EchoSense: Personal Surrounding Safety Detection System

Team Advisor: Professor Hung Cao and Floranne Ellington
Andrew Cassidy (Hardware and Firmware Developer)
Jerry He(Android Application Developer)
Kohsuke Hirano(Android Application Developer)
Isaac Yen (Firmware Developer)

Abstract:

EchoSense is a personal surrounding safety detection system. Its goal is to alert cyclists of approaching cars. There are two objectives for EchoSense development. The first object is to develop a hardware prototype that uses the LIDAR module to scan the distance between a cyclist and an approaching car. The hardware calculates velocity using the set of distances and sends a Bluetooth Low Energy attribute value 1 to the Android application if the approaching car travel too fast. The second objective is the Android application development. The application read the Bluetooth attribute value from the hardware and notifies the cyclist of the approaching car if the value is 1. Both objectives were developed in parallel to complete a functional prototype before the deadline on December 6. The hardware is fully functional. The software is 80% complete, as more work is needed to enable the application to continuously scan signals from the hardware and notify the cyclist when the attribute value is 1.

Introduction:

The number of cyclists on the road has increased steadily in recent years. [1][3] Consequently, fatal road accidents are more likely to involve cyclists. [2] Some of these accidents are attributed to inattentive cyclists, reckless drivers, or collisions between cyclists and drivers. Team EchoSense aims to build a device to alert cyclists of approaching vehicles so they know whether they are in danger. The team decided to build EchoSense, a Personal Surrounding Safety Detection System that reads the approaching vehicle's speed and alerts the cyclists through Bluetooth if the vehicle travel at high speed.

Initially, EchoSense wanted to use Ultrasonic Distance Sensor HC-SR04 to detect the approaching car. However, the team discarded that idea because of the ultrasonic sensor’s limited range. Instead, EchoSense chose LIDAR-Lite v3HP sensor to detect approaching vehicles.

Team EchoSense decides to develop the hardware/firmware and the Android application in parallel. The hardware/firmware uses the LIDAR sensor. The sensor uses a laser and time of flight measurement to determine its distance from an object. We calculate the velocity of an approaching vehicle by taking multiple readings of the car's distance from the device. The Android application reads the Bluetooth Low Energy attribute advertised

by the hardware to alert the cyclists of approaching cars. The team develops and tests the products incrementally to catch any existing bug and fix any issue that it encounters. The hardware is fully functional. The application is 80% completed. Team EchoSense still needs to implement a feature that allows EchoSense's application to continuously read the Bluetooth attributes from the hardware. The team strives to create an accurate personal surrounding safety detection system for cyclists on the road.

Methods

To increase team EchoSense's efficiency in developing EchoSense prototype, the team divided the project into two major parts: hardware/firmware development and the Android Application development. The team developed the two parts in parallel. It implemented and tested new codes incrementally to ensure ease of debugging and functional software implementations. The team also used a Gantt Chart to keep track of the deadlines, progress, and milestones.

Hardware

EchoSense's hardware consists of a Garmin LIDARLite v3HP Lidar module for rangefinding, and the OSHChip microcontroller module as the main controller. The OSHChip is based on the Nordic nRF51822 microcontroller, and has onboard Bluetooth LE support in addition to an ARM Cortex m0 CPU and 32 kilobytes of RAM. Power is supplied by a standard 9 volt battery, which is converted to 3.3V and 5V for the microcontroller and LIDAR module respectively using linear voltage regulators.

During development, we ran into some problems with the voltage regulators failing. After this happened twice, we added a pair of zener diodes to limit the voltages to within the tolerances of the OSHChip and LIDAR module to prevent any further damage.

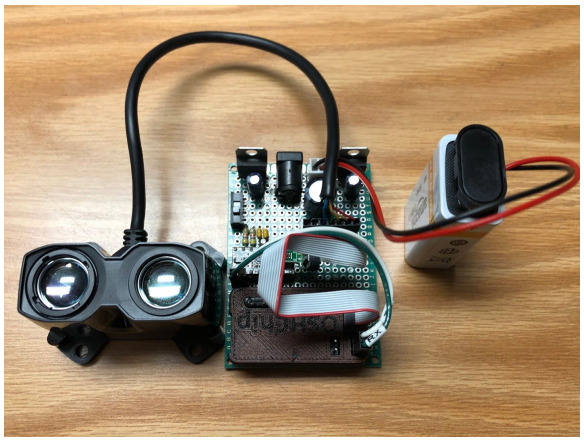

Figure 1: EchoSense prototype hardware

Firmware

EchoSense's firmware was written in C++ on the mbed framework, and uses the LIDARLite_v3HP library for communicating with the LIDAR module.

At startup, the LIDAR module and job system are both initialized, and the bluetooth broadcasting is started. The job system begins calling a tick function at a preset rate, currently set to once every 50ms, or 20hz. The tick function acquires the current reading from the LIDAR module, and uses it to calculate the velocity using the following formula:

$$Velocity = (\Delta distance) \div \Delta time$$

If the velocity is greater than the maximum cutoff speed of 60 km/h, EchoSense assumes that the velocity is a false positive. The average distance and the old distance between the approaching car and the cyclist will be assigned to the newly measured distance. However, if the velocity is greater than the trigger speed of 5 km/h and less than maximum speed 60 km/h, EchoSense will send a BLE attribute value of 1 to the EchoSense's Android application. The application will warn the cyclist of the approaching car.

Android Application

The goal of the mobile application for the EchoSense system was to notify cyclists when dangerous objects are approaching via the notification system that is built into the mobile phone. When the hardware issues an alert through Bluetooth, the mobile application will issue a notification to alert the cyclist.

The mobile application for EchoSense is currently available on Android devices. The application is written in Java on Android Studios IDE. It is compatible with any android device that is at API level 23 and above; this means that the android device needs to have an Operating System based on Android 6.0 and up installed. The android device will also need to have Google Play Services and Google services framework installed.

When the application is launched, it asks the user for bluetooth and location service access. The application requires bluetooth to communicate to the hardware, and it also requires location for BLE services. This allows users to control bluetooth connectivity from within the application.

The Android application is developed using the built-in platform support for BLE, short for Bluetooth Low Energy (android.bluetooth) [4]. This built-in platform have all the classes and methods for connecting to BLE devices. The application utilizes these classes and methods to scan, connect, and communicate with BLE devices.

Each BLE device has its own GATT (Generic Attribute Profile) that defines the way data is transferred between devices. The EchoSense Android Application obtains the GATT profile from the EchoSense hardware when a bluetooth connection is made between the Android device and the EchoSense hardware. The application then obtain characteristic information from the GATT profile; the characteristic contains values set by the EchoSense hardware. The application can then read from the characteristic; when the application sees a rising edge from the characteristic, it outputs a sound notification and vibration (if supported by the device) to alert the cyclist.

The development of this application faced a lot of difficulties. The first iteration of the application was able to turn on the bluetooth functionality of the Android device but was not able to scan for nor connect to the EchoSense hardware. As it turns out, there are two different Bluetooth API for android; one for Bluetooth Classic and one for Bluetooth Low Energy. The first iteration of the EchoSense application used

the API for Bluetooth Classic which does not support Bluetooth Low Energy.

The second biggest difficulty on the Android application development is reading the characteristic. A lot of time were spent on debugging reading characteristic as the application was not getting the right value from the characteristic or it was not continuously reading from EchoSense Hardware.

In the future, it will be a wiser idea to utilize the Nordic Bluetooth Android Library to develop the app as this library have better support for the hardware for the EchoSense hardware since the hardware is made by the same company.

Prototype Results

EchoSense has completed its hardware and hardware. If the team moves a fast-moving object toward EchoSense's sensor, the device sends an attribute value 1 to the connected mobile device. The team tests the attribute value using the mobile application nRF Connect. If an object is moving faster than 5 km/h but less than 60 km/h toward EchoSense, the attribute value changes from 0 to 1 on the nRF app. Team EchoSense also take false-positive into consideration in that if something suddenly moves sideways into EchoSense's range, EchoSense will not alert the cyclist of that object. This false-positive elimination feature is implemented through maximum velocity condition, in that the sideway moving object will not trigger the EchoSense's notification because its movement will be perceived as greater than the maximum speed of 60 km/h in the EchoSense. That way, EchoSense will only detect the fast-approaching car and eliminate all other false positives.

Summary

This report showcases the design of a personal safety system for cyclists that utilizes the LIDAR module. The design uses the LIDAR module in conjunction with a Bluetooth module as well as an Android application to notify the cyclist. The LIDAR module captures the approaching object behind the cyclist and calculates its velocity. If it determines the object is approaching too fast, it will send a signal to the cyclist's mobile phone via Bluetooth to alert them. The figure below showcases the basic idea of this design.

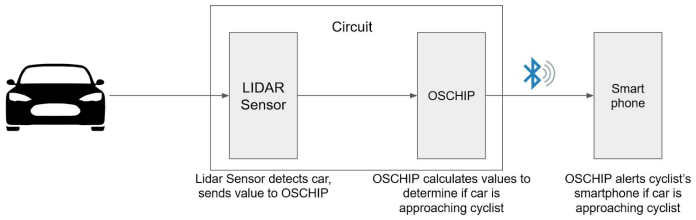

Figure 2: OSHChip operation diagram

The main objectives were met in this design. The LIDAR module is able to determine whether the object is approaching too fast by calculating its velocity using a set of captured distance from the approaching object. The LIDAR module then outputs a signal via Bluetooth and it is successfully received by the Android application. The only problem remains is for the Android application to continuously scan for signal from the LIDAR module. The team will solve this problem in the winter quarter.

Conclusion

This report examined the development of a personal safety system for cyclists. The two main objectives of this design so far were to develop a working prototype for the hardware and a functional software that can detect the signal from the hardware. So far, a majority of both objectives were met. The prototype for the hardware is fully functional; it is able to detect approaching objects and determine whether it is safe or not. The software is also 80% functional as it can receive the signal from the hardware. The only steps left are to do more real world testing for the hardware, and to finish the Android application so that it can continuously scan for signal from the hardware and notify the user when it receives the correct signal.

Acknowledgement

Team EchoSense would like to thank our advisor Floranne Ellington and Professor Hung Cao for giving advice to the team on building the prototype EchoSense. The team would also like to thank Philip Freidin at Fliptronics for providing the OSHChips, and the EECS159A instructional staff for letting the team borrowing the Power Supply in the lab. Lastly, Team EchoSense would like to thank every member on the team for completing the prototype.

References

[1] Steve I. "Bicycle accidents in the United States people powered movement", People Powered Movement, 2019. [Online]. Available: https://www.peoplepoweredmovement.org/bicycle-accidents-in-the-united-states/. [Accessed: Nov 6 2019].

[2] L. Watson and M. Cameron, "Bicycle and motor vehicle crash characteristics," Monash University Accident Research Centre, Melbourne, Victoria, 2006.

[3] "Road safety factsheet," *The Royal Society for the Prevention of Accidents*, Nov-2017. [Online]. Available: https://www.rospa.com/rospaweb/docs/advice-services/road-safety/cyclists/cycling-accidents-factsheet.pdf. [Accessed: 07-Dec-2019].

[4] "Bluetooth Low Energy Overview," *Android Developers,* N.d. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le. [Accessed: Oct 30 2019].

Appendix

Technical Standards

UART: EchoSense's design team chooses UART standard to enable serial monitoring for debugging. EchoSense is compliant to the UART standard because it is capable of serial monitoring.

I2C: EchoSense's design team chooses I2C connections to enable serial monitoring. EchoSense is compliant to I2C because its circuit connection is based on I2C standard.

BLE: EchoSense's design team chooses Bluetooth Low Energy standard for communication between the device and mobile application because nRF51822 is

compatible with Bluetooth Low Energy. EchoSense uses Mbed BLE source code to develop its notification services to be compliant with the BLE standard.

Constraints

The major constraints that EchoSense faced were interoperability, power, and scheduling.

The mobile app is a key component of EchoSense. The interoperability problem stemmed from EchoSense' initial inability to develop a mobile application that scans and read BLE information advertised by EchoSense. To overcome this constraint, the mobile app development members used Nordic Bluetooth Android Library to implement functions that read the BLE information from EchoSense.

Another problem EchoSense faced was power supply in that EchoSense's voltage regulator failed and the battery overloaded the microcontroller. To prevent this problem from happening again, the design team added additional diodes to limit the power supply's voltage.

The last constraint was scheduling. Each EchoSense's members have different schedules and assignments. This reduces the team's development time for EchoSense. All members of EchoSense partitioned some of their time in the week to prototype EchoSense and split the tasks so the team can develop its hardware, firmware, and the mobile application in parallel in this limited development time.

These major constraints will prevent the team from completing its prototype if they are not fixed. It is necessary to overcome these constraints.

Security Issues

Currently, EchoSense only broadcasts a single bit of information, depending on weather or not a vehicle is approaching from behind. This is not making use of any kind of encryption or built in security features of the Bluetooth LE protocol. The only security concern with the prototype as it currently exists would be a hacker spoofing the signal to prevent the rider from being alerted.

As EchoSense is further developed, it will likely have configuration data and more information being broadcast over the Bluetooth bands, some of which could be sensitive information. We plan to begin using the built in Bluetooth LE security features if required.