March 2018

ICE Detection for LED Headlights

Cameron Back
Worcester Polytechnic Institute

Ian Nolan
Worcester Polytechnic Institute

Thomas Vining
Worcester Polytechnic Institute

ICE Detection for LED Headlights

A Major Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

Submitted by:

Cameron Back
Ian Nolan
Thomas Vining

Date: March 10, 2018

Report Submitted to:

Professor Stephen Bitar, Worcester Polytechnic Institute

Abstract

This project focused on building an efficient system for detecting and melting ice that may form on an LED headlight during inclement weather. Although LED headlights draw less power and provide more efficient light than their competitors, they do not produce enough outward heat to melt ice. The proposed solution utilizes an Indium Tin Oxide (ITO) film to produce heat along the surface of the headlight, as well as infrared (IR) sensors for detecting ice, and temperature sensors to determine freezing conditions. System operation is managed by a microcontroller which also allows for CAN (Controller Area Network) bus integration. During testing, the prototype proved it could perform as designed as well as interface with a vehicle's CAN bus.

Acknowledgements

The team would like to express our sincere gratitude to the following people, organizations, and institutions for their help and support throughout our project:

- Professor Bitar, Instructor, WPI
- Professor McNeill, Department Head, WPI
- The New England Center for Analog and Mixed Signal Design (NECAMSID)
- Mr. Joseph Gerardi, General Manager, Innovative Dynamics Inc (IDI), 2560 N Triphammer Rd, Ithaca, NY 14850

Executive Summary

LED headlights come with more energy efficiency and brighter lights than their competitors, however, they do not produce enough outward heat to melt ice off of a headlight lens. When using LED headlights in winter weather it is possible to have obstructed headlights that could lead to unsafe driving conditions. To melt the ice along the lens exterior, an Indium Tin Oxide (ITO) film was laid along the inside, which could produce heat uniformly along its surface when an electric current is passed through it. The drawback to using this system however, was that it drew 18 watts of power when activated, so it could not be left on for prolonged periods of time if the car was not running, or it would risk draining the car battery.

To ensure that the ITO was only on when necessary, a control system was created using a microcontroller to implement a programmable control system. The system worked by first using temperature sensors for determining if ice formation was possible. If the temperature was below 40°F, it would then check an infrared (IR) sensor for obstructing ice. If ice was detected, then the ITO would be triggered to deal with the ice, but as soon as the IR sensor no longer detected any obstruction, the ITO would be shut off to conserve energy. These IR sensors incorporate a simple IR LED and transistor pair, to sense the presence of obstructions within a certain proximity. The LED emits IR outward from the bulb. Meanwhile, the transistor is sensitive to IR light, and can complete a circuit whenever IR is detected. Therefore, whenever objects appear in front of the pair, the transistor will receive the IR from the obstruction reflecting waves from the LED. Finally, for multiple microcontrollers to be able to communicate with each other in a car, a Controller Area Network (CAN) was used. This system uses two data lines: CAN High (CAN-H), and CAN Low (CAN-L). By taking the differential between the two lines, a data stream is created.

Throughout the development of the final prototype, both the ice sensor and CAN bus system underwent many changes. First, a test had to be performed to ensure that an IR sensor could properly detect the presence of ice on the exterior of a lens. This was done by placing ice cubes on the exterior of the mock headlight enclosure, and testing the responsiveness of a prebuilt proximity IR sensor. After proving that IR was viable, a new IR sensor was developed from a simple IR LED and transistor pair. Upon testing this simple design, four of these LED transistor pairs were then combined into a final design to test four separate points along the headlight surface area, thus increasing the likelihood of detecting ice before it spread too far throughout the lens.

In parallel to the ice sensor's development, the CAN bus system was being utilized to emulate an automobile's communication system. By having two microcontrollers communicating with each other over the CAN lines, it is possible to have a headlight node receive information from daylight, external temperature, and other sensors found inside of a car. The first attempt to create a CAN bus system involved connecting a teensy with a CAN transceiver to an Arduino Uno with a CAN shield. After testing each individual component separately and then together as one system it was determined that the Teensy and Arduino Uno could not form their own CAN. The next design involved connecting two Arduino Unos with CAN Shields to form a CAN system. This new design however did not have easily accessible CAN TX and RX lines, so a CAN transceiver was connected to the CAN lines.

Once development of the two systems was complete, the IR sensor was integrated into the CAN bus system, along with two temperature sensors to simulate both the headlight sensor, and the car's existing onboard sensor.

Then, to improve the ice detection sensor, the IR LEDs were arranged in pairs to increase the energy efficiency of the system. An OR chip was then added to convert the four signals of the ice sensor into one output. This was done to reduce the required amount of connections to the Arduino Uno. A MOSFET was added to the ITO and ice sensor circuit, so the Arduino Uno would be able to control the additional power draw.

Each Uno has code uploaded to it so it can perform the task that would be associated with operating the headlight ice detection system and operating the external temperature. The headlight Uno starts off its code by asking for the external temperature from the other Uno. After receiving a temperature below 40°F, the headlight Uno checks its own temperature sensor for the same condition. Once both sensors are below 40°F then the Uno activates the ice sensor and waits a second for a positive ice reading. Following a positive reading the Uno then powers the ITO until the ice sensor no longer detects any ice.

The final prototype combines the ITO heating film, two analog temperature sensors, two microcontrollers, two CAN shields, an ice detection sensor, and a fully assembled PCB. After fully assembling the final prototype the power draw from the idle to the ice detection state goes from 700 to 950 mW, and in the heating state the power draw goes up to 18.95 W.

There are two main advantages to adding this system as opposed to other LED headlight ice management systems, such as the 2016-2017 project. The first is energy efficiency, since this system uses temperature sensors along with an IR sensor it is able to conserve power when it is cold out and there is no ice obstruction, unlike its competitors. The second advantage of this system is the convenience for the user, as not having any system requires the user to actively track the condition of their headlight lens. The final prototype can be applied to other

applications such as heating windows, outdoor lights, and other clear systems that are exposed to fog/ice buildup.

However, there were a few potential issues that still need to be addressed before applying this project to an automobile's headlight system. The first issue was that the 2016 project's ITO testing procedure did not attempt to melt ice, and a simple test of melting ice in a cold environment could prove its functionality. The second potential issue was that the IR sensor would likely be triggered by sunlight and other IR sources such as halogen headlights. A possible solution to this problem would include using the car's ambient light sensor, optical filters for the transistors, changing the angle between LED and transistor, or modulating IR signal through the IR LED. The third potential issue is the lack of error checking and redundancies. This could be solved by determining the signs of a defect, keeping a log of temperature readings, and additional hardware for redundancies. The last issue is in regard to using an Arduino Uno along with its CAN library to communicate with a car's CAN system. In order to verify that this system is compatible with a particular car, it would require researching CAN protocols and identifiers from the car's manufacturer.

Authorship Page

Section Title:	Section		Author	Editor
Abstract			Cameron Back	All
Acknowledgements			All	All
Executive Summary			All	All
1 Introduction				
	1.1		Cameron Back	Ian Nolan
	1.2		Ian Nolan	Thomas Vining
	1.3		Thomas Vining	Cameron Back
2 Background				
	2.1		Cameron Back	Ian Nolan
	2.2		Ian Nolan	Thomas Vining
	2.3		Ian Nolan	Thomas Vining
	2.4		Cameron Back	Ian Nolan
	2.5		Thomas Vining	Cameron Back
3 Methodology				
	3.1		Cameron Back	Ian Nolan
	3.2		Cameron Back	Ian Nolan
	3.3		Cameron Back	Ian Nolan
	3.4		Thomas Vining	Cameron Back
	3.5		Ian Nolan	Thomas Vining
	3.6		Thomas Vining	Cameron Back
4 Results and Analysis				
	4.1		Cameron Back	Ian Nolan
	4.2		Thomas Vining	Cameron Back
	4.3		Ian Nolan	Thomas Vining
	4.4		Cameron Back	Ian Nolan
5 Conclusion				
	5.1		Cameron Back	Ian Nolan
	5.2		Ian Nolan	Thomas Vining
	5.3		Thomas Vining	Cameron Back
	5.4		Cameron Back	Ian Nolan
Appendices			All	All

Table of Contents

Table of Figures

Table of Tables

Table of Acronyms

Full Names:	**Acronyms:**
Acknowledgement bits	ACK
Controller Area Network	CAN
Cyclic Redundancy Check	CRC
Chip Select	CS
End Frame	EF
Forward Looking Infrared	FLIR
Infrared	IR
Indium Tin Oxide	ITO
Master In Slave Out	MISO
Master Out Slave In	MOSI
Receive	RX
Serial Clock	SCLK
Thin Film Devices	TFD
Transmit	TX

1 Introduction

LED headlights come with more energy efficiency and brighter lights than their competitors, however, they do not produce enough outward heat to melt ice off of a headlight lens. When using LED headlights in winter weather, it is possible for ice to obstruct the light being emitted by the headlights, which could lead to unsafe driving conditions.

1.1 Detecting and Melting Ice

In order to melt the ice on the exterior of the lens, an Indium Tin Oxide (ITO) film was placed along the interior, following the design used by a previous MQP group. An ITO film produces heat uniformly along its surface whenever current is induced through it. By placing it along the headlight, it is able to melt away all of the ice on the outside of the lens. The drawback of the ITO system is that it draws around 18 watts of power, meaning it cannot be left on for extended periods of time while the car is turned off or else it will risk draining the car battery. The previous design attempted to remedy this issue by incorporating a temperature-based control system and a simple hysteresis loop [1]. However, this meant that the ITO would still be on constantly during freezing temperatures, regardless of whether ice was present or if the car was running. To increase the efficiency of the system, a more effective control system was needed. To accomplish this, a series of new sensors and a digital control system were incorporated.

First, Infrared sensors were included in order to sense the presence of ice on the exterior of the lens. These sensors incorporate a simple IR LED and transistor pair, which can sense the presence of obstructions within a certain proximity. The LED emits IR outward from the bulb. Meanwhile, the transistor is sensitive to IR light, and will complete a circuit whenever IR is detected. Therefore, whenever objects appear in front of the pair, the transistor will receive the IR from the obstruction reflecting waves from the LED.

Then, to further improve upon the control system, a microcontroller was introduced into the system in order to allow for a programmable control system. The controller stores the code for controlling the ITO output, and additional sensors. The additional sensors include two temperature sensors and one ice detection sensor which are all connected to the microcontroller. Then finally, two microcontrollers were used to simulate the process of communicating within the automobile. This system was intended to prove that the design could communicate with, and be integrated into a car's existing sensors and systems.

1.2 Controller Area Network

In order to have multiple microcontrollers to communicate with each other in a car, a Controller Area Network (CAN) is used. This system has two data lines to transmit data to from one microcontroller to all other microcontrollers simultaneously. The two data lines are CAN High (CAN-H) and CAN Low (CAN-L), and by taking the differential between these two lines, a data stream is created. When the differential goes to a high value then it is translated as a dominant bit, whereas a zero-value differential translates to a recessive bit. There can only be one active transmitter at a time and if two devices attempt to communicate with each other, then the devices enter a stage of arbitration. Most automobiles today use CAN bus systems to allow for communication between sensors [2]. Creating a system that uses CAN bus protocols will make it easier to incorporate it into automobiles.

1.3 Emulating a CAN System

To model the CAN line, two Arduino Unos with CAN bus Shields were attached by two CAN bus lines. The CAN Library software was downloaded to drive the system. First the system was tested by sending data from an analog temperature sensor on one Uno to the other across the CAN bus line. Then by looking at the lines using a logic analyzer, the team determined that CAN bus messages were being sent and received across the system. Code was then created to utilize the ice and temperature sensors to determine when to turn on the ITO. Two separate code modules for each controller were designed, compiled and loaded onto each Uno. Together, these two boards worked together using CAN bus to send data across to one another.

2 Background

This chapter discusses the research material found through Google Scholar, IEEE Xplore, Engineering Village, Gartner, and other online resources. The first section looks at a previous MQP project that focused on melting ice on a headlight enclosure, and the next few sections cover the current methods for Ice detection, and CAN bus research and interface methods.

2.1 Efficient LED Headlight Heating System (2016-17)

Previously, another MQP had been done in the same field from 2016-2017, titled: Efficient LED Headlight Heating System. Their project aimed to develop a system for melting the ice off of an LED headlight enclosure. They began by creating a realistic headlight enclosure for testing, and then they developed plans for how to heat the surface. The main design goal was to heat the front lens from approximately 0°F to 65°F within a 10-minute window.

Their initial attempt involved transferring heat given off by the LED and its heatsink to the inside surface of the front lens by blowing the air forward with a simple fan. After testing however, this option was proven to be incapable of heating the lens within the 10-minute window, meaning that a new approach needed to be developed. So instead of the fan, they incorporated an ITO film into the circuit, which would apply uniform heat along the inside of the front lens whenever a current was run through it. After modifying their original circuit to incorporate the ITO film, they went on to test the new prototype using a forward looking infrared (FLIR) camera. The ITO prototype was then proven through thermal testing to be capable of heating the glass within the desired time frame, achieving the overall goal for the project. Upon seeing the potential of the original ITO heating element, the team decided to build on what the previous group had done, and to further their design. From the previous project: the headlight enclosure, ITO film, and all of the thermal data relating to these two components were reused.

2.1.1 The headlight enclosure

The headlight enclosure was designed to test the different solutions proposed in their project. It was modeled after a Toyota Camry headlight, with measurements of 6” x 5.5” x 4.5”. To simplify the design and simulations, the test enclosure design was built to be a 6” cube with a hole in the back face for the placement of the light bulb. The materials selected were intended to model the materials of an actual headlight as well. Five of the sides were constructed out of acrylic and the lens is polycarbonate to replicate the material of a standard car’s lens. The model of the enclosure is shown in Figure 1.

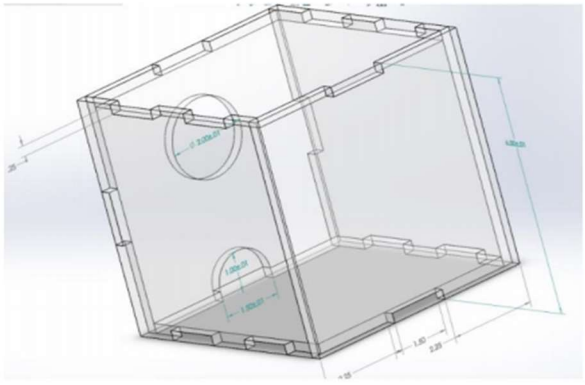

Figure 1 SolidWorks model for the enclosure [1]

2.1.2 The ITO film

The ITO film was acquired from Thin Film Devices (TFD) as a custom order. ITO films are transparent, electrically conductive, and have been applied to several other applications from flat-panel displays, smart windows, polymer-based electronics and heating. As a heating film, it is commonly used in the cockpit windows of airplanes as a de-icer, and in camera lenss to keep the lens from fogging up.

2.1.3 System block diagram

For their prototype circuit, they designed a purely analog system, shown in Figure 2, which took the outputs of two temperature sensors, and activated the heating film (switching from voltage low to voltage high) if either of them sensed temperatures below 35°F (temperature low). After the heating film is activated, it will remain on until a temperature of 50°F or higher (temperature high) is read. Thus, completing the hysteresis loop in Figure 3.

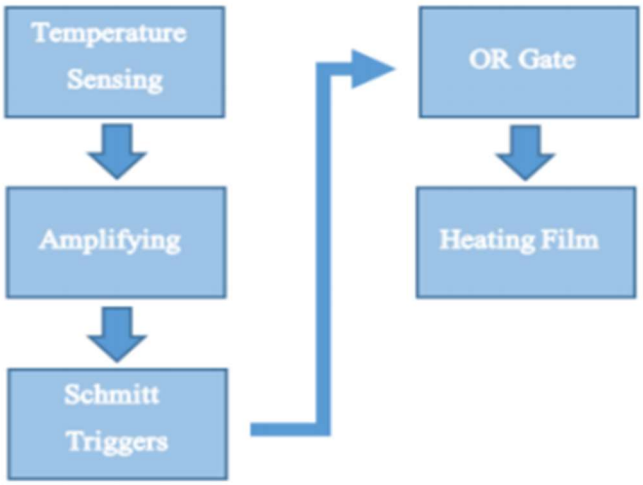

Figure 2 System block diagram [1]

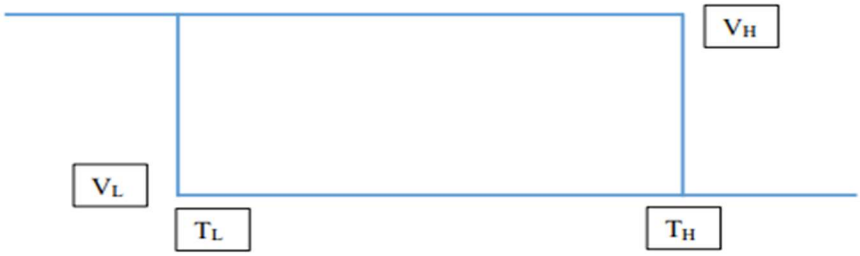

Figure 3 Hysteresis loop, voltage (VH, VL) and temperature high & low (TH, TL) [1]

2.1.4 Thermal testing results

For the final stage of their project, the team tested their system by placing it in a freezer, and waited until the temperature of the device was at 0°F before activating their system. They then began powering the device using a 12 V battery, and waited a period of 10 minutes. At that point, they began using a forward looking infrared (FLIR) camera to measure the final

temperature across the surface area of the lens. The result, shown in Figure 4 below, confirmed that the system could heat the front of the lens to 65°F in a 10-minute window. Table 1 and Figure 5 then show the chart of the temperature vs time of the front lens across the 10-minute window, once the ITO film began heating its surface.

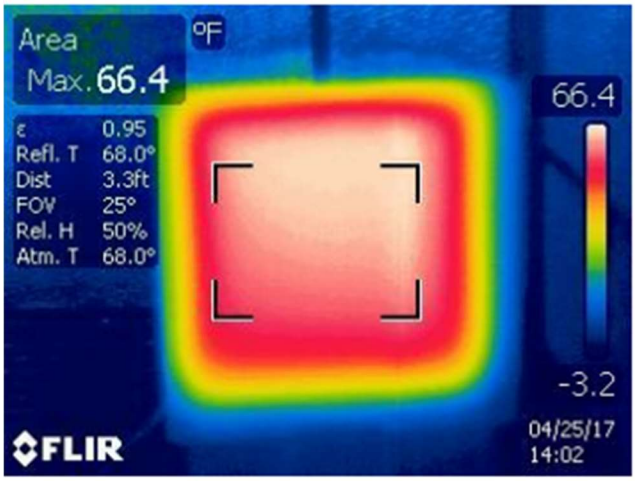

Figure 4 FLIR image of the ITO enclosure prototype [1]

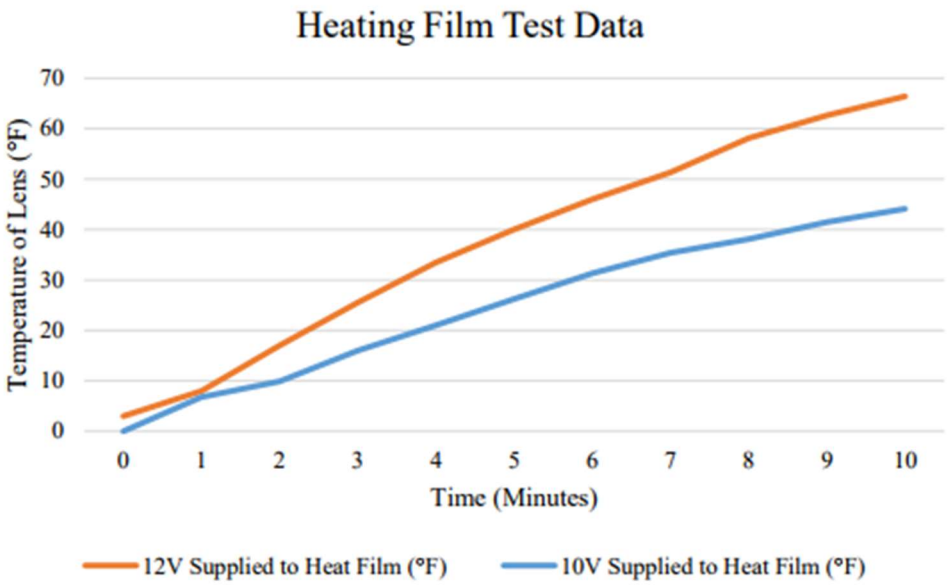

Figure 5 Plot of temperature vs time of the front lens [1]

Table 1 Chart of temperature vs time of the front lens [1]

Time (Minutes)	10V Supplied to Heat Film (°F)	12V Supplied to Heat Film (°F)
0	0	3.1
1	6.8	8.0
2	9.9	17.2
3	16.0	25.5
4	21.0	33.5
5	26.2	40.0
6	31.3	46.1
7	35.4	51.3
8	38.1	58.1
9	41.5	62.6
10	44.1	66.4

2.1.5 Their recommendations

Their project then concluded by leaving many recommendations for future changes and enhancements for their system. Most notable, were suggestions for increasing the efficiency in their design, as well as adding some element of control to their project. First, their design was connected directly to the battery, so it would operate regardless of whether or not the car was running or the lights needed to be on. By their estimates, when running the ITO constantly without the engine running, the prototype would drain a fully charged 12V 45Ah car battery in approximately 14 hours. Second, they noted that the system was activated whenever temperatures were below freezing on the surface of the lens, but not all below freezing weather conditions would cause ice to form on the headlight lenss. To solve this, they suggested an element of control be added to the system to ensure that the ITO was active only when absolutely necessary. To build on this concept, they also suggested incorporating a secondary method for sensing ice to ensure that the system would be able to plan its running times as efficiently as possible.

2.2 LED Basic Info and Market Trends

LED headlights are on the rise in the US. By the year 2020 is its projected that 20% of US cars will be equipped with LED headlights [3]. This study does not even include the potential of new car technologies involving LED headlights. In Figure 6, it becomes clear that LED headlight (headlamp) technology is at the peak of its hype cycle.

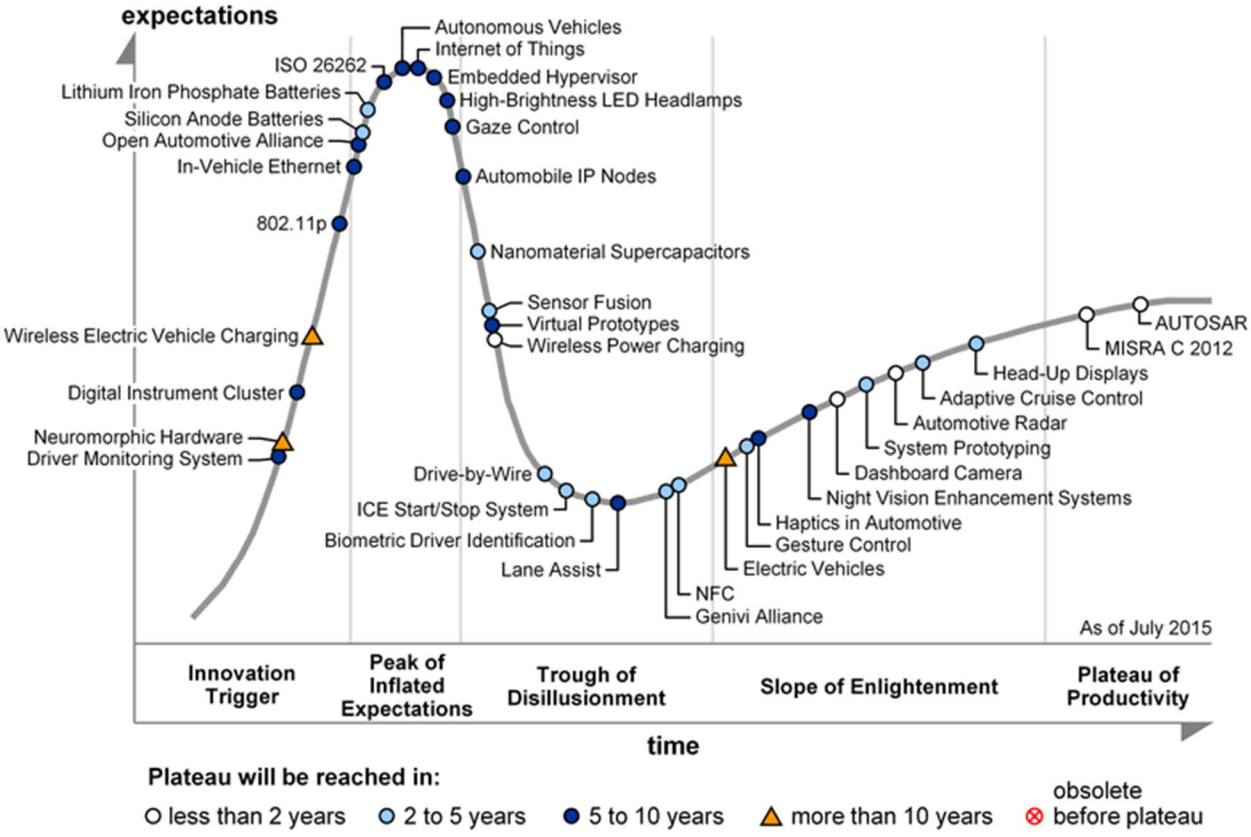

Figure 6 Hype Cycle for Automotive Electronics, 2015 [4]

Furthermore, LED headlights are also being developed as method for cars to communicate to other transportation systems. An example of this is the Car2Car communication system which relies on LED lights due their high ability to control them [5]. These new applications for LED headlights will boost the presence of LED headlights on the streets. With this increase in usage, there will be more drivers exposed to conditions where the current LED headlight structure will be less effective than halogens.

2.3 Current Problems with LED Headlights

Despite their efficiency, LED headlights still have four major problems compared to halogen headlights. The first problem is the amount of light that is projected from the headlight. An LED bulb can produce 3,000 lumens while a halogen bulb produces only 1,400 lumens [6]. Because of this, LED headlights are often referred to as being too bright and having the capability to blind oncoming drivers, however, Audi has been addressing this problem by implementing their LED Matrix. The matrix uses a series of LEDs and turns off the bulbs that would otherwise blind oncoming drivers. Another problem with LED headlights is the high color temperature of the light. An LED headlight is around 6,000K, which is more intense than the 5,500K for daylight. Halogens on the other hand output a 3,000K color temperature (Figure 7) and drivers have become accustomed to color temperatures ranging from halogens to daylight.

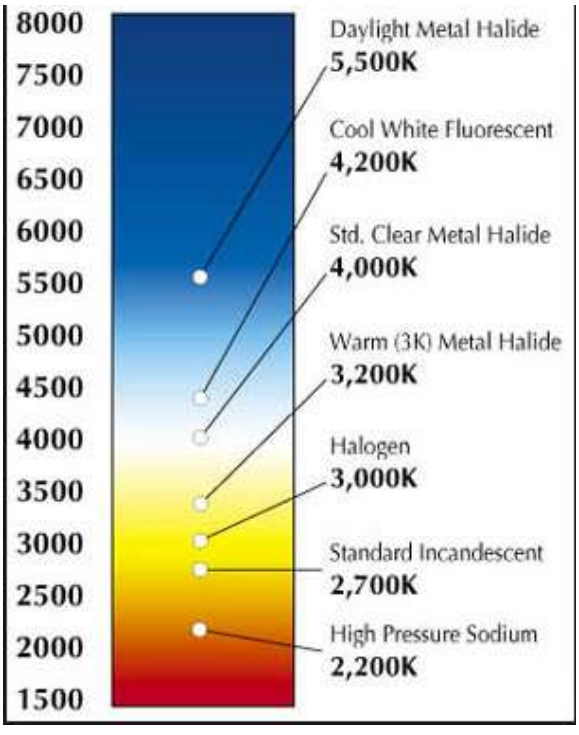

Figure 7 Temperature color chart [7]

Some LED headlight upgrade kits reduce their headlights color temperature by encasing their bulb in a color tinted film to change from a LED blue to a yellow. The third problem is the higher cost of LEDs. This problem may not continue for long though, since the price of LEDs is on a downward trend. As this continues, it is also expected that more products will begin incorporating them.

The last problem with LED headlights is the lack of heat that is emitted by the diode as a form of infrared radiation. To be clear, while LED circuits are more efficient than halogens, they do still produce heat, and that is why a heat sink is a necessary component of the LED headlight. Due to the lack of heat being emitted towards the lens of the headlight, ice can buildup on the front of the lens and that build can reduce driver visibility (Figure 8). Rigid Industries makes a heated LED headlight for Jeeps that turns on the heating element when the outside temperature of the car is below 50°F [8].

Figure 8 Snow accumulation on LED headlight [1]

But before examining how to detect and reduce ice buildup on the lens, it is important to first understand the factors that lead to the formation of frost and ice buildup in the first place. Temperature is a key part in the natural formation of ice. Whenever the ambient temperature falls below 32°F there is a chance of ice formation. However, due to a car's inability to retain heat, ice

can still form in patches at temperatures as high as 42°F. Another important factor in ice formation is the presence of water. Humidity, slush, rain, and snow are different conditions that can coat a headlight in water that later forms into ice. Wind speeds are another factor in the formation of ice. The stronger the winds the more difficult it is for air to condense and form a layer of ice on the car [9]. Therefore, a car traveling at a faster rate of speed has more air acting on it, making it more difficult for ice to form.

2.4 Ice Detection Methods

To run the system more efficiently, the ITO film only needs to be active when ice is present. To accomplish this, the team aimed to use a sensor to detect the presence of ice on the surface of the lens. The team then began researching all methods of ice detection that currently existed, to determine the best sensor for the project. During this process, the team contacted a technology developer called IceSight, whose main area of expertise was in ice protection systems.

2.4.1 IR sensor technology

One method for sensing ice buildup, developed by IceSight, uses IR technology to sense ice on a roadway surface. As Figure 9 shows, the IR sensor works by shooting an IR beam at the road. Each road condition: dry, damp, and wet, will reflect the light differently, so by measuring how much is reflected back, the sensor is able to detect different types of hazardous driving conditions.

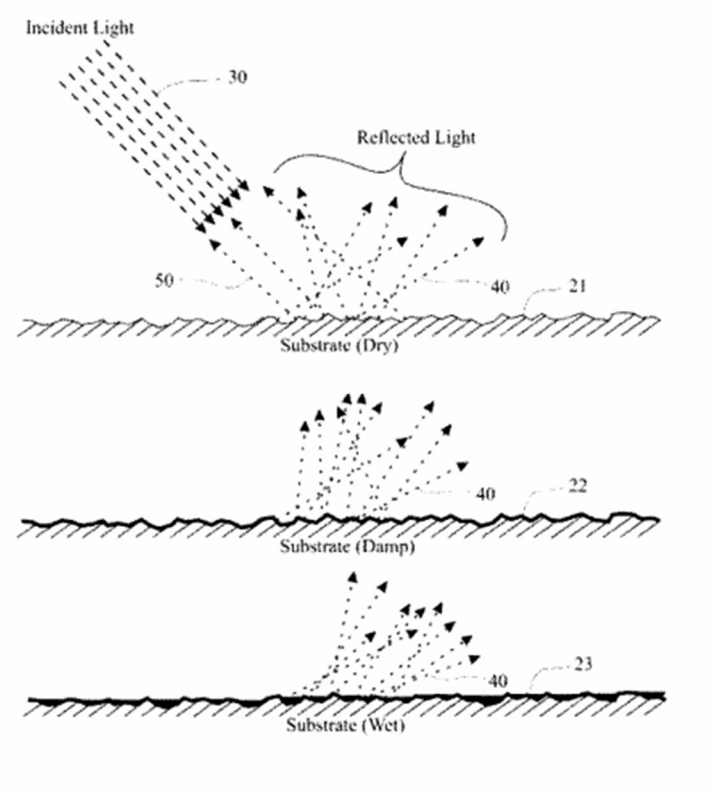

Figure 9 IR beam sensing [10]

The limitation of this specific device was that it needed a solid surface to reflect most of the IR beam back to be able to sense the differences in surface moisture. Since the headlight ice detection system needs to be able to sense ice obstructions through glass and polycarbonate, and this device is only designed for use on asphalt and concrete, it was not the best option for this project. However, IR sensing was still considered as a possible solution throughout the project.

2.4.2 Capacitive sensor

Capacitive sensors operate by sensing the change in charge between two plates, based on the material between them. They operate by reacting to changes in capacitance due to the inclusion of materials in between the two capacitive plates. Each material has a unique dielectric constant, which varies based on the material's conductivity. Since the characteristics of water are known, a capacitive sensor could sense whenever there was any water/ice obstructing the lens.

2.4.3 Microwave Interferometer sensor

While there are plenty of plausible options for surface ice detection, there are even more systems being developed to detect ice in many other applications, which may one day change the way ice is detected electronically. For example, at Baylor University they are developing a Microwave Interferometer sensor for use in aircrafts, which will be able to determine the volumetric and mass fraction concentration of either water or ice crystals within a cloud [11]. This application does not help with detecting ice on a given surface, but is still an interesting new technology being developed in the field.

2.5 CAN Bus Research

CAN bus is one of the systems used by US, and European cars to send information to all the components. Connecting to CAN bus gives the heated headlight the benefit of having access to information from other components on the system and also being able to give access to other components. This allows for greater control over the Headlight heater and the ability to be easily integrated into more advanced systems in the future.

2.5.1 CAN bus overview

Created in 1986 by Robert Bosch, it was originally designed to reduce the amount of wiring required in producing automobiles. It reliably sends messages and can be modified; these factors have led to becoming one of the standards in the US and is the only standard used in European cars. The use of CAN bus has even moved on to be included in boats, trains and also in industrial controls such as conveyor belts [12].

CAN bus architecture varies between iterations but the basic concept remains the same. Most CAN bus designs use two lines to carry data through the device: a CAN-H, and a CAN-L. The differential between these two lines is 2V the dominant state (logic low) and is 0V for the

recessive state (logic high). A typical 5V CAN transceiver's has the CAN-L line go from 1.5 to 2.5V and the CAN-H line go from 2.5 to 3.5V. The CAN controller is the device that handles all of the CAN protocols and acts as the middleman between the micro controller and CAN transceiver. One of these protocols is the arbitration process, which is whatever message ID had the highest priority gets to transmit first. Then if two or more messages share the same ID they transmit their message until each device "shuts up." A device is actively talking when it is in the dominant state (2V differential) and when the other is in a recessive state it quiet. As soon as one device speaks over another device, the quiet device waits for the other to finish its message before transmitting its. The next part of the CAN bus protocol is its message structure (Figure 10) which includes synchronization, multistage error checking system, and indication that a message has finished.

Figure 10 CAN bus message structure [2]

The CAN bus message structure is broken into 7 categories. The first category is the Start Field (SF), it is always dominant to show the start of the message. The second part is the Message Identifier indicates the priority of the message, the lower the value the higher the priority. The Control field has information about the number of fields that are in the Data field and it used by other receivers to check that they received all the information. The Data field contains all the information the transmitter wanted to share. The next field is the Cyclic Redundancy Check (CRC), it is message that is used for error checking. The acknowledgment (ACK) field is where the receivers of the message flip the bit to indicate that they have received the message. The End Field (EF) is the last error checking field and indicates the end of the

message [2]. The use of dominant states prevents messages from colliding and disrupting each other, instead the message with the highest priority (the most zeroes or maximum voltages) will supersede the lower priority message. With this hierarchy in place, CAN bus is very stable in terms of timing, as there is a very low chance for a message to be lost due to collisions with other messages.

As shown below in Figure 11, all components in the CAN bus system are connected on these two lines CAN-H and CAN-L. Due to this, instead of having components send messages directly to another device, each component in the system will receive each message sent from anything connected to CAN bus. A resistor (usually 120 Ω) at each end of the wire prevents echo of the signal from coming back to the transmitting device. Sensors and other components are not connected directly to these two CAN bus lines, instead there is a CAN Transceiver unit that deals with receiving and transmitting the signal on the CAN Bus line, and a CAN Controller that deals with taking the signal and interpreting what the device needs to do, as well as send out any commands to other modules.

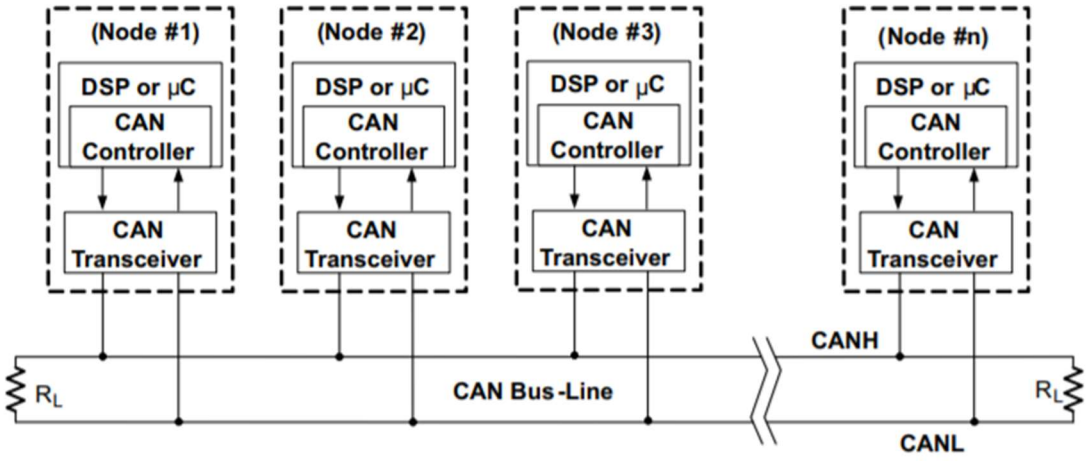

Figure 11 CAN Bus system [13]

An Important detail to mention is the power for both the system itself as well as the modules connected to it. The sensors and devices hooked up to CAN Bus do not receive power directly from CAN bus, instead they are powered off of the car's power circuit. The electrical generator or the battery of the car are used to drive this circuit which then modulates the electricity down to the required need of the component. CAN Bus itself requires power from the power circuit to run the transceivers, as they have to send out and receive signals along the wire. A typical system uses 5V for the transceiver components to run, but there do exist lower power 3.3V systems that significantly lower watt consumption.

Another important component now common in CAN bus systems is the ECUs or Electronic Control Units. These computer devices contain the logic required by the components in the system to function. Throughout the CAN bus wiring in the car there are several of these units to monitor performance of other module or control their use. A good example would be automatic doors in cars. The door itself is powered by the car's electric generator or battery, but the actions it takes are controlled by an ECU; it prevents the door from operating in unsafe conditions such as when the car is moving, or an object is detected in the way of the mechanism. To accomplish this task of directing the other components, the ECU connects to and communicates to other sensors and modules over CAN bus. Using the information, it receives, it can then transmit commands to drive the car components. Altogether, this system of separate but connected car computer devices form what is commonly referred to as the car's computer system.

2.5.2 OBD II - On Board Diagnostics II

OBD II is the computer diagnostic standard found in all car on-board computers sold in the United States. It is a computer system that allows for the user of the car it provides error codes and sensor data (in real time). An example could be a bad air filter for air codes or an 02 sensor Helps describe fuel mixture). The user would plug in a scanner tool to the OBD II and request information on the component, and the OBD II would communicate with the correct module through CAN bus and return the information. While this system is connected to the CAN bus system, this does not mean CAN bus uses OBD II. Instead the computer connects to the CAN bus system and obtains the requested information from the user. This system allows for cars to run different wiring protocols, such as different forms of CAN bus or even other systems entirely, with wildly different communication systems, that can all be accessed by the user through the same interface.

3 Methodology

This section is separated into the three different phases of development of the IR ice sensor. The next section focuses on testing CAN bus connectivity and assembling the final prototype. The final section goes into the steps of testing the final prototype.

3.1 Phase 1 IR Obstacle Avoidance Sensor Module

Upon reviewing the many possible methods, IR obstruction sensors were chosen first as the possible method for sensing ice on the lens exterior. These sensors differed from the prementioned IceSight sensor, because they emit a scattered beam of IR from a simple LED, and only measure the presence of objects, instead of emitting a focused beam of IR from a laser, which required a lot more precision and would likely have had issues passing through the lens. To determine the viability of IR sensing ice buildup on the surface of a lens a few tests needed to be performed.

An initial IR test was attempted using an IDUINO Reflective IR obstacle detection/avoidance sensor. This device emits IR waves from an LED and measures the amount of IR received with an IR phototransistor placed beside the emitter. This product was designed specifically for connectivity to Arduino boards like the one used in this project. When an object is present near the bulb, IR is reflected back, and the sensor is able to detect the beams. When power is provided to the sensor, an indication light is activated, as seen in Figure 12. Then when enough IR photons are reflected back, a second indication light is activated signaling the presence of an object, as seen in Figure 13. Generally, IR passes through glass and polycarbonate like the front lens of the headlight, so the lens will not obstruct the beam. This system was slightly more sophisticated than the minimum system needed to detect ice, however having a pre-constructed, pre-tested system was imperative since this experiment was intended as a proof

of concept rather than a final solution. In theory, if ice reflected the IR back, then a cheaper and more customizable system could be built for detecting the presence of ice on the surface of the lens.

For testing whether or not this approach was valid, the sensor was initially tested for basic functionality, and then placed into the mock headlight enclosure created by the previous MQP team in 2016-2017. From within the enclosure a series of tests were run to determine whether using an IR sensor was in fact a valid approach for detecting ice on the surface of the headlight lens.

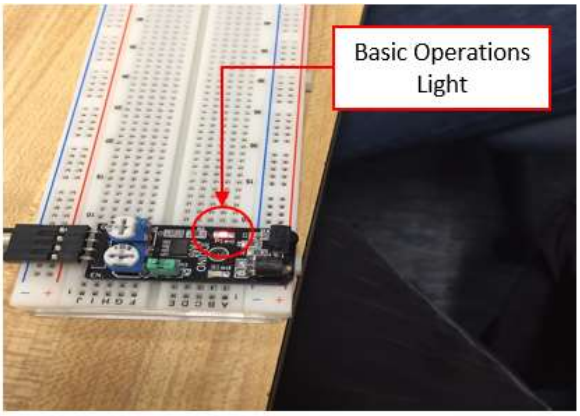

Figure 12 Basic operations light

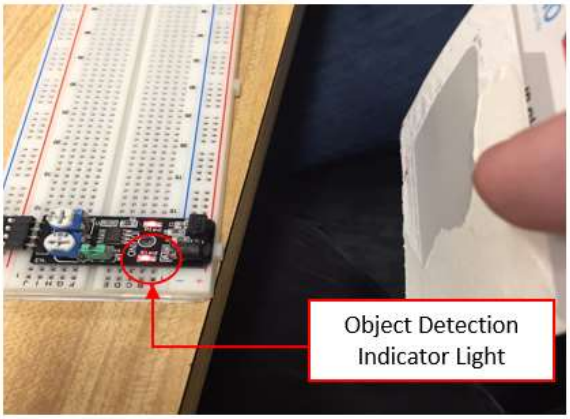

Figure 13 Object detection Indicator light

3.1.1 IR sensor basic functionality testing

Experimentation of the IR Obstacle Avoidance Sensor began by first ensuring the device received was operating as expected, and by gathering some information on how it functioned. It was discovered that higher voltage yields more accuracy (greater indication distance). The operating current was then tested by setting up the digital multimeter in series with input voltage from the DC power supply into the sensor. Current and indication distance were then both measured with the device inside of the headlight enclosure to get some data from the sensor. The results are tabled below.

Table 2 Initial IR testing values

Voltage	Current (mA)	Indication Distance (cm)
3.0	0.0001	2
3.3	0.0001	5
3.5	0.0001	7.3
3.6	0.0001	7.5
3.7	0.0001	9.1
3.8	0.0001	22.8
>3.8	0.0001	Detects ITO Film

This data shows that the voltage input increases the sensitivity of the device. If a voltage of more than 3.8 V is applied to the sensor, it ends up detecting the ITO surface, leaving the indication light always active. This was because the current passing through the LED increased as the input voltage increased, which in turn caused the intensity of the LED's output to increase.

More IR being emitted meant that more IR was being reflected off of an object, which lead to a more sensitive system. It was then noted that varying the input voltage affected the sensitivity of the device. The voltage needed to be adjusted to the point that the sensor would no longer be activated by the ITO and still be to detect ice buildup on the exterior of the lens.

3.1.2 IR sensor ice cube experiment

A similar experiment was then run, using ice cubes on the surface of the lens. During this experiment, Ice cubes of roughly 3/4 square inch area ice and about 1/4 inches thick were used to simulate surface ice. The starting value of 3.3 V was found to be too low to sense the ice cube. The next test voltage was 3.5 V, and as Figures 14, 15 and Table 3 show, the sensor was sensitive enough to pick up the presence of ice. During experimentation, it was perceived that the clarity of the ice played a big factor. If the ice was perfectly clear, then the sensor was not able to detect it. This would not be a problem in frost/ice buildup on a headlight however, since perfectly clear ice does not build up on cars, and even if it did, then the light of the headlights would not be obstructed and there would be no need to melt the ice anyway.

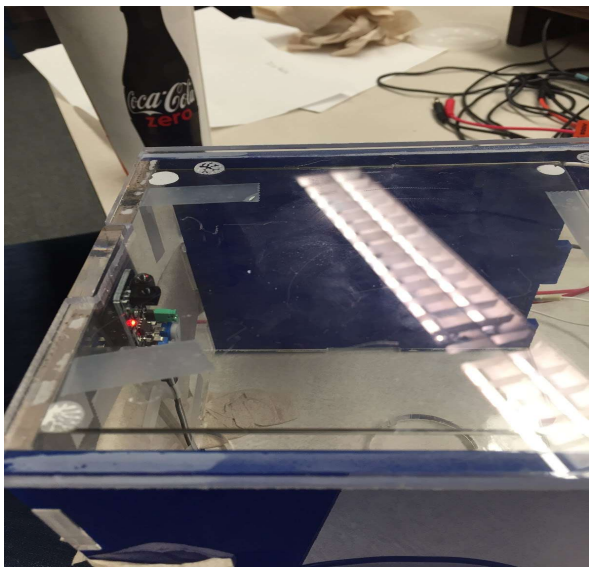

Figure 14 IR Ice testing setup

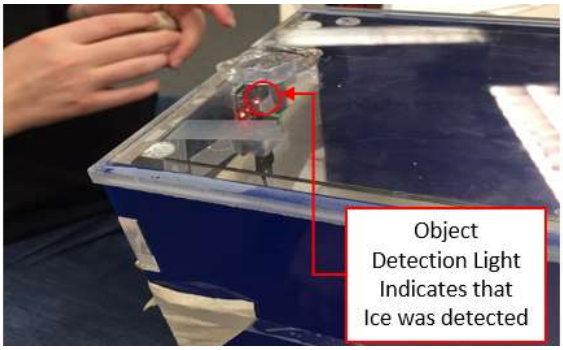

Figure 15 3.5V IR ice cube test

Table 3 Ice cube IR sensor test

Voltage (V)	Ice detected?
3.3	No
3.4	Yes
3.5	Yes
3.6	Yes
3.7	Yes
≥3.8	Detects ITO

3.1.3 IR sensor fogged lens experiment

The next test performed was to simply fog up the lens by breathing on it, and visually inspect whether the IR sensor could detect that presence. As Figure 16 shows below, this assumption turned out to be correct, meaning that as long as the lens was obstructed by something cloudy enough, and which reflected IR, the IR sensor would be able to detect its presence. Overall, these tests yielded very positive results, since they proved that IR sensors could detect the presence of ice on the front of headlight lens. Following these tests, IR sensors were accepted as the primary method for detecting ice on the lens exterior.

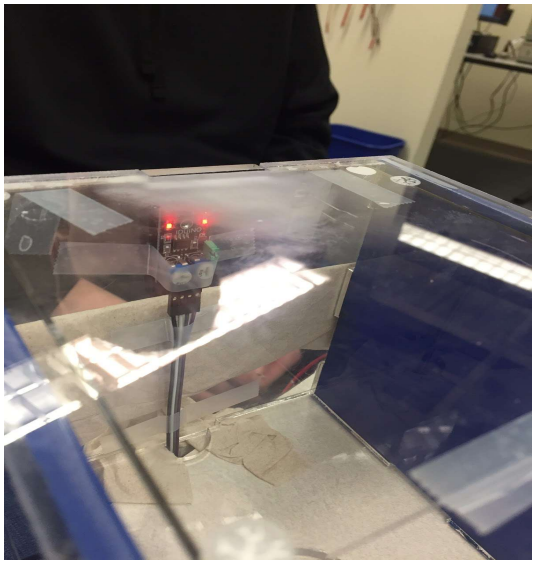

Figure 16 Fogged lens test

3.1.4 What causes IR to sense ice/fog

The next stage of the design process was to understand as much as possible about how IR passes through or is reflected by ice. There are three different varieties of IR: Near IR, Mid IR, and Far IR. This sensor uses Near-Infrared, which passes through water instead of being absorbed. What makes ice block visible light, is not the ice itself, but rather the impurities within it [14]. These impurities generally reside towards the center of ice. Water free of minerals and impurities always freezes first, which pushes air bubbles and contaminants toward the center [15].

IR passes through ice differently than visible light, based on its properties, impurities, and what spectrum of IR is being used. Generally, IR beams are reflected by electrically conductive surfaces, such as aluminum, and absorbed by nonconductive materials such as plastics. The IR bulbs being used in the ice detection system emit near IR waves. As Figure 17 shows below, near IR beams like the ones emitted by the IR LEDs in proximity sensors, will pass through the water and ice, and reflect off of the contaminants, thus triggering the transistor, indicating the presence of the ice.

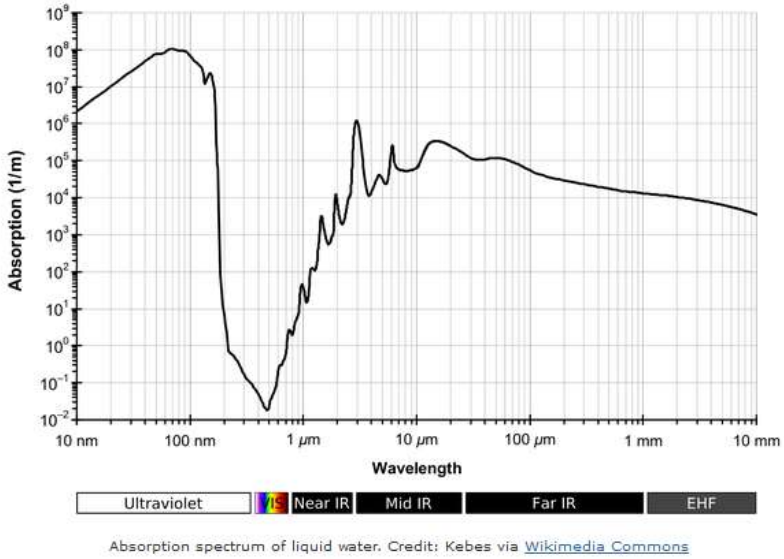

Figure 17 Light absorption of water given different spectrums [14]

3.2 Phase 2 Custom IR transistor sensor

The next phase of the project was to develop a sensor that could more efficiently detect ice, and could detect ice on multiple points along the headlight surface by using a more conservative design and number of components. To accomplish this, SUNKEE IR Transistors, and IR LEDs, along with a Texas Instruments CD4071BE quad two input OR gate chip were used as components in order to create a circuit that would pass a voltage signal whenever IR light was reflected off of the ice. The original design incorporated four separate sensors found midway along the edges of the front lens as seen in the figure below.

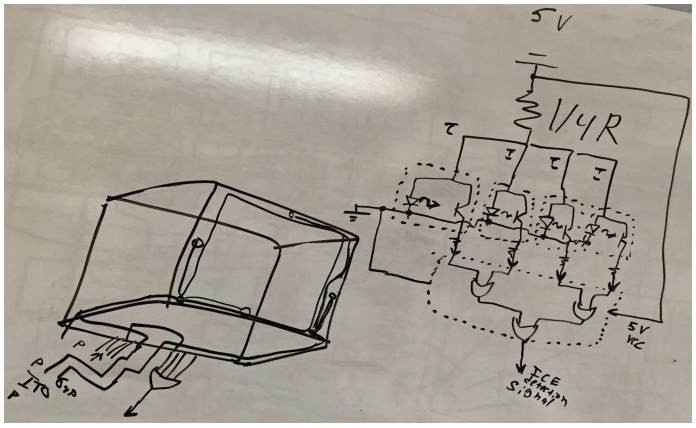

Figure 18 Mock circuit diagram for new IR ice detection system

3.2.1 Basic surface detection test

Once the components arrived, they were assembled into a simple ice detection system, and tested under similar conditions to the original IR sensor. The system was first tested using a simple solid surface, to determine if the circuit could detect an object within its specified range. When designing the test circuit shown in Figure 19 below, design ratings of the IR transistor and LED were used when choosing resistors, while keeping in mind that there would be a 0.7V drop across the two LEDs. Thus, the voltage drop across the resistor which influenced the overall current would be 4.3V instead of 5V. Knowing this, resistors values were estimated in order to create the currents close to the rated currents for the components of 60 mA for the LED and 100 mA for the Transistor. For the first wave of testing, resistances were increased by a factor of 10, to see if a lower current could be used for the circuit.

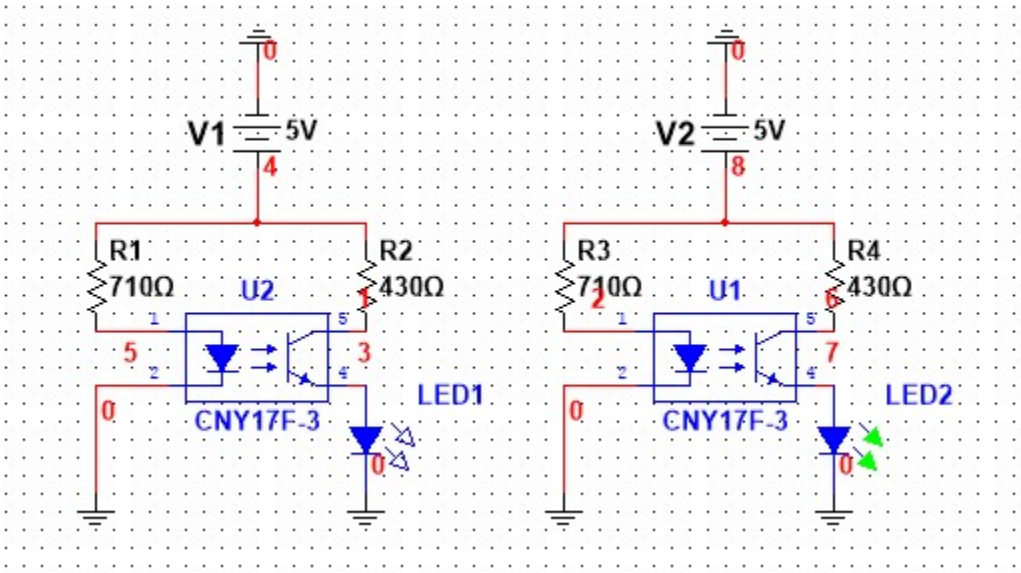

Figure 19 Test circuit used

Upon applying voltage to the system, shown in Figure 20, the circuit was immediately able to function as planned. It was first perceived that the transistor was able to detect the proximity/opacity of an obstruction based on the amount of voltage it allowed through the transistor. As the results in Table 4 show, the voltage across the transistor gradually increased as

an object moved further away from the transistor, and voltage across the indicator LED gradually became lower. Also, as objects moved away the voltage dropped across the resistors fell as well, meaning that the current traveling through them was being lowered by the transistor.

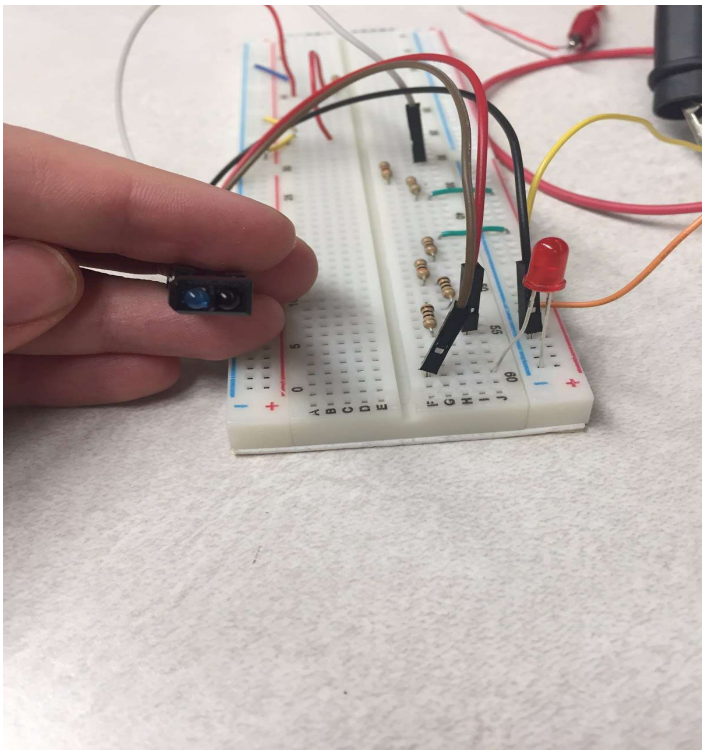

Figure 20 Testing proximity of object detection

Table 4 Transistor voltage testing data

Input Voltage (V)	Sensor distance (mm)	Detected?	Voltage across transistor (V)	Voltage across Resistors (V); (420 Ω)	Voltage across indicator LED	Total (V); (calculated)	Current (mA); (calculated)
5.1	direct obstruction (0)	Yes	3.19	0.25	1.83	5.27	0.5952380952
5.1	10	Yes	3.22	0.14	1.81	5.17	0.3333333333
5.1	12	Yes	3.25	0.11	1.8	5.16	0.2619047619
5.1	14	Yes	3.27	0.09	1.8	5.16	0.2142857143
5.1	16	Yes	3.3	0.07	1.79	5.16	0.1666666667
5.1	no obstruction (500+)	No	3.37	0.02	1.75	5.14	0.04761904762

3.2.2 Testing through the mock headlight lens

Following the initial experiment, a basic qualitative test was then performed, in order to determine if the new sensor could still detect objects on the outside of the headlight lens. The initial testing resistances were also replaced with lower values of 96 Ω for the IR LED, and the 43 Ω for the IR Transistor, in order to induce a higher current through the system. These resistance values were achieved by placing a set of resistors in parallel and series to create those specific values as seen in the circuit diagram shown in Figure 21. This was done to increase the sensitivity of the device. 43 Ω was chosen with the expectation of producing the maximum amount of current through the transistor, being ten times smaller than the original test circuit. However, 96 Ω was placed in the circuit by accident, when attempting to create 71 Ω. A 51 Ω resistor was intended to be placed in parallel with two 100 Ω resistors, however a 510 Ω resistor was unintentionally placed instead. This mistake was discovered following this experiment, and was immediately fixed for future tests. However, this did not affect the outcome or results of this experiment, since the purpose of this test was to learn more about how the components were functioning within the system.

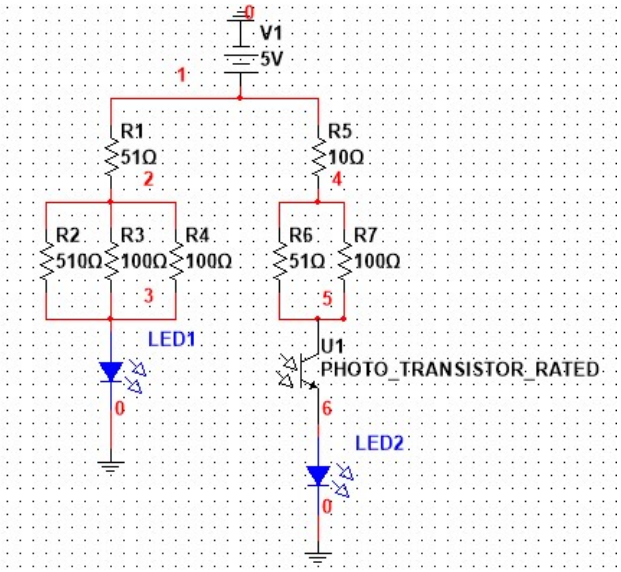

Figure 21 Circuit diagram for IR system for ice detection

As Figure 21 shows, The LED was always on, meaning a slight amount of IR light was being reflected off of the ITO film and headlight lens, however the LED became brighter as objects came closer to the transistor.

Figure 22 Testing within the headlight enclosure (obstructed and nonobstructed)

A second test of the LED/transistor components was then run, using the same resistance values, but without the headlight enclosure, as seen in Figure 23. In this new situation, the LED had a more noticeable change when an object moved closer. LEDs would then remain mostly off when no objects were in range.

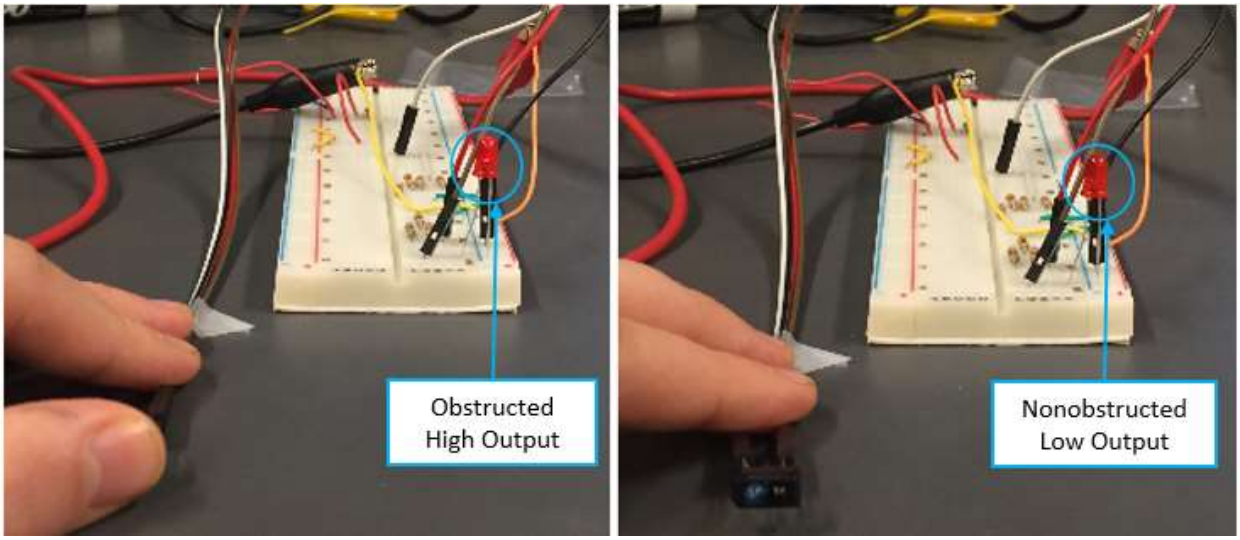

Figure 23 Testing outside of headlight enclosure (obstructed and nonobstructed)

3.2.3 IR Transistor sensor circuit analysis

Upon perceiving the slight difference in LED brightness and overall current output, a full analysis of the system was run both inside and outside of the enclosure. The results, seen in Tables 5 and 6 below, show that when placed inside of the enclosure, the greatest change between the presence or absence of an object actually occurs within the current across the transistor line. This makes sense, considering that the transistor will be limiting the current through the system. As Figures 24 and 25 show, the presence of the enclosure lens plays a major role in the current passing through the transistor, since the enclosure reflects some of the IR back the current increases when the sensor is placed in the enclosure. Regardless, these figures also show that there will be a decrease in voltage of about 0.20 volts when an object is present. This means that when designing the system for detecting obstructions such as ice, the current limiting factor of the transistor will be the element that needs to be measured. This will be done by first converting the current change to a measurable voltage change, then using the voltage difference to detect the change in current.

On a separate note, it was also proven through this analysis that the IR LED line of the circuit remains unchanged regardless of the presence of the enclosure, or any other obstruction. The current through 38.5 mA and 1.272V.

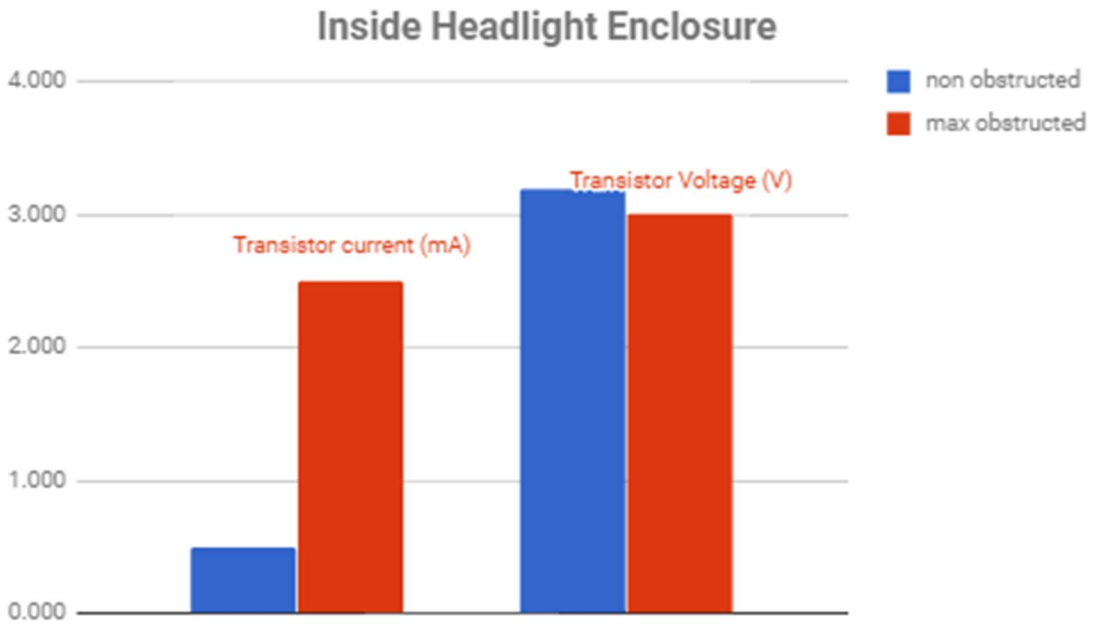

Figure 24 Effect of an obstruction on the IR phototransistor inside the enclosure

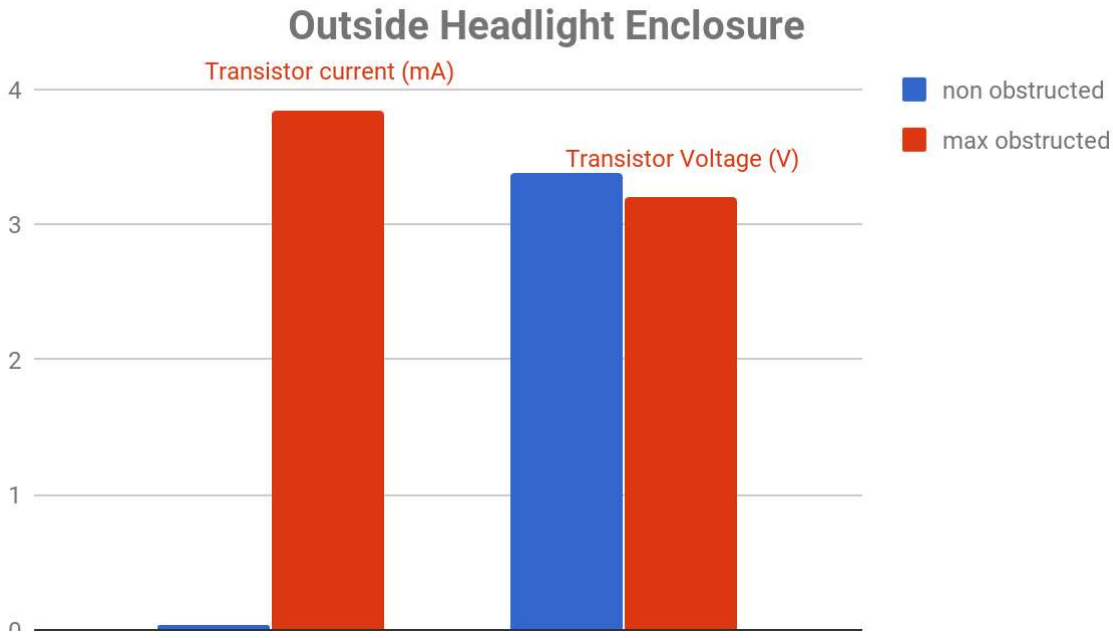

Figure 25 Effect of an obstruction on the IR phototransistor outside the enclosure

Table 5 Circuit analysis outside of the mock headlight testing enclosure

		Outside Enclosure			
	Transistor		**non-obstructed**	**max obstructed**	
	transistor current	(mA)	0.003	3.850	<- fluctuates (1 - 4.5)
	transistor voltage	(V)	3.38	3.20	
43 Ω	**transistor side resistor voltage**	(mV)	0.00	15.00	
	Indication LED voltage	(V)	1.64	1.82	
	Total Voltage	(V)	5.02	5.04	
	IR LED		**non-obstructed**	**max obstructed**	
	IR LED current	(mA)	38.50	38.50	<- all IR LEDs are within 0.2 V of this value
96 Ω	**IR LED side resistor voltage**	(V)	3.77	3.77	
	IR LED voltage	(V)	1.272	1.272	
	Total Voltage	(V)	5.04	5.04	

Table 6 Circuit analysis inside of the mock headlight testing enclosure

		Inside Enclosure				
	Transistor		**non-obstructed**	**max obstructed**	difference	
	transistor current	(mA)	0.500	2.500	2.000	<- fluctuates (1 - 3.0)
	transistor voltage	(V)	3.19	3.00	-0.19	
43 Ω	**transistor side resistor voltage**	(mV)	18.00	122.00	104.00	
	Indication LED voltage	(V)	1.82	1.89	0.07	
	Total Voltage	(V)	5.03	5.01		
	IR LED		**non-obstructed**	**max obstructed**		
	IR LED current	(mA)	38.50	38.50		
96 Ω	**IR LED side resistor voltage**	(V)	3.77	3.77		
	IR LED voltage	(V)	1.272	1.272		
	Total Voltage	(V)	5.04	5.04		

3.3 Phase 3 Ice Sensor Final Design

In the final phase of the ice sensor's development the energy efficiency of the system was increased. The connections between the Arduino Uno and the ice sensor was reduced. Lastly a MOSFET was added to the ice sensor circuit to allow the Uno to be able to control the power to the ice circuit.

3.3.1 The Quad-transistor design

Once it was determined that a single IR transistor sensor could detect ice buildup on a headlight lens, a system was developed for incorporating multiple IR LED/transistor modules for better coverage of the headlight lens. The system, shown in Figure 26 below, aimed to minimize the number of inputs and outputs to the Arduino and to keep the circuit as simple as possible. A 62 Ω resistor was placed in series with the LED line for testing, however since the LED's output plays an important role in determining the sensitivity of the device, the resistor was designed to be interchangeable, so that the device would be customizable to a variety of headlights and other applications.

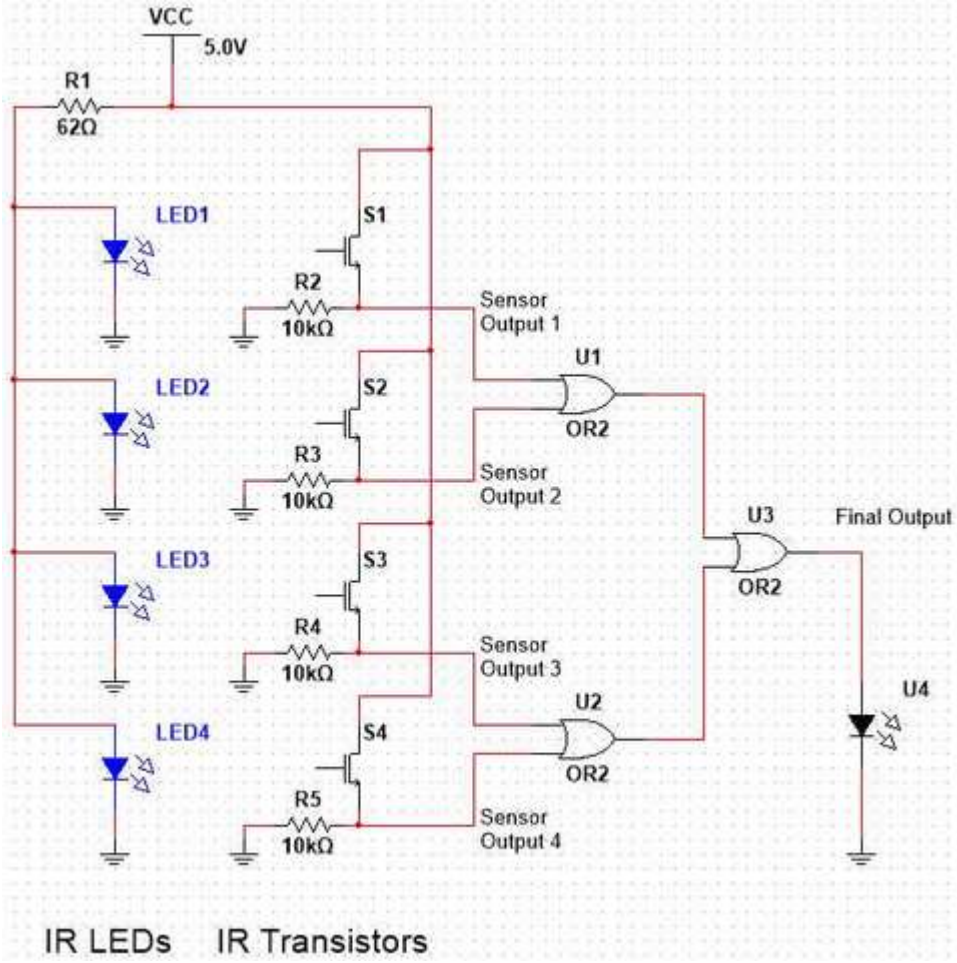

Figure 26 Quad-Ice sensor circuit diagram

Upon testing, it was discovered that the resistance preceding the IR transistor line had no effect in the overall output of the sensor, so that resistance was removed from the system. Finally, a resistor was placed after each of the IR transistors, in order to be able to measure voltage outputs off of each transistor. Those outputs were then connected to a series of OR gates in order to create a single output, indicating that ice was detected by any of the four sensors. For the construction of the OR gate system, a Texas Instrument (TI) CD4071BE OR gate chip was used. This chip, seen in Figure 27 below, was selected because of its 5V power requirement. This meant that the OR chip could powered completely off of a 5V source, negating the need for an additional power supply.

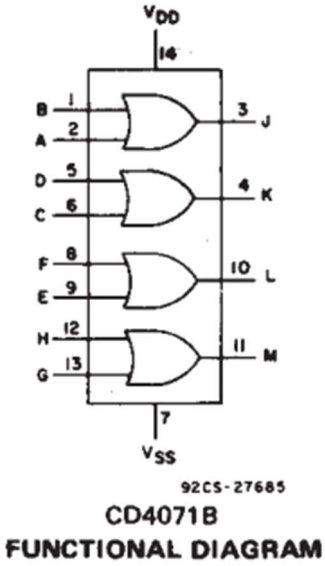

Figure 27 TI CD4071BE OR gate layout [16]

Upon constructing the test circuit on a breadboard, it was confirmed that the prototype design could produce the necessary output when any of the IR transistors were triggered. Without an indication LED, the final output at the end of the OR gate was 5.06 V, as seen in Figure 28. With a red indication LED, the output was 2.00 V. Since these voltage outputs could be read by the Arduino, this means that the prototype sensor will be compatible with the digital side of the project.

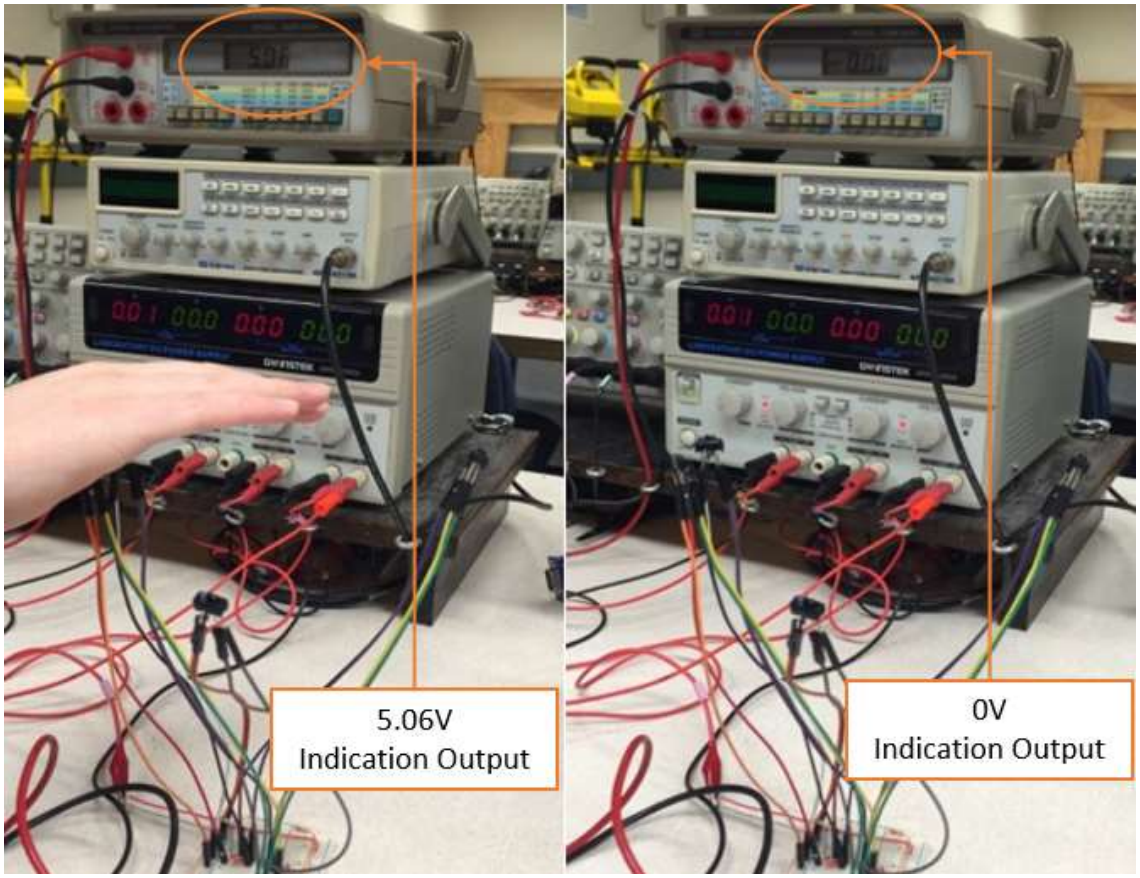

Figure 28 Quad-Ice sensor; obstruction vs. non-obstruction

3.3.2 Improving system efficiency

Finally, to improve upon the overall system, two main changes were made. First, a capacitor was placed between the power input and ground pins of the OR gate chip, in order to reduce noise from the input power. Second, as shown in Figure 30 below, the system was altered, so that instead of having four LEDs in parallel (Figure 29), a system of two parallel lines of two LEDs in series was used. Since each LED has a voltage drop of about 1.8V then placing two LEDs in series causes a 3.6V drop. By implementing the new system, the resulting voltage drop across the resistor went from 3.2 to 1.4V. The resistive value was then changed to 53 Ω to have the same current flowing through each LED (13 mA). When this new setup was implemented the current into the system went from 70 to 40 mA, which is almost half the power of the previous system.

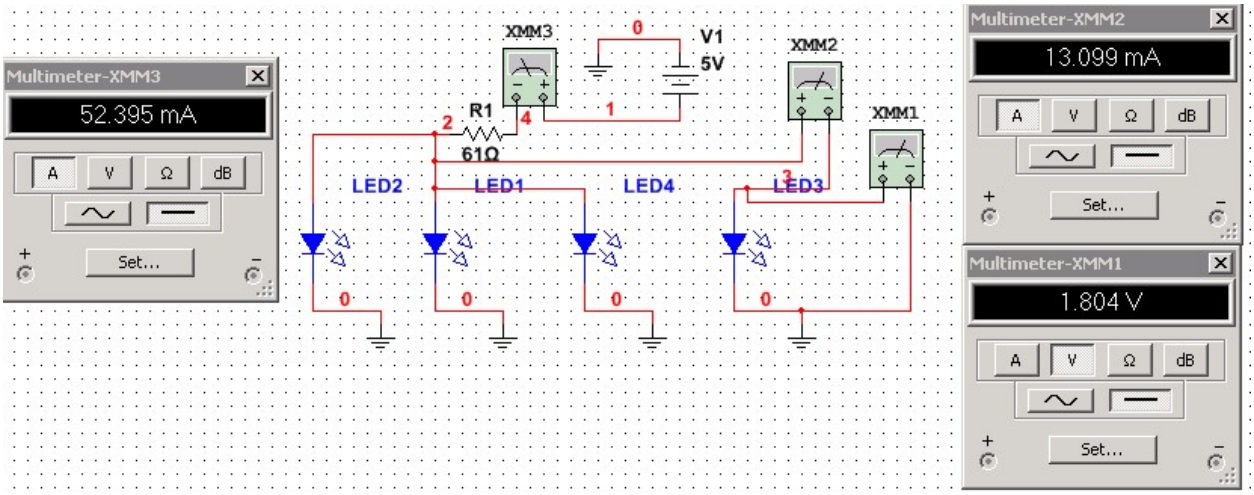

Figure 29 Initial design and power simulation

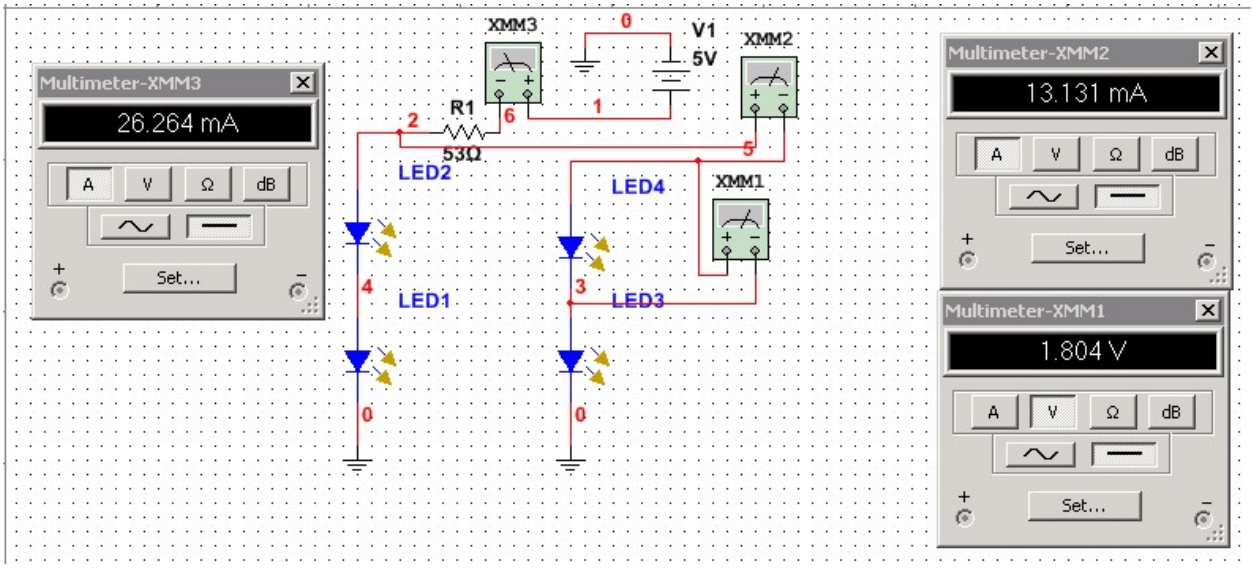

Figure 30 Secondary design and power simulation

3.3.3 Implementing the sensor on a PCB

Once the design was completed, and the team was satisfied with the results, the design was then converted into a PCB layout using the Express PCB program. The final PCB design, shown in Figure 31, utilizes two layers (red is top and green is bottom) to control the ice sensor, the ITO, and provides 8 four-pin terminals to connect/power multiple devices. The fully assembled PCB can be seen in Figure 32.

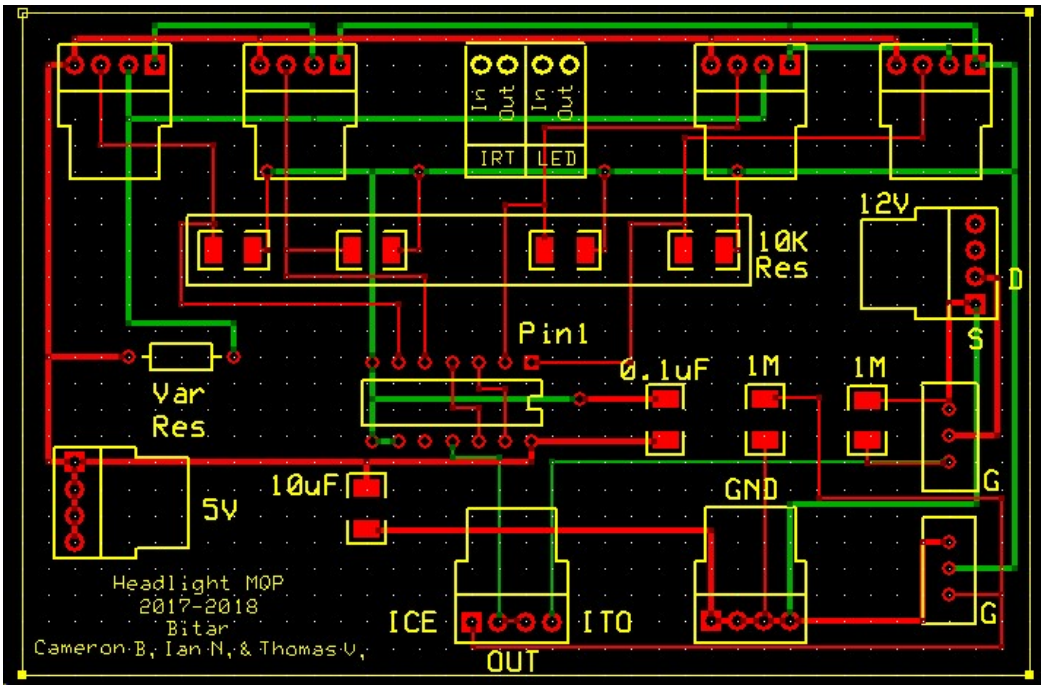

Figure 31 PCB design

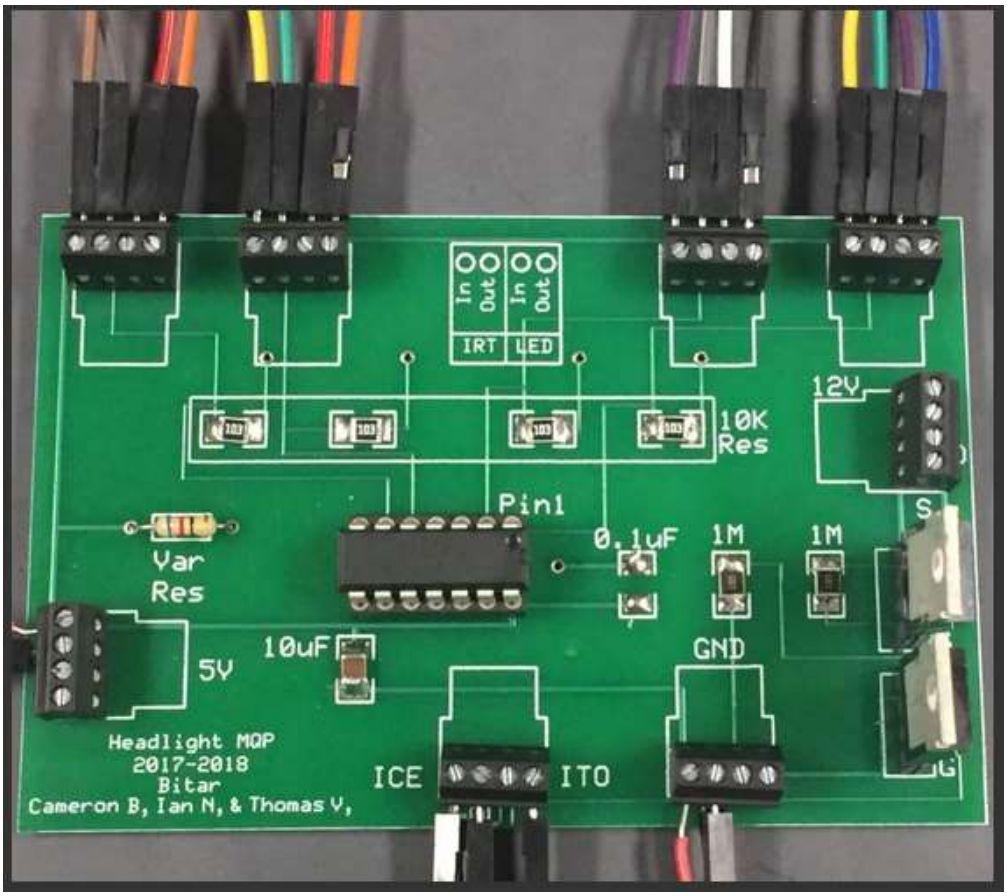

Figure 32 PCB final

3.4 CAN Bus Testing

In this section the team developed a series of test to verify that a CAN bus system was created. The first system uses an Arduino Uno with a CAN shield connected to a Teensy with a CAN transceiver. After discovering issues with the first setup, two Arduino Unos with CAN shields were connected to emulate a CAN system. This system can be seen in the figure below.

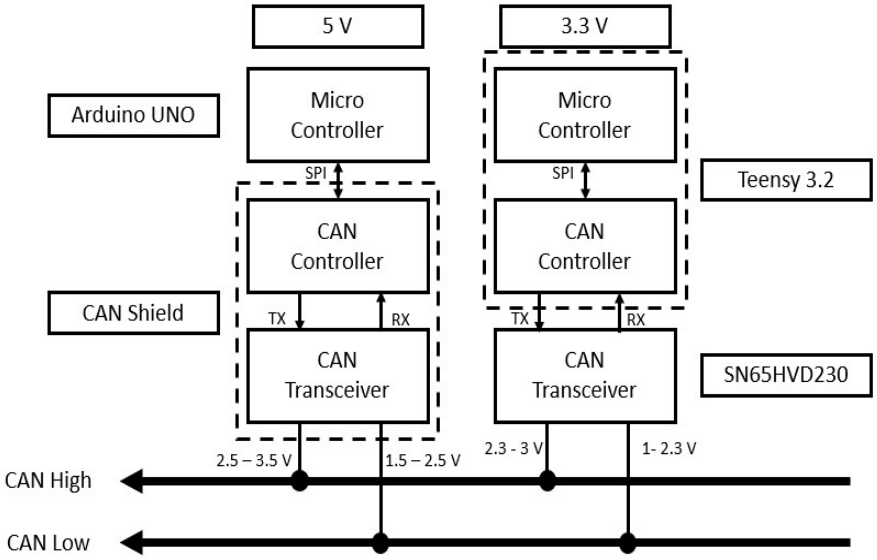

Figure 33 Diagram for first test system

3.4.1 Creating a CAN bus network: first attempt

After acquiring the initial component order for the CAN bus network and downloading a CAN bus library for Arduino with Teensy add-on, the first configuration was built to test if the two microcontrollers would be able to connect to each other, as seen in Figure 34.

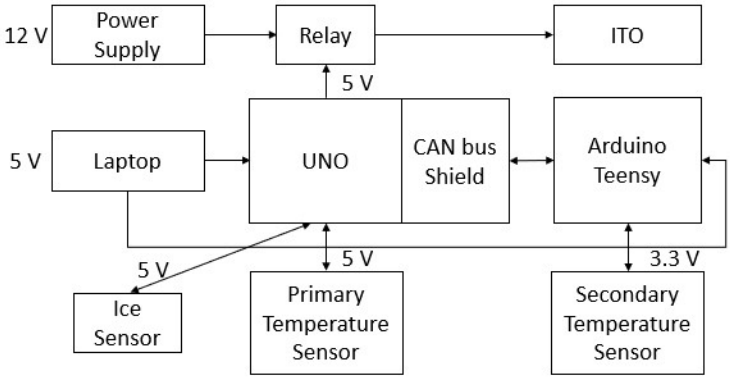

Figure 34 Experimental Design for setting up the CAN bus network

After each of the components were connected, and loaded up with the CAN bus programs, it was tested and ended up failing. At this point, a full examination of each component was required to debug the system, starting with the CAN bus Shield.

3.4.2 Testing the output of the CAN bus Shield

The first part to check about the CAN bus shield was if a signal was being produced, and if that signal matched up with CAN bus protocol. After using an oscilloscope to check the output from the Shield it was discovered that there was no output at all. The 2.5V coming from the CAN bus line was determined to be the default signal sent out by the CAN lines. After some research, it was discovered that one of the pins was improperly referenced in the code, the SPI_CS_PIN was set to 9, which on certain boards the SPI_CS pin is on the tenth pin. Without the CS pin being properly addressed then the master device cannot initiate any communication to the slave device. After this change, the board was able to output the signal from the CAN library test code. Now the signal itself, needs to be analyzed to determine if it fits CAN bus protocol. A signal outputted from the CAN Shield onto the CAN lines can be seen in Figure 35.

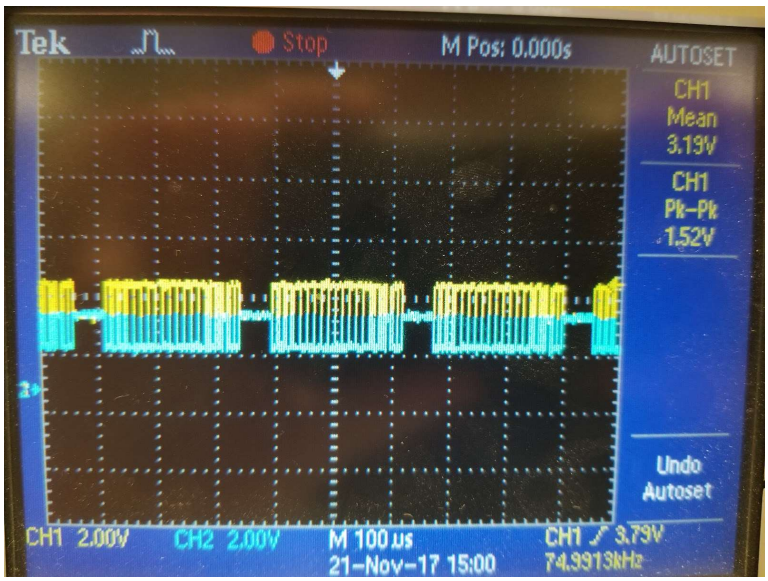

Figure 35 Output from the CAN Shield

A quick examination of the signal, showed that the two signals, CAN H and CAN L, are reflections of each other over the 2.5V line which is expected. The pk-pk of the CAN H and CAN L waves was around 1.5V for each. Initially the team had incorrectly surmised that this was incorrect as it was thought that the CAN lines would reach the high and low Voltage rails of 5V and 0V. However, after reviewing the datasheet for the transceiver, it was found that voltages on the lines matched the expected values.

3.4.3 Testing outputs of the Teensy

A discovery of two different issues had hindered initial progress in receiving signals from the Teensy. The first was the use of improper libraries to write to the Teensy. Due to this, the ports were not even being written to. The second problem discovered with the Teensy, although saying CAN bus compatible, was misleading in that it has a CAN bus controller on it, but it needs a transceiver to communicate on the CAN lines. The part was ordered to connect the Teensy to CAN bus. The CAN flex library that was needed to program the teensy was downloaded. However, being a different library from the CAN shield's one, some work was required to prepare the two devices to communicate properly.

After attaching the transceiver to the Teensy and doing some initial testing, it was found that the CAN bus line was now severely distorted. After some debugging, the problem was discovered to be the jumper. When the jumper was removed on transmitter, the signals on the CAN bus line returned and the team was able to view the output waves coming from the teensy. However, if we manually add 150 Ω (two 300 Ω resistors in parallel) the signal does not become distorted. The team theorized that it might be a problem in the traces, the board's schematic is misleading, and/or the wrong resistor could be in the circuit. It was determined that in the end, it was indeed simply shorting the CAN H and CAN L wires, as can be seen in the Figure 36.

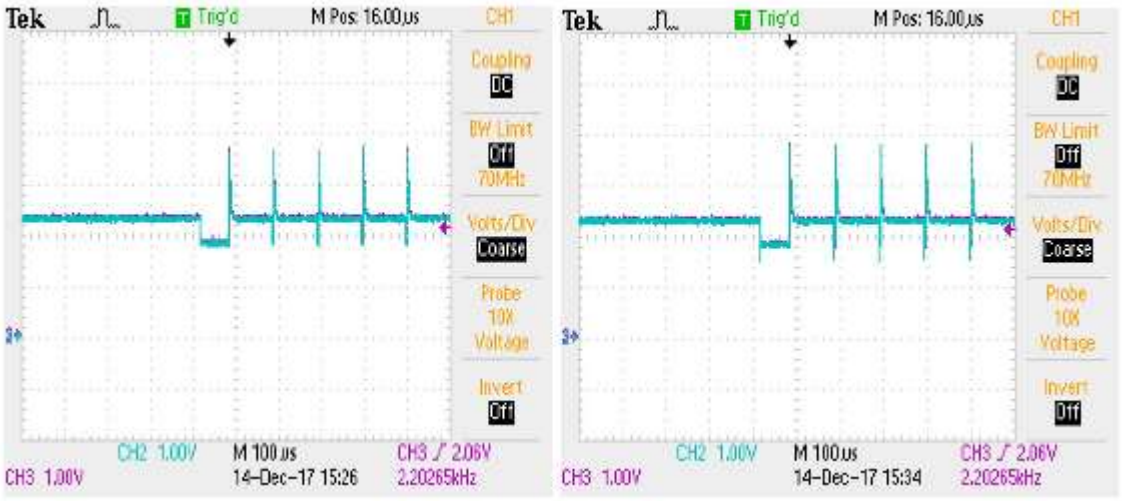

Figure 36 Teensy CAN bus Transmitter w/ jumper(left) & shorted CAN lines (right)

At this point, both the CAN bus shield and the CAN bus mini breakout board are transmitting the proper signals on the CAN lines. The CAN bus shield has 1.5 to 2.5V on CAN-L, and on CAN-H line the voltages ranged from 2.5 to 3.5V. For the mini breakout board, the voltages ranged from 1.0 to 2.3V on CAN-L and on CAN-H the voltages range from 2.3 to 3V. Both transceivers have a differential of 2V and their reference voltages are around 2.5V (2.5V for shield and 2.3V for mini). Since both transceivers are built to work with the standard 5V transceivers in traditional CAN bus systems and they are both performing properly according to their specs, the team believes that the transceivers are properly communicating to one another.

The next part to check in the system was the CAN TX and RX lines between the CAN controllers and the transceivers in transmission mode. When the module is in transmission mode the can controller passes in the message using a square wave from 0 to V_{cc}. From the shield the 5V square was successfully passed through to the transceiver and from the teensy a 3.3V square wave was outputted. Since the transceivers are also reading the message that is being transmitted to for arbitration purposes, error detection, and looking for the acknowledgment bit. Given these

reasons the RX line should be an exact copy of the TX line. For both the shield and the mini this was the case.

When the transceivers are in receive only mode, the TX line should stay constantly at the V_{cc}. When both systems are in receive mode the RX line is properly shows the differential on the CAN low and high lines. Also, both systems' CAN controllers are outputting Vcc on the TX line. The TX line should eventually flip the acknowledgement bit in the message to show that a node received a full message. However, since neither device has been able to collect a full message from the other there appears to be a problem between the CAN controllers.

As an extra step to verify the problem is with the CAN controllers, the SPI connection was looked at by on the CAN bus shield (CAN controller on teensy only has TX RX lines). The MCP2515 has two buffers where it can store a message based on the filters the user sets and one other buffer that is used to assemble messages. When a message is sent into the one of the readable buffers a flag for that buffer can be read to signal that there is readable data in that buffer. The microcontroller then continually polls for the flag by using the SPI connection. The first step is to have the microcontroller force chip select (CS) too low to indicate that the master device would like to communicate with the slave device, this can be seen in the figure below.

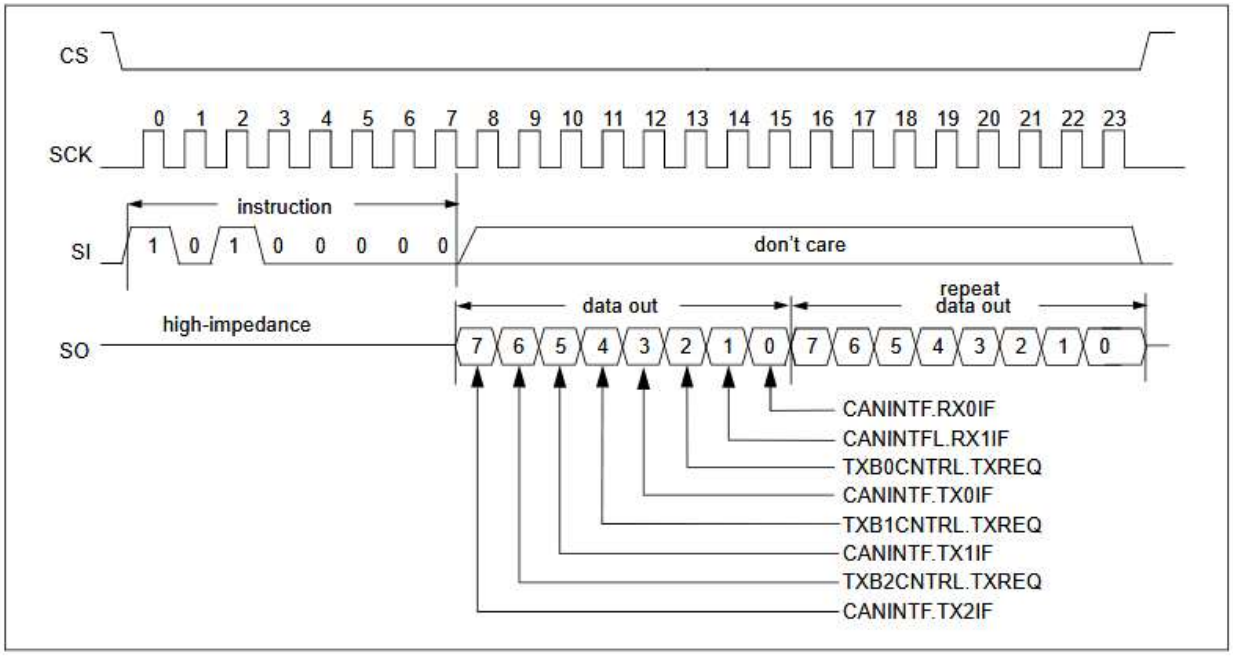

Figure 37 Read status instruction for MCP2515 [17]

The next step is for the microcontroller to pass in the instruction command and for the CAN controller to output data that shows which buffers have a message in them. When looking at these pins on the oscilloscope it can be seen that two devices are communicating properly, the output from the CAN controller indicates that it has not received the proper information to form a full message. The next things to explore are different modes on the can controller, different baud rates, and trying to get another can shield for testing.

When connecting two CAN bus shields together they worked perfectly without a termination resistor. This helps us confirm that there may be a compatibility issue between the Arduino system and Teensy system. To further confirm this some readings were taken on the Tx and Rx lines when the shield is receiving messages from another shield and when the it is receiving messages from the teensy. This test shows that the teensy is trying to transmit a message when it is in receive mode.

With the two CAN bus shields being able to transmit and receive messages across the CAN H and CAN L lines, code was developed for the two boards to model a Car ECU monitoring the external temperature and our headlight node shown below in Figure 38. The Headlight node would transmit a request message across the line to ask for a temperature and then begin listening on the line for a response. The Car ECU would be waiting for a transmission and upon receiving the temperature request, will take a reading from the sensor connected to it, place the reading into the message and then send it back to the headlight node that is waiting for it. While these tests demonstrate the ability to send and receive messages across CAN bus to communicate between two devices, further research into the wave will have to be done to determine whether other devices such as a car will be able to communicate with this system.

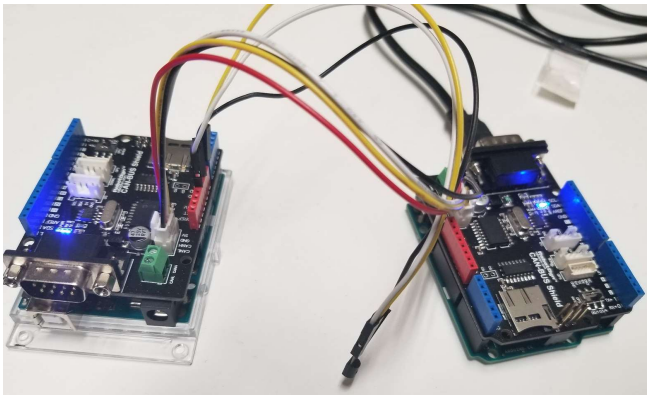

Figure 38 Two CAN bus shields forming a working CAN bus system

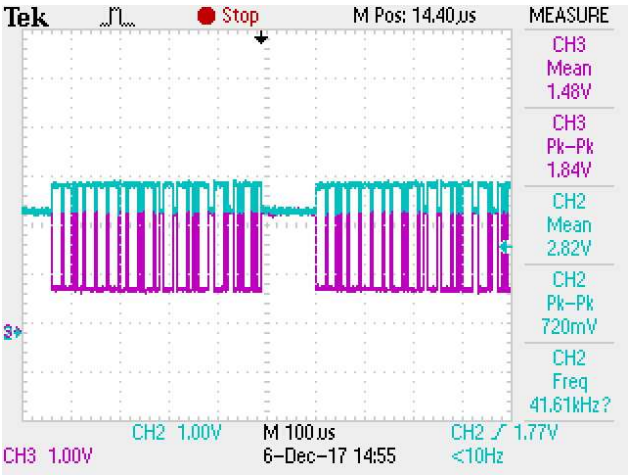

Figure 39 Teensy transmitting to Arduino (CAN High and Low)

3.4.4 Logic analyzer

To verify that the CAN bus shields were sending and receiving CAN bus messages, a logic analyzer was needed to take a look at the individual bits. Since there was not an external clock to connect to the logic analyzer the team had to use the internal clock and over sample the incoming messages. Another issue is that the shields do not have accessible CAN RX and TX lines. Therefore, a CAN transceiver was used to translate data from the CAN bus lines into a data stream for the logic analyzer. The setup for this is depicted below.

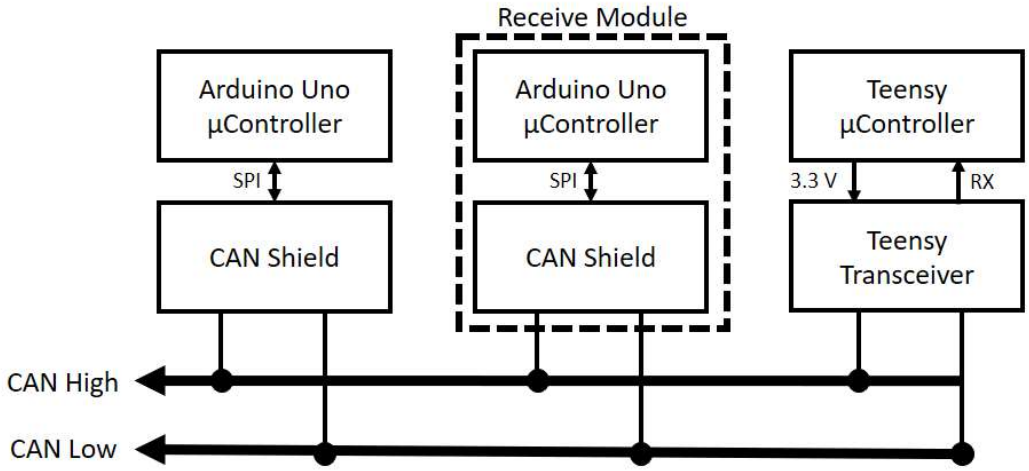

Figure 40 Depiction of CAN test circuit

In order to properly over sample the message the data must be collected at twice the speed of the transmission. The first step was to use the oscilloscope to figure out the bit rate. The Uno was coded to send a series of bits alternating between 0 and 1 in the data field so it would be easier to determine the bit rate. The bit rate is equal to about 526 kHz, as seen in the figure below.

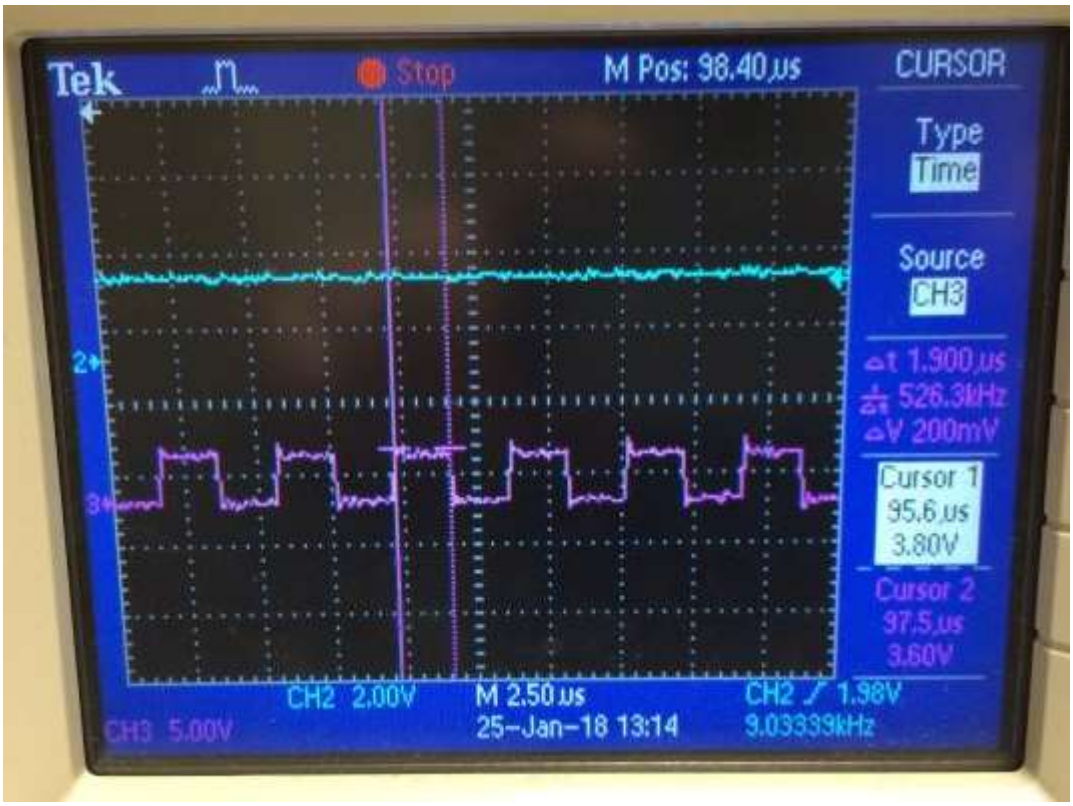

Figure 41 CAN bit rate on RX (Bottom)

After determining the bitrate of Uno's CAN Shield system, the next step was to choose the right clock frequency for oversampling. The clock frequency chosen was 2 MHz since it is greater than twice the bitrate. The first course of action was to investigate the communication problems between the Teensy and the CAN shield. The setup of this experiment and possible test points are depicted in the figure below.

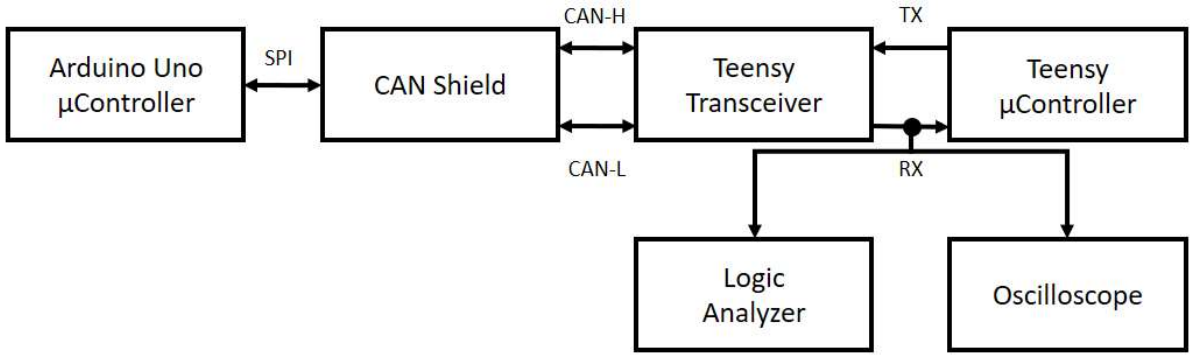

Figure 42 Connecting logic analyzer and oscilloscope to RX

Some initial observation on an oscilloscope revealed that Teensy's TX line was active while the device was in receive mode, instead of staying railed high at 3.3V. The team took images both of the TX and RX lines of the transceiver, first with the TX line connected to the Teensy, and then with 3.3V to force it high. When the TX line was connected to the Teensy it was found to be affecting message on the CAN bus lines. The TX was going down to 0V while the CAN Shield was transmitting the data field and not while the acknowledgement field. Since the Teensy is in receive mode there is no reason for the device to attempt to transmit other than when acknowledging that it has received a message, thereby flipping the acknowledgement bit. With this discovery it was concluded that the Teensy was the one responsible for the breakdown in communication between itself and the CAN Shield. The breakdown in communications is shown in the figures below.

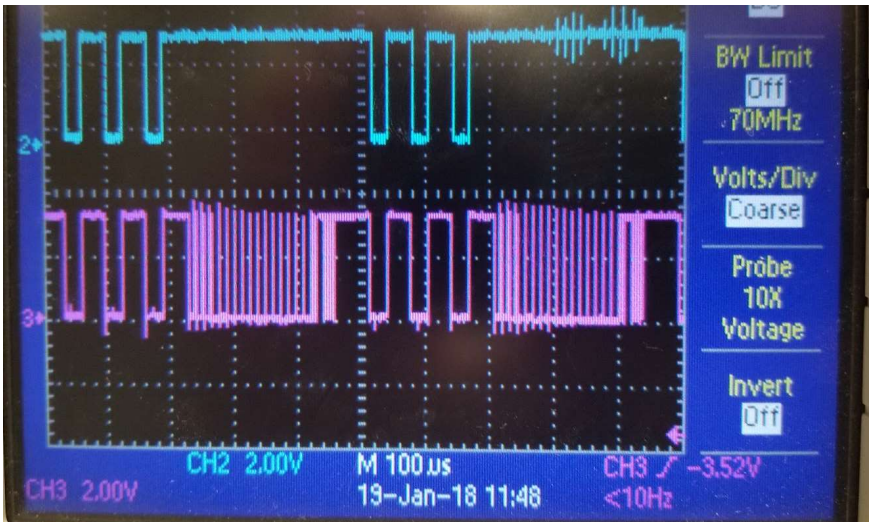

Figure 43 TX(top) and RX(bottom) of the Teensy in Receive mode

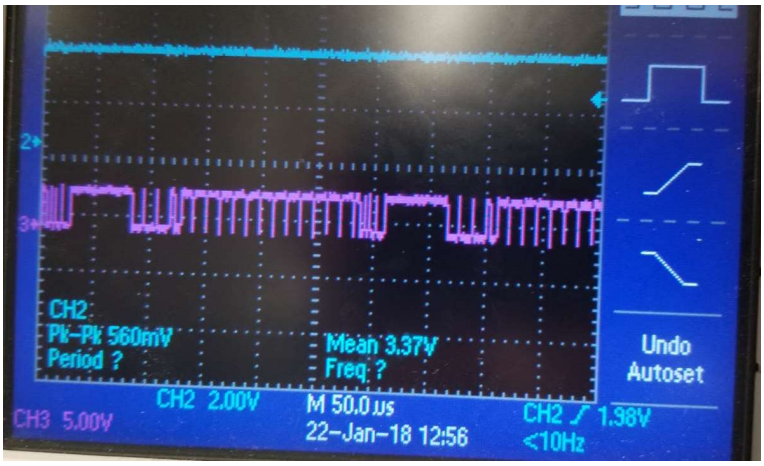

Figure 44 RX (bottom) of the Teensy in Receive mode with the TX line railed to 3.3V

First an image of the CAN message was taken with the two shields communicating with each other, and thus in theory flipping the Acknowledgment bit. For the next test, the receiving shield was disconnected to then take an image of the message without the Acknowledgment bit flipped. Taking a look at the end part of the two messages shown in the figure below, it was immediately clear that they were different; the receiving shield was definitely changing the message. The one difference between these two messages was that there was a one followed by a zero bit at the far right of the CAN message that had been acknowledged. This corresponds to the structure of a CAN bus message, where the acknowledgement bit is stored after the CRC in a two-bit field. The zero bit on the right side of the two bits of the acknowledgement bits is flipped by the receiver to indicate that the message has been received. With this knowledge the team was able to confirm that the CAN bus protocol was functioning properly.

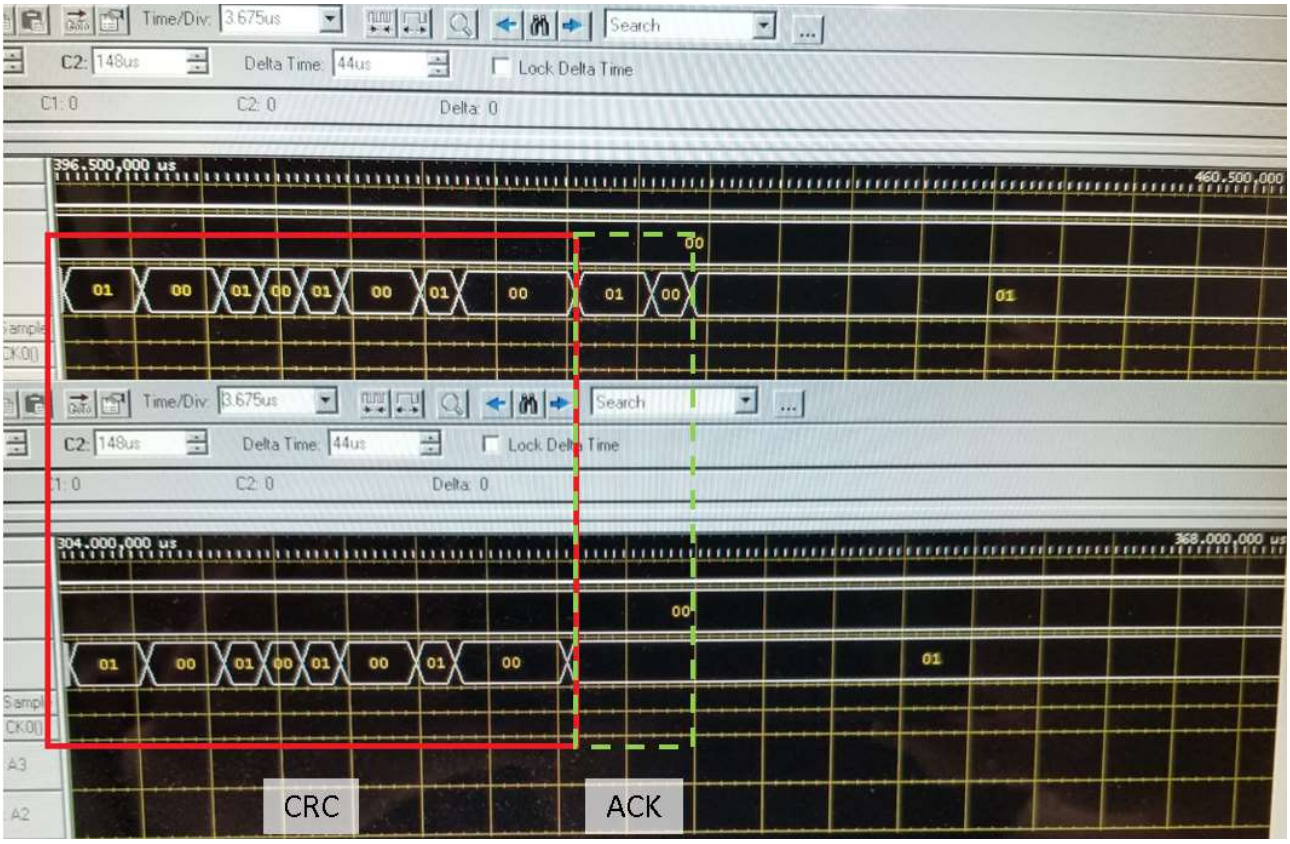

Figure 45 Receiver connected (top) disconnected (bottom)

3.5 Integrating the Ice Detector into the CAN Bus System

There are four major components that need to be connected to one another. The first component is the headlight microcontroller that serves as the decision maker in when to poll for information and what devices to activate based on information provided. The second component is the ice detection system which consist of the IR sensors and temperature sensor for the headlight. The third component is the heating element which contains the ITO and power MOSFET for switching. The last component is the car's microcontroller as it has the external temperature and information whether the headlights are on (not daytime running lights).

The first component is the middleman between all of the other components of the system. The microcontroller uses two (ADC) analog to digital pins to interrupt data and uses the digital pins to power the devices within the ice detection system. The power MOSFET that controls the

ITO has its gate connected to a digital pin of the microcontroller. The car's microcontroller is then connected to the other one through the use of the CAN bus high and low lines.

3.6 Testing Complete Prototype

The prototype includes several components that all have to work individually, then have to be able to communicate to one another and finally develop a system that accomplishes the following conditions:

More energy efficient than halogens (standby and in use)

Fully automated ice detection and management system

Ability to communicate with host car via CAN bus

Durable, repeatable, and functions in real life conditions*

System status/awareness for error detection*

* Reach goals

From these conditions a series of test are developed. The first test is to measure all the voltage and current requirements when the system is in standby and fully powered. The next test is to put the system into a series of artificial conditions to simulate the situations the system is expected to handle. The next test is to have both the headlight microcontroller and the car's CAN bus system to be able to successfully communicate with one another. The durability/reliability test will be testing how the ice detection system handles a continuous workload. The self-error checking system will have to be aware of when the system has lost one of its sensors and be able to report system conditions through the CAN bus system.

4 Results & Analysis

This chapter is separated into four major sections. The first section focuses on the testing the improvements of the IR ice sensor. The second section is focused on testing the digital side of the project. Then, the last two sections focus on a general overview of the final prototype and any potential problems that could occur in the system.

4.1 Improvements and Testing of Ice Sensor

This section focuses on three different aspects of the ice sensor. The first subsection goes into detail on increasing the coverage of the ice sensor. The second subsection is about increasing energy efficiency of the system and the last subsection is about adding additional hardware to reduce connections to the Arduino.

4.1.1 Quad IR emitters and transistors

The purpose of the IR emitter and transistor pair is to operate as an infrared sensor, and be able to detect when an IR reflective object appears within proximity of the sensor. The intensity of the IR LED is dependent on the current passing through the bulb. Meanwhile, the IR transistor is sensitive to IR light, and is able to complete a circuit whenever IR is detected. Therefore, whenever objects appear in front of the pair, the IR transistor will receive the IR from the obstruction deflecting IR waves from the LED.

In this instance, these pairs are used to detect the presence of ice on the outside surface of the headlight lens. More accurately however, they are detecting the presence of contaminants and air bubbles trapped inside of the ice, since these pairs are emitting near IR, which passes through water, rather than being reflected. While this may seem like a drawback, it should be noted that if uncontaminated ice were to ever appear on the surface of the lens, visible light would still be able to pass through, and there would be no need for the ice to be dealt with. In order to be

completely certain that an IR sensor would be able to detect ice, a simple test was performed using an ice cube and the IR sensor shown in Figure 46 below.

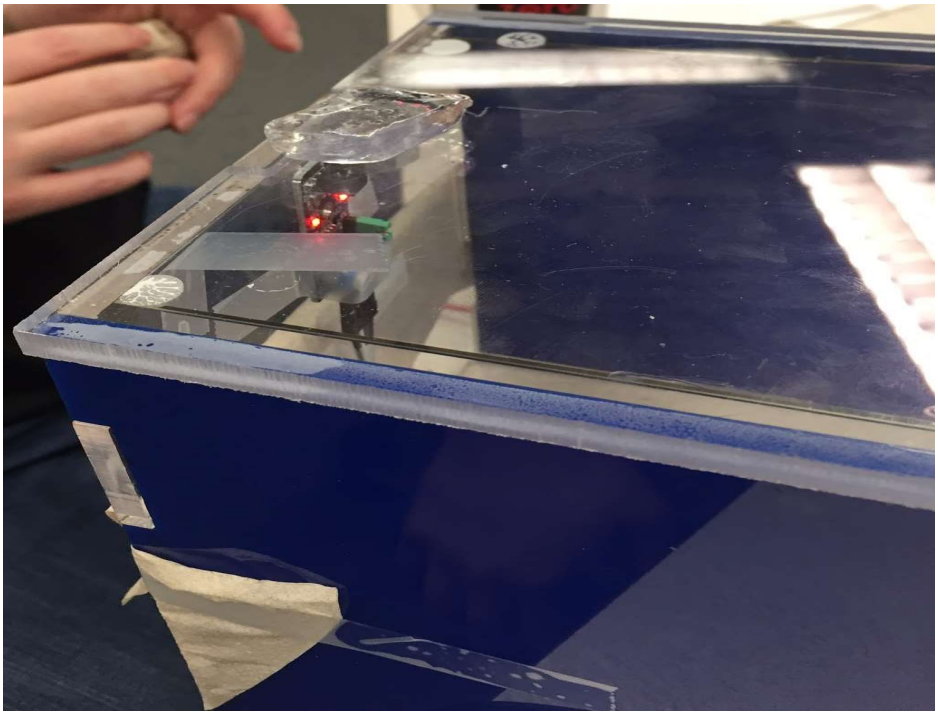

Figure 46 IR Ice cube test

Once it was proven that a single IR sensor could detect ice buildup on a headlight lens, a system was developed for incorporating multiple IR LED/transistor modules for better coverage of the headlight lens. The system, shown in Figure 47 below, incorporated OR gates in order to consolidate four outputs into one, which would indicate that one or more of the sensors had detected the presence of an obstruction. This system was also tested, and as expected, was able to produce a 5V output, proving it was capable of performing as designed.

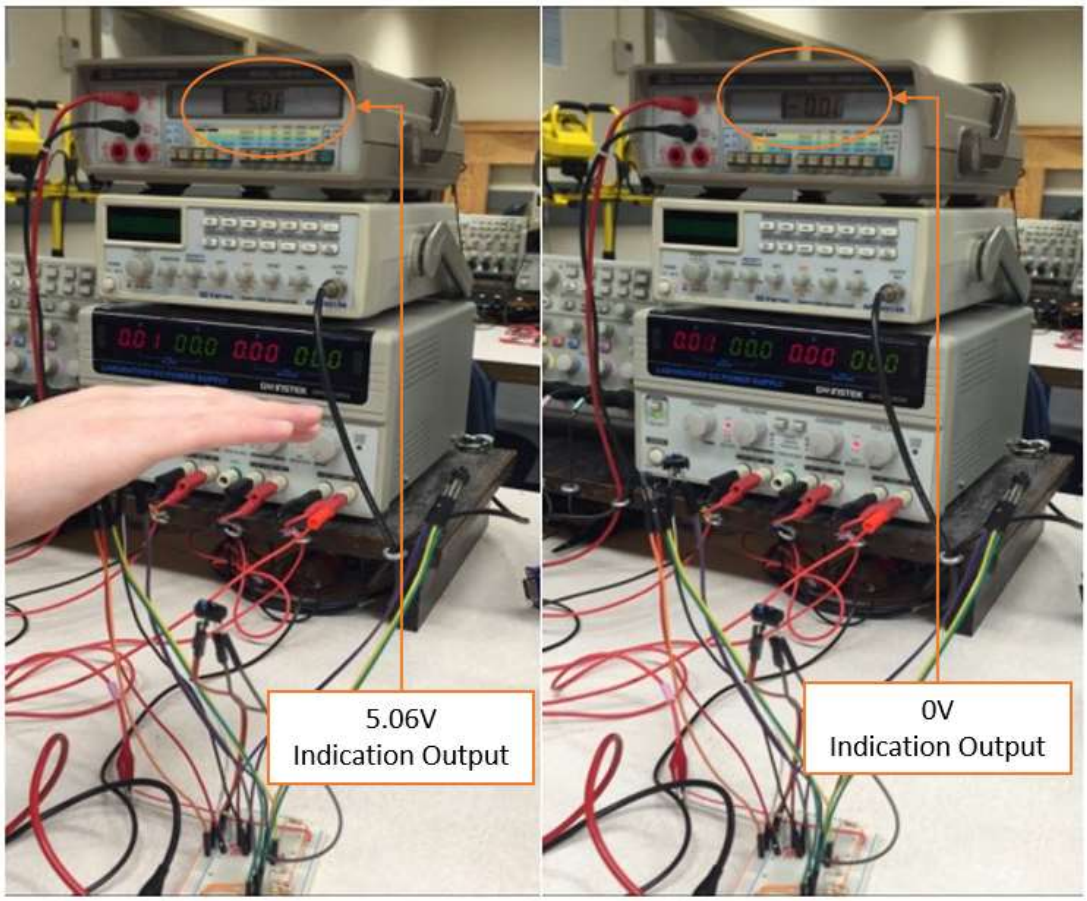

Figure 47 Quad-Ice sensor; obstruction sensitivity of the IR sensor

One very important discovery to note was that when testing the IR sensor, the sensitivity of the device was dependent on the current provided to the LED. This result is actually understandable, since it is the LED's IR light which will be triggering the transistor. By increasing its output, the LED increases the detection range of the transistor. The trick in this system, is to find the proper balance for the particular lens being used. If too little IR is emitted from the LED, then the IR will not be reflected enough by the obstructions, but if too much is emitted, then there will be enough IR reflected off of the lens to trigger the sensor. The system is calibrated to the point that it only turns on when an obstruction is detected in the immediate vicinity of the sensor through the lens.

4.1.2 Incorporating multiple ice sensors

The final quad-sensor design, shown in Figure 48 below, runs off of a 5V input power source and uses 26 mA when fully activated, and is activated by a signal from the Arduino, which triggers a MOSFET switch, completing the connection between the sensor and the shared ground of the complete system. The 5V rail is used to power two parallel circuits: the LEDs being on one, with the Transistors and OR gates being the other.

The LED circuit incorporates a 53 Ω resistor immediately following the 5V rail, in order to limit the output of the LEDs, since their output has an effect on the sensitivity of the sensor. The four LEDs are then arranged for optimal power use, with the LEDs being placed in two series pairs. Each led has a voltage drop of 1.804 V, so when two are placed in series that voltage drop doubles to about 3.6 V. Thus, by incorporating this design instead of placing four LEDs in parallel, the voltage dropped across the resistor is only 1.4V instead of 3.2V, and about 26 mA instead of 52mA based on simulations. With this system, the LED circuit is able to minimize power loss, while still allowing for 13 mA of current to be flowing through each LED.

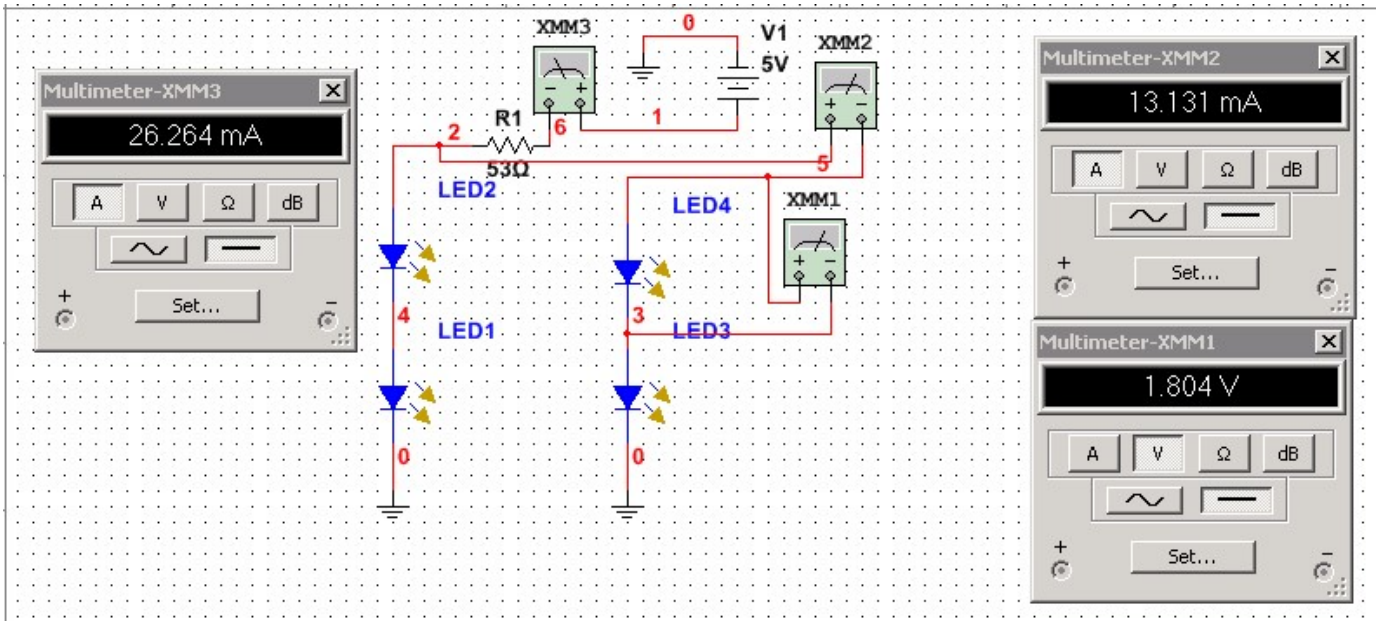

Figure 48 LED circuit diagram and power measurements

The transistor circuit does not require any resistance in series with the 5V rail. Instead, 5V is applied directly to four transistors in parallel. The emitters of the transistors are each connected through their own 10 kΩ resistor to ground. The voltage between the transistor output and the resistor are then used as four separate output signals to indicate the presence of an obstruction whenever the transistor has been activated. After it is activated, the transistor allows current to flow through the resistor, inducing a voltage drop and indication signal spikes from 0V to roughly 5V. These outputs are all then wired into the input terminals of two OR gates, whose outputs are then put through one third and final OR gate. The output of the final OR gate is used as the final output of the system, and will read roughly 5.06V whenever one or more of the sensors detects an obstruction (transistors are triggered). This output can then be read by the Arduino, and incorporated into a larger logic system for detecting ice buildup on a headlight.

4.1.3 MOSFET switching and logic gates

The key differences between using a relay module and a MOSFET for switching the ITO, is that a MOSFET is a simpler and cheaper option. The MOSFET is a RFP30N06LE N-Channel Power MOSFETs with a max rating of 60V & 30A, and a 2V threshold voltage. The Uno only outputs 5V on all of its digital pins and has a max current of 40 mA per pins. These specs are below the maximum ratings for the MOSFET and outputs a voltage greater than the threshold voltage. By adding MOSFETs to the Ice detection and ITO circuits the Uno is able to control these devices while also having a negligible power difference.

There are four IR transistors in the ice detection circuit and the ITO should be activated if one of the sensors goes off. In order to reduce the connections between the Uno and the ice detection system a OR gate was utilized. The CD4071BE is apart of the Texas Instrument 4000 series of CMOS OR Gates and the chip includes four dual input OR gates. The OR chip was also

chosen since the required input voltage was 5V and can be powered off of an Uno. The power difference of powering the OR chip off the Uno is negligible.

4.2 Testing CAN and Code for Arduinos

The following subsections are focused on testing the Arduinos' system. The first subsection focuses on verifying that the system is properly emulating a CAN system. The second subsection focuses on the code to manage the temperature sensors, ice sensor, and ITO.

4.2.1 CAN connectivity

With the logic analyzer, each bit on a CAN bus message could be isolated. With over one hundred bits in each message, it was determined that the most efficient method of verifying would be with the acknowledgement bit. The Acknowledgment part of the message makes up two bits near the end of the CAN bus message. This bit starts low when the message is sent out, and is flipped by the receiver. Once the transceiver finds that the Acknowledgment bit flipped, it knows that the message has been successfully received.

When analyzing, differences exist between the two messages, confirming that the CAN bus message had been changed by the receiving node. Taking a look closer, the part that has been changed was the last two bytes of which contain both the acknowledgement bit and the end frame. From the logic diagram below in Figure 49, the following code was developed

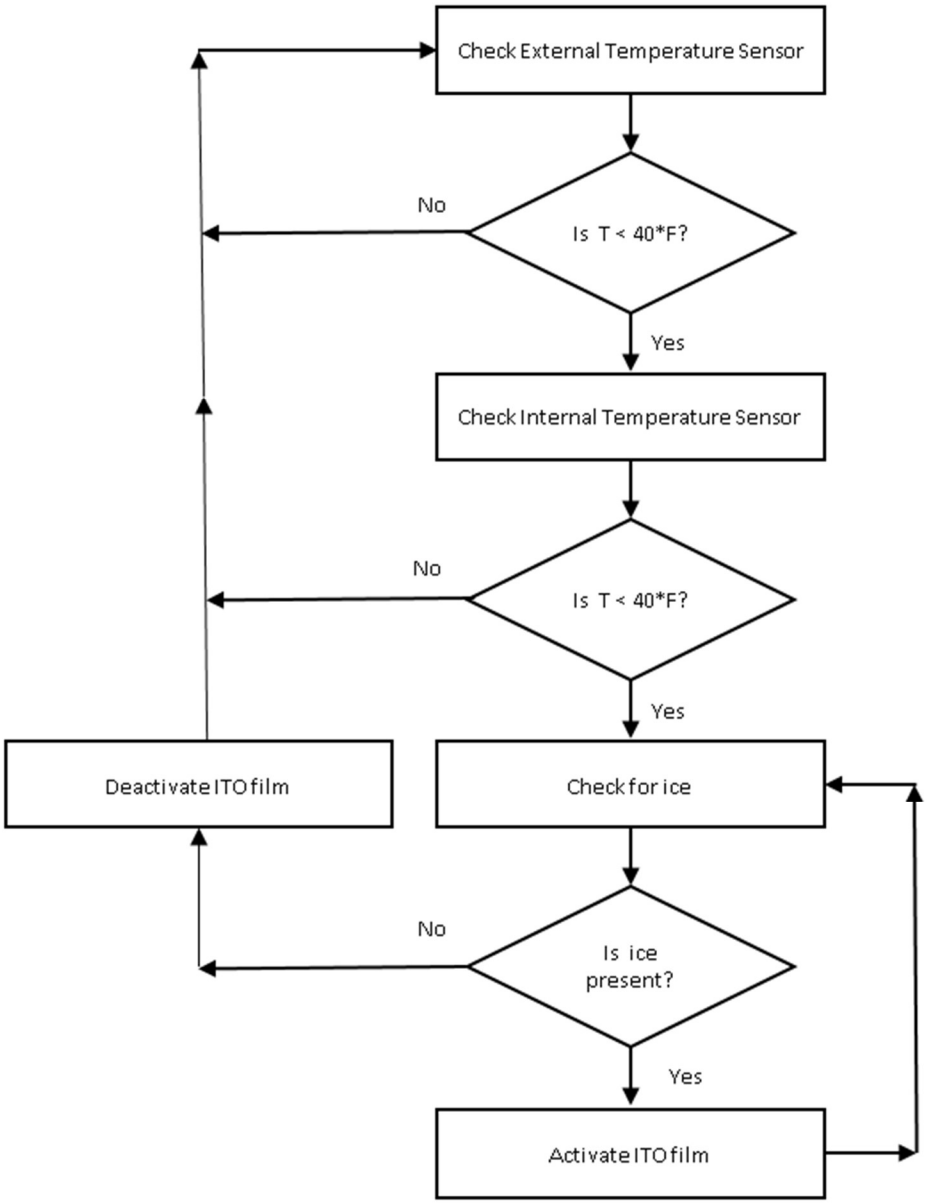

Figure 49 Logic diagram for ice sensing

The Arduino Uno represents the headlight node, and as such contains the code that regulates all of the system; this code can be viewed in Appendix A. When converted into code, this is represented by three separate conditionals, two ifs, and one while loop. The first conditional is checking the external temperature represented by the second Arduino Uno with an analog temperature sensor attached. The headlight Uno sends a message through CAN bus to request a temperature reading from the second Uno, which sends Temperature data back through

CAN bus. If this returned temperature is below 40 degrees Fahrenheit, then it goes onto the second if statement, if not it will continually poll the external temperature sensor. The next conditional deals with the analog sensor on the headlight Uno. After reading this on-board sensor, if it is above 40 degrees Fahrenheit, the headlight Uno will return to polling the external temperature. If the headlight temperature sensor is below 40 degrees Fahrenheit, then the code moves onto the while loop. The condition to stay in the loop is that the ice sensor detects ice. The Uno will turn the sensor on by outputting 5V to the ice sensor's MOSFET and as long as 5V is outputted from the ice sensor to the Uno it will enter the loop. In this loop the ITO is activated by a 5V output that uses a MOSFET to control the ITO. As long as the ice sensor detects ice the code remains in this loop. Once it stops detecting ice, it will exit the loop, turn off the ITO, and go back to polling the external temperature sensor.

4.2.2 Code to regulate system

The I2C analog temperature sensor was used to obtain temperature readings to regulate parts of the code. The sensor takes in 5V from one of the analog pins from the Arduino, and gives a varying voltage reading. This voltage reading is then converted to a temperature based on the formula seen in Appendix A. This reading can then be used by the code to determine how to proceed through the logic. There are six analog/digital pins on the Arduino Uno which can both be used as inputs or outputs. Labeled A0 through A5, the following set up was used for the team's pins:

A0 - I2C temperature sensor output signal

A1 - Ice sensor output signal

A2 - Ice sensor input voltage

A3 - ITO MOSFET Switching Voltage

The CAN library was used to have the Arduino Unos communicate with each other through CAN bus protocol. The library comes with several CAN bus test codes to check the functionality of the user's CAN bus system. The send and receive_check test programs were used to understand and create the code for the team's final prototype. CAN library contains only a few functions that can be called, with CAN.begin, CAN.sendMsgBuf, and CAN.readMsgBuf being the ones used the code. Most of the processing occurs with the primary functions calling other functions.

4.3 Final Prototype Overview

The final prototype has three different states that it can be in. The first state is the idle position, only the two Arduinos and their temp sensors are enabled. This position is only enabled when both temperature sensors read temperatures greater than 40°F. The power draw in this state is approximately 700 mW, 5V & 140 mA. In the test for ice stage both temperature sensors read below 40°F and now the ice detection system is activated. The addition power consumption in this stage is about 250 mW, 5 V & 50 mA, and adding this to the power consumption stage brings it up to 950 mW. This is the shortest state since it activates the ice detection system long enough for one reading or about 1 second. If ice is not detected then the system reverts back to the idle position, however if ice is detected then the system enables the heating system. In the final state the power consumed by the ITO is about 18 W, 12 V & 1.5 A. The power cost at each state is outlined below, in Table 7. Since the heating state requires the most power the system checks at 2-minute intervals to determine that ice is present, otherwise the system reverts back into the idle state.

Table 7 Power consumption based on operating states

State	Total Power drawn	Additional power drawn
Idle position	700 mW	NA
Test for ice	950 mW	250 mW
Final state (ITO)	18.95 W	18 W

The system's effectiveness is determined on keeping the sensors mounted and the headlights clean. As long as the headlight does not have caked on containments, everything is mounted properly, the ice detection system is calibrated for the lens, and all sensors are functional then the system will operate properly.

An overall depiction of the system is shown in Figure 50 below. The headlight Uno uses two MOSFETs to control the ground lines from the ice sensor and the ITO. The pins that are used to control these devices are A1 and A2, respectively. The Car Uno communicates through CAN bus and is powered by the Car Uno through a Seeduino Groove connector. The temperature sensors are connected using 5V, ground, and A0 for each Uno. The Ice sensor and ITO are both powered by their respective power supplies. The ice detection system outputs to pin A3 on the headlight Uno to indicate the presence of ice.

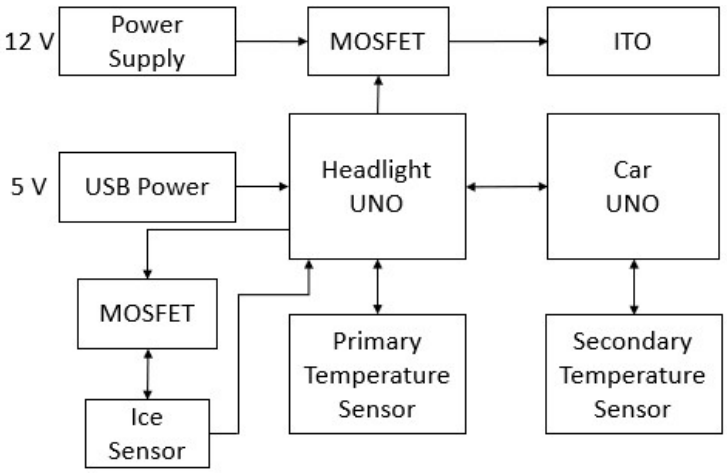

Figure 50 Diagram of final prototype

4.4 Potential Issues

One potential flaw lies in the ITO method for heating the lens. This project focused on providing a better control system for the ITO, thus lowering the overall power needed for a de-icing system. All information regarding the feasibility of the ITO system for ice melting was acquired from a previous MQP project. In their experiment, they placed the prototype headlight enclosure in a walk-in freezer unit, and waited until its temperature matched the 0°F ambient temperature. They then began delivering power to their ITO circuit, and measured the temperature of the lens after 10 minutes of the ITO applying heat across lens. The measured temperature was 66.4 °F, which did meet their initial goals. Despite this however, these testing conditions may not have been a perfect representation of what a real-world headlight would be dealing with.

First, the headlight was not dealing with any ice buildup on the surface of the lens. Because ice has a very high specific heat, it would be much more difficult for the ITO to increase the surface temperature of the lens with ice being present. Second, the headlight will be facing wind chill when the vehicle is moving. With wind chill included in the system, it is even more difficult for the ITO to heat the lens. With these two conditions present, the time needed to heat the lens, and melt ice, are unknown.

Another potential area for improvement lies with the IR ice sensor. The purpose of the ice sensor is to improve the efficiency of the system by ensuring the ITO is only operating, and drawing power, when ice is present. However, the IR sensor used is actually just detecting the contaminants in the ice. This means that other things, such as dirt could also trigger the sensor. This could sacrifice the efficiency of the system. However, since the ice sensor is only checked

when freezing temperatures are detected, non-ice contaminants will only be a problem during times when freezing temperatures are present.

A second potential flaw in the current IR ice sensor system is that the transistor is triggered by all types of IR. For example, the IR emitted by a human body would also trigger the ice sensor. Since the sun emits IR, the sensor could be triggered by its rays during daylight hours. Because some cars require headlights to operate during the day, this could be a problem for the current IR sensor system.

The final potential flaw in the system, is that it is always possible for a sensor or microcontroller to wear out over time. If this occurs, those elements may not be able to provide the needed inputs to the Arduino needed to initiate the ITO. The code does not have any built-in safeguards to determine if a sensor is functioning properly or to notify the operator that there is a problem. The ice detection sensor in particular does not have a built-in error detection system and there is not enough data to show what the most common problem is. At this stage of the prototype's development it is only possible to mark the most extreme conditions this unit could be exposed to and to flag/report any data that is outside of that range.

5 Conclusions & Recommendations

This chapter first looks at the improvements that have been made on the previous year's project. The second section goes into the potential impact of this project. Then, the final two sections include remarks about working with the Arduino CAN Library and recommendations to improve the final prototype.

5.1 Improvements on Previous Year's Project

Overall, this project was a great improvement over its predecessor from 2016-2017. The former design focused solely on testing the viability of the ITO for heating the lens, and used a simple hysteresis loop to decide when to activate it. This design proved to be incredibly inefficient however, since it meant that the ITO would be on during all times when it was exposed to freezing temperatures, which is not necessary. This year's project removed most of these inefficiencies by incorporating a better system for deciding when to activate the ITO. By using an IR sensor along with the onboard temperature sensors, the ITO will only be active when it is cold enough and an obstruction is detected.

5.2 Potential Impact of Project

There are two main advantages to adding this system as opposed to other LED headlight ice management systems, such as the 2016-2017 project. The first is energy efficiency, since this system uses temperature sensors along with an IR sensor it is able to conserve power when it is cold out and there is no ice obstruction, unlike its competitors. Since this system also includes car connectivity, it is only powered when the headlights are required and it uses the car's preexisting temperature reading as a conditional. This means that the heating element will not drain the car's battery when the car is off. The second advantage of this system is the convenience for the user, as not having any system requires the user to actively track the

condition of their headlight lens. The final prototype does not just stop at headlights though, as the IR can also detect fog buildup on the lens. This means that it can be applied to other applications such as heating windows, outdoor lights, and other clear outdoor systems. This technology could be applied to automated vehicles to keep headlights, windows, and external sensors free of ice.

5.3 CAN library

Further testing would be required to prove that the microcontroller is able to interface with a car's CAN bus system. This would in term require substantial research into the exact message structure to obtain the correct data from the car's system. After gaining an understanding of CAN bus, it would require further research into a particular car to gather the information of how to access its CAN lines and to find relevant codes.

The CAN library used the by the team would only be recommended to users with sufficient understanding of CAN bus. Although creating a CAN bus system between two Arduinos is simple, it is difficult to use the CAN library with other CAN emulation systems.

5.4 Recommendations

One potential issue is that the 2016 project's ITO testing procedure did not definitively prove that the ITO could melt ice in real world driving conditions. To prove that the ITO is able to melt ice at an acceptable rate, a new test needs to be created, where the system is exposed to real world driving conditions, such as pre-existing ice buildup, wind chill, and ongoing precipitation.

Another potential issue is that the IR sensor would likely be triggered by sunlight and other IR sources such as halogen headlights. To work around this dilemma, the system could be modified to incorporate the car's onboard ambient light sensors to determine if the sun is out, and be made to run only off of temperature sensors during that time. However, for states that require daytime headlights, then future iterations of the system could also incorporate small optical filters on the transistor lenss. These filters would help reduce over saturation from the sun and help identify differences from IR reflected from obstructions. Another potential way to overcome this problem is to modulate the IR signals and to have the transistors activate only after receiving a reflected signal. The last suggestion to help improve the accuracy of the IR system is to experiment with different angles between the IR emitter and phototransistor to find an optimal angle. An optimal angle would have less interference from external IR sources and would also provide more coverage of the headlight lens.

The last concern in this project is with the lack of error checking when receiving information from the sensors. A possible solution for this sensor problem is to determine the signs of a defected unit and to create additional hardware/software options to have the Uno identify the problem. A hardware solution could involve adding switches to the ice sensor to indicate a problem in the circuit. A software solution could involve the microcontroller keeping a short log of previous readings and use that information to determine if a sensor is outputting abnormal data.

References

[1] A Major Qualifying Project *et al*, "Efficient LED headlight heating system," .

[2] *Automotive CAN Bus System Explained*. Available: https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski/.

[3] L. Franck and F. Gil-Castineira, "Using delay tolerant networks for Car2Car communications," in 2007 IEEE International Symposium on Industrial Electronics, 2007, .DOI: 10.1109/ISIE.2007.4375013.

[4] J. F. Hines, "Hype cycle for automotive electronics, 2015," Gartner Inc., Jul 29,. 2015.

[5] L. Franck and F. Gil-Castineira, "Using delay tolerant networks for Car2Car communications," in 2007 IEEE International Symposium on Industrial Electronics, 2007, .DOI: 10.1109/ISIE.2007.4375013.

[6] B. Popa. (-01- 21T19:41:32+00:00). Battle of the Headlights: Halogen vs. Xenon vs. LED vs.Laser vs. Conversion Kits. Available: https://www.autoevolution.com/news/battle-of- the-headlights-halogen- vs-xenon- vs-led- 26530.html.

[7] Anonymous (). Color Temperature: What it is, and what you need to know. Available: http://lumicrest.com/colour-temperature/.

[8] Anonymous (). Rigid industries: 7" Round Heated Lens w/PWM Adaptor. Available: https://www.rigidindustries.com/led-lighting/55004

[9] US Department of Commerce, NOAA. (). What Causes Frost?. Available: https://www.weather.gov/arx/why_frost.

[10] J. J. Gerardi and T. Banas, "(12) United States Patent," US 8,248,256 B1, Aug 21, 2012.

[11] B. R. Jean et al, "A microwave interferometer sensor for ice cloud detection and measurement," in 2017, Available: http://ieeexplore.ieee.org/document/7999544. DOI: 10.1109/MetroAeroSpace.2017.7999544.

[12] W. Chen et al, "Wireless extensions of CANBUS in industrial applications," in 2011, Available: http://ieeexplore.ieee.org/document/6098693. DOI: 10.1109/BMEI.2011.6098693.

[13] Texas Instrument, “3.3-V CAN Bus Transceivers,” SN65HVD230 datasheet, March 2001 [Revised July 2015].

[14] T. Maciel, "Walking On Water: Physics of Clear Ice," 2014. Available: http://physicsbuzz.physicscentral.com/2014/12/walking-on-water-physics-of-clear-ice.html.

[15] B. Radford. (Jan 9,). What Makes Ice Cubes Cloudy?. Available: https://www.livescience.com/32407-what- makes-ice- cubes-cloudy.html.

[16] Microchip, “High-Speed CAN Transceiver,” MCP2551 datasheet, July 1999 [Revised March 2002].

Appendix A: Code for Headlight Module

```
// demo: CAN-BUS Shield, send data
// loovee@seeed.cc

#include <mcp_can.h>
#include <SPI.h>

// the cs pin of the version after v1.1 is default to D9
// v0.9b and v1.0 is default D10
const int SPI_CS_PIN = 10;

MCP_CAN CAN(SPI_CS_PIN);// Set CS pin

void setup()
{
   pinMode(16, OUTPUT);//pin A1 instantiated as an input to power the Ice sensor
   pinMode(17, OUTPUT);//pin A2 instantiated as an input to power the ITO

   Serial.begin(115200);//setting speed for serial monitor for testing

   while (CAN_OK != CAN.begin(CAN_500KBPS))// init can bus : baudrate = 500k
   {
      Serial.println("CAN BUS Shield init fail");
      Serial.println(" Init CAN BUS Shield again");
      delay(100);
   }
   Serial.println("CAN BUS Shield init ok!");
}

unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 1};//data for the message to request the external
temperature from the temp node
```

```
void loop()
{

    if(checkExtTemp() == true)
    {

      if(checkIntTemp() == true)
      {

          while(checkForIce() == true)
          {
            activateITO(true);
          }

          activateITO(false);
      }
    }

    delay(1000);//wait for one sec between repeating the loop

}
/*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 * checkExtTemp
 * inputs:
 * outputs: boolean
 *
 * desc: sends a request across the CAN bus line for a external temperature reading. It then waits
until it receives it. When it does
 *       it will then check if the temperature is below frost levels(less than 40 degrees F). If it is,
then true is returned, else
```

```
 *       false is returned.
 * ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 */
boolean checkExtTemp(){

   boolean iceChance;
   int extTemp = 0;

   // send request for temperature reading:  id = 0x22, standrad frame, data len = 8, stmp: data buf
   CAN.sendMsgBuf(0x22, 0, 8, stmp);
   int exitFlag = 0;
   while(CAN_MSGAVAIL != CAN.checkReceive())//waits for return signal
   {
      exitFlag++;//increment flag to record amount of loops
      if(exitFlag > 1000)//if it goes above a certain amount, assume need to resend, go out and
enter the loop again
      {
        return false;
      }
   }
   unsigned char len = 0;
   unsigned char buf[8];//char array to hold the ext temp

   CAN.readMsgBuf(&len, buf);// read data,  len: data length, buf: data buf
   unsigned int canId = CAN.getCanId();
   //Serial.print(canId, DEC);
   Serial.println();
   //if(canId ==
   extTemp = buf[7];

   Serial.print("The current external temperature is: ");
```

```
   Serial.print(extTemp);
   Serial.println();

   if(extTemp > 40)//if temperature is over
     iceChance = false;
   else
     iceChance = true;

   return iceChance;
}

/*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 * checkIntTemp
 * inputs:
 * outputs: boolean
 *
 * desc: reads the temperature sensor connected to the arduino. If the temperature reading is
below frost levels(40 degrees F) then
 *       a true boolean is returned, else a false boolean.
 * ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 */
boolean checkIntTemp(){

   int sensorPin = 0;
   int intTemp = 0;
   boolean iceChance;

   //getting the voltage reading from the temperature sensor
   int reading = analogRead(sensorPin);

   // converting that reading to voltage,
```

```
    float voltage = reading * 5.0;
    voltage /= 1024.0;
    float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit 500 mV
offset
    float temperatureF = 1.8*temperatureC+32;
    intTemp = temperatureF;

    Serial.print("The current internal temperature is: ");
    Serial.print(intTemp);
    Serial.println();

    if(intTemp > 40)
      iceChance = false;
    else
      iceChance = true;

  return iceChance;
}

/*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 * checkForIce
 * inputs:
 * outputs: boolean
 *
 * desc: Takes in a reading from the Ice sensor. If a digital high is read then ice is considered to
be there and this function returns
 *       true. Else it returns false.
 * ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 */
boolean checkForIce(){
```

```
  boolean icePresent;
  int iceSensorPin = 1;

  digitalWrite(16, HIGH);
  delay(500);
  int reading = analogRead(iceSensorPin);
  digitalWrite(16, LOW);

  float voltage = reading * 5.0;
  voltage /= 1024.0;
  Serial.print("The reading on pin A1 is: ");
  Serial.print(voltage);
  Serial.println();

  if(voltage > 1.5)
     icePresent = true;
  else
     icePresent = false;

  return icePresent;
}

/*////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 * activateITO
 * inputs: boolean
 * outputs:
 *
 * desc: based on the boolean value passed into this function, the ITO film will be powered. If a
true boolean is passed in, then the
 *       ITO is turned on, else it is turned off.
 * ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
 */
void activateITO(boolean on){

  if(on == true)
  {
    //turn ITO on
    digitalWrite(17, HIGH);
  }
  else
  {
    //turn ITO off
    digitalWrite(17, LOW);
  }

}



// END FILE
```

Appendix B: Code for Emulating Car's CAN System

```
// demo: CAN-BUS Shield, send data

// loovee@seeed.cc


#include <mcp_can.h>

#include <SPI.h>


// the cs pin of the version after v1.1 is default to D9

// v0.9b and v1.0 is default D10

const int SPI_CS_PIN = 10;


MCP_CAN CAN(SPI_CS_PIN);                                    // Set CS pin


void setup()

{

    Serial.begin(115200);//start the serial monitor for testing


    while (CAN_OK != CAN.begin(CAN_500KBPS))              // init can bus : baudrate = 500k

    {

        Serial.println("CAN BUS Shield init fail");

        Serial.println(" Init CAN BUS Shield again");

        delay(100);

    }
```

```
    Serial.println("CAN BUS Shield init ok!");
}

unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 0};//char array to hold message data
int flag = 0;
int current_temp = 0;
int sensorPin = 0;
void loop()
{
    // waiting for a message with x22 on it

    while(flag = 0)
    {
      unsigned char len = 0;
      unsigned char buf[8];
      if(CAN_MSGAVAIL == CAN.checkReceive())            // check if data coming
      {
        CAN.readMsgBuf(&len, buf);    // read data,  len: data length, buf: data buf

        unsigned int canId = CAN.getCanId();
        flag = 1;
      }
   }
```

```
    //getting the voltage reading from the temperature sensor
    int reading = analogRead(sensorPin);

    // converting that reading to voltage,
    float voltage = reading * 5.0;
    voltage /= 1024.0;
    float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit 500 mV
offset
    float temperatureF = 1.8*temperatureC+32;
    current_temp = temperatureF;
    stmp[7] = current_temp;
     Serial.println(current_temp);
    CAN.sendMsgBuf(0x11, 0, 8, stmp);


}

// END FILE
```

Appendix C: Checklist for Troubleshooting Teensy with CAN

☒ Checked for common ground (shared one power source)

☒ Jumper on CAN transceiver

☒ Different message IDs

☒ Two shields and one Teensy

☐ Two Teensys

☐ Viewed CAN shield's TX on the logic analyzer

☒ Added termination resistance

Appendix D: CAN bus Messages

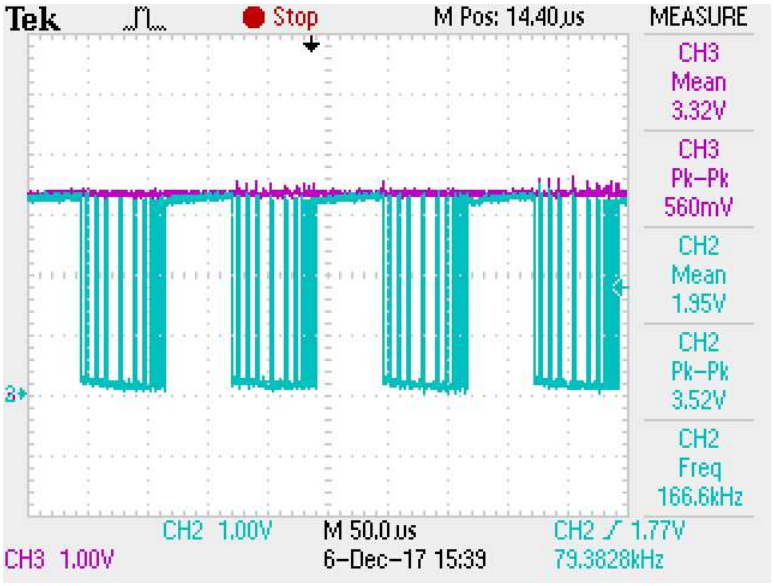

Figure 51 Teensy receiving from Arduino (RX and TX)

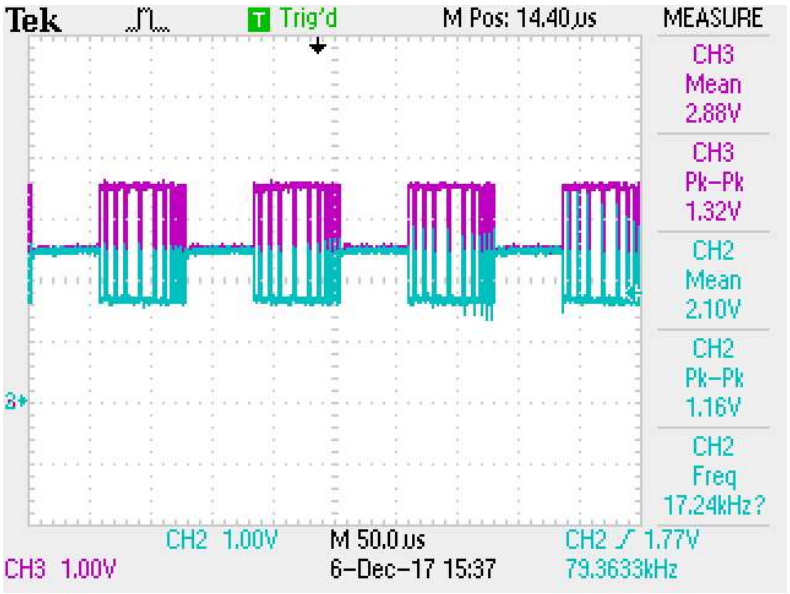

Figure 52 Teensy receiving from Arduino (CAN High and Low)

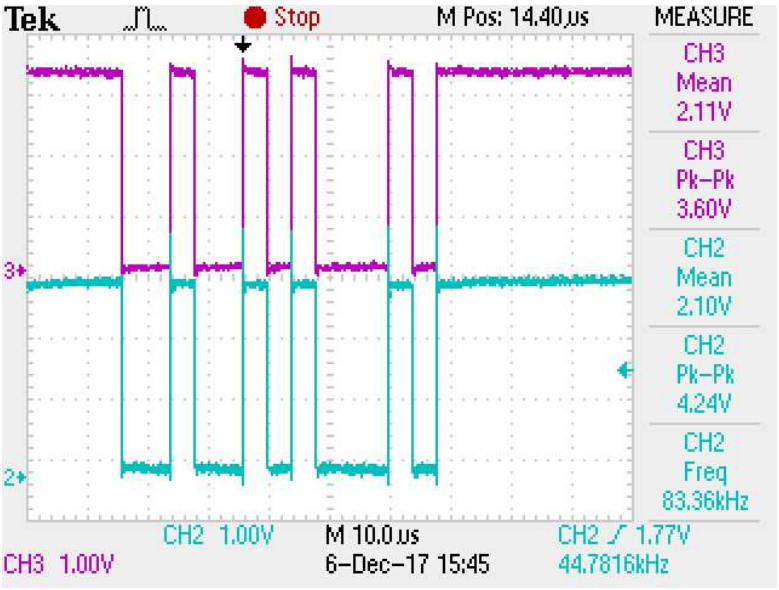

Figure 53 Teensy transmitting to Arduino (RX - Blue and TX)

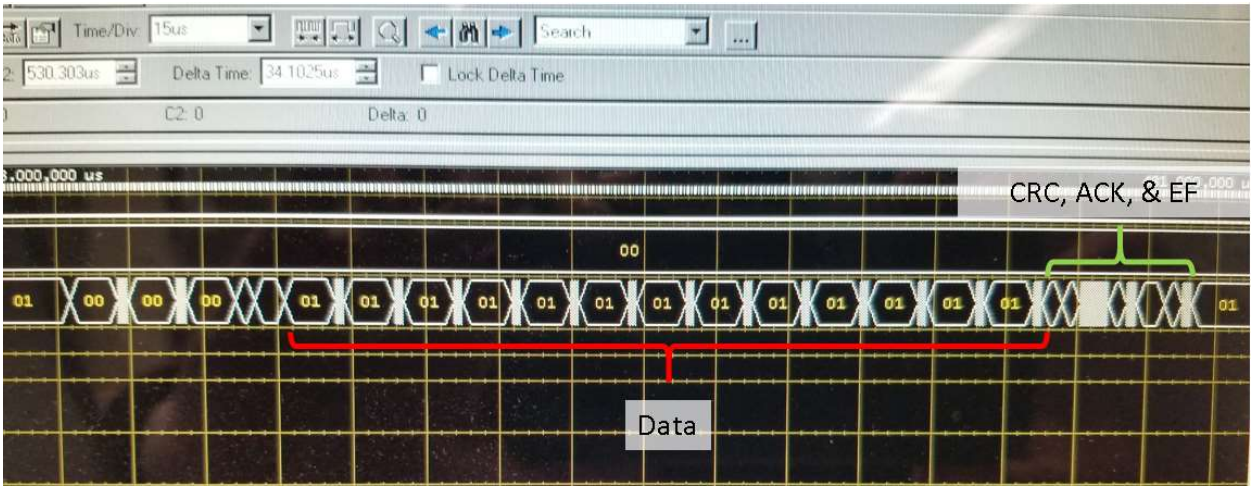

Figure 54 Full CAN bus Message on Logic Analyzer

Appendix E: Final Schematics

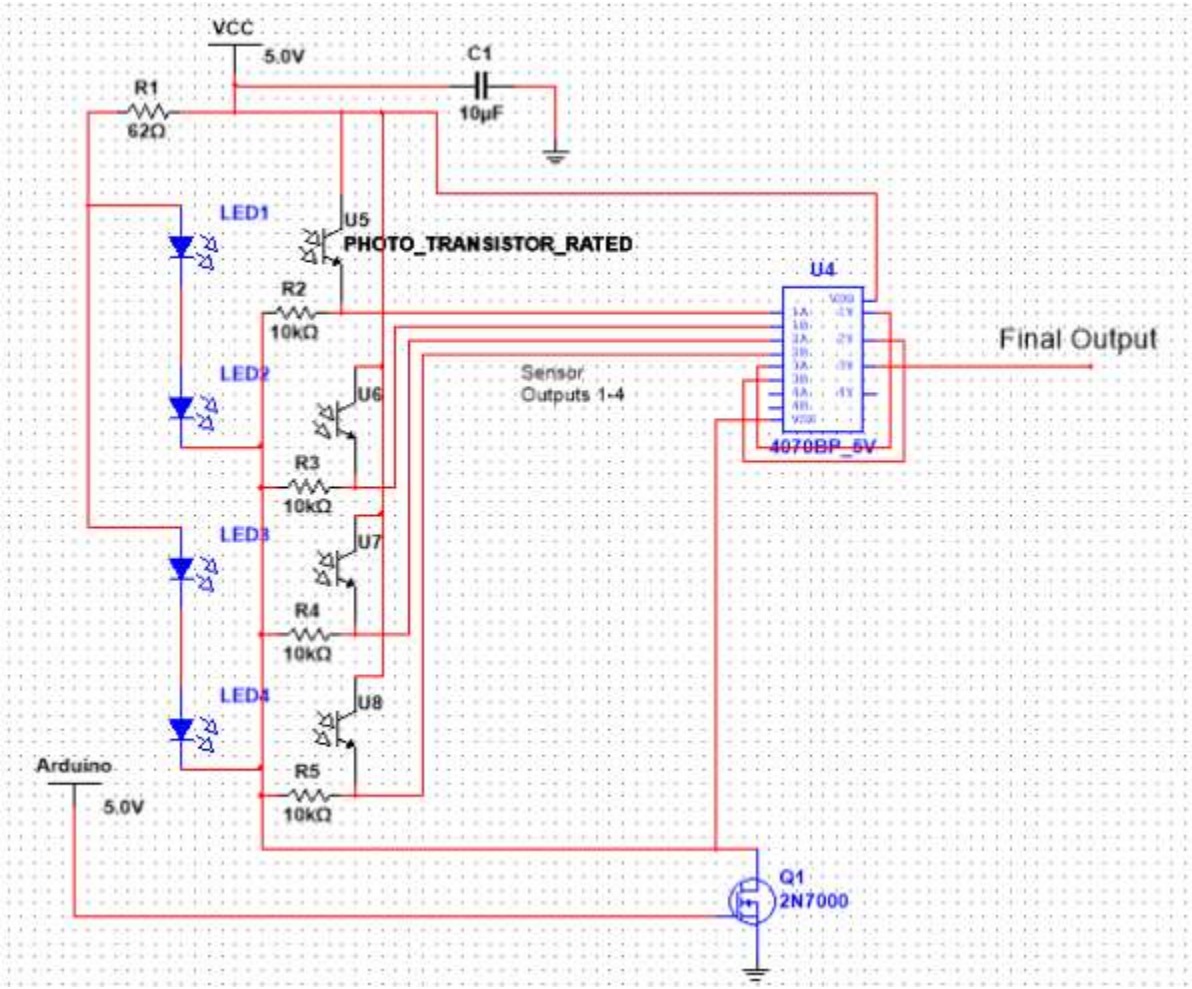

Figure 55 Final schematic for ice sensor

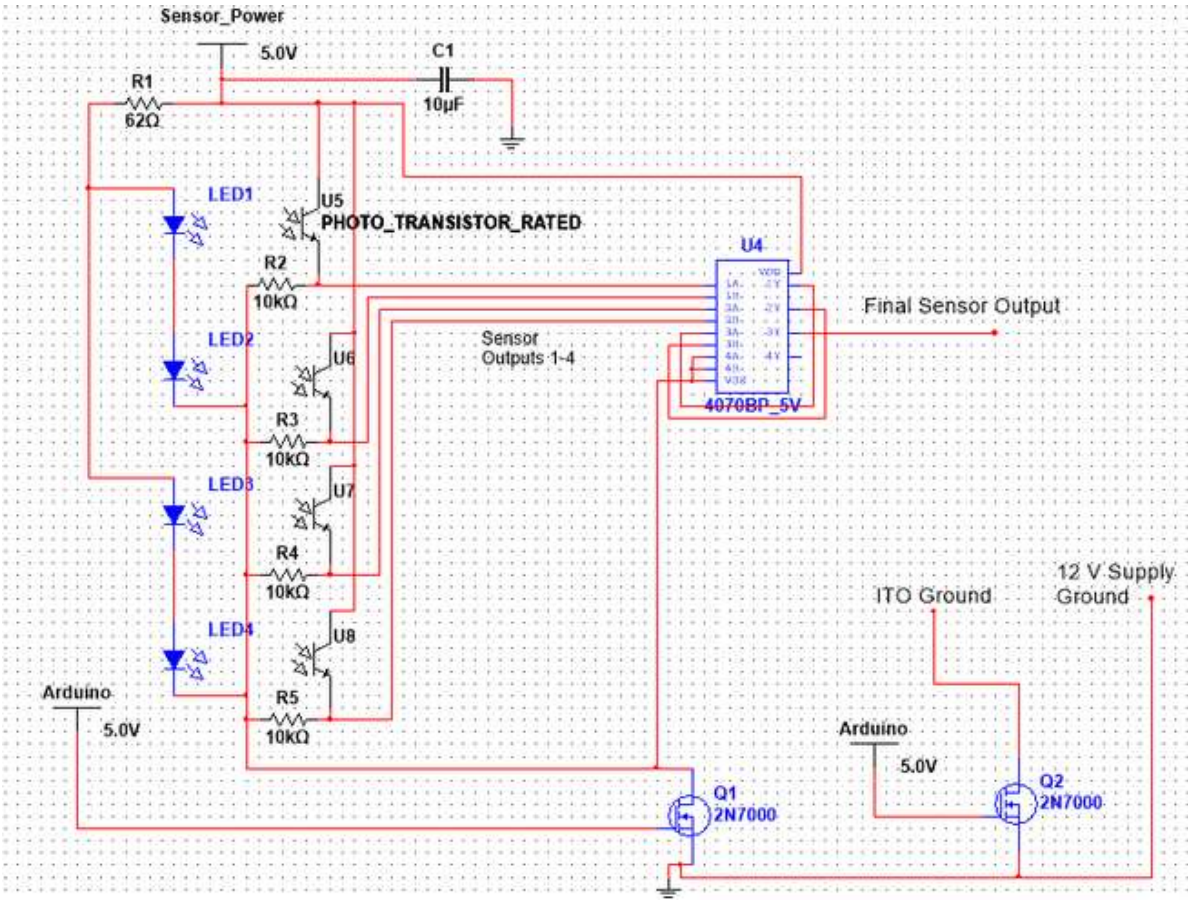

Figure 56 Final Schematic for PCB