Electronic Deer Warning System

David Zhuo **Computer Engineering**
davidzhuo@live.com
Anlang Lu **Electrical Engineering**
anlanglu001@gmail.com

Andrew Danowitz **Faculty Advisor**
adanowitz@calpoly.edu

California Polytechnic State University
San Luis Obispo, CA
Computer Engineering Department
December 2016

Abstract

Deer-vehicle collisions (DVCs) are extremely dangerous, often injuring or even killing drivers. Unfortunately, this form of automotive accident is commonplace in the United States. According to the NHTSA, DVCs result in 200 human deaths a year.[2]

Despite these deadly incidents, there currently are no deployed federal or state systems for preventing DVCs. There are many consumer electronic deer deterrent products, but their long-term effectiveness is questionable.[3] In fact, there does not appear to be much research into electronic deer deterrent systems. Aside from constant audio output and electric shock, no other means of electronic deterrent exist. Even if fixed deterrents were effective at repelling deer, these would further divide shrinking animal wildlife ecosystems, so this would not be a viable ecological solution. As such, different methods of preventing DVCs need to be explored.

One option would be to warn deer of incoming vehicles. Deer are intelligent and are capable of understanding potential harm. This project is to provide an easy means to experiment with different deer warning methods by delivering a prototyping system. As such, the system should be simple to understand and widely extensible.

Contents

Figures

Tables

Introduction

Deer-vehicle collisions (DVCs) are extremely dangerous, and can injure or even kill drivers. Unfortunately, this form of automotive accident is commonplace in the United States and shows no signs of disappearing. As our country expands urban and suburban developments into forests and undeveloped land, these types of accidents will become increasingly common. According to the National Highway Traffic Safety Administration (NHTSA) General Estimates system, fatal crashes involving animals increased by 39% over a 6 year period, from ~111/yr between 1992 and 1995 to ~154/yr between 1998 and 2001.[1] By 2010, DVCs resulted in 200 human deaths a year.[2]

Despite the growing frequency of these deadly incidents, there currently are no deployed federal or state systems for preventing DVCs. There are many consumer electronic deer deterrent products, but their long-term efficacies are questionable.[3] Most consumer products emit a constant audio output or simply deliver an electric shock, but deer quickly acclimate to these deterrents and learn to ignore them.[4a] There does not appear to be any known means of adequately repelling deer electronically and automatically. As such, different methods of preventing DVCs need to be explored.

One option would be to warn deer of incoming vehicles. Deer are intelligent and are capable of understanding potential harm. Using Pavlovian training, deer can learn to associate a stimulus with danger. If some electronically generated tone is sufficient as a stimulus, it would be possible to warn deer automatically of incoming cars.

The goal of this project is to provide an easy means for researchers to experiment with different deer warning methods by delivering an electronic deer warning prototyping system. To prevent ecosystem fragmentation, this would not be a fixed deer deterrent. Instead, it would be a flexible system that warns deer only when there is an approaching vehicle.

The primary means of detecting a moving vehicle is via seismic vibration. It is known that deer can hear ultrasonic tones up to 30 KHz (inaudible to humans), so the system will create these sounds with piezoelectric buzzers.[4b] The firmware was designed to be simple to understand, with helpful abstractions provided to future programmers. This is important so that researchers are not hampered by the code and understanding the system, and can instead focus on research and parsing data. Additionally, the hardware circuitry involved with data collection and power management was created to be as power-efficient as possible so that the system will require as little maintenance as possible. By providing a low-power hardware platform, the system can be deployed in a remote location and collect data for longer periods of time and create larger sample sets for observing the effect of certain deer warning systems.

System Design

Since this is the first iteration of the electronic deer warning prototyping system, there was no previous version of this project to base our work on. As a result, much of the work is foundational, rather than the implementation of a final product. For this project, work was divided between software design and hardware design with little mutual dependency. Currently, the system remains in two parts, but the interface between the two parts was clearly defined from the beginning and constantly refined so that integration between the two systems is not anticipated to be a challenge.

System Capabilities

The hardware for this project is divided into four parts. First is the power electronics, which provides power to the system with a high efficiency. The system is designed with power saving in mind, consuming less than 1 mW during normal operation. The system is capable of lasting up to a year with a reasonably sized battery. Second is the seismic sensor circuitry, this circuit amplifies and filters the seismic sensor input to produce a 0 - 3.3 V signal for the ADC. Third, there is a driver circuit for the piezoelectric buzzer, which can produce an ultrasonic noise. Lastly, there is the microcontroller for scheduling operations and interpreting data.

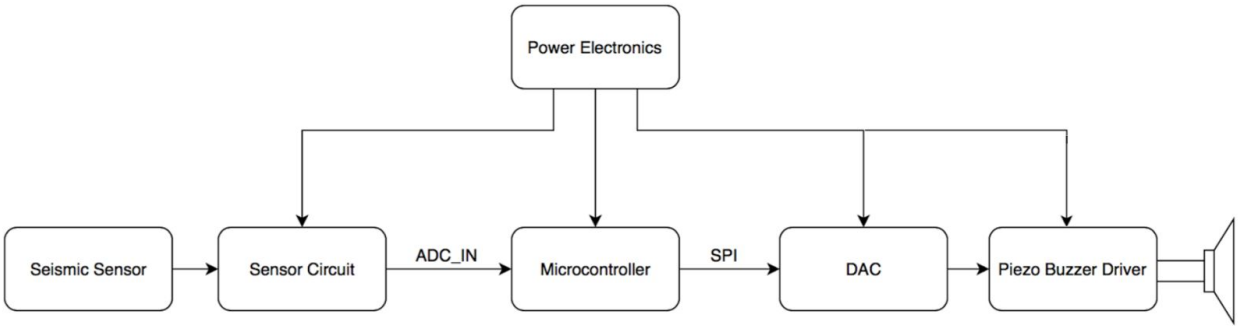

Figure 1a: System Block Diagram

The software portion of the project runs on the STM32F401RE Nucleo development board, shown in Figure 1. The board has a 84 MHz ARM Cortex-M4 chip with 512 KB Flash memory and 96 KB SRAM. Integrated into the development board is the ST-LINK/V2-1 programmer. An indication LED is provided for rudimentary information display and a user button is provided for system and board debugging.

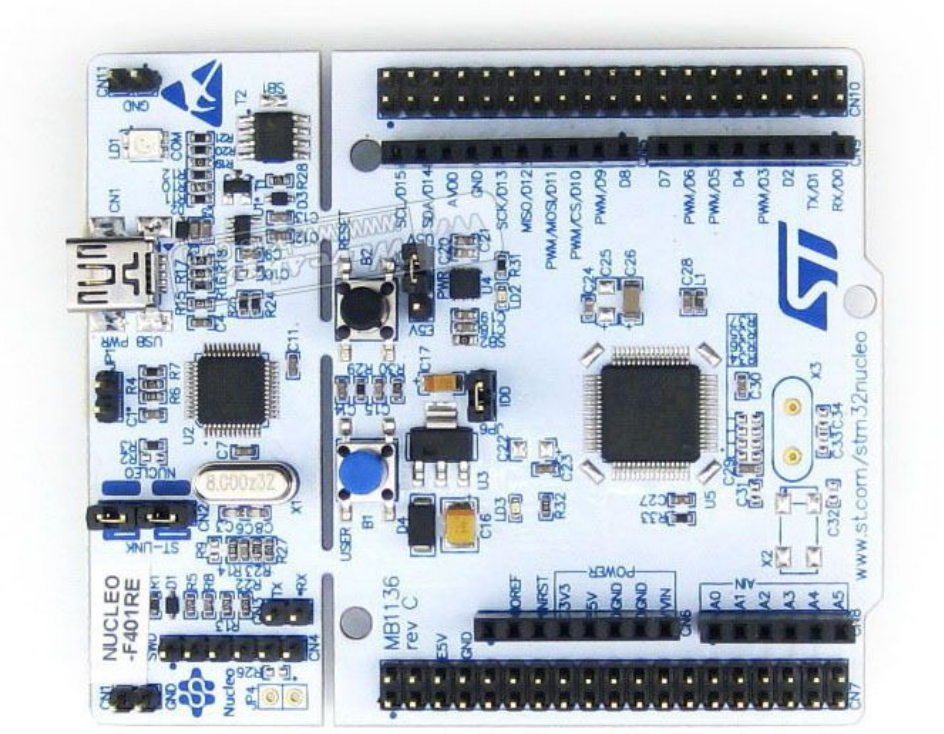

Figure 1b: STM32F401 Nucleo

The Cortex-M4 chip comes with a floating-point unit and is capable of performing complex fast Fourier transform (FFT) and other digital signal processing methods. There is an on-chip 12-bit analog-to-digital converter (ADC), and it is capable of sampling at hundreds of KHz synchronously, and much faster with direct memory access (DMA). Additionally, the chip can communicate with other ICs and devices via the Inter-Integrated Circuit (I^2C) protocol, Serial Peripheral Interface (SPI), or Universal Asynchronous Receiver/Transmitter (UART) protocol at a variety of speeds. At the moment, the chip interfaces with a digital-to-analog converter (DAC) using SPI to create output waveforms for driving a piezoelectric buzzer. The microcontroller is capable of entering a sleep state where it consumes ~2.1 μA.

Firmware & Software

The firmware driving the system is a finite state machine (FSM). This operating model can be represented as a flowchart, as shown in Figure 2. It is straightforward and a staple of software engineering with a well-defined input and output scheme. In the code, this is simply a switch-case structure in a persistent loop. Each state describes a set of actions the system will take.

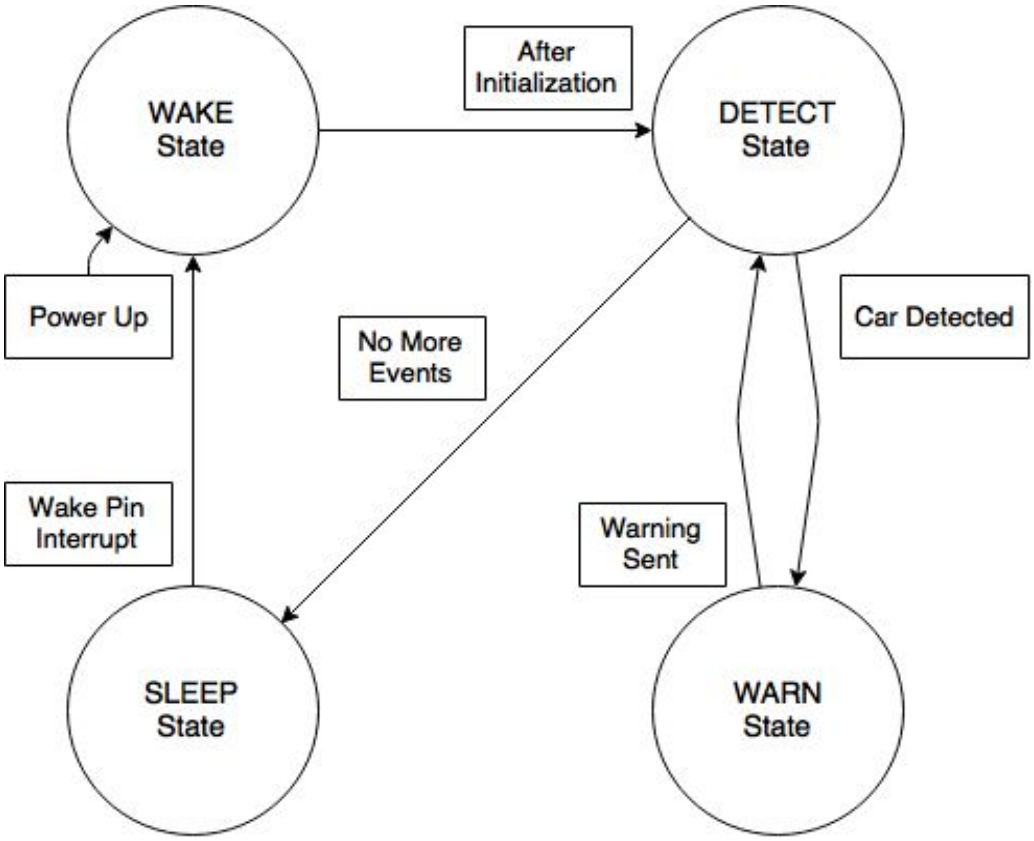

Figure 2: State Machine Flowchart

Program Flow

Upon device power up, The WAKE state initializes the STM32F4's system clock and each of the on-chip hardware peripherals, and it allocates dynamic memory for computations. If the project is compiled with DEBUG_MODE defined, the UART module and LED indicators are also enabled with the rest of the system.

After initialization, the device transitions directly into the DETECT state. Here, the board collects samples from the on-chip ADC, which is connected to the seismic sensor circuitry. After a sufficient amount of samples have been collected, a complex FFT is performed on the dataset and its magnitude is calculated. In future iterations of this project, there will be a function that matches the measured seismic signature with the seismic response of roads to cars to determine whether or not the sampled seismic waves indicate there is a car nearby. Currently, the firmware simulates this function and simply reads from a static integer array that simulates a car approaching.

In the case a car is not detected, an internal count will increment. If a car is detected, the count is reset and system enters the WARN state. Here, the system performs deer-warning activities. This is where future developers will modify the output to control the sound-generating hardware. For now, there is placeholder code for controlling an external DAC via SPI. After issuing a warning, the system transitions back to the DETECT state.

When the internal count reaches the limit for missed detections, the count is reset and the system moves to the SLEEP state. The board prepares itself for low-power sleep by turning off the system clock and enabling the wake pin. When the sensor hardware detects significant seismic activity on the road, it will pull the wake pin high and the board will transition back to the WAKE state.

Sampling & Fast Fourier Transform

Based on research on car seismic vibrations at Georgia Institute of Technology, seismic waves generated by a car can be detected up to 100 meters away. [5] A car traveling at 45 mph will cover this distance in about 5 seconds. It is a requirement to warn deer before this time. Presently, our microcontroller can collect samples and perform an FFT in less than 4 seconds. However, this does not include time for matching signals to cars or for allowing deer to react. Ideally, this system would be faster and various ways of speeding this up are discussed in the Future Work section under “Improving Detection Time”.

To detect an incoming car during the DETECT state, the on-chip ADC samples the output of the sensor circuitry. Analysis of the signal is performed in the frequency domain rather than in the time domain.

With synchronous sample collection, 84 MHz clock operation, sample and hold time of 480 clock cycles, and divide-by-6 clock prescaler, the ADC samples at around 29 KHz. To divide the sample rate by numbers aside from powers of 2, the system downsamples further by performing additional dummy ADC reads between actual samples. The real sampling rate is acquired by taking the number of measured samples divided by the time reported by the HAL_GetTick(), which is a time generated by timer interrupt.

Currently, the system has a sample window of approximately 3.995 seconds and collects 1024 samples. This results in a real sampling rate of 256 Hz, so the Nyquist frequency is 128 Hz. Given 1024 samples, the FFT length is 512. The frequency-bin resolution is 0.25 Hz per bin. This should result in a reasonably accurate frequency measurement. Given that the useful seismic spectrum of a car lies between 1-30 Hz[5], this should be enough frequency resolution for the primary range as well as a few harmonics above.

It is important to accurately sample lower frequency components of the seismic wave. Capturing 8 periods of the lowest frequency component should mitigate effects from the rectangular windowing function imposed by the sampling method. However, sampling 8 periods of a 1 Hz wave would be unreasonable long, so 2 Hz was chosen as the new lowest frequency component. 8 periods of a 2 Hz signal is 4 seconds.

There are caveats to this approach. If this sampling scheme is not accurate enough, then the number of samples and the sampling rate can be increased. However, care should be taken to minimize the time for this computation so that the system can warn deer of an incoming car. The sampling window should not be expanded, and the number of samples (size of the FFT calculation) should not increase dramatically.

Code Organization

Because the firmware is designed as an FSM, the code describing each state is modular and should not require much modification to reuse in a completely different system. All of the files have a clearly defined purpose, as shown in Table 1

Table 1: Source Files

File	Function
main.h	Important constants and peripheral header libraries
main.c	State machine and system initialization code
states.h & .c	Describes the code that will execute in each state and how each state transitions to the next
detect.h & .c	Collects seismic samples, performs FFT and calculates magnitude
warn.h & .c	Controls system output for creating warnings
utility.h & .c	A set of useful and miscellaneous functions, such as debug printing or making HAL_Delay more power efficient

The C header files describe how to use each of the functions in comments, important constants and enumerations for operation, and justifications for the mathematical operations. The headers are intended to be clear enough to understand the functions without reading the C code. All of these source files are located in the project subdirectory ‘src/deer_deterrent_firmware/’

Version Control

To keep a log of software development and set the groundwork for future collaborators, all of the software is stored and managed with the Git version control system. The project Git repository contains all of the code needed to create the program for the microcontroller and either contains or describes where to get all of the software needed to compile the program.

The root of the project repository contains a README document that describes the purpose of each important project directory. Digging into each of these project directories will be another README that elaborates on its purpose and describes how to update the code or approach that section of the project.

Libraries

The software uses the Cortex Microcontroller Software Interface Standard (CMSIS) library for hardware-accelerated digital signal processing and complex math on ARM microprocessors.

Also, CMSIS provides board definitions for controlling hardware directly. This information is provided by the microcontroller vendor, which, in this case, is STMicroelectronics.

Additionally, STM provides a hardware control library called the Hardware Abstraction Layer (HAL). The API is portable across any STM32 microcontroller, and the project code that uses the HAL API is mostly compatible with any other STM32 system, with exceptions coming from different peripheral configurations on different boards. Roughly half of the project code involves the HAL API, but since the hardware calls are abstracted, no additional HAL code should need be written and the software should be increasingly portable across STM32 systems.

CMSIS runs directly on the hardware, but also provides a high-level abstraction for a developer to use. HAL uses the low-level parts of CMSIS for some register definitions. Figure 3 illustrates how CMSIS and HAL relate to each other.

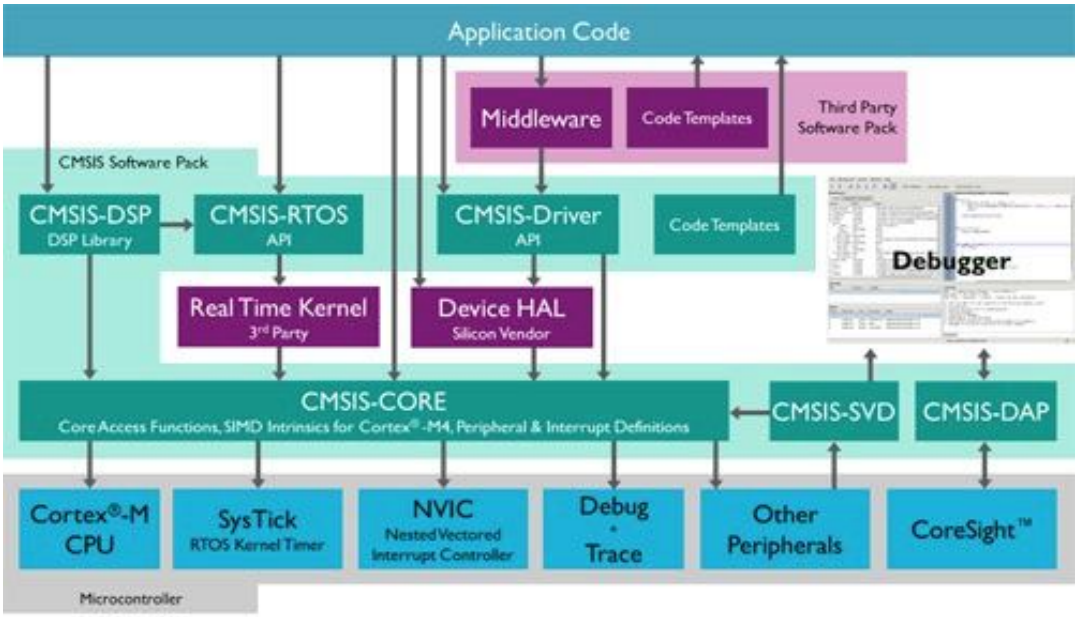

Figure 3: How the HAL library, CMSIS library, and our application code relate to each other

Included in the Git repository are makefiles for building each of these libraries and detailed instructions for building them.

For the HAL library, running ‘make clean && make’ within ‘Drivers/’ should regenerate the static library, ‘libstmf4.a’. Whenever any new peripheral is enabled, such as I^2C or I^2S, this library will need to be recompiled. In ‘Drivers/Makefile’ starting at line 22, compilation flags can be uncommented to include certain peripherals.

For the CMSIS library, 3 separate makefiles are supplied. ‘Makefile’ is the top level makefile that manages execution of the sub-makefiles. ‘Makefile.inc’ contains a set of definitions for

compilation, and ‘sub.mk’ compiles object files for each source directory in the CMSIS library. The Makefiles for this can be found in ‘Drivers/CMSIS_Makefiles’ and includes a script that installs sub.mk in ‘Drivers/CMSIS’ as well as instructions for building CMSIS. The generated library is ‘libdsplib_lm4f.a’.

Moving forward, both the CMSIS library and the HAL library can be updated by downloading the latest source files and recompiling.

Compilation & Deployment

To compile the software portion of the project, a few software development packages are needed beyond standard C compilation and build tools:

1. GNU ARM Embedded Toolchain
2. Texane’s open-source STMicroelectronics ST-LINK tool

The GNU ARM Embedded Toolchain is required to compile the C code to machine code that the STM32F4’s ARM processor can execute. The ST-LINK tool is used to transfer the machine code from the programmer’s computer to the development board.

A makefile is provided to build and deploy all of the software for MacOS Sierra and Ubuntu 16.10. It is located in src/deer_deterrent_firmware/

To compile and link the code, use either ‘make release’ or ‘make debug’. The ‘make release’ option compiles the code without the UART module and without the -ggdb debug flag and is intended for full deployment into the field. The ‘make debug’ option is intended for development and for reading system status during code execution, so it is highly recommended that code be tested in debug mode while developing. Currently, UART messages are sent with system initialization, state transitions, and FFT outputs.

To deploy the software, execute ‘make flash’ to move the compiled object code to the STM32F4 board.

Hardware

Overview

At the current stage of development, the hardware stages are shown in Figure 4.

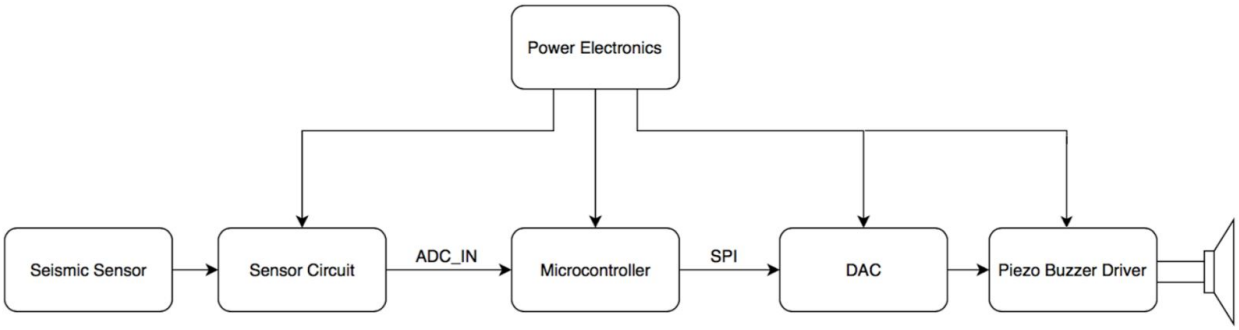

Figure 4: System Hardware Block Diagram

The power electronics circuit provides a 3.3V rail and a 1.8V rail to the rest of the system at around 85% efficiency.

The sensor circuitry filters the output of the seismic sensor with a cutoff frequency of 100 Hz, and amplifies the signal by 500 times.

The ADC samples the input signal and sends the signal to the microcontroller for signal analysis. If a car is detected the microcontroller sends a waveform to the piezo buzzer driver through a DAC to produce an ultrasonic sound.

Circuit Design Considerations

Piezo Buzzer Driver

For the first prototype, the functionalities of the piezo buzzer driver is basic. A common source amplifier amplifies a waveform from the microcontroller, and drives the buzzer with battery power, because battery rail has the highest voltage in the system without requiring additional circuitry. The schematic is shown in Figure 5.

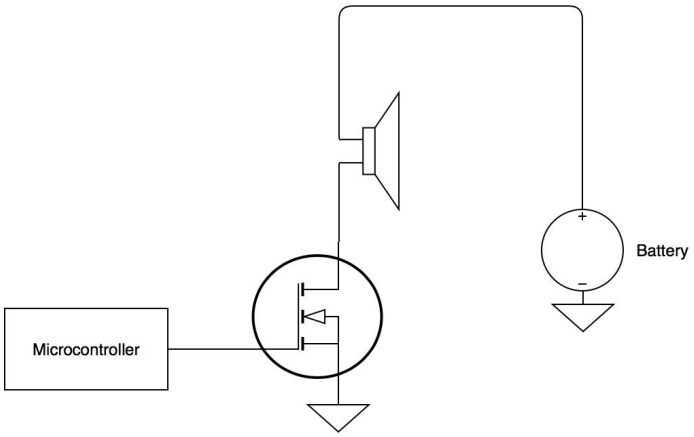

Figure 5: Piezo Buzzer Driver Schematic

The only customizable part of the waveform driving the piezo buzzer at this moment is the frequency of the waveform. A DAC solution for this driver was implemented. However, because of the SPI communication required for the DAC, a high enough frequency waveform (~30 kHz) was not possible. The SPI implementation was slow due to a combination of issues, which are addressed in the Future Work section under “Improving SPI & UART Transmission”.

Currently, the driver will be capable of producing a tone. During future development, if a specific waveform is required, an analog circuit should be considered to transform the existing square wave into other waveforms (i.e. bandpass filter to produce a sine wave).

Seismic Sensor Circuit

The seismic sensor circuit is the most sensitive part of the system. The seismic sensor produces a small voltage difference when vibration is detected. Experiments are required to characterize the exact signal strength for a car driving on a road. However, it would be reasonable to assume the sensor would output less than 5 mV for a typical vehicle. Due to

the nature of a relatively small input signal, a circuit designed for low noise and high gain is implemented, as shown in Figure 6.

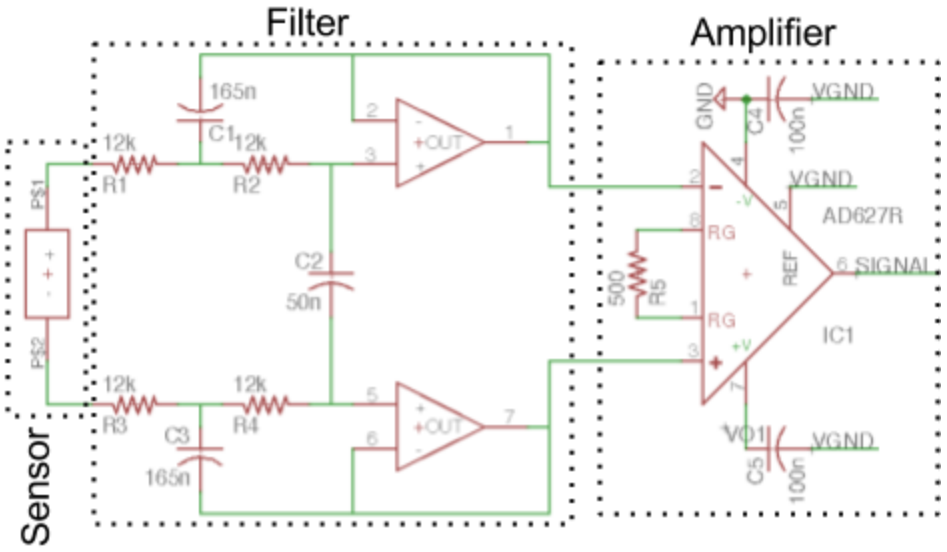

Figure 6: Filter and Amplifier Schematic

The sensor block represents the seismic sensor module. The output of the sensor is fed into a Sallen-Key lowpass filter, which includes the two op-amps and their supporting passive components shown as the filter block in Figure 6. The resistance and capacitance values are chosen for a Butterworth response with a cutoff frequency of 100 Hz. The reason Butterworth response is desired is for a Butterworth filter, the frequency response is flat in the passband. This particular butterworth filter is designed to provide unity gain for any frequency below 100Hz. The simulation result for this filter is shown in Figure 7.

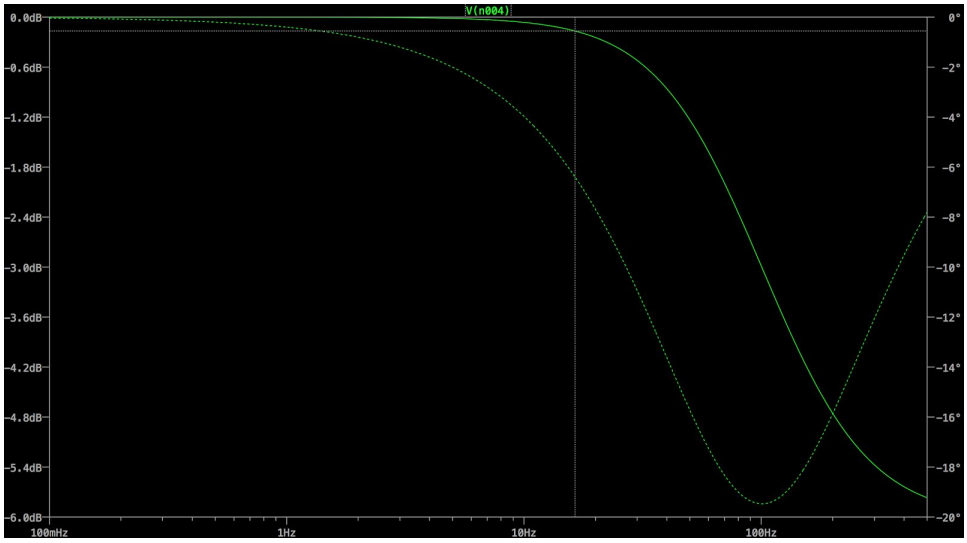

Figure 7: Frequency Response of Sallen-Key Filter in Schematic shown in Figure 6

The cutoff frequency is chosen at 100 Hz due to research indicating that most car caused road vibration is below 100 Hz, in fact averaging around 15 Hz[5].

An active filter topology (Sallen-Key) is chosen because of its high input impedance. The seismic sensor is not a perfect voltage source, therefore a high input impedance is required to prevent loading. The drawback for an active filter topology is that it requires a power supply. However, since a gain stage is required to make the signal useful for the microcontroller, the same power supply can be used for the filter stage. A low power op-amp is also selected to make the power consumption disadvantage insignificant. The op-amp selected for this project is the TSV522AIST, which has a typical power consumption of 200μA. The op-amp is rail-to-rail in order to provide maximum signal resolution. This particular amplifier is also automotive grade certified. This could be beneficial for this project because the application of this project is to deploy the system on a public road. Automotive grade reliability is highly desired for parts selected.

The filter circuitry is designed to be differential. This differential topology requires 1 extra op-amp than its single ended counterpart. The benefit of this topology is a very high common mode noise rejection. If in future development, if this implementation adds unnecessary complexity, it can be redesigned to be single ended. The only change required for this is to change C2 to 100nF and remove the bottom half of the filter circuitry. If this change is made, an analysis of the resulting common mode noise is highly recommended.

The amplifier stage implementation is a simple off-the-shelf instrumentation amplifier, AD627. Similar to the filter circuitry, its differential nature provides very high common mode rejection. At the same time, it provides a large gain with low drift. The amplifier typically consumes ~300μW of power, providing rail-to-rail output. This amplifier is not automotive grade, but is commonly used in medical devices. As mentioned before, automotive grade parts are desired, however, it was difficult to find a part with as low power consumption as this IC as well as automotive grade certification. For this prototype this part will be reliable enough. In future development, however, this part should be replaced by more reliable parts. The gain of this amplifier can be easily adjusted by replacing R5. 500 Ω is chosen for the time being. However, actual experiments on public roads is required to determine the best gain for this amplifier. The equation used to find the proper resistance can be found on page 15 of the datasheet for this amplifier [18]:

$$R = 200k\Omega / (gain - 5) \qquad [1]$$

Power Electronics

The seismic sensor circuitry requires a dual rail power due to the op-amp and the instrumentation amplifier. The output of the circuitry is sent to the onboard ADC for the microcontroller. Therefore, the output voltage range should be the same as the input range for the ADC which is 0 to 3.3V. In a dual supply amplifier setup, the output voltage would

swing above and below the reference. For the microcontroller to share the power supply with the sensor circuitry, the sensor circuitry must be referenced to 3.3/2 = 1.65V. Therefore the power electronics circuit is designed to supply a 3.3V rail and a 1.65V rail. A dual output switching regulator, MIC23250, is used with its adjustable output configuration, as shown in Figure 8.

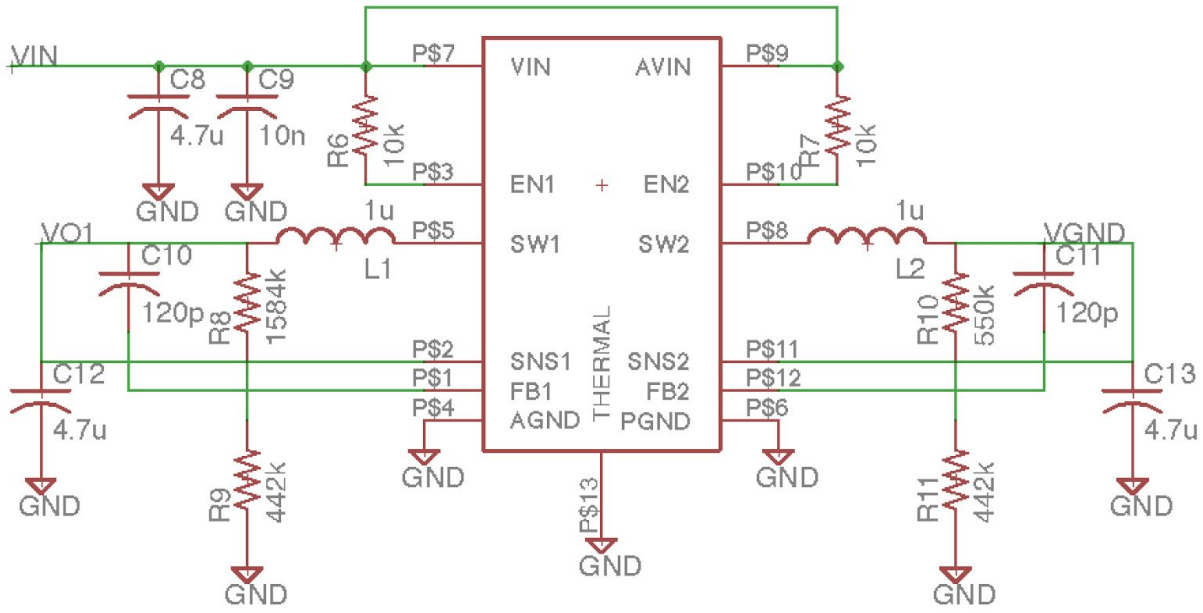

Figure 8: Power Electronics Schematic

The regulator IC supports 2 output rails. SW1 and SW2 pins are the outputs. SW1 is used to generate the 3.3V output and SW2 is used to generate the 1.65V output. EN1 and EN2 are the enable pins for the outputs. In this revision, power will not be shut off since the rails supply power to the microcontroller. Therefore the enable pins are pulled high permanently. In future revisions, if, due to power saving reasons, a rail needs to be shutdown at any time, a redesign of the PCB is required to route EN1 and EN2 to a GPIO pin.

C12, C13, L1 and L2 are the LC filters to smooth out the output voltage rails. These values are recommended by the regulator IC's datasheet[19] to be 1 µH and 4.7 µF. These recommendations are not the best values for all applications. This system draws under 3 mA for nominal operation, whereas these values are recommended for systems drawing up to 400 mA per rail. Currently these values are acceptable since there is no observable peaks on the output rails. However, in future developments as the circuit changes, peaks might be observed on the power rail. If this is the case, consider adding extra capacitors in parallel with C12 and C13.

R8, R9, R10 and R11 make up the resistive divider that through pins FB1 and FB2 dictates the output voltage. Refer to page 11 of the datasheet for this information [19]. To summarize, FB pin holds its voltage at 0.72 V through an internal comparator. Using SW1 as an example, to adjust the output voltage of SW1, use a resistive divider so that R8 + R9 ~= 1 MΩ, and desired Vout = 0.72 * (R8 / R9 + 1).

CAUTION: This IC is very sensitive to ESD. When handling this PCB, ESD precautions are highly advised. In future development, if the output voltage is not as expected, the first step in debugging should be to check if the regulator IC is damaged. To do this, measure the FB pins to see if the voltage is around 0.72V. The most convenient test points for FB pins are at the terminations of R8, R9, R10 and R11. If the FB pin is not 0.72 V, replace the IC. Refer to the PCB Layout Considerations section of this documentation for precautions when replacing the IC.

This IC has very high advertised efficiency, which is the main reason it is chosen for this system. It has advertised 94% peak efficiency, characterized at ~200 mA. However, at lower power ranges its efficiency is not as good as the peak efficiency. During operation, the seismic sensor circuitry draws around 800 μA. At this current level, with an input of 5.2 V, and output rails at 3.3 V and 1.65 V, the efficiency was recorded to be around 84%. More tests for power consumption will be required once the entire system is integrated, since efficiency will improve as more power is consumed.

PCB Layout Considerations

General Considerations

The PCB layout is shown in Figure 9.

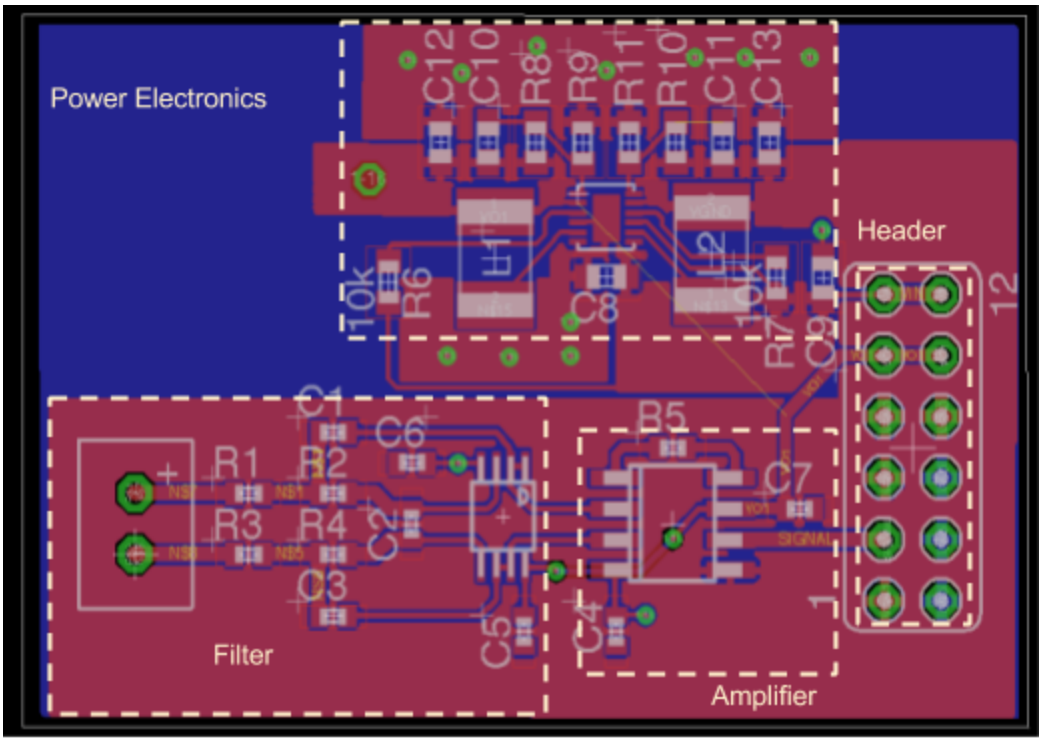

Figure 9: PCB Layout

The power electronics occupy the top of the board, the sensor circuitry occupies the bottom of the board. The piezo buzzer driver is currently on a separate breadboard.

The PCB is 2 layers, blue is bottom layer, and red is top. The bottom layer has only one short trace, and the rest of the layer is filled by GND plane. This is the 0V rail. The top layer has divided planes of GND, 1.65V rail and 3.3V rail. Refer to the layout file for details of those planes.

Headers on the right are used to interface with the board. Refer to the schematic for the exact pinout. To summarize, from left to right, on the top row is Vin, the 2.7-5.5V input from the battery; the second row is the 3.3V rail, use a jumper wire to connect one of the two pins to the header on the top left of the board (in the Power Electronics portion) as shown in Figure 10. This is required because the signals are not connected in the layout file in order to keep the GND plane continuous. This is mostly to protect the rest of the circuit from the switching noise of the power supply; The third row is 1.65V rail, this is labeled VGND in

schematic, not to be confused with GND which is the 0V rail; In the second to last row, the left pin is labeled SIGNAL, this is the output to the ADC. The rest of the pins on the right (bottom 3 rows) are all extra GND references. The pins directly above and under the SIGNAL are unconnected in this revision, and can be used for any extra functionalities if needed.

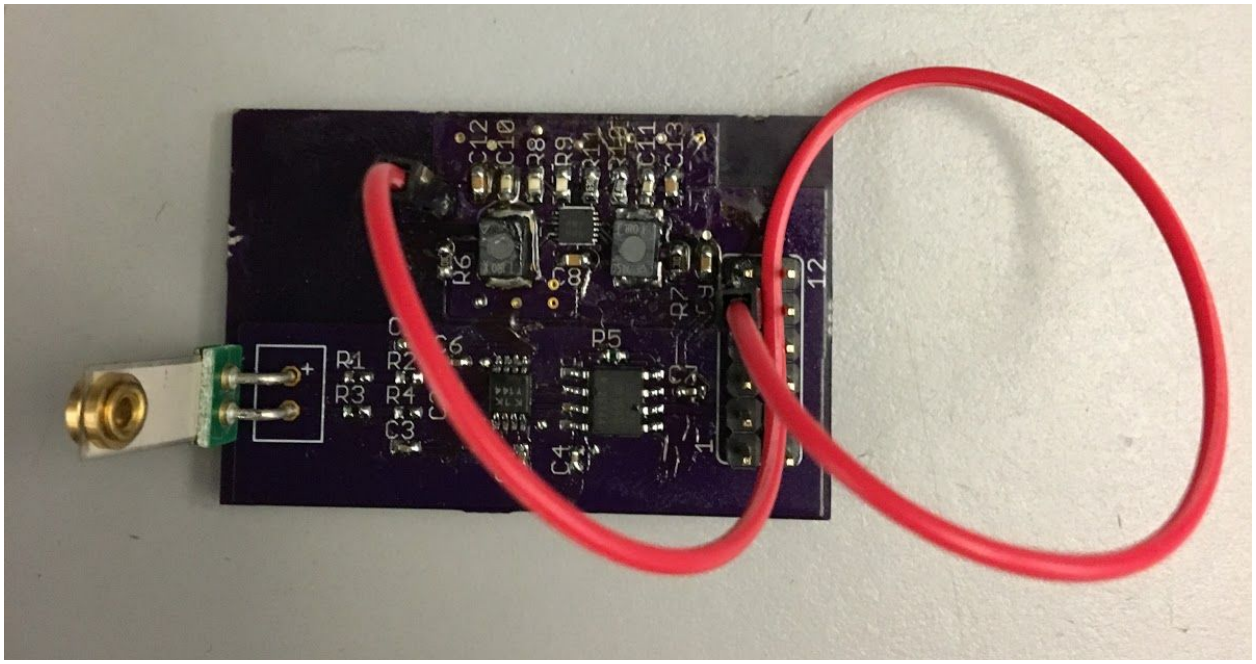

Figure 10: PCB with 3.3V Rail Connected

Modification Considerations

To check if the power IC is damaged, measure the voltage at top of R8 and R10. It is very likely the IC is damaged if either of these voltages do not read 0.72V. In event of the power IC being damaged, replace it using a heat gun. Use 30% max setting fan speed, and around 350 degrees celsius setting. Be aware that L1 and L2 do not have as high of a temperature rating as the IC itself, it is highly recommended to remove L1 and L2 first if the power IC is being replaced. Kapton tape is useful to keep all the other components in place while using the heat gun on the IC, it also shields other components from excessive heat, preventing damage to surrounding components.

To adjust gain of the seismic sensor circuitry change R5 using the equation:
R = 200kΩ / (gain - 5).

To adjust output voltage of either power rails, replace R8 or R10 instead of R11 and R9.

When purchasing replacement resistors, keep in mind that the power circuitry uses 0603 resistors, and the filter circuitry uses 0402 resistors. The 0402 pads on this PCB are

designed with extra space to make soldering easy, and it should not be difficult to try to solder 0603 resistors on 0402 pads.

Conclusion

No prior work for a system like this exists. Because of that, we were encouraged by the fact that we were creating a fresh system without any baggage from prior work. Developing a platform that outlasts many senior project cycles required a much longer view for each design decision. There were many challenges encountered while working on this system, such as cleanly dividing work between two team members such that mutual dependence was minimized. In doing so, the risk of diverging projects grew so meetings were held almost daily to keep the projects tightly coupled.

Major design decisions across a wide range of subdisciplines within both computer engineering and electrical engineering had to be weighed, and often we did not have the answers. Many meetings were spent debating the merits of different approaches. Often, the answers to our design questions were to choose sensible defaults, but make it simple for future developers to change or expand the system.

Future Work

Reliability Improvements

Currently not all parts are automotive grade. This is not an urgent issue right now since the system is aimed to be a platform for research and further development. However, the end goal of this project is to design a system to be deployed on public roads. This goal dictates the necessity for automotive grade parts. During future developments, this should be kept in mind, and designers should eventually move towards all automotive parts.

Piezo Buzzer Driver Improvements

Currently the piezo buzzer driver can only output square wave. To be able to output different waveforms, the simplest solution is to use a DAC. However, the DAC that was implemented was too slow for ultrasonic frequencies due to its SPI communications. However, a square wave of desired frequencies can be generated through the microcontroller. With simple analog filtering, a square wave can be turned into other waveforms. The next step should be to determine the waveforms that might be of interest and attempt to create those waveforms with analog filtering.

Any supporting code for generating waveforms of interest should go into warn.c. There are already functions defined that roughly emit sine and square waves. Improvements for

existing functions should go into emit_xxx() and additions should follow the same format as outlined in the switch-case structure in Figure 11.

```
// warn.h

typedef enum Waveform
{
    SQUARE, TRIANGLE, SINE
} Waveform;


// warn.c

void emit(Waveform w, float32_t frequency, int duration)
{
    switch (w)
    {
        case SQUARE:
        emit_square(frequency, duration);
        break;
        case TRIANGLE:
        emit_triangle(frequency, duration);
        break;
        case SINE:
        emit_sine(frequency, duration);
        break;
    }
}
```

Figure 11: Format for output waveform code in warn.h and warn.c

Explore Manufacturing Processes

The electronics system could be prone to failure due to the frequent vibration. Explore the possibility of implementing the circuit on flex PCB. Flex PCBs are much more resistant to vibration, and have a much smaller footprint.

Increasing System Service Life

Currently, the system has no plans for generating energy for extending the service life. Consider using solar panels to reduce maintenance intervals.

Car Characterization

A major part of this project is car detection. Currently, the data collection and processing parts of this project have been implemented. However, we do not yet have samples of the system capturing the seismic waves generated by a car. As a result, we do not have a seismic profile of a car, or a good way of matching samples to cars.

A future group will need to install the system by the road and collect enough positive samples of a car passing by and negative samples of random other objects moving by. Each of these datasets can be condensed by taking the FFT of the wave and generating a seismic spectrum.

One way to generate a reference car profile is by averaging the FFT spectrum of the positive samples. Then, to perform matching, the variance of the L2 norm can be computed. If the variance or distance is low enough, then there is some likelihood a car is nearby.

Another way to generate a reference profile is by training a neural network to discriminate between positive and negative samples. Then, instead of using the FFT as the profile, weights encoded in the network are the profile. To match a car, one simply needs to reimplement the neural network on the microcontroller and seed the weights with those found via training.

Improving SPI & UART Transmission

The HAL library function calls currently used in this project for SPI and UART are synchronous. Because these are blocking calls, the STM32F4 ARM processor executes transmission instructions line by line or waits for transmission confirmation before executing more code.

The system is capable of transmitting SPI and UART signals asynchronously, via interrupt or direct memory access (DMA). These alternatives allow the ARM processor to continue executing code while it waits for a transmission confirmation from the UART or SPI module. Asynchronous calls have the potential to improve power efficiency by reducing operating time, and the overall speed of the system.

In addition, the chip select line of the SPI module is not enabled. If the current GPIO solution is replaced with the SPI peripheral's NSS line, then each SPI transmission would be faster.

Faster SPI transmissions would enable higher resolution waveform output and unlock higher frequencies.

Using Power Efficient Standby Modes

The delay between the DETECT and WARN states is the inefficient HAL_Delay. Despite being slightly more efficient than the default implementation with cycle counting, this is still a non-ideal way of pausing code execution. STM documentation describes other sleep and standby modes that can be used.

Some experimentation with different sleep/standby modes is required to minimize energy consumption while maintaining correctness of code execution. These different modes disable clocks, peripherals, and memory, which may affect timing, device state and

availability, and stack and heap data. Additionally, some care should be taken to ensure that +/- 0.1 second accuracy is maintained to ensure the WARN state is entered soon after the DETECT state if a car is detected.

Disabling Unused Peripherals to Decrease Energy Use

None of the components are disabled or powered down in state transitions. For example, upon entering the SLEEP state, only the system clock is stopped. However, during this state, the SPI and ADC module can each be disabled.

In fact, each state only requires the use of a subset of the available system peripherals, so each state transition can call its own device initialization code. Table 2 describes the needed peripherals for each stage.

Table 2: Peripherals used in each state

State	**Peripherals**
WAKE	Power control clock, System clock
DETECT	Power control clock, System clock, ADC, GPIOB
WARN	Power control clock, System clock, SPI, GPIOB
SLEEP	Power control clock

In addition to disabling each of these peripherals, their respective clocks can be turned off.

Improving Detection Time

The sample window is currently 3.995 seconds. This is too long for an early warning system. The system can potentially detect a car up to 100 meters away. A car traveling at 45 mph will cover this distance in about 5 seconds. Ideally, this time should be cut down to give deer more time to respond.

One way to improve the response time is to decrease the size of the sample window. If fewer than 8 periods of the 2 Hz component is determined to be accurate enough to profile a car, simply decreasing the sample window should suffice. Additionally, if 2 Hz is not significant to profiling at all, that can be eliminated. If this is the case, then capturing 8 wavelengths of the lowest significant wavelength should decrease the sample window. (e.g, if the lowest significant frequency is 5 Hz, then 8 periods of it is 1.6 seconds.)

Additionally, if post processing techniques exist to improve the captured seismic signal with a shorter sampling window, then they should be explored.

References

Deer Information

[1] http://wildlifecontrol.info/wp-content/uploads/2016/04/Deer-Vehicle_factsheet1.pdf
[2] http://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1002&context=vpc18
[3] http://benthamopen.com/contents/pdf/TOTJ/TOTJ-4-87.pdf
[4a] http://edis.ifas.ufl.edu/uw371
[4b] http://onlinelibrary.wiley.com/doi/10.2193/2006-326/epdf

Road Vibration

[5] http://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=2172&context=icchge

Software Sources

[6] ARM GCC - https://launchpad.net/gcc-arm-embedded
[7] St-link Utility - https://github.com/texane/stlink
[8] CMSISv4 DSP Library - https://github.com/ARM-software/CMSIS
[9] CMSISv5 DSP Library - https://github.com/ARM-software/CMSIS_5
[10] CMSIS Makefile Sources (Albeit, modified) - http://kernelhacks.blogspot.com/2013/01/cmsis-dsp-software-library.html

Public References

[11] STM Nucleo - http://www.st.com/en/evaluation-tools/nucleo-f401re.html
[12] STM Nucleo Pinout - https://developer.mbed.org/platforms/ST-Nucleo-F401RE/
[13] STM32F401 - http://www.st.com/en/microcontrollers/stm32f401re.html
[14] MCP4921 DAC - http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf
[15]HAL Driver Documentation - http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf
[16] CMSIS Documentation - https://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php
[17] TSV522AIST Op-Amp - http://www.digikey.com/product-detail/en/stmicroelectronics/TSV522AIST/497-13195-1-ND/3522204

[18] AD627ARZ Instrumentation Amplifier - http://www.digikey.com/product-detail/en/analog-devices-inc/AD627ARZ/AD627ARZ-ND/819850
[19] MIC23250 Switcing Regulator - http://www.digikey.com/product-detail/en/microchip-technology/MIC23250-AAYMT-TR/576-3315-1-ND/1974133
[20] Reducing STM32F4 Power Consumption - http://www.st.com/content/ccc/resource/technical/document/application_note/13/0a/06/b9/1e/2f/4d/9d/DM00096220.pdf/files/DM00096220.pdf/jcr:content/translations/en.DM00096220.pdf

Appendix A: Bill Of Materials

Table 3: Bill of Materials

Qty.	Manufacturer	Manufacturer Part Number	Description	Unit Price (USD)	Extended Price
2	Murata	MA40S4S	TRANS 40KHZ ULTRASONIC	6.8	13.6
1	Texas Instruments	MSP-EXP430FR4133	LAUNCH PAD KIT MSP430FR4X MCU	14.53	14.53
1	TE Connectivity Measurement Specialties	1005940-1	SENSOR MINISENSE 100 VERTICAL	5.78	5.78
4	TE Connectivity Passive Product	RGP0207CHK1G0	RES 1.00G OHM 1/4W 10% AXIAL	0.87	3.48
2	CUI Inc.	CPE-122	BUZZER PIEZO 10V 24MM RADIAL	2.39	4.78
2	Mallory Sonalert Products Inc.	PT-2040PQ	BUZZER PIEZO 5V 22MM RADIAL	1.7	3.4
2	Texas Instruments	SN74LV1T34DCKR	IC BUFFER GATE SGL CMOS SC70-5	0.45	0.9
1			Breadboard	5	5
1	Microchip Technologies	MIC23250	Switching Regulator	1.62	1.62
1	Analog Devices	AD627ARZ	Instrumentation Amplifier	6.22	6.22
1	ST Microelectronics	TSV522AIST	Low Power Op Amp	1.02	1.02
2	TDK	B82432T1102K	Inductor	1.02	2.04
			Various Resistors	0	0
			Various Capacitors	0	0
1	OSH Park		Printed Circuit Board	10.52	10.52
				Subtotal	72.89
				Sales Tax	6.38
				Shipping	7.56
				Total	86.83

Appendix B: STM32F401 Nucleo Pinout

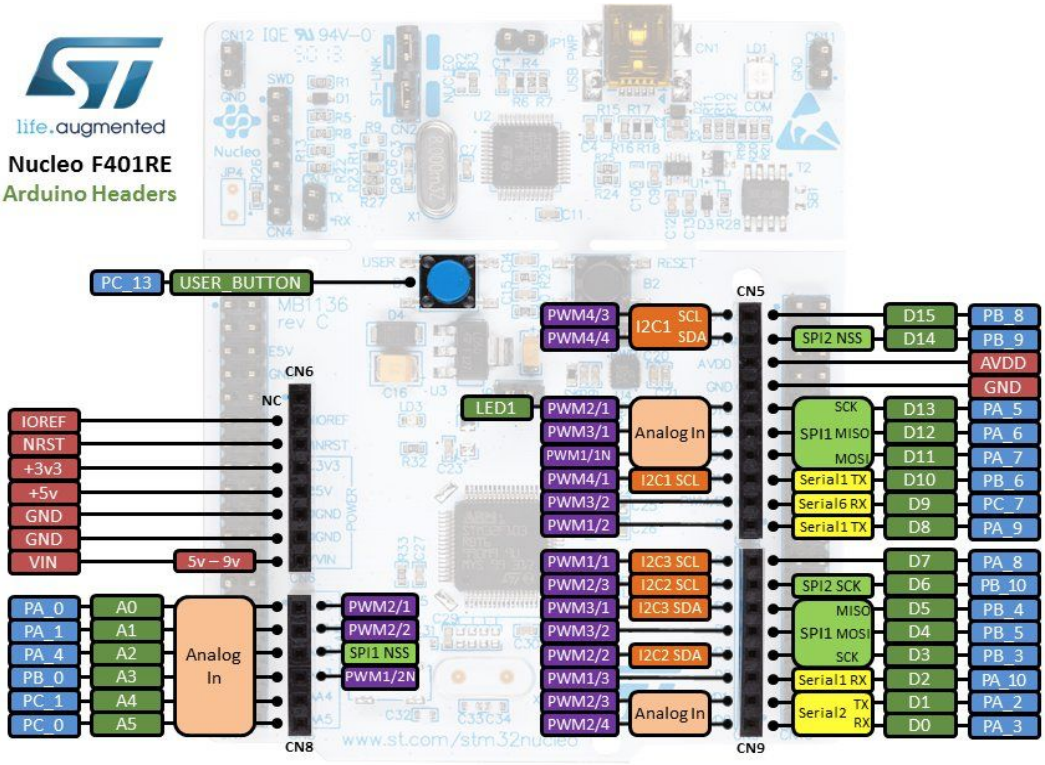

Figure 12: STM32F401 Nucleo Pinout

Table 4: Currently Used Pins

External	Internal	Function	Notes
A0	PA0	Wake Pin	
D1	PA2	UART TX	
D0	PA3	UART RX	
A2	PA4	SPI1 NSS	
D13	PA5	LED (LD5) Output	
D12	PA6	GPIO (CS)	This should be removed when NSS is used
A3	PB0	ADC Input	
D3	PB3	SPI1 SCK	
D5	PB4	SPI1 MISO	This isn't really being used. Feel free to remove it
D4	PB5	SPI1 MOSI	

Appendix C: Project Repository

This Git repository contains all of the code for the project and all of the schematic files
https://gitlab.com/davidzhuo/deer-deterrent