Giovanni Wong
A99419153
Daniel Knapp
A09365933

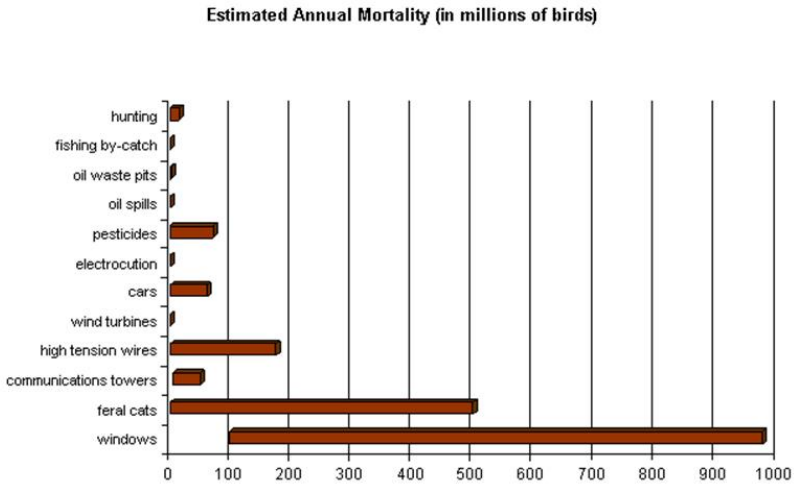
Angry Birds Final Report

Abstract

During migratory season, masses of birds fly long distances increasing the number of windows each bird will encounter. Birds can confuse the window's reflection of nature as the real thing causing many birds to die each year from collisions with windows. Birds play an important part of our ecosystem ranging from pollination to eating insects that are harmful to humans to helping in reforestation. There is a part of the UV spectrum that birds can see, but humans cannot. By putting UV films on windows we hope to significantly reduce the likelihood that birds will mistake window reflections as a path to more nature. In order to test the effectiveness of the films, an embedded system will be deployed to detect window vibration and verify that a bird was the cause. We suspect the tests to show a 50% decrease in the number of birds that collide with the windows that have the film versus the windows that do not.

Introduction

It is estimated that between 100 million and 1 billion birds die each year due to collisions with windows. That is significantly higher than than the deaths caused by their predators, cats, which cause up to 500 million a year or high tension wire collisions (aerial wires) with up to 174 million a year. These numbers are staggering as it means that windows are more dangerous to the lives of birds than even their predators. Steps can and should be taken to help reduce this number as the lives of birds play a non-trivial role in the lives of humans as well.



Some birds, like hummingbirds, help in pollinating plants. Pollination facilitates the reproduction of flowering plants and without animals like birds and bees helping along in this process, these plants (which includes crops) could see a huge decline. Many birds thrive off of eating insects and larvae. These insects consumed are often considered pests and are potentially hazardous to our way of life. An example of such an insect would be mosquitoes that suck our blood and can also pass along bloodborne pathogens like malaria. Eating these insects can also aid in reducing damage done to agriculture. Other birds consume seeds and help in dispersing plants by releasing the seeds in their droppings in various locations. Even birds that wade

in water can spread different types of fish by the fish eggs getting stuck to their legs. It is believed birds are useful as bio-indicators to detect harmful substances in the environment. An insecticide was banned in the US in 1972 called DDT which at the time was known kill large numbers of American Robins due to the poisoned insects that ate the plants covered in DDT. The danger of this drug would likely not have been discovered as quickly had birds not been there to take the fall.The US government has even taken steps to preserve the well-beings of birds dating back to the early 20th century that put an emphasis on protecting migratory birds. It is clear that birds have a high value to all of our lives.

The cause of bird deaths are due to the nature of glass. Glass can appear differently depending on a range of factors, including how it's fabricated, the viewing angle and the difference between interior and exterior light levels. Combining all these factors can change one's perception of what the glass looks like, such as a dark passageway, a mirror or even invisible. Because of these perceptions, birds usually collide into windows, mistaking them as potentially open paths, or a way to some vegetation they see. Because of this, tall buildings that happen to be in path of migration are a big problem.

One of the proposed solutions in reducing these deaths is to change the design the way glass is designed and laid out. Some buildings have external grids/lines that cover the glass to deter birds from flying into them. A study conducted on this design showed that covering as little as 5% of the glass can deter 90% of collisions under experimental conditions. This rule, called the 2x4 rule, has proven to be effective. However, in terms of economics, this is not feasible as this would be very hard to scale and reproduce.

Birds however can see into the ultraviolet(UV), a range considered unseeable by humans. Because of this, it has been suggested to incorporate UV reflective/absorbing patterns/films onto windows as a solution. However, the progress to incorporate this has been slow due to technical complexities, and a strong contrast of UV patterns is needed to be considered effective.

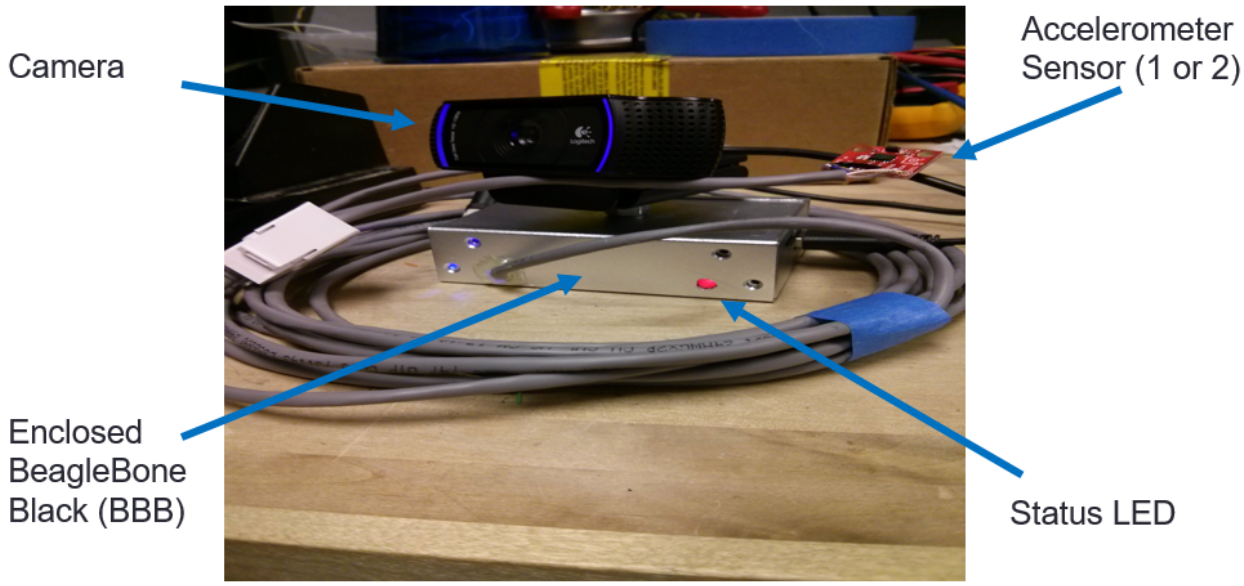
The researchers we have teamed up with from Minnesota and Pennsylvania have developed a UV film that may be the solution to this problem. However, in order to test this, they require an autonomous system that would monitor the effectiveness of the film. Since no human can constantly monitor a window 24/7 accurately, the engineers for exploration have been delegated to build the system that will collect data and reliably detect when a bird has come in contact with a window.

Technical Material

Overview of System

The embedded system in the project consists of a Beaglebone Black, which runs on the Linux Ubuntu, with Kernel version 3.14. Attached to the Beaglebone Black externally are the Logitech C920 HD pro logitech camera, 2 SparkFun Triple axis accelerometer breakout board(ADXL345), DS1307 Real Time Clock breakout board, and a basic Sparkfun 5mm LED. To protect the board and clock from any potential environmental damage, the board is protected within a metal casing. The shape and design of the casing

does not prohibit the accessibility of the external ports from the board, ensuring that the board is fully usable. The casing also has 2 holes, one for the LED and one for the ADXL connecting to the ports of the board. The web camera is attached to the casing by a screw and held together with a bolt. The board is then powered by an external power source, which then will actively start monitoring the windows by taking 20 pictures per second, which is then stored into a queue like data structure. Once the queue gets full, it will actively pop off the oldest pictures off the queue to make more space to put new pictures. The accelerometer is the acting sensor that sends data to the board, which is usually placed on the window held by tape. If a collision were to be detected, the data collected from the vibrations of the window will then trigger the system, and the pictures taken from the camera will be converted into a video format and stored into a SD card, with frames prior to the collision up to after. The method that the system detects a collision is through a autoconfigured threshold value, in which the previous retrieved data values are compared against the current values, and if the difference is greater than the threshold, data is stored onto the SD card.



Beaglebone Black

The BeagleBone Black is a low-cost, community-supported development platform for embedded systems. Similar to the Arduino Uno, the BeagleBone Black is an more expensive alternative solution that provides much better system specs in comparison to the Arduino. Equipped with the AM335x 1GHz ARM Cortex-A8, 512 MB DDR3 RAM, 4GB on-board flash, 3D graphics accelerator, and 2x PRU 32bit microcontrollers, the BeagleBone black is capable of being a portable computer, similar to the RaspBerry pi. The BeagleBone is capable of running a variety of operating systems, such as Ubuntu, Angstrom and Android, just to name a few. For these reasons, the Beaglebone Black was chosen as the main hardware board for this project.

For this specific project, there were 2 types of BeagleBone Black boards: type B and C. Type B came preinstalled with the Ångström distribution while type C with Debian. Because the Angry Birds project was an ongoing legacy project, the Ubuntu distribution was flashed on all existing boards. For the sake of keeping the project in line with our schedule, the boards were not flashed back to Debian, as Ubuntu proved to be a difficulty to enable and disable certain hardware features, which will be outlined in the challenges faced and future work section.

The communication interface used between the ADXL and BeagleBone Black was I2C. The reason we chose I2C over SPI was mainly because of wiring and complications of the SPI protocol. The SPI protocol requires 5 wires, 2 being ground and VCC while I2C only requires 4 with the ground and VCC.In addition, if more devices were to share the SPI protocol, a complicated procedure of wiring and setup would be required in order for both devices to work, which made I2C more appealing. I2C and SPI are only meant for short distances, as specifications of the project required us to extend the range of the protocol, prompting us to make certain hardware configurations in order to make the system work. In addition, the clock also uses I2C.

We used an open source library for controlling and enabling certain features of the Beaglebone Black. The library enables the BeagleBone Black to read analog input, using gpio pins and communication with other devices over SPI, I2C and uart. The library is written in C++ and was already integrated into the project from day one of when the class started.

We also implemented a feature where the board can either be in debugging mode and either use 1 or 2 sensors. This allows the researchers more freedom for their setup depending on the situation, and allows us to find out any potential problems within the system.

ADXL 345

The ADXL345 is a ultralow power 3-axis accelerometer with high resolution measurement up to +- 16g. The high resolution enables the accelerometer to measure changes of less than 1.0 degree when tilted.The accelerometer currently operates at 25 kHz, which gathers about 200 data sets every second.Taking advantage of this feature, the accelerometers will be used to detect the vibrations of the window in the event of a collision.

In order to filter out potential noise(non collisions detected as collisions), we developed a dynamic threshold algorithm that will collect data from the accelerometer for 30 seconds. During these 30 seconds, the previous value is subtracted by the current value, absolute value from the calculation to go to the index of the array. We then increment the value at the index and repeat this process. After the collection is done, we walk through the array and obtain the highest value threshold in which there is no incremented values in the next 2 indexes after the current index. For example: If current index is 10, which has a value of 3, and index 11 and 12 are both 0, then index 11 is set as the threshold. The reason behind this is because we account for randomness/error while the accelerometer is collecting data. Since the accelerometer is very cheap, the quality of accuracy may not be as good as a higher grade accelerometer. Thus, using this method we hope that the randomness of the collected data will fall within a certain range, which statistically is very high for a device this cheap. In addition, the algorithm allows non human interaction to set a threshold, which benefits the researchers since they may not have the technical capability to set the threshold themselves, which requires modifying the main program file itself.

We also utilized a open source library that enables us to specifically modify hardware configurations on the ADXL itself, and also setting up the appropriate settings for I2C protocol. This library was developed specifically for the BeagleBone black, which in turned worked out very well.

DS1307 Real Time Clock

The DS1307 is connected to the BeagleBone black through I2C. Its purpose is it keep track of time in the event that a board is reprogrammed or even powered off. The DS1307 helps by timestamping the pictures

when the camera is continuously loading into the queue, and also timestamping the event of the collision. This is crucial for our project since it is mostly data logging, which is stored onto the SD card.

Logitech C920 HD

The web camera, which is a logitech C920 HD, is capable of taking 15 megapixel pictures at 1920 x 1080 resolution. The camera connects to the system via USB,and the system will only work when the camera is connected. While the system is running, the camera continuously takes 20 pictures per second, and stores the pictures into a queue. When queue capacity is reached, it takes the oldest pictures and removes it from the queue to free more space. The reason for this setup is save memory space, 4 GB of onboard storage is very small in the grand scheme of things, considering that if we saved every picture, it would use all the memory very quickly. By using this setup, we save space and are more efficient of what gets saved and what gets deleted.When a collision is detected, the system will take pictures from the queue from prior to the collision up to a few seconds after. The system then turns these pictures and converts them into a small video for ease of viewing. The movie is then timestamped and saved into a folder along with the data collected from the detected collision into the SD card.

LED

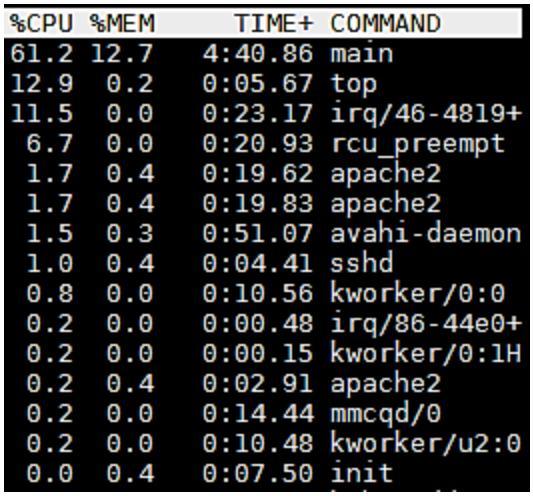
The LED is a standard regular 5mm LED that is connected to the system. The LED is connected to the GPIO pin of the board, which control the rate at which the LED blinks. Under normal conditions, the LED will blink at a rate of 1 blink per second, while under collision detection roughly 5 blinks per second. After the data is stored in the SD card after detecting a collision, the LED goes back to the normal blinking rate.This LED also serves as system indicator, notifying the user the status of the system. If the LED is not blinking at the proper rate, it could potentially mean that the system is not working as intended, and something is not functioning properly. Also, the LED not being on can serve as an indicator that the system is not on.

Milestones

Our group had 2 major milestones for the quarter that we wanted to accomplish. From those major milestones, we broke them down into weekly milestones that should be accomplished by the end of the week. The 2 major milestones were to first have more than 1 sensor working with a single BeagleBone Black (BBB). This deadline was to be on May 8th. The second major milestone was to be able to wirelessly communicate between a single BBB and a single sensor. This deadline was to be on June 5th. The following paragraphs will be our major milestones broken down into weekly granularity.

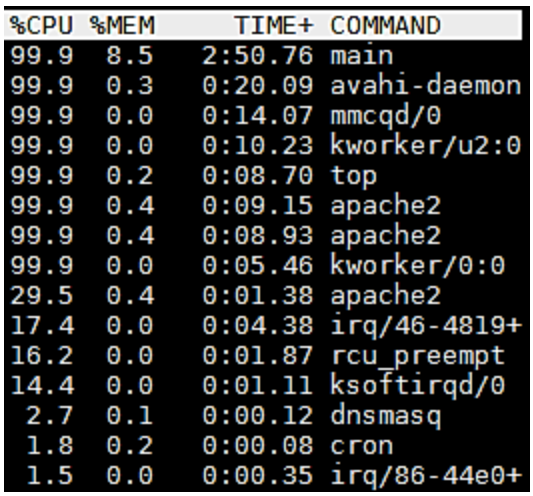
Under the first major milestone, we wanted to first do research on what parts we needed to accomplish the first major milestone. This was to be done by the end of week 3 (4/17). Some of the parts we already had, so we also just took some time to familiarize ourselves with our sensor, the ADXL345. Since this is an ongoing project, we also started looking into an issue we were having with a lagging operating system. When running our main program, that does the detection for when a bird collides with a window, after a few minutes the whole operating system on the BBB would completely bog down to the point of being completely unresponsive. We were using a linux program, “top”, that shows the running processes and their cpu usage among other things to help and pinpoint what could be causing the issue. Once our detection program was running for a few minutes the system would bog down and “top” would report that multiple processes were

simultaneously using 99% of the cpu. Clearly this program was not going to be useful in pinpointing what the problem was.



%CPU	%MEM	TIME+	COMMAND
61.2	12.7	4:40.86	main
12.9	0.2	0:05.67	top
11.5	0.0	0:23.17	irq/46-4819+
6.7	0.0	0:20.93	rcu_preempt
1.7	0.4	0:19.62	apache2
1.7	0.4	0:19.83	apache2
1.5	0.3	0:51.07	avahi-daemon
1.0	0.4	0:04.41	sshd
0.8	0.0	0:10.56	kworker/0:0
0.2	0.0	0:00.48	irq/86-44e0+
0.2	0.0	0:00.15	kworker/0:1H
0.2	0.4	0:02.91	apache2
0.2	0.0	0:14.44	mmcqd/0
0.2	0.0	0:10.48	kworker/u2:0
0.0	0.4	0:07.50	init

Normal Program Operation

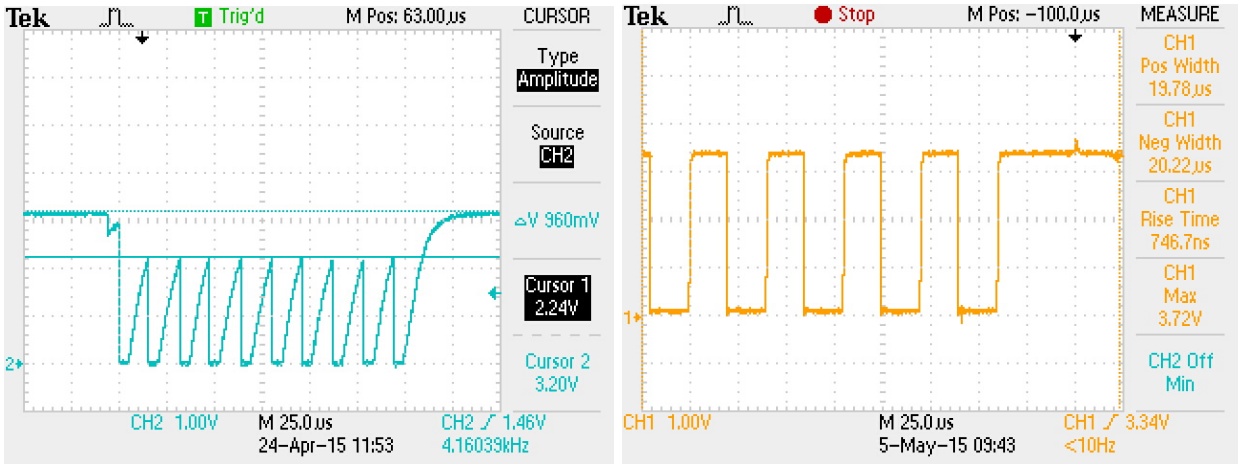


%CPU	%MEM	TIME+	COMMAND
99.9	8.5	2:50.76	main
99.9	0.3	0:20.09	avahi-daemon
99.9	0.0	0:14.07	mmcqd/0
99.9	0.0	0:10.23	kworker/u2:0
99.9	0.2	0:08.70	top
99.9	0.4	0:09.15	apache2
99.9	0.4	0:08.93	apache2
99.9	0.0	0:05.46	kworker/0:0
29.5	0.4	0:01.38	apache2
17.4	0.0	0:04.38	irq/46-4819+
16.2	0.0	0:01.87	rcu_preempt
14.4	0.0	0:01.11	ksoftirqd/0
2.7	0.1	0:00.12	dnsmasq
1.8	0.2	0:00.08	cron
1.5	0.0	0:00.35	irq/86-44e0+

Excessively Bogged Down

For week 4 (due 4/24) Daniel and Gio wanted to sync up in terms of what hardware and software is being used in the system to get a good understanding of how it all ties together. This was important because Daniel had already been working some on the project for half of a quarter and could disseminate some useful information. In terms of what was actually accomplished, Daniel and Gio spent some time looking through our code to try and pick out what the problem could be. Gio went to get additional help by talking to his operating systems professor in the hopes of getting advice about how we can determine what the problem is with the lagging and fix it. Fortunately, a graduate student who had previously been actively leading the Angry Birds project had a go at freshly installing an operating system on a BBB. His testing showed that the system was able to run our detection program for 20 minutes with no problems. Daniel followed up by creating a new bootable micro-SD card to flash a system with a fresh operating system. After flashing the system and installing the necessary third-party libraries as well as the Angry Birds code, he was able to confirm that it would run at least 15 minutes with no problems. This was proof enough that using a fresh version of the operating system would work since on any system that was cloned, it would bog down no later than running the detection program for 5 minutes. About 8 more systems were tested successfully in this way.

For week 5 (due 5/1) we wanted to start assembling a system with 2 sensors with minimal cord length between the BBB and our sensor having now received any necessary parts. However, our focus this week ended up coming down to fixing some faulty communication between the BBB and the sensor in the current setup with only 1 sensor. We prioritized this issue over our goal because without being able to successfully communicate with 1 sensor, we will have no way of knowing that the setup we are planning to build is working. After talking to a few colleagues about how to solve the issue, we chose to look at the communication lines on an oscilloscope to try and see how well they follow the I2C protocol that is being used. Doing this showed a clear problem. The SCL line of the I2C bus that is supposed to act as the clock for synchronizing who talks when was completely jagged when this line is supposed to house digital signals. From this information, we were able to deduce that we needed to reduce the effective resistance of the pull-up resistors on the I2C line since we saw on the oscilloscope that the rise-time of the signal was too large.



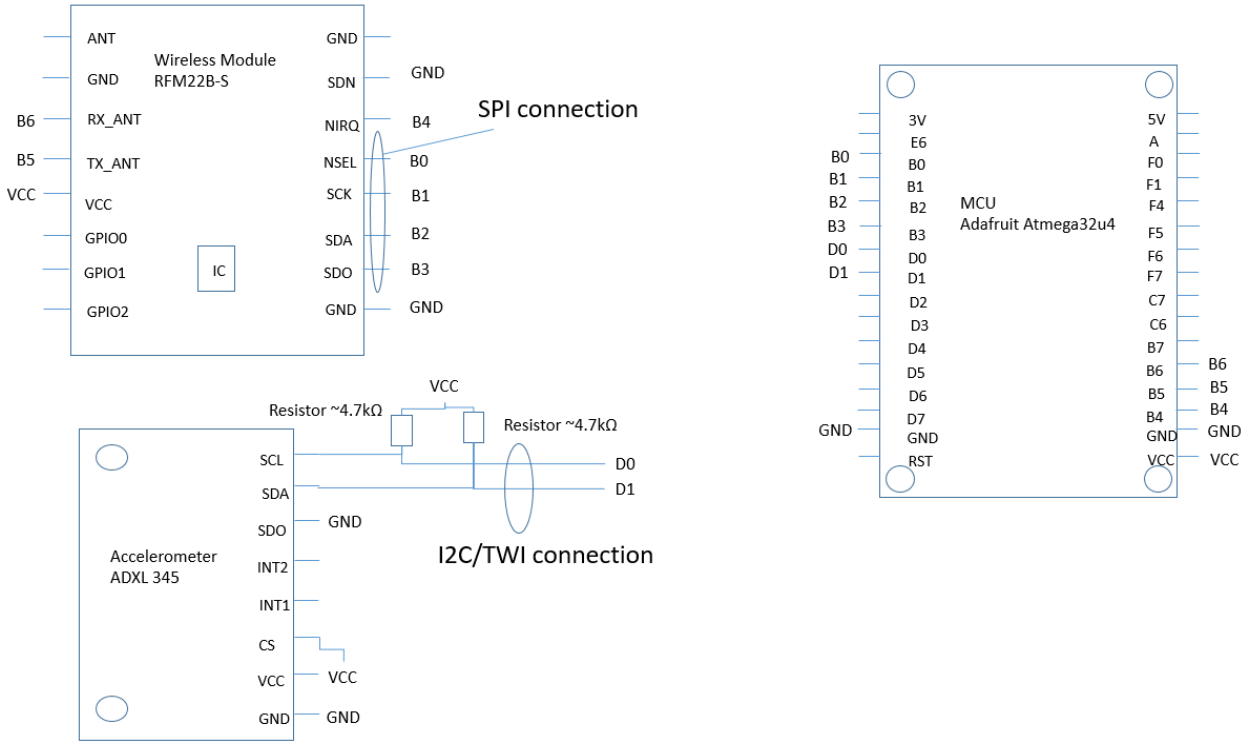
* The figures above show before and after screenshots looking at the SCL (clock) line of the I2C bus. The important things to notice are the shape of the signals and their peak-to-peak values. Going up or down 1 grid line vertically is 1V where the grid lines are the dotted gray lines. On the left we see the signal before adding the external pull-up resistors. When attempting send bits, we see the signal only reaches 2.24V and is very jagged for communication that is supposed to be digital. On the right hand side we see the SCL line after the pull-up resistors are added. It's clear to see that the signal rises to the peak ~3.3V quickly unlike before and thus the wave looks much more like a square wave which is desired.

For week 6 (due 5/8) we wanted to have our first major milestone completed by having 2 sensors with 1 BBB at the normal 16 feet of cord length. Since we didn't really get to achieve our goal for the previous week, we had to somewhat merge it with this week's goal. Instead of testing the system at a short distance first though, we just went ahead with the full 16 feet of cord when making the 2 sensor version of the system. First we used the newly acquired 1k ohm resistors as pull-up resistors for the SCL and SDA lines of the I2C bus. Since there are internal pull-up resistors of 1.5k ohms for each of these lines already in the BBB, we calculated the effective resistance of the pull-up resistors to be 0.6k ohms. This would also increase the power consumed when communicating over I2C by approximately 2.5 times, but since the power consumption is so small (5mA with the external pull-ups) it doesn't appear to be a problem. Additionally, we were able to meet our major milestone this week. The 2nd sensor that we are using is connected to the same pair of I2C lines as the 1st sensor with the difference being we have the address pin on the sensor tied to Vcc rather than GND like we have on the 1st sensor. Since the sensors have different addresses they are able to be connected to the same bus with the BBB deciding which sensor it wants to talk to. Naturally we had to add some code for this new sensor that followed suit with how we talked to the 1st sensor. By the end of this week, we were essentially back on track for our milestones.

For week 7 (due 5/15) we planned to begin work on our 2nd major milestone starting by doing research on the components we would need to achieve wireless communication between the BBB and a sensor. This week accomplished our goal (with the help of the grad student that was previously leading the project). We decided on the ATmega32u4 breakout board for the microcontroller that would be wired to the sensor and an RF board to tell the BBB when a vibration is detected. For the RF board, we go the RFM22B transceiver module. For a power source we wanted to think about what would be easiest for a researcher to recharge or change out. To keep things simple, we decided to go with a waterproof battery pack with space for 3 AA batteries and to use a voltage regulator to bring the 4.5V down to the 3.3V we need to power the sensor and RF board. We would hook everything up on potentially a prototype board for this class to at least get a good proof of concept. If we had time we would maybe come up with plans for a custom PCB as well as a good

enclosure to keep it all safe from the rain as the components on the sensor side will be on the outside of a window.

For week 8 (due 5/22) the goal was to try and find or write our own code to put on the microcontroller that will be able to talk to the RF board and to solder everything together that will go with the sensor. We were able to get everything soldered together according to our schematic that we had made as we wanted to. We were also able to find some libraries we thought we could use to have the microcontroller send and receive through the RF board. We didn't get to actually programming the microcontroller, but we thought that we would still be on track for our milestone next week.



* Schematic for what will be hooked up to the sensor for wireless communication

Week 9 and 10 will be lumped together since we were essentially doing the same thing for both weeks without a ton of success. The end of week 9 (due 5/29) we wanted to test close range wireless communication between the BBB and sensor and test that the way we are supplying power to the sensor side works. Week 10 (due 6/5) we wanted to have the wireless communication working with the sensor components on the outside of a window and the BBB some distance away. During these weeks we spent basically all of our time doing research into why the microcontrollers I have won't program and how SPI can be enabled on the BBB so that it can talk to the RF board. It turns out that there is probably some weird timing issue with programming the microcontroller. People on some forums state that the microcontroller works best when not on USB 3.0 ports and is connected through a USB hub. What finally seemed to work consistently that we found out on week 10 was if you hit the reset button on the microcontroller board (which is supposed to run the bootloader and get it ready for programming) and wait about 14 seconds until just before the microcontroller exits the bootloader, we can program it successfully. On the other end with the BBB, we still haven't quite figured out how to enable SPI. There is a significant amount of information about how to do it online, but most of them are relevant for different of previous versions of operating systems. As such, many of the instructions given on how to do it are outdated and no longer applicable. All of the

instructions deal with changing these “device-tree” files that the operating system loads at boot time. That means messing something up in one of these files could potentially brick the system which is a bit disconcerting. Ultimately we’ve tried some of the solutions, but have not found one that we have been able to get to work yet leaving the wireless version of our system unfinished.

Conclusion

The Angry Birds project shows promise in its goal to be a reliable way to test the effectiveness of the UV films being developed to reduce bird deaths due to window collisions. It is hard to say how effective the films will be before doing real testing with them. However, if we find that the films are 50% effective, and they are applied to only 10% of buildings, we could still be saving between 5-50 million birds annually. If the films are as unobtrusive as we expect and are utilized in areas where window collisions are more likely to occur like in well-known bird migratory paths, this estimate could increase significantly.

As for some hardware improvements, the idea of having multiple versions of the Angry Birds system seems to make the most sense since every building is different. Completion of the wireless version of the system could be very useful for buildings that have double pane windows. It could even be quite effective in buildings have many small windows on a given wall that can all be viewed from one camera location. In that case there could potentially only be 1 BBB to upwards of 12 wireless sensors depending on the layout of the windows. In other locations, it may be best to only have 1 sensor connected to a BBB because there are no windows nearby. An improvement can also be made to theoretically have 4 sensors physically connected to 1 BBB which would require an intelligent way of enabling only 2 sensors at a time and switching which ones are enabled very quickly to effectively be able to monitor all 4 simultaneously. The benefit this has over the a wireless connection is mainly that there is no need for someone to change batteries and the sensors do not have to be waterproof.

Improvements can also be made more on the software side. Having a more intelligent and automated way of deciphering a bird collision from other things vibrating the window would prove useful in reducing the number of false-positives. One way to achieve that would be to get vibration data from a real collision of a bird with a window and see if there is a particular signature that separates a human’s tap on a window from the bird’s collision. This would require moving away from using an accelerometer since it would be extremely unlikely to give enough data points to do this kind of analysis. Having the right quality and electrical setup with a piezoelectric would likely be the way to go. Another way to reduce false-positives could be to do an analysis of how recently a collision was detected. This could help in the case where hail or potentially a heavy rain constantly hitting a window could trigger continuous detected collisions for long periods of time. It should be reasonable to assume that many birds will not sequentially fly into the same window over any length of time. The last idea currently considered would be to use a computer vision algorithm to look through the captured images and decide if a bird collision could have triggered the detection. This could be extremely useful in cases where windows are close enough to doors that a slamming of the door could cause the window to vibrate. An algorithm that just checks for significant movement in the pictures may be enough to weed out the vast majority of the false-positives that are not due to physical contact with the window.

References

http://www.sibleyguides.com/conservation/causes-of-bird-mortality/
http://www.abcbirds.org/abcprograms/policy/collisions/pdf/Bird-friendly_Building_Guide_WEB.pdf
http://www.flap.org/pdfs/birds-uv-light.pdf
http://www.fws.gov/migratorybirds/regulationspolicies/treatlaw.html
http://www.extension.iastate.edu/naturemapping/monitoring/importance_birds.htm