ECE 445

Spring 2024

Final Report:
Stick-On-Car Proximity Sensor

Shrijan Sathish, Raunak Bathwal, Aryan Damani

Table of Contents:

Introduction:

Many older cars lack proximity sensors that let the user know how close their car is to various obstacles, whether it be their garages, parking garage walls, or even curbs. Though this can be handled through or simply guessing, driver experience, or even tricks of knowing where to look in the rearview or side mirrors, it is always better to be sure. Given that this is a feature most modern cars come with pre-equipped and something users find useful, we believe that providing an easy, convenient, and relatively cheap solution to this issue will be beneficial.

Our proposed solution involves placing four groups of three ultrasonic sensors on the car with one group on each corner. These modules will then relay information back to the driver through radio signals from a transmitter to a central module. This central module will contain the receiver, a speaker, and four LEDs - each one corresponding to its respective sensor group. As the car approaches objects and the sensor detects them, the central module will light up the LED indicating which direction the object is coming from, and the speaker will start beeping at 1 Hz. As the obstacles get closer, this frequency will increase until it reaches a single tone, letting the driver know that they are less than one foot away from hitting an obstacle. Figure 1 provides a visual aid to easier illustrate our solution, and a block diagram of our electronic components is provided in figure 2:

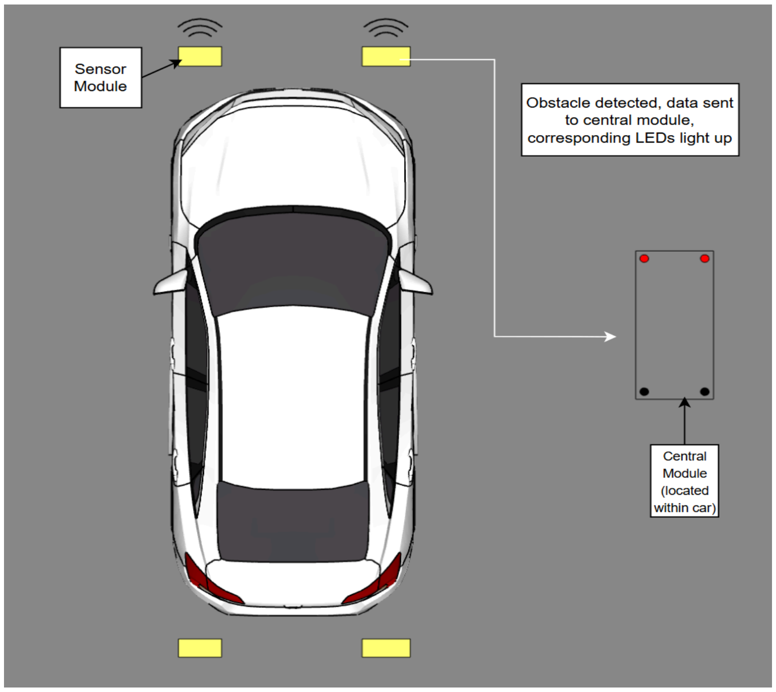

Sensor Module
Obstacle detected, data sent to central module, corresponding LEDs light up
Central Module (located within car)

Figure 1: Visual Aid of Solution

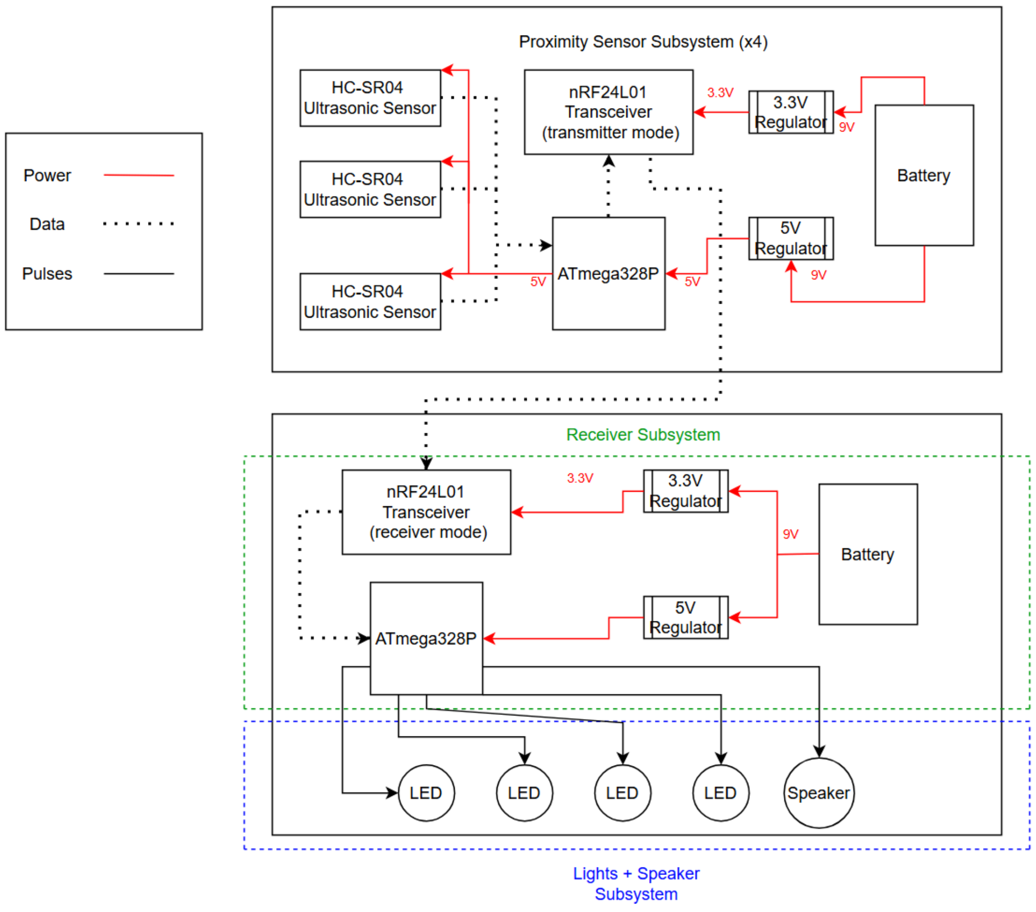

Power
Data
Pulses
Proximity Sensor Subsystem (x4)
HC-SR04 Ultrasonic Sensor
HC-SR04 Ultrasonic Sensor
HC-SR04 Ultrasonic Sensor
nRF24L01 Transceiver (transmitter mode)
3.3V
3.3V Regulator
9V
Battery
ATmega328P
5V
5V
5V Regulator
9V
Receiver Subsystem
nRF24L01 Transceiver (receiver mode)
3.3V
3.3V Regulator
9V
Battery
ATmega328P
5V Regulator
LED
LED
LED
LED
Speaker
Lights + Speaker Subsystem

Figure 2: Block Diagram of Solution

As seen in our block diagram, we have three subsystems in our solution - the sensor subsystem, the receiver subsystem, and the lights and speaker subsystem - all of which are outlined in more detail below:

<u>Sensor Subsystem:</u>

This subsystem uses an ATmega328P - AU as its microcontroller, drawing power from a 9 Volt battery passed through a 5 Volt linear regulator. It takes in data from three ultrasonic sensors, determines which sensor is returning the lowest value, and then sends it to the transceiver. This transceiver will receive power through a 3.3 Volt linear regulator and be set to the “transmit” mode so it can send the data to the central module. This is repeated four times and is located within each external sensor module that will be placed on the car.

<u>Receiver Subsystem:</u>

This subsystem is located within the central module, and also uses an ATmega328P - AU as its microcontroller. Once again, we supply power through a 9 Volt battery which is passed through both 5 Volt and 3.3 Volt linear regulators for the microcontroller and transceiver respectively. This transceiver will be set to the “receive” mode, and will be simultaneously receiving data from the other 4 transceivers for the microcontroller to process.

<u>Lights and Speaker Subsystem:</u>

This subsystem is also located in the central module and is connected to the microcontroller in the receiver subsystem. It consists of 4 LEDs and a small speaker which are all simple digital outputs. The lights will be powered on depending on which sensor is in range of an obstacle, determined by the receiver subsystem.

Design:

One decision we made was to split up the sensors into 4 separate modules with a main receiver module. One of our early designs had only 2 modules, one for the front and back each, but we decided to split it into 4 as that would allow the design to adapt better to various sizes of cars. Many design decisions we needed to make were related to the components that we chose for each function.

For the main distance sensors, we chose to use the HC SR04 ultrasonic sensor, as it could read up to the range that we needed accurately, which was about 3 feet, as well as having low enough tolerances and a high enough field of view according to both the datasheet and our testing after acquiring the sensors. Another factor that influenced our choice of distance sensor was cost, as our overall solution is built for older model cars, which would be targeting a user base that would be more sensitive about cost, so it was important our components would be lower cost as well. The sensors cost about ~$1.50 a piece, which also influenced our choice to use it in our design. An alternative we considered would have been a LIDAR sensor, as it is more accurate and could sense through screens, which is not possible with the ultrasonic sensor and means it cannot be fully weather proofed, but due to the high cost of the LIDAR sensors and the unneeded level of accuracy for our purposes, we chose to stay with the ultrasonic sensors,

We also chose to use the NRF24L01 modules to communicate between the sensor modules and the main receiver module due to the ability of it to simultaneously receive from multiple transmitters at the same time, which enabled us to have each sensor module operate independently and constantly transmit distance data, as well as allowing us to only need one receiver chip in the receiver module which let us use the same PCB for the receiver module that we were using for the sensor modules. Additionally, cost was also a factor in this decision, as the transceiver modules cost ~$2 a piece, which is helping us keep our cost for the whole system down. One potential alternative would have been using bluetooth, but it was not possible from our research to have a single receiver simultaneously receiving from multiple sources without interference and with low latency, and additionally, the cost was about 2-3x that of the NRF24L01 module, so we chose to use the NRF24L01.

<u>Schematic:</u>

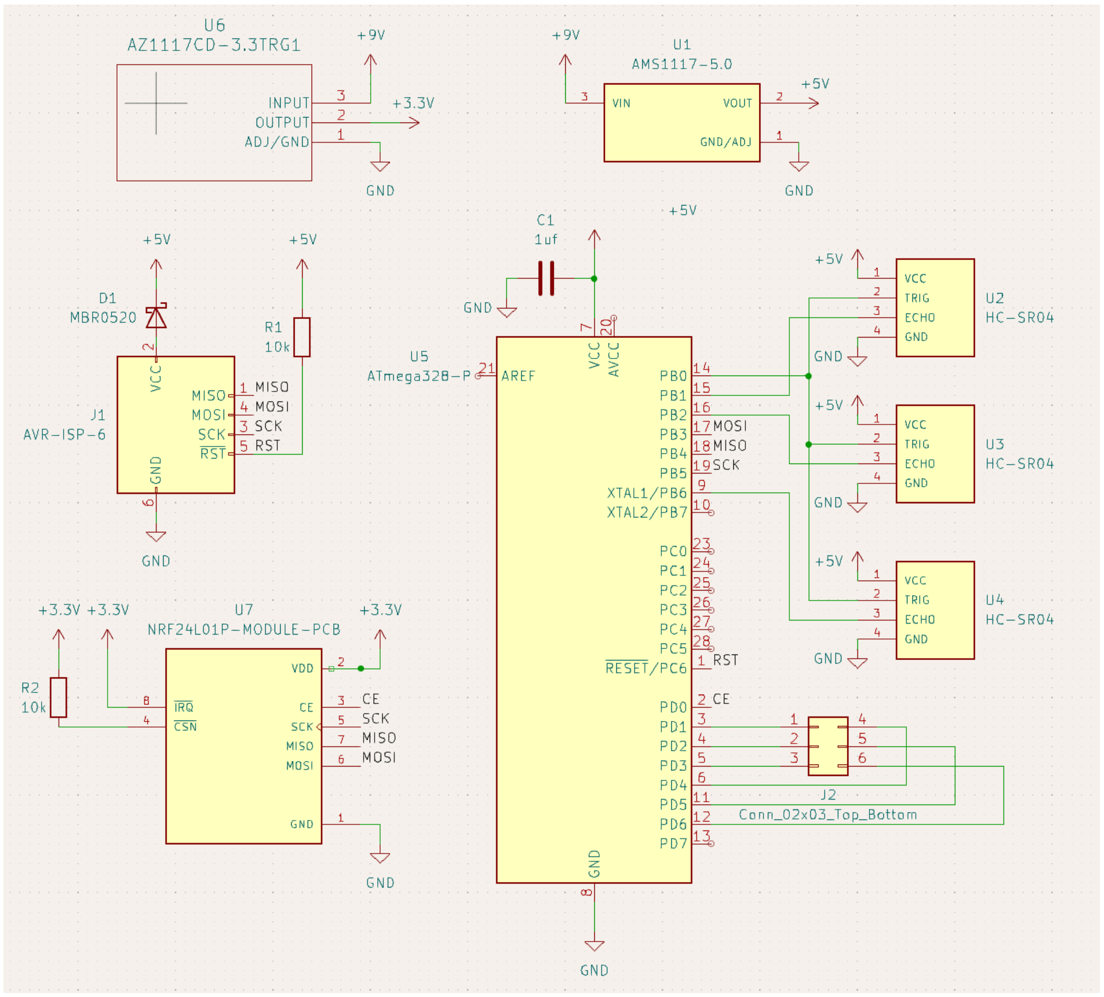


Figure 3 - Full Schematic for Sensor Module

Figure 3 above is our full schematic for our PCB. It consists of an ATMega328 microprocessor, with 3 ultrasonic sensors, a 5V regulator to step down the voltage from the input 9 volt battery, a 3.3 V regulator to supply power to the transceiver module, and the transceiver module itself. Due to the way we designed the PCB and the addition of 6 debug pins, we would also have been able to use the same exact PCB and design for our central receiving module, by attaching LEDs in place of the ultrasonic sensors and on the debug pin, as well as attaching a speaker to a debug pin, and changing the code.

PCB:

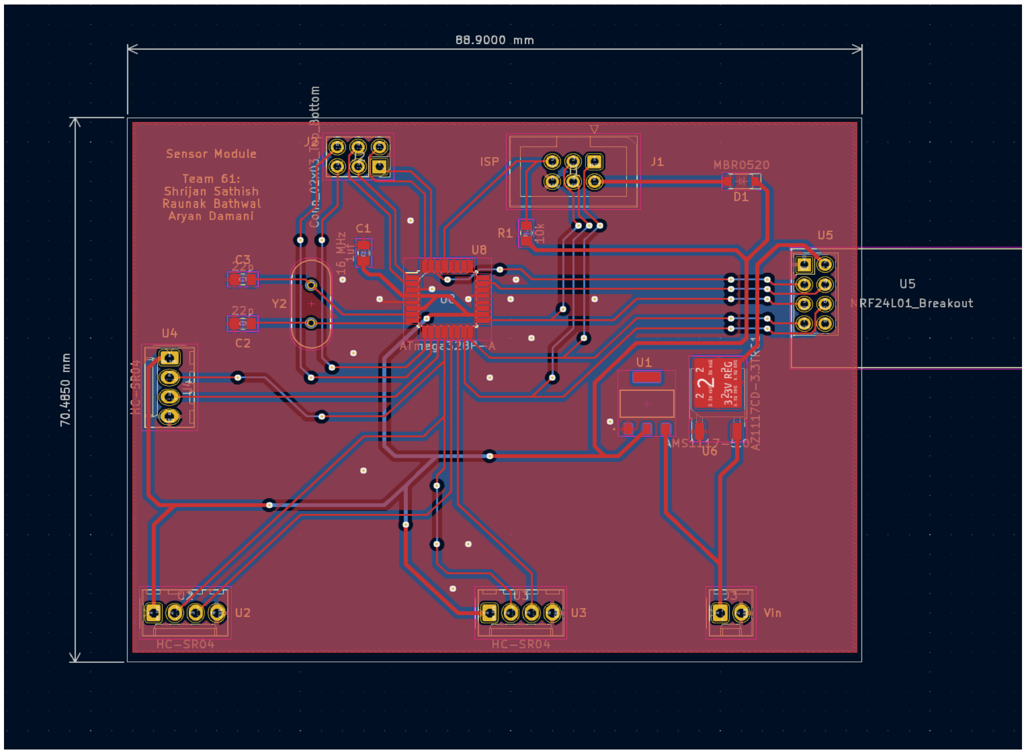


Figure 4 - Final PCB Design

Our PCB design is shown in figure 4 above, which was done with lots of empty space for the sake of debugging as we do not have any significant spatial restrictions on our PCB design. It was also recommended to place the receiver module off the edge of the PCB to minimize any potential signal interference from the other PCB connections.

CAD:

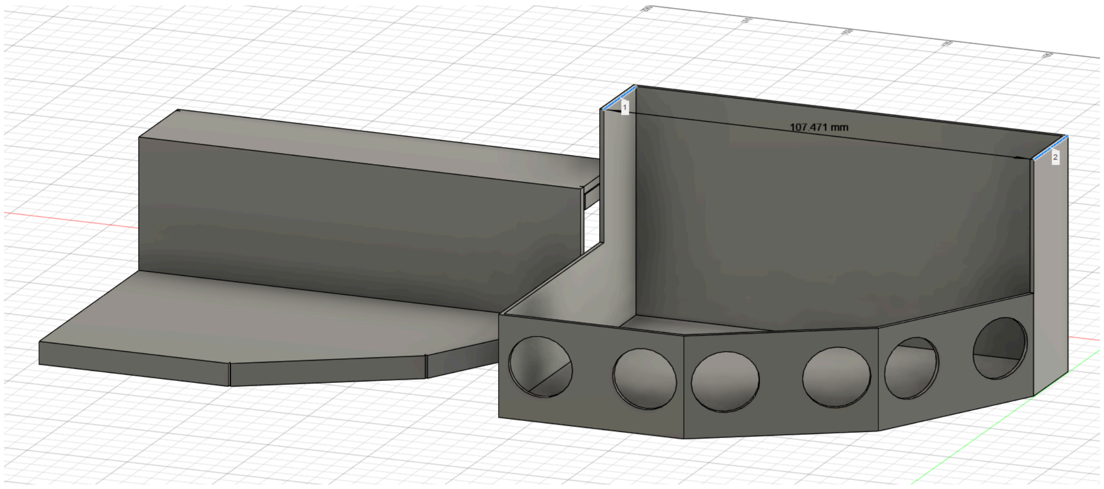


Figure 5 - 3D printed housing for sensor module

The housing allows 3 ultrasonic sensors to be placed within the cutouts, and the plastic allows some amount of weatherproofing to protect the internal components. All internal components, including the PCB, battery, and transceiver will be placed within this housing during operation.

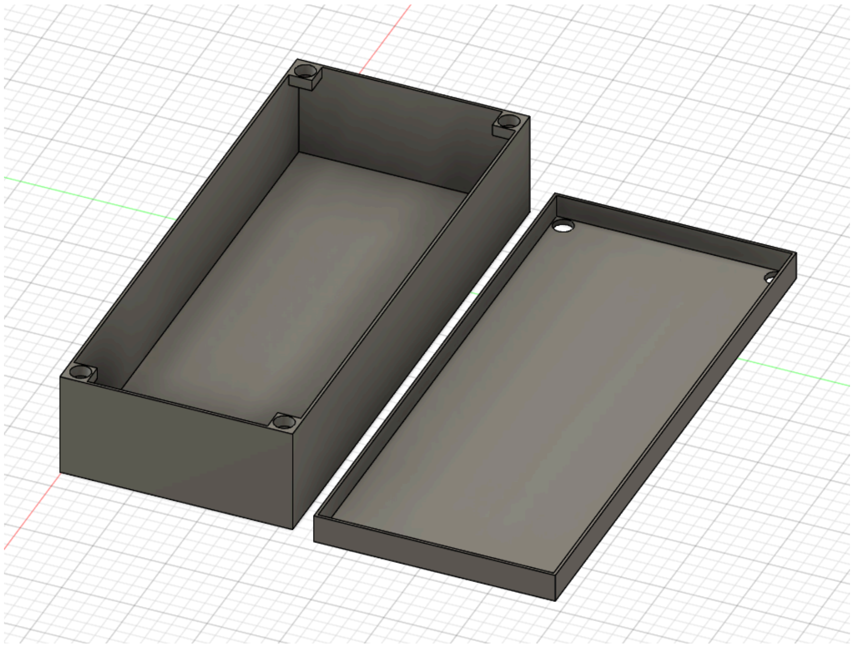


Figure 6 - 3D printed housing for central module

This will contain the PCB, speaker, battery, and the cutouts are for the 4 LEDs so that the driver can see them,

<u>Sensor Angle Considerations:</u>

One part of our design that was critical to its success is the proximity sensors' ability to detect objects in front of the car. Since each ultrasonic sensor has a detection cone of 30 degrees, we believe that having 3 per unit, each offset by 30 degrees will be a good way to sense 90 degrees, from the side of a given car to the front bumper of the car.

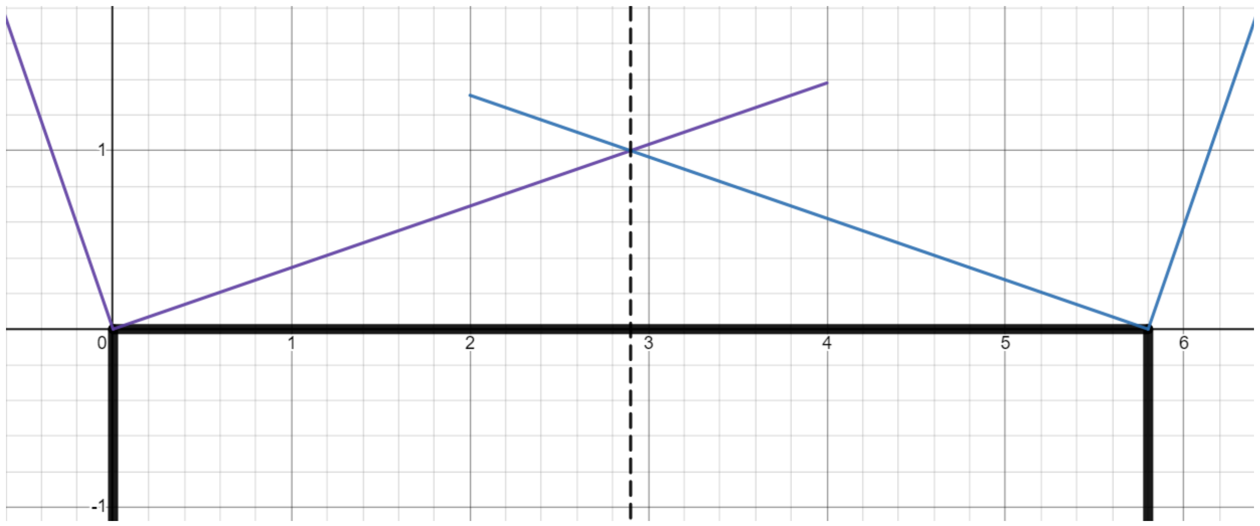


Figure 7: Graph plot outlining detection angles for 3 ultrasonic sensors placed together

Using a graphing application, and assuming an average car width of 5.8 feet, we can see that sensors placed at an angle such that they start detecting at a distance of 1 foot in front of the car allows us some leeway at the edges of the car as well, which could help with object detection. We would simply have to warn the user that once the "beeps" coming from the receiver reach a frequency that makes it sound like a constant tone, that an object is closer than one foot from the car.

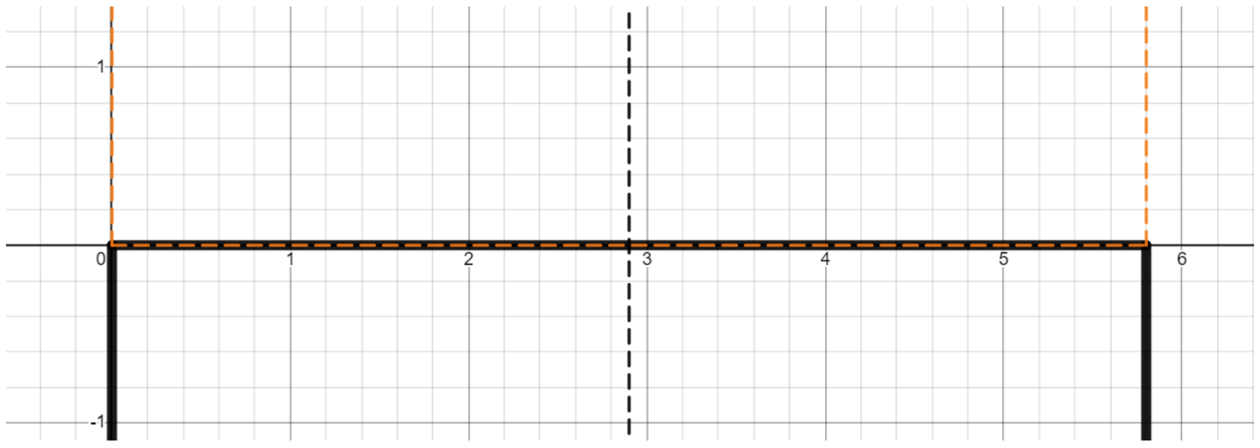


Figure 8: Detection angle (orange dashes) if ultrasonic sensors are aligned with the bumper of the car

If we sacrifice that little extra distance away from the edge of the car to detect objects that are strictly in front of it, the 90 degree detection area will allow us to measure objects right up to the bumper, but also only strictly in front of the car (as seen in the graph above).
This provides us with enough confirmation that regardless of the detection angle we choose, we can use three ultrasonic sensors per detection module in a way such that we will be able to detect obstacles in front of or behind the vehicle at any given time.

According to the HC-SR04 datasheet, the formula to calculate distance based on the time recorded by the ultrasonic sensor is as follows: Centimeters = ((Microseconds / 2) / 29). Using the graph seen in figure 4, we can calculate an example of what a threshold distance might look like. If we take one of the sensor modules placed on the left side of the car (relative to figure 4), and one of the ultrasonic sensors closest to the bumper, we can calculate the max distance ahead of the car it can detect with a 30 degree cone.
Using $tan(30) \ = \ 0.57735$, we can calculate the slope of a 30 degree angle and plot it on the graph:

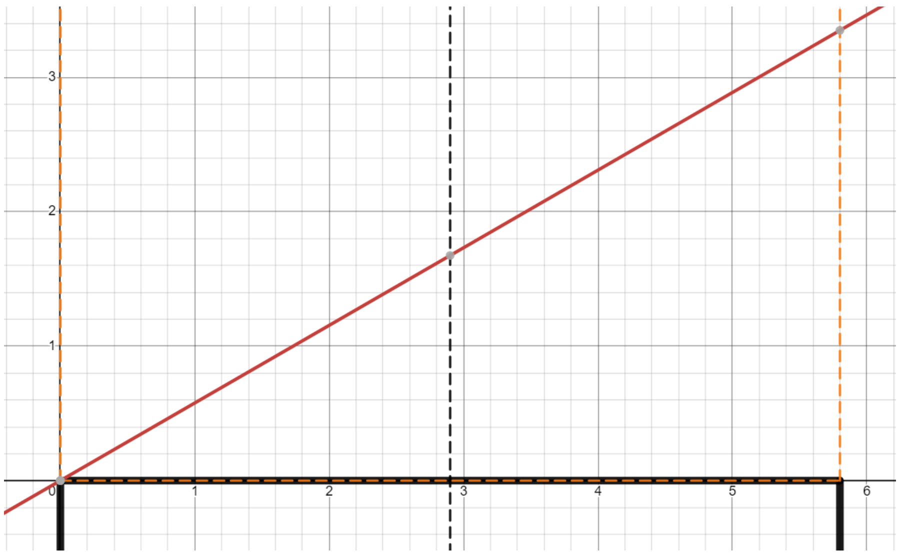


Figure 9: Max angle of detection of ultrasonic sensor aligned with bumper of car

As seen in figure 5, we have 2 intersection points at the midsection line of the car and the opposite edge of the car. By hovering over these points, we get a distance from the car of 1.674 feet at the midpoint, and 3.349 feet at the opposite end. Let's use the midpoint distance of 1.674 feet as an example, and convert this to 51.02352 cm. Using the formula from the datasheet, we can calculate that 51.02352 = ((Microseconds / 2) / 29), meaning that the threshold for microseconds would be 2959.36416. This time data can be used to trigger the microcontroller to send signals to our LEDs and speaker to flash and beep at a certain frequency, which ensures that our design will work as intended.

Verification:

After completing the design we had in mind and implementing it, we were able to conduct our checks and verification to ensure that the solution would work as intended. We did this in three parts, with a set of tests to perform for each subsystem.

Sensor Subsystem Verification:

The first subsystem we tested was the sensor subsystem, which was done for each of the four modules independently. Table 1 below shows our expectations for this subsystem:

Table 1: Requirements and Verification Table for Sensor Module

Requirements	Verification
The proximity sensor must be able to detect objects that come within 0-3 feet of the car continuously.	We will place an object at various points between 0 and 3 feet from the sensor and record the output from the sensor to check accuracy within +/- 5 cm.
It must be able to consistently update the receiver module about distance from obstacles and delay between sensor and receiver is minimal (less than human perceivable)	We will record the output from the receiver module while we put an object in front of the sensor and detect the delay continuously over 10 minutes, with an alert that triggers if the delay is over 50 ms at any time.
Sensor reading is continuous, and is able to detect within ~15 degrees on each side of the sensor, horizontally and vertically	We will place an object 15 degrees from the center of each sensor and record the output from the sensor to see if it is detected.

The first verification in Table 1 was conducted as figures 10 and 11 show below:

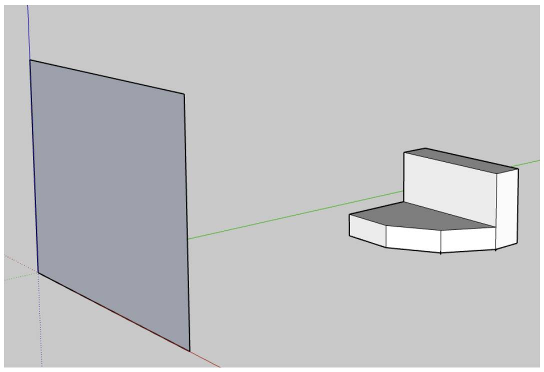


Figure 10: Simulated Verification of Object Detection

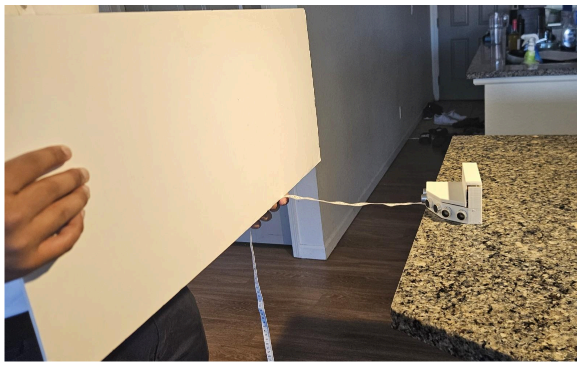


Figure 11: Actual Verification of Object Detection

After conducting these tests, we collected data by printing the sensor distance data to the serial monitor in the Arduino IDE, and organizing our results into Table 2 below. The data we collected proves the success of the sensors' accuracy within the module as well as the fact that the software is updating with the closest possible obstacle.

Table 2: Difference in Actual versus Measured Distance to Sensor Module

Actual Distance (cm)	Measured Distance (cm)	Difference (Measured - Actual) (cm)
5	6	-1
10	12	-2
15	17	-2
20	21	-1
25	26	-1
30	32	-2
35	36	-1
40	41	-1
45	47	-2
50	54	-4
55	58	-3
60	65	-5
		Average Difference: -2.083

As seen in Table 2, the average distance between the actual distance versus the measured distance from the object to the sensor was just over 2 cm, meaning that this surpassed our requirement for the sensors to be accurate within 5 cm of an object.

To verify that the delay between the transceivers was not humanly detectable, the success came from collecting no data. This is because the way we verified this was by setting an alert between the sensors that would print a message and terminate the program if the delay ever exceeded 50 ms. As we did not get any message like this for just over fifteen minutes, we deemed this part of the sensor subsystem as successful.

Lastly, we also needed to verify that the angle of detection was indeed 15 degrees from the center of each of the three sensors placed in each sensor module. To do this, we took each sensor we planned to use and tested them individually for their cone of detection; we placed an object 30 centimeters at various angles up to 15 degrees away from the center of the sensor. The data we collected is below in Table 3, showcasing a detection error of under 2 centimeters from the actual distance away, which we deemed as accurate enough since 2 centimeters is a negligible distance when applied to the scale of cars.

Table 3: Verification of Sensor Detection Angle

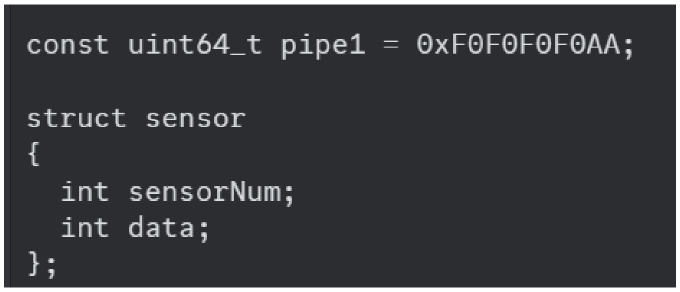
Angle (degrees)	Distance Measured (cm)
-15	29
-10	27
-5	29
0	30
5	28
10	28
15	29
Average	28.5714

<u>Receiver Subsystem Verification:</u>

Table 4: Requirements and Verification Table for Receiver Subsystem

Requirements	Verification
The receiver inside the car must be able to receive signals from the proximity sensor consistently and update the microcontroller with the new data	We will test in 2 conditions: open air and through the car. In either condition, the receiver must print the correct data from the sensor subsystem to the console. This will be done by holding an object at a known distance away from the sensor, and making sure this same distance is received at this subsystem.
The receiver must be able to discern between the different sensors located around the car	We will hold an object in front of each sensor's detectable range, one at a time in order to set up this verification. Then, using print to console messages, the receiver must accurately state from which sensor it is receiving data.

The verification of the receiver subsystem heavily depended on making sure it was receiving accurate data from each of the four sensor modules through the car, and that it could discern which sensor it was collecting data from at any given time. For the first verification that it was receiving data, we tested in open air and verified that the distances printing from the transmitter in each sensor module were the same values on the receiver side in the central module, which they were. We then moved on to testing this through a car by placing the sensor module in front of the car while keeping the receiver inside the car, and we noticed no change in delay nor data sent, meaning that this was a successful verification.



```
const uint64_t pipe1 = 0xF0F0F0F0AA;

struct sensor
{
  int sensorNum;
  int data;
};
```

Figure X: Sensor Struct in Code

We also sent our data as structs that contained labels on which sensor the data was from, as seen in figure X (1 = front left, 2 = front right, 3 = back left, 4 = back right). These labels were accurately printed when tested, meaning that we passed the second requirement and our receiver subsystem was working as intended.

<u>Lights and Speaker Subsystem Verification:</u>

Table 5: Requirements and Verification Table for Lights + Speaker Subsystem

Requirements	Verification
The LED on each corner (four total) must light up with its corresponding proximity sensor as a visual cue to the user	The four sensors will be placed in the same orientation as they would be on a car. Then, an object will be held in the detectable range of each sensor, one at a time. This should light up the corresponding LED in this subsystem.
There must be audible and increased frequency beeps as the car gets increasingly closer to an obstacle, giving an audio cue for distance	An object will first be held at a non-detectable distance threshold, and slowly move towards the sensors. At the same time, the speaker must produce beeps at a higher frequency, starting at 1Hz when first detected and increasing to a single tone when right in front of the sensor.

Lastly, we had to verify our third subsystem consisting of the lights and speaker that would notify the user of an obstacle. We did not collect data in tables for this set of verifications as they were fairly easy tests to conduct and observe success. For the first verification, we placed the four sensor modules in the same orientation as the LEDs on our central module. We then showed a hand in front of each sensor module - first individually, then multiple at the same time. When this was done, The corresponding LEDs lit up every single time with no detectable delay. The second test involved slowly moving an object towards the sensor, and making sure the speaker increased in frequency as the object came closer. When this test was conducted, the speaker made no noise at a distance beyond 3 feet, started beeping at 1 beep per second (1 Hz) when beginning to enter the threshold distance, and increased in frequency until it reached a single tone when the object was less than 0.5 feet away from the sensor. This proved the success of our lights and speaker subsystem.

Costs:

Labor:

The average salary for a computer engineering graduate from UIUC is $109,176, which equates to roughly $52 an hour. We each dedicated 10 hrs/wk per person for 9 weeks, which will give ($52/hr) * 2.5 * (10 hrs/wk * 9 wk) = $11700 total for labor per partner.

Table 6: Cost Calculations

Part	Price	Total w/o Lab-Owned	Total w Lab-Owned
HC-SR04 (Ultrasonic Sensor) x 12	1.3 x 12	15.60	15.60
NRF24L01 (Transceiver) x 5	2 x 5	10	10
PCB x 5	3.50 x 5	17.50	17.50
Red LED x 4	0.10 x 4	0.40	0
Speaker	2.10	2.10	2.10
9V Battery x 5	5.50 x 5	27.50	0
Mini Breadboard	1.00	1.00	0
Wire Connectors	5.00	5.00	0
3D Printed Filament	5.00	5.00	5.00
	Total:	84.10	50.20

If we were to attempt to make this project commercially viable, we use our estimate of around $84.10 + $11700 for parts + labor costs, we get a total of $11784.10 to make the product. Now if we want to mass produce this product, we would have to do research into the market where this product would be purchased. Around 20-30% of all cars driven in the US are older than 2010, and are not equipped with built-in proximity sensors, giving a very large potential user base for our product.

Schedule:

Table 7: Schedule of Work

Week 7	Design Review with TAs, Start PCB Design, Review part list
Week 8	Review part order and submit, Continue PCB design
Week 9	Spring Break
Week 10	Finalize PCB Design, Start Receiver build with RasPi for test
Week 11	Create Housing, Start module creation with PCB, Start Microcontroller programming for module
Week 12	Write receiver logic for output to lights and speakers, Complete receiver
Week 13	Test thoroughly and debug any functionality
Week 14	Add extra functionality if necessary
Week 15	Practice for demo

Conclusions:

Our design works as a suitable add-on for those who drive older model cars without built in proximity sensors to gain more knowledge about their surroundings so they can maneuver their vehicle with greater caution and precision than they would have been able to without our solution. The data is accurate enough for a driver to have it be of use to them, and the data is communicated in an intuitive way to the driver such that they would be able to maneuver their car safely and can rely on our system to do so. Our accomplishments include successful fabrication of a PCB that was able to be programmed and read sensor data while outputting values, but we ran into issues with using the transceiver module on the PCB so we were unable to use it for our final demo and instead replaced the PCBs with Arduino UNO to demonstrate functionality. Additionally, we succeeded in low latency data transmission over the air, with multiple transmitters all being able to send readable data without interference to a single receiver. Additionally, our main driver feedback system, the combination of LEDs and speaker, worked well to communicate distance data to the driver in a way that the driver could easily parse and adjust for. In the future, should we continue with this project, we would want to have our PCB work properly, which would allow us to have a much smaller form factor, which would enable easier mounting as well as decrease the drag created by the modules. We would also want to have a slightly smarter algorithm to process distance data, as occasionally the ultrasonic sensors do return garbage outlier data that we need to account for. Additionally, we would add power switches to each sensor module, to preserve battery life as well as make it more convenient, and have the central module be powered via usb from the car itself, to preserve power as well.

Ethics and Safety:

In accordance with the IEEE Code of Ethics (7.8, I - III), we guarantee that our group will uphold the highest standard of integrity and safety when designing and testing our proximity sensor.

Firstly, as a team we divided work equally and adhered to IEEE's Code of Ethics concerning treating all persons involved with fairness and respect. We enlisted the help of TAs and professors and had a harassment-free workplace when engaging in the design and engineering process (IEEE II, 7-9).

As our design obviously involves real vehicles, safety is quite a large concern and required rigorous testing to make sure that our design will not fail in any condition. The biggest safety concern arises from our sensor's ability to detect and transmit accurate distance data to the microcontroller in the receiver, which we promise to be fully transparent about (ACM 1.3). We made sure to only run simulation tests where no person or property can be damaged (ACM 1.2). An example of such a test would be to keep the sensors stationary and at a height and distance that would mimic a car, and move objects slowly towards them.

Ultimately, we plan to uphold the highest standards of design, safety, and integrity throughout this project, as seen in the IEEE Code of Ethics, the ACM Code of Ethics, and laws/regulations pertaining to the practical use of obstacle detection sensors on cars. We aim to produce a product that delivers an extra edge of safety and convenience to owners of older model cars while adhering to rigorous ethical and safety standards.

References

1. "IEEE Code of Ethics." *IEEE*, www.ieee.org/about/corporate/governance/p7-8.html. Accessed 8 Feb. 2024.
2. *Code of Ethics*, www.acm.org/code-of-ethics. Accessed 9 Feb. 2024.
3. ElecFreaks, "Ultrasonic Ranging Module HC -SR04," 2011. https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
4. "nRF24L01 Wireless RF Module," Components101, Apr. 30, 2021. https://components101.com/wireless/nrf24l01-pinout-features-datasheet