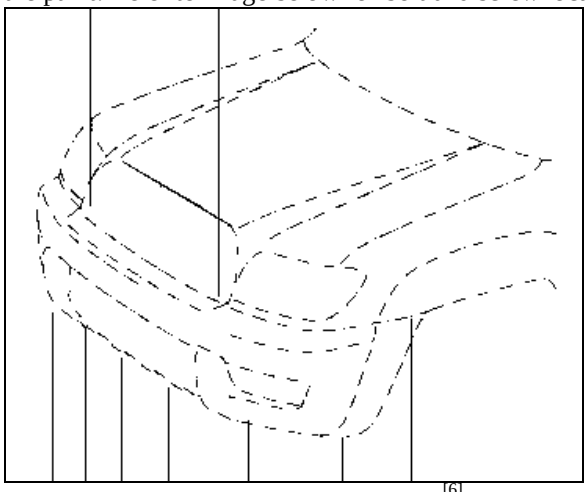


Figure 5. Locations of screw and bolts on bumper. [6]

Step 2.

Using a 10 mm socket, remove the three screws and one bolt on the side that hold the headlight into place. Refer to Figure 6 for fastener locations. Repeat procedure for other side. To disconnect wiring, press on the tab that is connected to the bulb and gently pull.

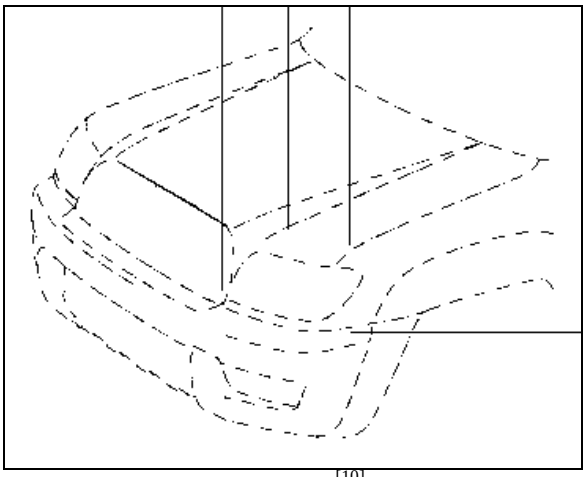


Figure 6. Headlight fastener locations. [10]

Step 3.

Using the screws provided, attach the box with the computer to the upper tie bar, as shown above. **Warning**: Make sure you stay away from any moving parts such as fans, belts, etc.

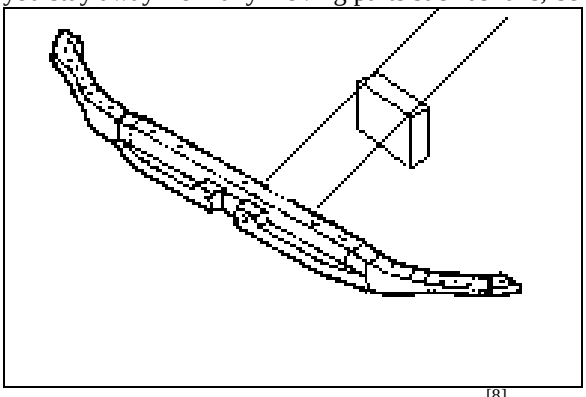


Figure 7. Approximate location of box on upper tie bar. [8]

Step 4. Connect both USB cables into the box previously installed; pull one to the left headlight wiring harness, the other one to the right. Using some zip-ties, secure wire to radiator support wiring harness. Connect the power cable and using the vehicle's wiring harness, connect to the battery leads. At this step, connect the USB Bluetooth adaptor in an available slot. **Warning:** Make sure none of the wires is touching any moving parts.

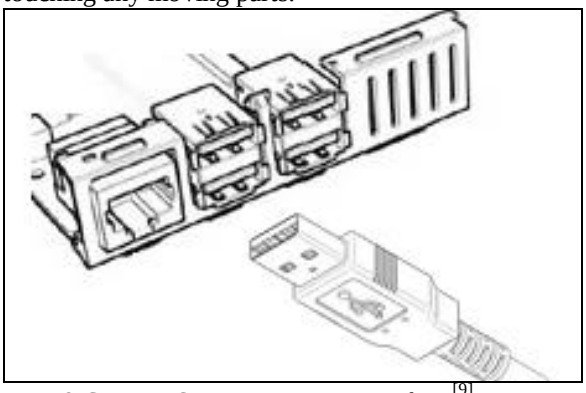


Figure 8. Connect USB wires into appropriate slots. [9]

Step 5.

Install the provided headlight using the bolts from the original headlight. Make sure to reconnect all vehicle wiring as well as

the extra cable coming from the computer. Repeat procedure for both sides. See image below for bolt and screw reference.

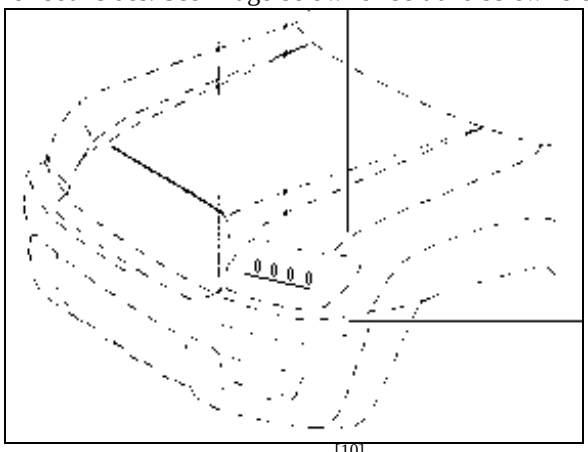


Figure 9. Bolt locations on headlights. [10]

Step 6.

Reinstall the original bumper, and installs the proper screws and bolts as they were removed. Use care when installing bumper to avoid damaging the paint or the newly installed headlights.

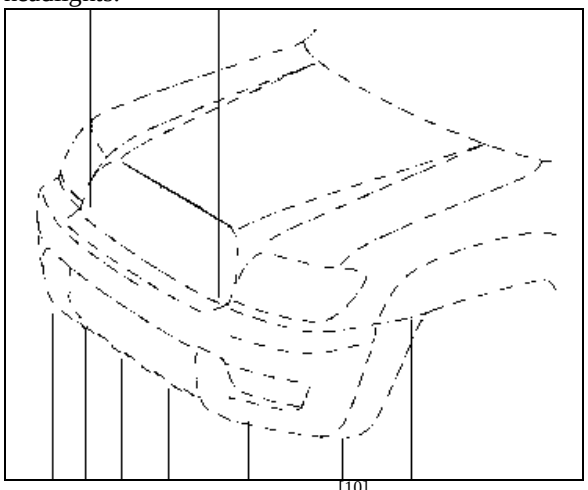


Figure 10. Bumper fasteners locations. [10]

Step 7.

Install the provided OBDII reader into the slot located under the front lower dash panel, near the brake pedals. See image above for reference. See Figure 10 for location of the OBDII slot.

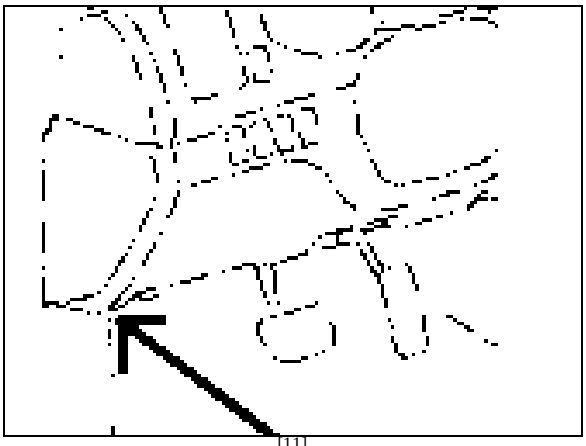


Figure 11. OBDII slot location. [11]

Step 8.

Prior to starting the car, make sure all connections are properly connected. If everything is connected, start the car, wait a few minutes for initialization and test basic work functions by turning on the headlights, high beams and turn signal. Proceed to driving on the road if all systems are working as expected.

References

[1]Outline of Toyota 4Runner front end. 2014.

[2]Parts.com, 2004 Toyota 4Runner SR5 V6 4.0 Radiator Support Diagram. .

[3]ItalTronic, Raspberry PI cover components. .

APPENDIX B: HARDWARE

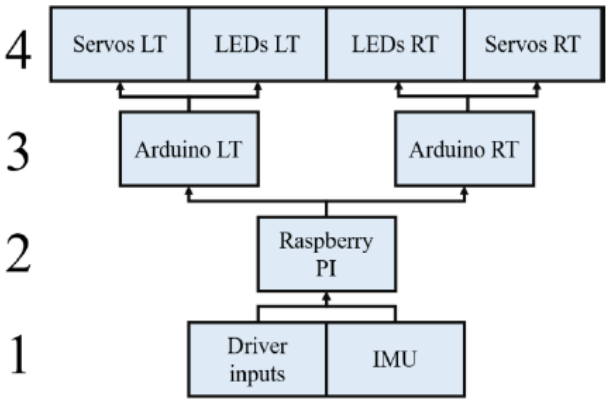


Figure 11: Hardware Block Diagram[12]

The figure above shows the block diagram for the intelligent headlights in relation to what goes into the actual headlight. Row 4 shows that the Left (LT) servos and LED go together since the servos control where the light of the LED goes. It's a similar concept for the right since the right headlight is a mirror image of the left side. Below Row 4 is Row 3 where the Arduino is used to control the servos that control the movement for the LED. It should be noted that there is only a single Arduino that control the Servos. In Row 2 the Raspberry Pi is in charge of processing the data for the sensors and based on the drivers condition will send data to the Arduino on what action to take for moving the servos. Lastly, in Row 1 The IMU is the sensor that gives the raspberry pi the information it needs on the drivers speed and angle. The drive input in Row 1 is meant to represent the driver needing to control the turn signal and high beams/low beams. This signal is fed into an Arduino through the means of a pushbutton. It should be noted that There are two Arduinos that are used for separate processes. The one Arduino that the raspberry pi communicates with based on the drivers conditions, and the Arduino that handles the user inputs for high beam/low beams and turn signal, as required by Federal code.

The next Figure that is being discussed is the LED Driver. The LED driver is a component that allows multiple LED's to be controlled at one. It changes the brightness of the LED's and regulates the current going through it. Below is the Diagram that shows the where to connect each pin on the LED Driver. It's the only component level diagram that was used in the final product.

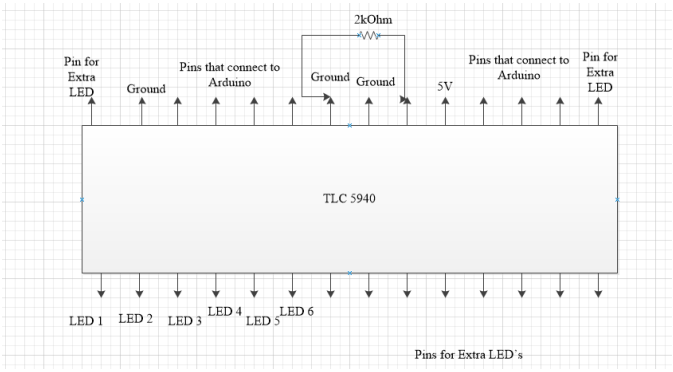


Figure 12: LED Driver Schematic[13]

Some data sheets that are used will be listed below:

Full Data sheet for the TLC 5940 (LED Driver) can be in the references as [37]

Full data sheet for High Output current on the LED Driver (TLC 5940) can be seen in the references as [37]
Full data Sheet for the IMU (BN0055) can seen in the references as [39]

The sheet for High output current was a possible way to increase the current of the LED's to make them brighter, but an alternative way is by decreasing the resistor that can be seen in the draw LED driver schematic above.
The datasheet for the IMU is listed as well since it was used to gather data on the turn angle for left, right, up, and down motions..

For the testing of the LED's and RGB Strip, were tested at a low temperature of -13.5 degrees Celsius and a high temperature of 62.78 degrees Celsius. They worked correctly at each of these conditions and were able to function as expected without a problem. For the endurance test the LED's and RGB Strip were left running for 14 hours. Each of the items worked as expected without a problem after the endurance test was over. The endurance test was to ensure that the component would work over long periods of time without problems.

The servos were tested at a low temperature of -18 degrees Celsius and a high temperature of 43 degrees Celsius. The servos were tested to see if they would be able to perform a sweep through various increments and they did so without a problem. The servos were also tested to see the maximum and minimum sweep as well as the angle necessary to move the LED units.

Lastly, the IMU was tested for consistence in collecting data. At different intervals the same testing condition was repeated to see if the data collected at the separate intervals was the same. The difference between the data collected was very minimal. The IMU was also testing under different voltages ranging from 1.6V to 3.6V which is the range of operation as indicated by the data sheet. The instrumentation was 0.2V until it reached 3.6V. Collecting data through this testing period showed only a small fraction of an error in the difference between the data points gathered. The errors weren't big enough to be a cause for concern.

Below are pictures of and RGB Strip, An LED, a servo, and the IMU.

Figure 13: RGB Strip[14]

Figure 14: LED [15]

Figure 15: Servo[16]

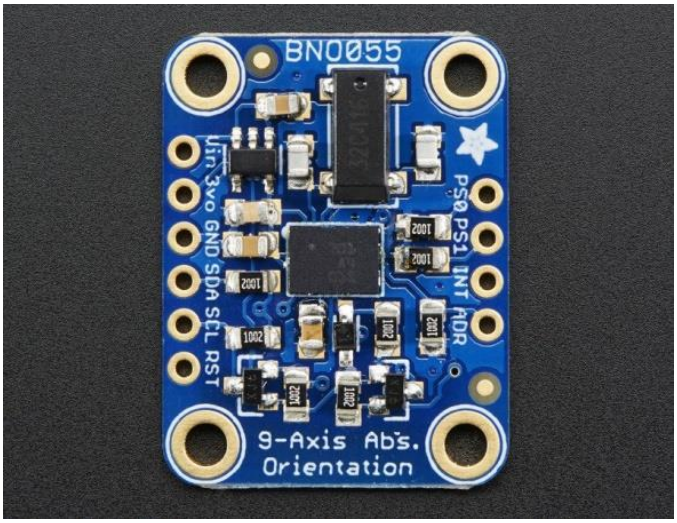


Figure 16: IMU[17]

APPENDIX C: SOFTWARE

The following Figure Shows a Block Diagram of the communication between the Raspberry Pi and Arduino.

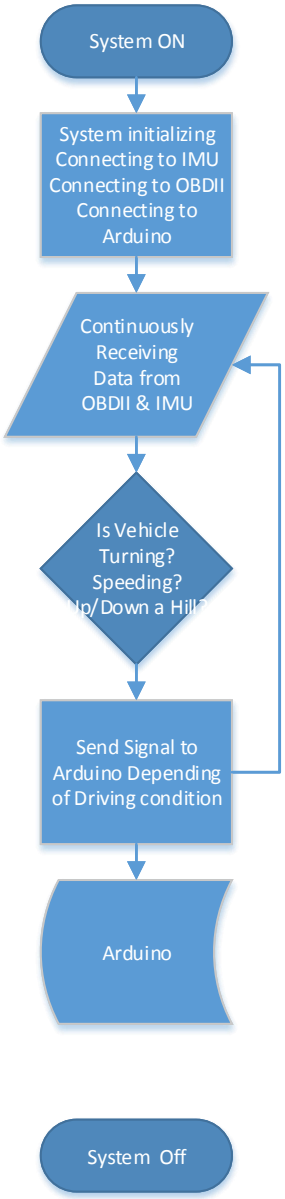


Figure 17: Raspberry Pi and Arduino Communication [18]

The following figure shows the block diagram Between Arduino and Servos/LED/Turn Signal.

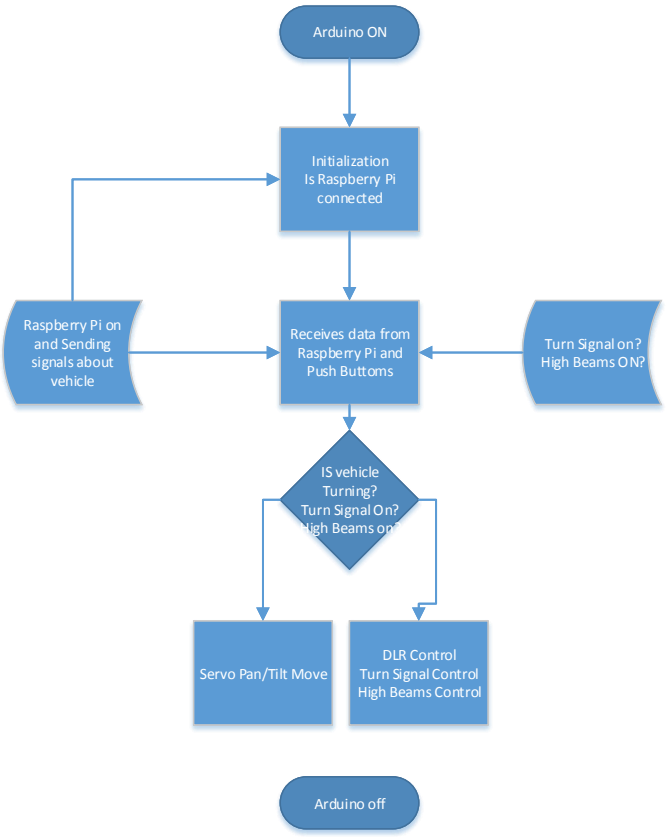


Figure 18: Arduino and Servo/Turn Signal/High Beam Control [19]

The following Python Code is a demonstration test in order to test the headlight system by feeding the program a set of data we gather using the IMU on actual run.

```
# Smart Headlight System
# Main System program
import logging
import sys
import time
import datetime
#from Adafruit_BNO055 import BNO055
import serial
#ser = serial.Serial('/dev/ttyACM0', 9600)
# Initialize Array
simArray = [] // array of numbers that allow us to see if vehicle is turning left or right
timeArray =[] // time stamp array
# Enable verbose debug logging if -v is passed as a parameter.
if len(sys.argv) == 2 and sys.argv[1].lower() == '-v':
    logging.basicConfig(level=logging.DEBUG)
z = 0
t = timeArray[0]
#y = 0
for i in range(len(simArray)):


    # Returns to current while loop if no data provided
```

```
    if z < simArray[i]:
        z = simArray[i]

    t = timeArray[i]
    # Configuration settings for each condition.
    if z <= .069 and z >= -.069:
        #ser.write('0') # send default setting to the arduino
            print(t)
            print('Vehicle is in default position')
    elif z >= .07 and z <= .19:
        #ser.write('1') # Slight Left Turn
            print(t)
            print('Vehicle is making a Slight Left turn')
    elif z <= -.07 and z >= -.19:
        #ser.write('2') # Slight Right Turn
            print(t)
            print('Vehicle is making a slight Right turn')
    elif z >= .2 and z <= .34:
        #ser.write('3') # Moderate Left Turn
            print(t)
            print('Vehicle is making a moderate Left turn')
    elif z <= -.2 and z >= -.34:
        #ser.write('4') # Moderate Right Turn
            print(t)
            print('Vehicle is making a moderate Right turn')
    elif z >= .35:
        #ser.write('5') # Sharp Left Turn
            print(t)
            print('Vehicle is making a sharp Left turn')
    elif z <= -.35:
        #ser.write('6') # Sharp Right Turn
            print(t)
            print('Vehicle is making a sharp Right turn')
    else:
        #ser.write('0') # bring vehicle to default position
            print(t)
            print('Vehicle is in default position')

    #y = y + 1
    # Sleep for a 1/10 before reading and sending info back to
the arduino.
    time.sleep(0.333)
```

The following code is used by the Arduino which allows us to control the movement of the servos based on the given signal provided by the Raspberry pi.

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

//Call servos
Adafruit_PWMServoDriver servo1_1 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo1_2 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo2_1 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo2_2 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo3_1 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo3_2 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo4 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo5 =
Adafruit_PWMServoDriver();
Adafruit_PWMServoDriver servo6 =
Adafruit_PWMServoDriver();

//Servo pinout
uint8_t servo1_1pin = 0;
uint8_t servo1_2pin = 1;
uint8_t servo2_1pin = 2;
uint8_t servo2_2pin = 3;
uint8_t servo3_1pin = 4;
uint8_t servo3_2pin = 5;
uint8_t servo4pin = 6;
uint8_t servo5pin = 7;
uint8_t servo6pin = 8;

//Min and Max values
#define SERVOMIN 150
#define SERVOMAX 600

//Full down angles (50)
int servo1_1LowestPosition = 110;
int servo2_1LowestPosition = 108;
int servo3_1LowestPosition = 115;
int servo4LowestPosition = 115;
int servo5LowestPosition = 105;
int servo6LowestPosition = 90;

//Slight down angles (60)
int servo1_1SLPosition = 100;
int servo2_1SLPosition = 100;
int servo3_1SLPosition = 100;
int servo4SLPosition = 100;
int servo5SLPosition = 100;
int servo6SLPosition = 90;

//Even with surface angles (70)
int servo1_1FlatPosition = 90;
int servo2_1FlatPosition = 80;
int servo3_1FlatPosition = 90;
int servo4FlatPosition = 90;
int servo5FlatPosition = 80;
int servo6FlatPosition = 90;

//Slight up angles (80)
int servo1_1SUPosition = 68;
int servo2_1SUPosition = 68;
```

```
int servo3_1SUPosition = 68;
int servo4SUPosition = 70;
int servo5SUPosition = 68;
int servo6SUPosition = 90;

//Full up angles (90)
int servo1_1UpPosition = 55;
int servo2_1UpPosition = 58;
int servo3_1UpPosition = 55;
int servo4UpPosition = 55;
int servo5UpPosition = 55;
int servo6UpPosition = 90;

//Zero position angles (4)
int servo1_2ZeroPosition = 90;
int servo2_2ZeroPosition = 85;
int servo3_2ZeroPosition = 95;


//Slight left position angles (3)
int servo1_2SLPosition = 105;
int servo2_2SLPosition = 100;
int servo3_2SLPosition = 100;


//Moderare left position angles (2)
int servo1_2MLPosition = 130;
int servo2_2MLPosition = 115;
int servo3_2MLPosition = 100;


//Full left position angles (1)
int servo1_2FLPosition = 145;
int servo2_2FLPosition = 125;
int servo3_2FLPosition = 115;

double pulselength = 0;

int raspInput = 0; //input condition from Raspeberry PI


//The four positions are zero, SL (slight left), ML (medium left), FL (full left)
//Servos are numbered from outside to inside, starting at 1
//Servos numbered _1 are up and down control, _2 left and right control


void setup() {
  //Begin all servos
  servo1_1.begin();
  servo1_1.setPWMFreq(60);
  servo1_2.begin();
  servo1_2.setPWMFreq(60);
  servo2_1.begin();
  servo2_1.setPWMFreq(60);
  servo2_2.begin();
  servo2_2.setPWMFreq(60);
  servo3_1.begin();
  servo3_1.setPWMFreq(60);
  servo3_2.begin();
  servo3_2.setPWMFreq(60);
  servo4.begin();
  servo4.setPWMFreq(60);
  servo5.begin();
  servo5.setPWMFreq(60);
  servo6.begin();
  servo6.setPWMFreq(60);
  yield();

  //Serial begin
  Serial.begin(9600);
  Serial.println("Ready!");

}

void loop() {
  // put your main code here, to run repeatedly:
  //if (Serial.available() ) {
  //raspInput = Serial.read() - '0';
  //delay(2000);
 // raspInput = Serial.parseInt();
 /*
  * Code for serial to get 2 bytes instead of 1
  *   convert ASCII to integer, add, and shift left 1 decimal place
     integerValue = ((incomingByte - 48) + integerValue);
  */
 char buffer[] = {' ',' '}; // Receive up to 2 bytes
 while (!Serial.available()); // Wait for characters
 Serial.readBytesUntil('n', buffer, 2);
 raspInput = atoi(buffer);
 Serial.print("RaspInput: ");
 Serial.println(raspInput);

  switch (raspInput) {
    case 51:
      signal51();
            Serial.println("[51]");
      break;
    case 52:
      signal52();
             Serial.println("[52]");
      break;
    case 53:
      signal53();
            Serial.println("[53]");
      break;
    case 54:
      signal54();
             Serial.println("[54]");
      break;
    case 61:
      signal61();
```

```
          Serial.println("[61]");
  break;
case 62:
  signal62();
          Serial.println("[62]");
  break;
case 63:
  signal63();
          Serial.println("[63]");
  break;
case 64:
  signal64();
          Serial.println("[64]");
  break;
case 71:
  signal71();
          Serial.println("[71]");
  break;
case 72:
  signal72();
          Serial.println("[72]");
  break;
case 73:
  signal73();
         Serial.println("[73]");
  break;
case 74:
  signal74();
          Serial.println("[74]");
  break;
case 81:
  signal81();
          Serial.println("[81]");
  break;
case 82:
  signal82();
          Serial.println("[82]");
  break;
case 83:
  signal83();
          Serial.println("[83]");
  break;
case 84:
  signal84();
          Serial.println("[84]");
  break;
case 91:
  signal91();
          Serial.println("[91]");
  break;
case 92:
  signal92();
          Serial.println("[92]");
  break;
case 93:
  signal93();
          Serial.println("[93]");
    break;
  case 94:
    signal94();
            Serial.println("[94]");
    break;
  default:
    zeroPosition();
  break;
  } // end of case
 // } // End if serial avaiable


} // End of void loop

void signal51()
  {
   setServo(servo1_1LowestPosition,
          servo1_2FLPosition,
          servo2_1LowestPosition,
          servo2_2FLPosition,
          servo3_1LowestPosition,
          servo3_2FLPosition,
          servo4LowestPosition,
          servo5LowestPosition,
          servo6LowestPosition);
  }


void signal52()
  {
   setServo(servo1_1LowestPosition,
          servo1_2MLPosition,
          servo2_1LowestPosition,
          servo2_2MLPosition,
          servo3_1LowestPosition,
          servo3_2MLPosition,
          servo4LowestPosition,
          servo5LowestPosition,
          servo6LowestPosition);
  }


void signal53()
  {
   setServo(servo1_1LowestPosition,
          servo1_2SLPosition,
          servo2_1LowestPosition,
          servo2_2SLPosition,
          servo3_1LowestPosition,
          servo3_2SLPosition,
          servo4LowestPosition,
          servo5LowestPosition,
          servo6LowestPosition);
  }


void signal54()
```

```
 {
  setServo(servo1_1LowestPosition,
        servo1_2ZeroPosition,
        servo2_1LowestPosition,
        servo2_2ZeroPosition,
        servo3_1LowestPosition,
        servo3_2ZeroPosition,
        servo4LowestPosition,
        servo5LowestPosition,
        servo6LowestPosition);
 }

void signal61()
 {
  setServo(servo1_1SLPosition,
        servo1_2FLPosition,
        servo2_1SLPosition,
        servo2_2FLPosition,
        servo3_1SLPosition,
        servo3_2FLPosition,
        servo4SLPosition,
        servo5SLPosition,
        servo6SLPosition);
 }


void signal62()
 {
  setServo(servo1_1SLPosition,
        servo1_2MLPosition,
        servo2_1SLPosition,
        servo2_2MLPosition,
        servo3_1SLPosition,
        servo3_2MLPosition,
        servo4SLPosition,
        servo5SLPosition,
        servo6SLPosition);
 }


void signal63()
 {
  setServo(servo1_1SLPosition,
        servo1_2SLPosition,
        servo2_1SLPosition,
        servo2_2SLPosition,
        servo3_1SLPosition,
        servo3_2SLPosition,
        servo4SLPosition,
        servo5SLPosition,
        servo6SLPosition);
 }


void signal64()
 {
  setServo(servo1_1SLPosition,
        servo1_2ZeroPosition,
        servo2_1LowestPosition,
        servo2_2ZeroPosition,
        servo3_1LowestPosition,
        servo3_2ZeroPosition,
        servo4SLPosition,
        servo5SLPosition,
        servo6SLPosition);
 }

void signal71()
 {
  setServo(servo1_1FlatPosition,
        servo1_2FLPosition,
        servo2_1FlatPosition,
        servo2_2FLPosition,
        servo3_1FlatPosition,
        servo3_2FLPosition,
        servo4FlatPosition,
        servo5FlatPosition,
        servo6FlatPosition);
 }


void signal72()
 {
  setServo(servo1_1FlatPosition,
        servo1_2MLPosition,
        servo2_1FlatPosition,
        servo2_2MLPosition,
        servo3_1FlatPosition,
        servo3_2MLPosition,
        servo4FlatPosition,
        servo5FlatPosition,
        servo6FlatPosition);
 }


void signal73()
 {
  setServo(servo1_1FlatPosition,
        servo1_2SLPosition,
        servo2_1FlatPosition,
        servo2_2SLPosition,
        servo3_1FlatPosition,
        servo3_2SLPosition,
        servo4FlatPosition,
        servo5FlatPosition,
        servo6FlatPosition);
 }


void signal74()
 {
  setServo(servo1_1FlatPosition,
        servo1_2ZeroPosition,
        servo2_1FlatPosition,
```

```
            servo2_2ZeroPosition,
            servo3_1FlatPosition,
            servo3_2ZeroPosition,
            servo4FlatPosition,
            servo5FlatPosition,
            servo6FlatPosition);
  }

void signal81()
  {
    setServo(servo1_1SUPosition,
            servo1_2FLPosition,
            servo2_1SUPosition,
            servo2_2FLPosition,
            servo3_1SUPosition,
            servo3_2FLPosition,
            servo4SUPosition,
            servo5SUPosition,
            servo6SUPosition);
  }


void signal82()
  {
    setServo(servo1_1SUPosition,
            servo1_2MLPosition,
            servo2_1SUPosition,
            servo2_2MLPosition,
            servo3_1SUPosition,
            servo3_2MLPosition,
            servo4SUPosition,
            servo5SUPosition,
            servo6SUPosition);
  }


void signal83()
  {
    setServo(servo1_1SUPosition,
            servo1_2SLPosition,
            servo2_1SUPosition,
            servo2_2SLPosition,
            servo3_1SUPosition,
            servo3_2SLPosition,
            servo4SUPosition,
            servo5SUPosition,
            servo6SUPosition);
  }


void signal84()
  {
    setServo(servo1_1SUPosition,
            servo1_2ZeroPosition,
            servo2_1SUPosition,
            servo2_2ZeroPosition,
            servo3_1SUPosition,
            servo3_2ZeroPosition,
            servo4SUPosition,
            servo5SUPosition,
            servo6SUPosition);
  }

void signal91()
  {
    setServo(servo1_1UpPosition,
            servo1_2FLPosition,
            servo2_1UpPosition,
            servo2_2FLPosition,
            servo3_1UpPosition,
            servo3_2FLPosition,
            servo4UpPosition,
            servo5UpPosition,
            servo6UpPosition);
  }


void signal92()
  {
    setServo(servo1_1UpPosition,
            servo1_2MLPosition,
            servo2_1UpPosition,
            servo2_2MLPosition,
            servo3_1UpPosition,
            servo3_2MLPosition,
            servo4UpPosition,
            servo5UpPosition,
            servo6UpPosition);
  }


void signal93()
  {
    setServo(servo1_1UpPosition,
            servo1_2SLPosition,
            servo2_1UpPosition,
            servo2_2SLPosition,
            servo3_1UpPosition,
            servo3_2SLPosition,
            servo4UpPosition,
            servo5UpPosition,
            servo6UpPosition);
  }


void signal94()
  {
    setServo(servo1_1UpPosition,
            servo1_2ZeroPosition,
            servo2_1UpPosition,
            servo2_2ZeroPosition,
            servo3_1UpPosition,
            servo3_2ZeroPosition,
            servo4UpPosition,
```

```
            servo5UpPosition,
            servo6UpPosition);
  }
 void zeroPosition()
  {
    setServo(servo1_1FlatPosition,
            servo1_2ZeroPosition,
            servo2_1FlatPosition,
            servo2_2ZeroPosition,
            servo3_1FlatPosition,
            servo3_2ZeroPosition,
            servo4FlatPosition,
            servo5FlatPosition,
            servo6FlatPosition);

  }

/*void slightLeft()
  {
    setServo(servo1_1SLPosition,
            servo1_2SLPosition,
            servo2_1SLPosition,
            servo2_2SLPosition,
            servo3_1SLPosition,
            servo3_2SLPosition,
            servo4SLPosition,
            servo5SLPosition,
            servo6SLPosition);
  }

void mediumLeft()
  {
    setServo(servo1_1MLPosition,
            servo1_2MLPosition,
            servo2_1MLPosition,
            servo2_2MLPosition,
            servo3_1MLPosition,
            servo3_2MLPosition,
            servo4MLPosition,
            servo5MLPosition,
            servo6MLPosition);
  }

void fullLeft()
  {
    setServo(servo1_1FLPosition,
            servo1_2FLPosition,
            servo2_1FLPosition,
            servo2_2FLPosition,
            servo3_1FLPosition,
            servo3_2FLPosition,
            servo4FLPosition,
            servo5FLPosition,
            servo6FLPosition);
  }
*/
```

```
void setServo(int a, int b, int c, int d, int e, int f, int g, int h, int
i)
  {
    pulselength = map(a, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo1_1.setPWM(servo1_1pin, 0, pulselength);
   // Serial.println(a);
    //Serial.println(pulselength);
    pulselength = map(b, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo1_2.setPWM(servo1_2pin, 0, pulselength);
    //Serial.println(b);
    //Serial.println(pulselength);
    pulselength = map(c, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo2_1.setPWM(servo2_1pin, 0, pulselength);
   // Serial.println(c);
    //Serial.println(pulselength);
    pulselength = map(d, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo2_2.setPWM(servo2_2pin, 0, pulselength);
    //Serial.println(d);
    //Serial.println(pulselength);
    pulselength = map(e, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo3_1.setPWM(servo3_1pin, 0, pulselength);
    //Serial.println(e);
    //Serial.println(pulselength);
    pulselength = map(f, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo3_2.setPWM(servo3_2pin, 0, pulselength);
   // Serial.println(f);
    //Serial.println(pulselength);
    pulselength = map(g, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo4.setPWM(servo4pin, 0, pulselength);
    //Serial.println(g);
    //Serial.println(pulselength);
    pulselength = map(h, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo5.setPWM(servo5pin, 0, pulselength);
    //Serial.println(h);
    //Serial.println(pulselength);
    pulselength = map(i, 0, 180, SERVOMIN, SERVOMAX);
// converts pulse lenght to dregrees
    servo6.setPWM(servo6pin, 0, pulselength);
    //Serial.println(i);
    //Serial.println(pulselength);
  }
```

Finally, the below code is used to control the Turn signal as well as the Daylight Running lights.

```
#include <Tlc5940.h>
#include "LPD8806.h"
```

```
#include "SPI.h" // Comment out this line if using Trinket or Gemma
#ifdef __AVR_ATtiny85__
 #include <avr/power.h>
#endif

const int buttonPin = 2; //pin of button controlling low beams
const int buttonPin2 = 4; //pin of button crontrolling high beams
int buttonState1 = HIGH;
int lastButtonState;
int buttonState2 = 0;
int ledState;

unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

int nLEDs = 32;

int dataPin  = 7;  //pin of button controlling turn signal
int clockPin = 8; //pin to control RGB strip
int buttonPin3 = 12; //pin to control RGB strip
int buttonPress3;
LPD8806 strip = LPD8806(nLEDs, dataPin, clockPin);


void setup()
{
pinMode(buttonPin, INPUT);
pinMode(buttonPin2, INPUT);
 Tlc.init(0);
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000L)
  clock_prescale_set(clock_div_1); // Enable 16 MHz on Trinket
#endif

pinMode(buttonPin3, INPUT);
  // Start up the LED strip
  strip.begin();

  // Update the strip, to start they are all 'off'
  strip.show();

delay(100);
}

void loop()
{
  int i;
int reading = digitalRead(buttonPin);  //read a button press
buttonState2 = digitalRead(buttonPin2);
buttonPress3 = digitalRead(buttonPin3);
if (buttonPress3 == LOW)//if button is pressed turn signal activates
{
  colorWipe(strip.Color(255,  30,   0), 50); //DRL's amber
  colorWipe(strip.Color(0,  0,   0), 50); //DRL's off

  delay(1);

}
if(reading != lastButtonState)
{

  lastDebounceTime = millis();
}

if((millis()- lastDebounceTime)>debounceDelay)
{
  if(reading != buttonState1)
  {
    buttonState1 = reading;
  if(buttonState1 == HIGH)
  {
   ledState = !ledState;

  }
}
}
 if (ledState == HIGH && buttonState2 == HIGH )  //if button 2 pressed than low beams on
 {

 Tlc.update();
Tlc.set(1, 50);
 for (i = 0; i< 32; i++)
  {
  strip.setPixelColor(i ,0,  0,   0);  //  DRL off

  strip.show();
  delay(1);
  }
}
delay(1);



  }

else if (ledState == HIGH && buttonState2 == LOW) //if button 2 pressed turn high beams on
  {

Tlc.update();


Tlc.set(1, 4095);

 for (i = 0; i< 32; i++)//DRL's off
{

  strip.setPixelColor(i ,0,  0,   0);  //  yellow 127-64-0
  // orange 255-160-0
  strip.show();
```

```
  delay(1);
}

delay(10);

  }

else if (ledState == LOW)//if buttons not pressed than the
headlights are off
 {

Tlc.update();

Tlc.set(1, 0);
delay(1);//10

 for (i = 0; i< 32; i++) // DRL's on only iff the headlights are
off
{

  strip.setPixelColor(i ,64,  64,   64);

  strip.show();
  delay(1);
}


 }
lastButtonState = reading;
}


void colorWipe(uint32_t c, uint8_t wait)
{
  int i;

  for (i=0; i < strip.numPixels(); i++)
  {
     strip.setPixelColor(i, c);
     strip.show();
     delay(30);

       }

 }
```

APPENDIX D: MECHANICAL

Project consists of several mechanical components: the LED units, and the LED assembly. There is also the filler panel holding everything together, as well as providing visually appealing design features.

i. LED Unit

Consists of three parts:

1. U shaped steel bracket (Fig. 20)

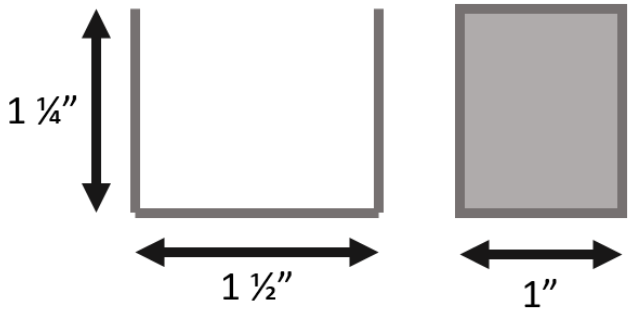


Figure 20. LED unit U shaped steel frame[20]

2. Aluminum centerpiece with steel rods to attach to steel bracket. (Fig. 20)

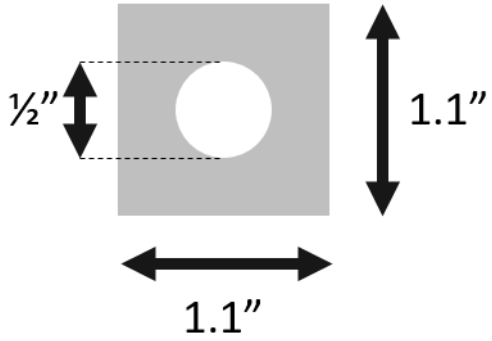


Figure 21. Centerpiece that holds the cover with lens and the LED[22]

3. Plastic cover with lens (Fig. 21)

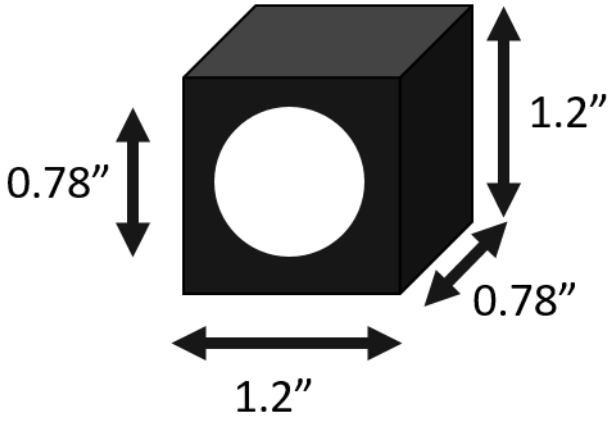


Figure 22. Cover with lens, lens goes into hole in front. Back side is hollow. [23]

ii. LED array

Consists of am aluminum plate, with holes drilled for servos. See Fig. 4 for details.

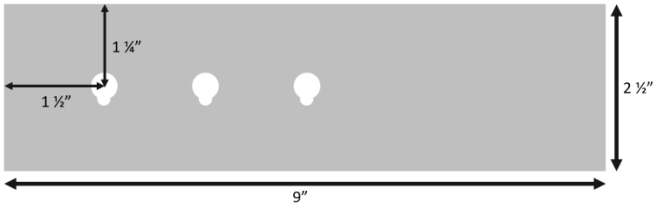


Figure 23. Plate for LED array. Aluminum, holds all LED units together. [24]

iii. Filler Panel

The filler panel is a fiberglass piece that the LED array is attached to and the entire assembly is then placed into the headlight housing. See Fig. 21.

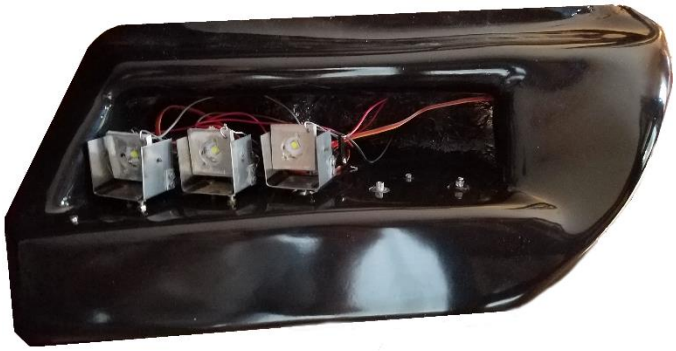


Figure 24. Filler panel with LED unit assembly partially installed. [25]

APPENDIX E: VENDOR CONTACTS

N/A

APPENDIX F: RESUMES

Alisha Rosario

Objectives

My objective is to get an entry level position as an Electrical Engineer and apply my knowledge towards a company with a good reputation and successful business venture.

Education

- California State University, Sacramento

Bachelor of Science in Electrical Engineering (May 2017)
GPA: 3.3

- Solano Community College

Associates of Arts in Mathematics (May 2012)

Technical Experience

- Senior Design (Fall 2016-Spring 2017)
  - To reduce the amount of accidents at night, the visibility of the driver will be increased through the means of self-adjusting headlights
  - Servos, LED's, and Sensors are key components that are programmed to act accordingly
- Advanced Analog Integrated Circuits Lab (Fall 2016)
  - Drew and Simulated a Latching Comparator using Mentor Software
- Electronic Circuits Lab I & II (Fall 2015 - Spring 2016)
  - Built circuits on a breadboard and tested the circuit for behavior using the oscilloscope.
  - The function generator and Voltage source was used to send signals to the circuit.
  - Simulated Circuits using Multisim Program

Skills

- Programming: C#/C++, Visual Basic
- Problem Solving
- Communication
  - Group Processing Module through Tau Beta Pi (October 2016)
- Teamwork Skills
  - Team Leader for a semester
  - Worked with groups on projects

Achievements

- Member of Tau Beta Pi Engineering Honor Society
  - California State University of Sacramento
- RLDC (Regional Leadership Development Conference) (2011)
  - Made a Business Plan with a group
  - Talked with Professionals at Microsoft about Business Plan
- Member of MESA (Mathematics, Engineering, Science, Achievement)
  - Napa Valley College (2011- 2012)
- Member of SHPE (Society of Hispanic Professional Engineers)
  - Napa Valley College (2011- 2012)

Juan Chico

OBJECTIVE:
To enter California's high technology workforce, and make significant contributions to the field.

EDUCATION:
In progress: BS, Computer Engineering · CSU Sacramento · Dec 2017

Courses:
Electronics Advanced Computer Organization Assembly Architecture
Network Analysis Data Structures & Algorithm Development Programming Methodology
Circuit Analysis Computer Interfacing
Advanced Logic Design Systems Programming

PROJECT EXPERIENCE:
16-bit Single cycle Processor:
Member of a two person team, I help develop and implement a 16-bit single cycle processor unit using MIPS architecture. To accomplish this task I had to develop many of the components that made our five stage pipeline design using behavioral modeling Verilog. The purpose of this project was to implement store/word operations, integer arithmetic and branching. Simple hazard detection and forwarding was used to avoid hazards caused by the instructions.

Android software implementation with a Micro controller:
Member of a four person team, I help design and implement a toy vehicle controlled via an android phone.
Using an Arduino as the car's micro controller, I was in charged in the development and design of an application that would accomplish this task. Using blue tooth for connectivity, I developed an application using the phone's accelerometer to drive the vehicle. The application functions included driving the vehicle forward, backward, steering and turbo.

Data System Conversion:
I was in charge in a system conversion for two Dental offices. Two accomplish this task, I had to install a network of computers at each work station. A server was use for the synchronization and communication between these computers. In order for these computers to manage and maintain patient's records, a data conversion had to be done. Using a digital X-ray machine was also used in order to digitally take patients X-rays and place them on the patient?€ ?s dental records.

Web Developer:
For the past few years I was in charge in developing and maintaining websites for small businesses. The tools I used to accomplish this included knowledge in HTML and PHP. Software such as MySQL and CMS was also used.

Quiz Show Buzzer:
I was in charge in developing a system to create a quiz show buzzer for one of my engineering clubs. I created a user interface using C#, in which it displayed a question and would display a number between 1 thru 4 depending on which buzzer was click first. In order to achieve, I used four buzzers which all of them were connected to an Arduino micro controller. The Arduino was used to decide which of the four buzzers were click first and send out a signal to the interface created using C# to display which buzzer clicked first.

Check-in System:
I was in charge in developing a system to check-in students for one of my engineering clubs. In order to achieve this, I used a Magnetic Stripe ID Swiper for which students would use their school ID to check-in. The ID Swiper was link to a C program I made to grab the raw information from the students ID and filter out the students name and ID number. After filtering the information needed, the program then placed the information into an Excel sheet with a time stamp.

Safe Rides America Android APP:
I am currently working on an android application that markets and operates the mobile-app-based transportation network. The Application allows consumers to submit a trip request, which is routed to Safe Rides America taxi drivers.

KNOWLEDGE AND SKILLS:
Computer Languages:
Verilog · C · Java · C++ · C# · X86 Assembly · HTML · PHP · SQL

Systems:
Linux · Windows 8 · Windows 7 · Windows XP

Software:
ModelSim · Cadence PSpice · MultiSim · Eclipse · Visual Studio · Matlab · Xilinx ISE · MS Office

Tools:
Oscilloscope · Function Generator · Logic Analyzer

WORK EXPERIENCE:
Teacher Assistant CSUS MESA 3/14 - 4/14
Document Analyst Old Republic Title 3/15 to present
Computer Repair Technician Self-Employed 3/06 to present

ACTIVITIES AND ACCOMPLISHMENTS:
• Web Master, Society of Hispanic Professional Engineers
• Active Member, Association for Computing Machinery
• Active Member, Competitive Robotics
• Captain, High School Soccer Team

Konstantin Komarchuk

I am currently a student, majoring in Electrical Engineering, with concentration in Digital/Analog Electronics. Looking for an entry level/internship position to get experience in the industry while still in school as well as apply acquired knowledge right in the field.

Educational Experience

California State University, Sacramento — Sacramento, CA
Electrical and Electronic Engineering — (2015-Present)
Expected Graduation: 2017

Previous Courses:
Network Analysis
Electromechanical Conversion
Applied Electromagnetics
Signals and Systems
Electronics I

Introduction to Microprocessors,
Introduction to Feedback Systems
Computer Hardware Design
Electronics II
Robotics
Modern Communication Systems
Product Design Project I (Senior Project)

Currently taking - Spring 2017:
Product Design Project II (Senior Project)
Engineering Economics
Digital Control Systems
Physical Electronics (Semiconductor Physics)

American River College Sacramento, CA
Associates of Science, Mathematical and Physical Science
Associates of Arts, Social Science
Graduated: 2015

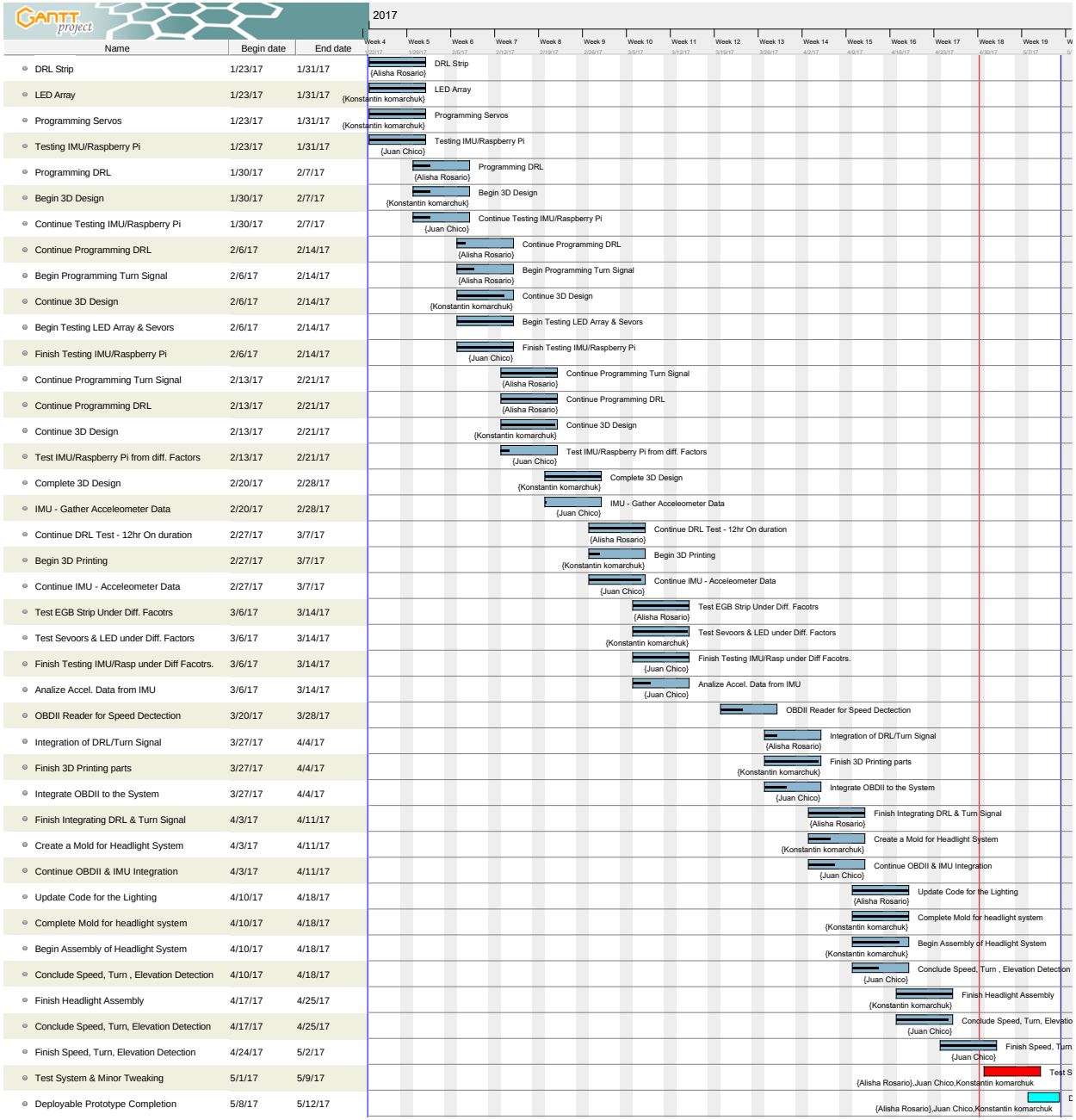
Professional Experience
B & J Body Shop & Towing — Rancho Cordova, CA
Body/jury man Assistant — 2013-Present

Personal Skills
I am fully fluent in two languages, English and Russian, as well as have basic understanding of Belarussian and Ukrainian.

C/C++/C#, Visual Basic, HTML, CSS, PHP, MultiSim, Advanced Design System (ADS), MS Office, PSPICE

Responsible leader of church groups and events, including camps, teaching and conferences. (Second Slavic Baptist Church, North Highlands, CA)

Gantt Chart


2017
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
Week 17
Week 18
Week 19
Name
Begin date
End date
DRL Strip 1/23/17 1/31/17 {Alisha Rosario}
LED Array 1/23/17 1/31/17 {Konstantin komarchuk}
Programming Servos 1/23/17 1/31/17 {Konstantin komarchuk}
Testing IMU/Raspberry Pi 1/23/17 1/31/17 {Juan Chico}
Programming DRL 1/30/17 2/7/17 {Alisha Rosario}
Begin 3D Design 1/30/17 2/7/17 {Konstantin komarchuk}
Continue Testing IMU/Raspberry Pi 1/30/17 2/7/17 {Juan Chico}
Continue Programming DRL 2/6/17 2/14/17 {Alisha Rosario}
Begin Programming Turn Signal 2/6/17 2/14/17 {Alisha Rosario}
Continue 3D Design 2/6/17 2/14/17 {Konstantin komarchuk}
Begin Testing LED Array & Sevors 2/6/17 2/14/17
Finish Testing IMU/Raspberry Pi 2/6/17 2/14/17 {Juan Chico}
Continue Programming Turn Signal 2/13/17 2/21/17 {Alisha Rosario}
Continue Programming DRL 2/13/17 2/21/17 {Alisha Rosario}
Continue 3D Design 2/13/17 2/21/17 {Konstantin komarchuk}
Test IMU/Raspberry Pi from diff. Factors 2/13/17 2/21/17 {Juan Chico}
Complete 3D Design 2/20/17 2/28/17 {Konstantin komarchuk}
IMU - Gather Acceleometer Data 2/20/17 2/28/17 {Juan Chico}
Continue DRL Test - 12hr On duration 2/27/17 3/7/17 {Alisha Rosario}
Begin 3D Printing 2/27/17 3/7/17 {Konstantin komarchuk}
Continue IMU - Acceleometer Data 2/27/17 3/7/17 {Juan Chico}
Test EGB Strip Under Diff. Facotrs 3/6/17 3/14/17 {Alisha Rosario}
Test Sevoors & LED under Diff. Factors 3/6/17 3/14/17 {Konstantin komarchuk}
Finish Testing IMU/Rasp under Diff Facotrs. 3/6/17 3/14/17 {Juan Chico}
Analize Accel. Data from IMU 3/6/17 3/14/17 {Juan Chico}
OBDII Reader for Speed Dectection 3/20/17 3/28/17
Integration of DRL/Turn Signal 3/27/17 4/4/17 {Alisha Rosario}
Finish 3D Printing parts 3/27/17 4/4/17 {Konstantin komarchuk}
Integrate OBDII to the System 3/27/17 4/4/17 {Juan Chico}
Finish Integrating DRL & Turn Signal 4/3/17 4/11/17 {Alisha Rosario}
Create a Mold for Headlight System 4/3/17 4/11/17 {Konstantin komarchuk}
Continue OBDII & IMU Integration 4/3/17 4/11/17 {Juan Chico}
Update Code for the Lighting 4/10/17 4/18/17 {Alisha Rosario}
Complete Mold for headlight system 4/10/17 4/18/17 {Konstantin komarchuk}
Begin Assembly of Headlight System 4/10/17 4/18/17 {Konstantin komarchuk}
Conclude Speed, Turn , Elevation Detection 4/10/17 4/18/17 {Juan Chico}
Finish Headlight Assembly 4/17/17 4/25/17 {Konstantin komarchuk}
Conclude Speed, Turn, Elevation Detection 4/17/17 4/25/17 {Juan Chico}
Finish Speed, Turn, Elevation Detection 4/24/17 5/2/17 {Juan Chico}
Test System & Minor Tweaking 5/1/17 5/9/17 {Alisha Rosario},Juan Chico,Konstantin komarchuk
Deployable Prototype Completion 5/8/17 5/12/17 {Alisha Rosario},Juan Chico,Konstantin komarchuk