Linear Localization of Microvibration Source with Piezoelectric Elements

Joshua Send
Torrey Pines High School

Introduction

This project was born out of the idea to scale down earthquake detecting systems into something smaller and more applicable to everyday life. Seismographs, earthquake detecting equipment, find the center of an earthquake by using time delays between P (compressional; fastest) and S (shear; second fastest) waves to calculate distance, and then use three such measurements to triangulate the position of the epicenter. In the beginning, simply knocking on wood and feeling the vibrations arrive at the other end provided enough motivation to start the project. I eventually got to work and obtained some vibration sensors.

This project would have been nearly impossible without the assistance of an old oscilloscope I chanced upon. With a little help, I soon had most of the controls figured out and was working with it most of the time thereafter. The oscilloscope was a great help in figuring out what time and voltage scale I needed to work with, which later also translated into the analog-digital converter options and coding optimizations. I soon realized that the initial approach was too complex, and that I needed to simplify the design. It would be too difficult to implement the full scheme – to locate the source of a micro-vibration in a plane using three sensors– with the limited time available. This led to the elimination of the third sensor, and the assumption that all future vibrations would be located on the line between the two remaining sensors in a long and thin material, greatly reducing complexity.

Purpose

The purpose of the project is to show it is possible to use time delays of small scale vibrations in solid materials to localize their source by building a working system to accomplish this task.

Materials

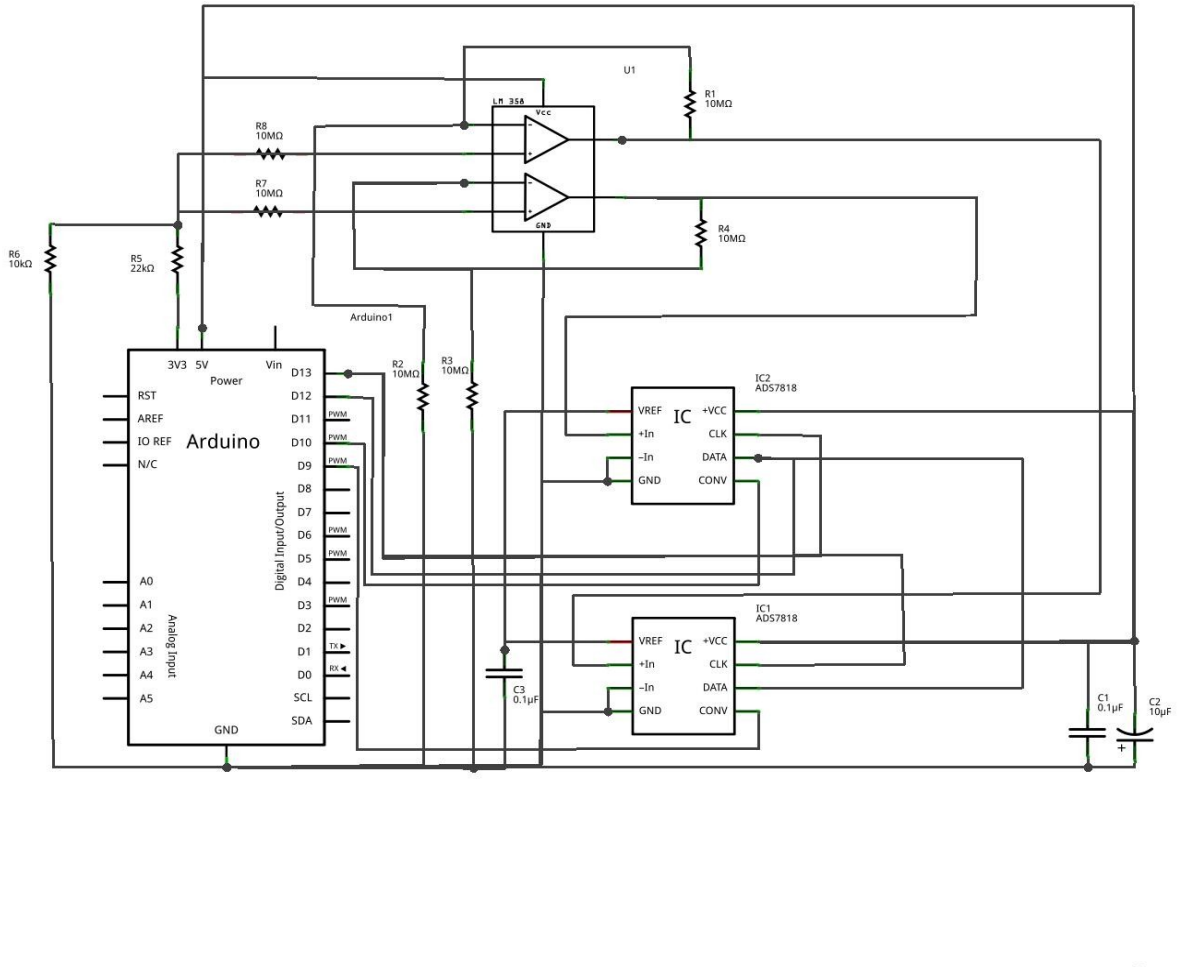
- 1 x Arduino Uno r3
- 2 x Burr-Brown ADS7818 Analog Digital Converter
- 1 x TI LM158n Low Power Dual Operational Amplifier
- 2 x Measurement Specialties, Inc. Piezo Film Sensor
- 2 x 0.1 μF ceramic capacitor
- 1 x 10 μF electrolytic capacitor
- 8 x 10 MΩ Resistors
- 1 x Protoboard
- 1 x 1" diameter PVC pipe
- 2 x Coaxial cable
- Oscilloscope and male BNC plugs with Coax connectors
- Arduino IDE
- Python 2.7
- Male to Female USB cable
- Roll of solid copper Hookup Wire
- Velleman VTSS5 soldering iron
- RadioShack 60/40 Light-Duty Rosin-Core Solder
- Adhesive Tape

- 2 x 2"x4"x7.25" Wood Beam
- 6 x 2" Wood Screws
- 2"x12"x32" Wood Slab
- Drill
- Permanent Marker

Procedure

All procedures performed by student

Overall schematic for electronics:



Drawing 1: Final Schematic

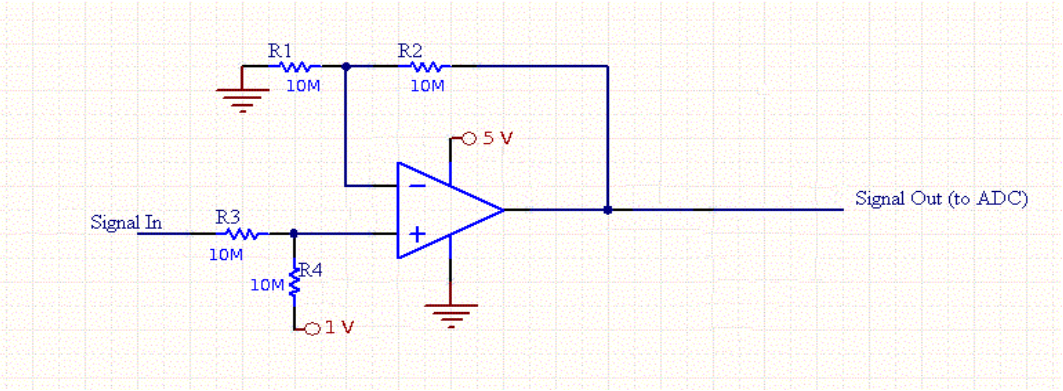
1. Constructed Pipe Holder
   - Drilled 1.5" hole in center of the two 2"x4"x7.25" pieces of wood using drill
   - Screwed above pieces to 2"x12"x32" slab using 2 wood screws for each wood piece so that holes face each other. One wood piece was placed on each end 12” ends of the slab
   - Put PVC pipe through holes
   - Fasten pipe in the holes with 2" wood screws

Image 1: Pipe holder – Photo by: Joshua Send

2. Tested for basic Information
   - Stripped Coaxial cables, soldered Piezoelectric Elements to end of each cable
   - Connected male BNC plugs to coax cables using connector
   - Set up oscilloscope
   - Connected coax cables to oscilloscope to test the Piezoelectric elements and determined time scale, voltage amplitude, etc. after taping piezoelectric sensors to pipe
3. Assembled Electronics
   - Connected corresponding pins on Burr-Brown Analog-Digital (AD) converters to Arduino; used protoboard and solid copper wire for connections

Arduino pin #	12 bit Burr-Brown AD pin #
+5 V	8 $+V_{cc}$
13	7 CLK
12	6 DATA
10/9	5 CONV
Analog GND	4 GND

   - Decoupled AD pin 1 (V_{REF}) with 0.1 µF ceramic capacitor to ground (as described in AD datasheet)
   - Connected AD pin 3 (-IN) to ground (as described in AD datasheet)
   - Decoupled $+V_{cc}$ to ground with 0.1 µF ceramic capacitor (as described in AD datasheet)
   - Connected resistors and voltage sources according to this schematic:



Drawing 2: Schematic for Op-amp setup

- Removed BNC connector, soldered wires onto coax cables, connected Coax shielding wires to ground, and center wire of each Coax to the 'Signal In' in the schematic above
- Connected op-amp output to analog input of ADC's

4. Completed the Hardware set-up
   - Connected Arduino to computer over Serial cable (male to female USB)
   - Taped piezoelectric sensor to pipe, ~54 cm apart
     - Taped sensor so that longer side is parallel to the length of the pipe (leads facing away from center)
   - Placed whole setup away from appliances creating electrical interference
5. Software – Arduino variant of C++ (compiled with AVR C++ compiler)
   - SPI library parameters
     - SPI.setBitOrder(MSBFIRST);
     - SPI.setClockDivider(SPI_CLOCK_DIV2);
     - SPI.setDataMode(SPI_MODE0);
   - Collected data from AD 1, then AD 2
     - For speed optimization, program controls registers directly, e.g. [code snippet]
       - PORTB &= B11111011; // pull pin 10 of arduino LOW (ADC sample)
       - PORTB |= B00000100; // pull pin 10 of arduino HIGH (ADC off)
       - Replicated above for pin 9 of Arduino for second AD
     - Program asks for data with SPI library and collect
   - Calculated total for set of specified length
     - For speed optimization, subtracted last and add newest values (rather than add up whole array each time)
   - Checked each new measurement for signal larger than threshold above average
     - For speed optimization, rather than divide by the length of the set (for average), multiply new value by length of the set and subtract from total.
     - Program takes above number and compares it to the threshold multiplied by the length of the set

Findings

This project collected an abundance of data, and it is easy to collect more. Appendix B has all the data used in this section, and will be referred to by parts for each graph.

An image is provided to help with the interpretation of the data. The black box is the PVC pipe tested on.

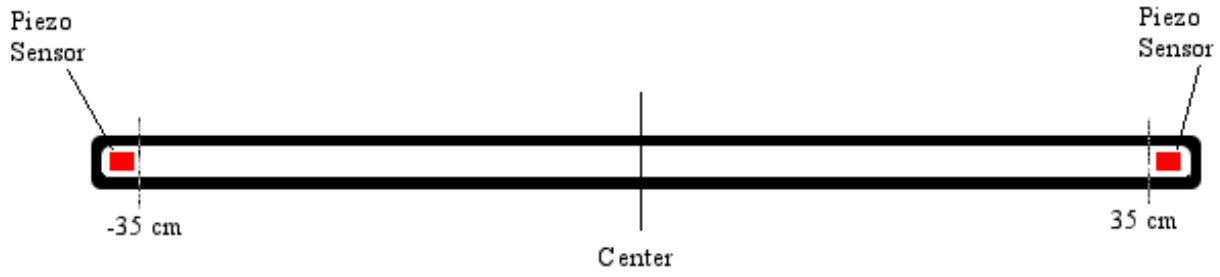


Figure 1: Reference Image

Accuracy

A summarized version of data collected from hitting the center ("Center"), at about -35 cm ("Left"), at about 35 cm ("Right"), about -12 cm ("Half Left"), and about 10 cm ("Half Right") measured from the center of the pipe.

Location	# Trials	Mean in cm	Standard Deviation in cm	Expected Value (actual) in cm
Center	15	-1.96	0.92	~ 0
Left	15	-26.89	0.93	~ -29
Right	15	29.73	0.63	~ 29
Half Left	15	-18.69	3.46	~ -16
Half Right	15	16.49	0.65	~ 16

Table 1: Statistical Information at Select Locations

The majority of the locations work well. The point where the data was the least accurate was the location 'Half Left'. Not only is the mean more than 2.5 cm off, the standard deviation is far larger than any of the other measured locations.

One other thing to note is that the leftmost trials and the rightmost trials were done at about 29 cm from the center (rather than the full available 35 cm). While testing the system, it was found that the system rarely even registered impacts further away than 29 cm. One suspicion is that the screw used to hold the pipe in place is absorbing or diverting the vibrations from their intended courses.

Note: for the data used to construct this table, see: Appendix B; sections 1-5.

Visual Representations of Single-point Testing

Visualizations of the a trial happening, in the order of Left, Half Left, Center, Half Right, Right.

Note: All the data used in this section can be found in: Appendix B; section 7.

The Blue line is the left Piezoelectric sensor output, while the Red line is the right Piezoelectric sensor output.

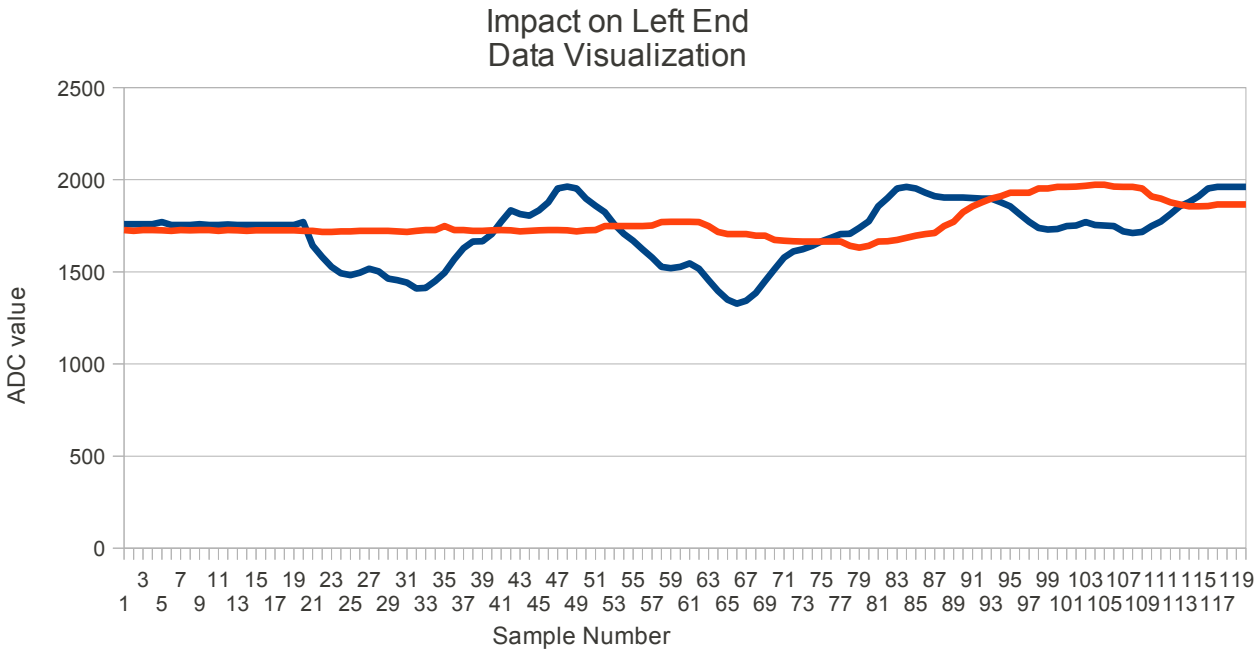


Figure 2: Left Data Graph

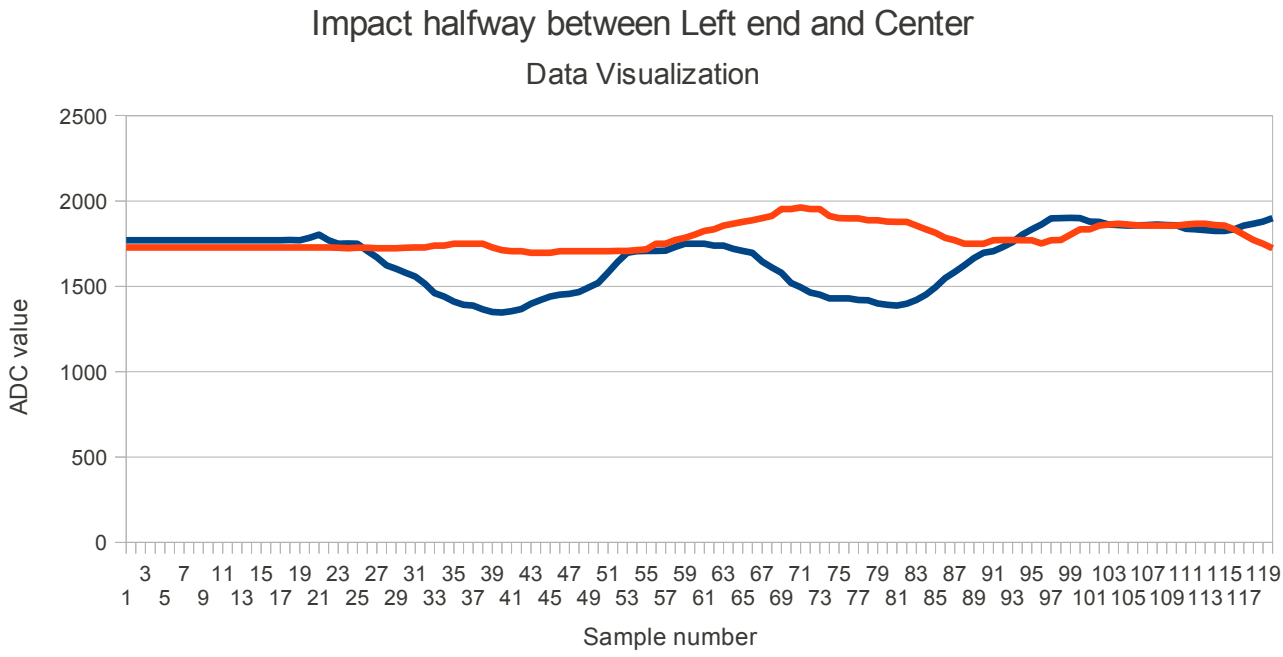


Figure 3: Half Left Data Graph

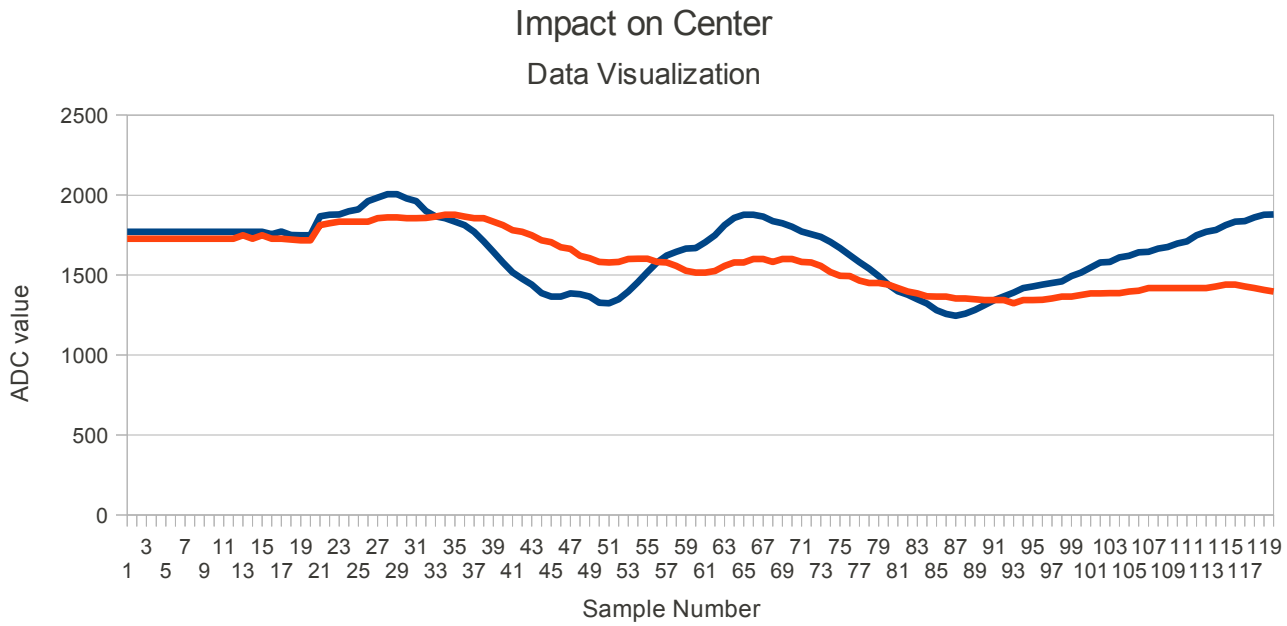


Figure 4: Center Data Graph

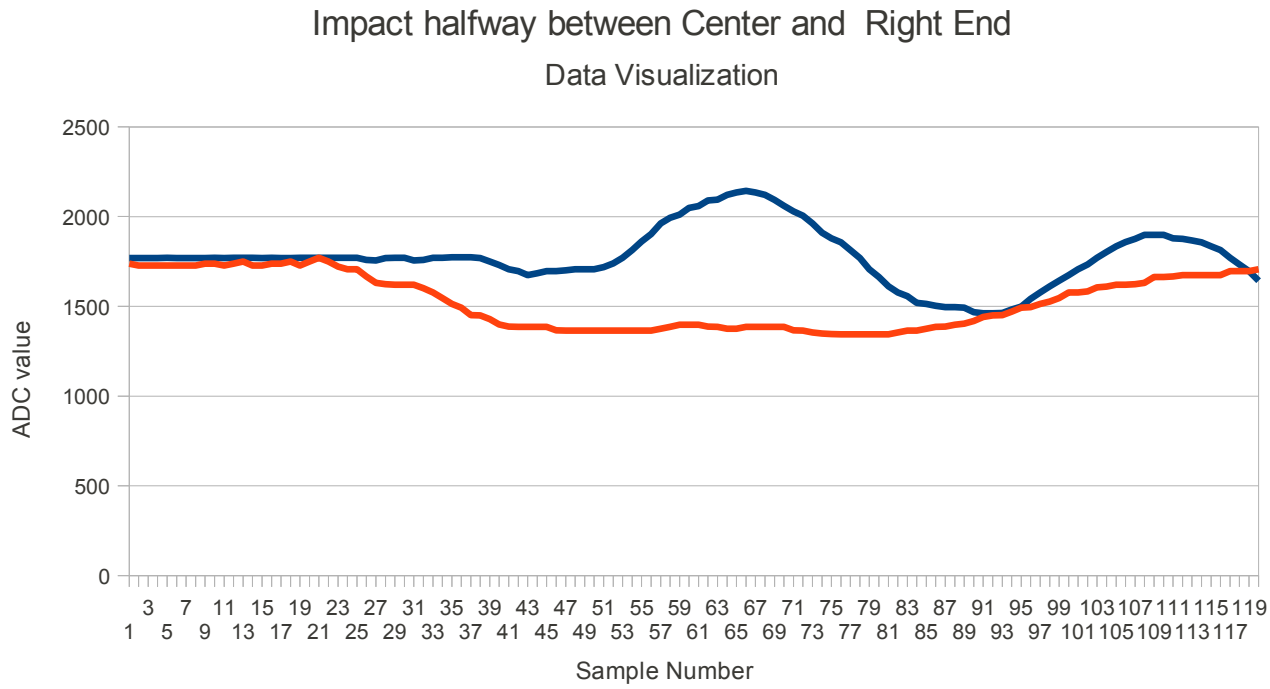


Figure 5: Half Right Data Graph

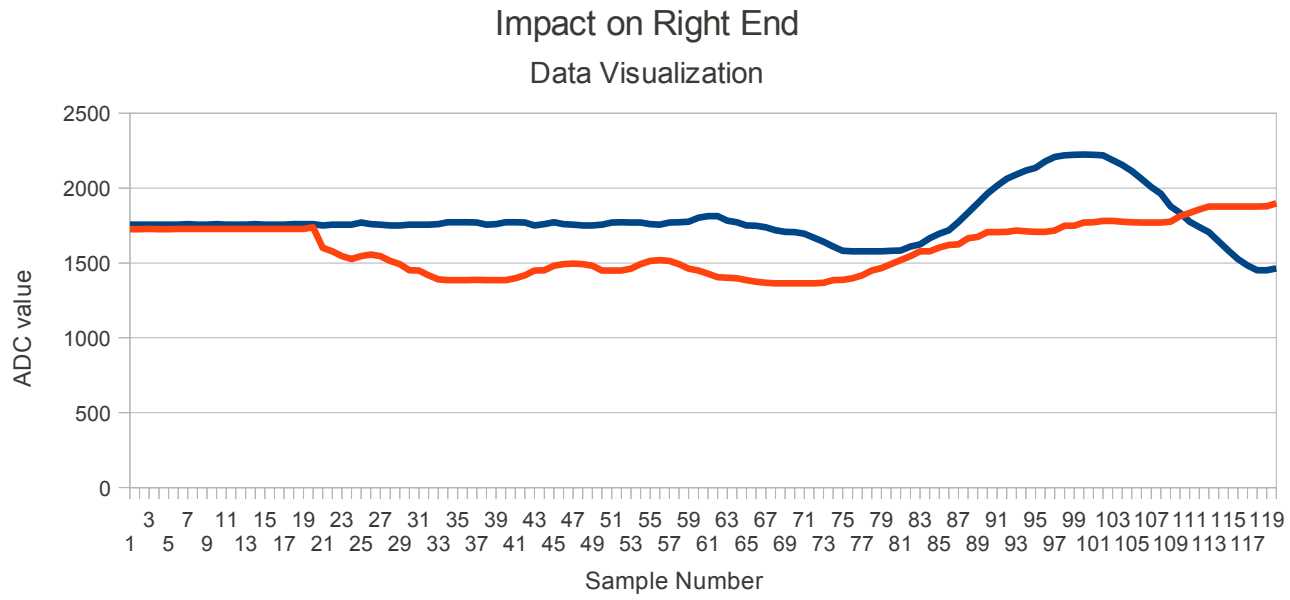


Figure 6: Right Data Graph

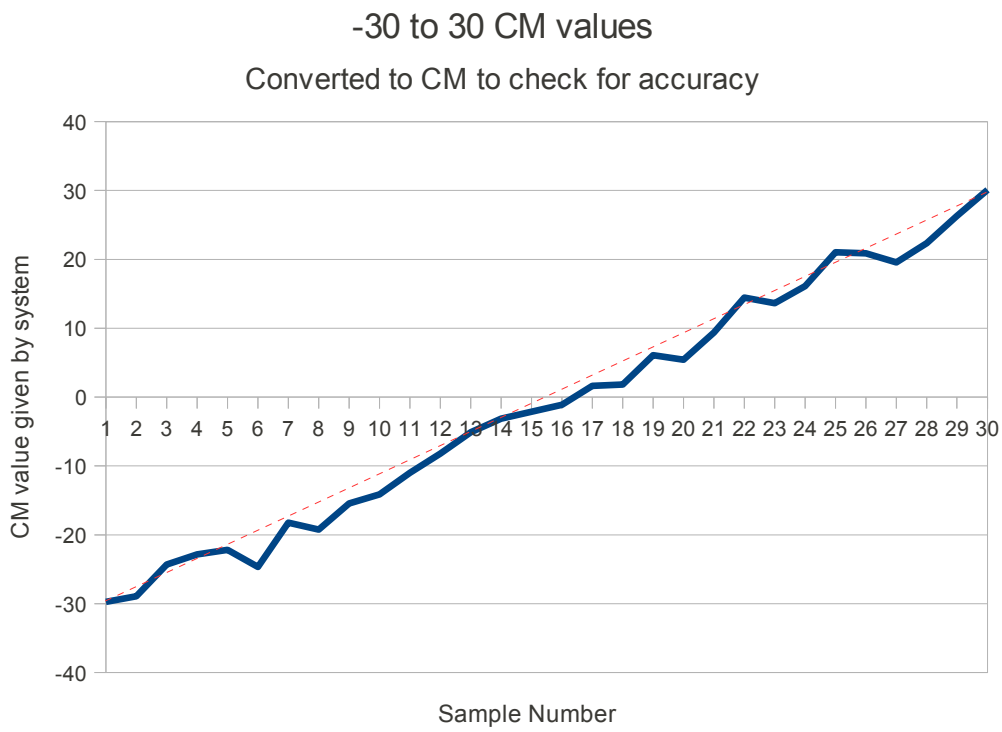
The interpretation of the these graphs is actually fairly simple: the x-axis is the sample number, which increases correspondingly with time. The y-axis is the value given by the analog-digital converter. As time passes (going to the right), the lines deviate from the normal course (seen on the left side of the graphs) and these changes represent vibrations detected by the piezoelectric sensors. The software on the Arduino detects the first deviation in one of the lines, stores the time that it occurred at, then scans the second line until it sees a deviation. By knowing the the time for the deviation in the first line, and the time for the deviation in the second line, a delay can be calculated. The delay is then converted into a distance from the center.
Using this logic, a delay of 0 microseconds means that the impact was equidistant from both sensors. This is perfectly demonstrated in Figure 4, where the deviation in both lines occurs almost simultaneously.

The graphs above are of only one trial from a set of 15. They are fairly good representatives of the ideal graph, since they were chosen on the basis of how close their result was to the actual (expected) value.

<u>Visual Representations of Entire Range of Measurements</u>

Another test was moving along the pipe with impacts spaced 2-cm apart, starting from the very left (~ -30 cm) and to the very right (~30 cm).
Ideally, this graph should be a line. Deviations represent inaccurate data where the system could not quite determine the correct location.

Note: This figure was constructed from 30 trials of data, but not all at the same point as in the previous graphs. In each trial, the impacts were spaced 2 cm apart
Note: Data used to construct this graph can be found in: Appendix B; section 6.



Ideally, this graph should be a straight line from -30 to 30 cm (red dotted line). While this graph isn't quite straight, it is still accurate +-2 cm in the worst case. The center region and the right end of the graph seem especially accurate compared to the others.

Conclusions

In general, this project was a success. The concept was proved to be viable and working, to a large extent. From the Results section, we can see that at the 5 chosen points where repetitive trails were run, the system returns quite consistent and accurate results, with the exception of the site labeled 'Half Left', where the standard deviation is too large to be very useful.

As noted once before earlier, in the two regions closest to the piezoelectric sensors do not register with the system. This is assumed to be due to the screws used to hold the pipe in place. Here is a visualization of the problematic areas.

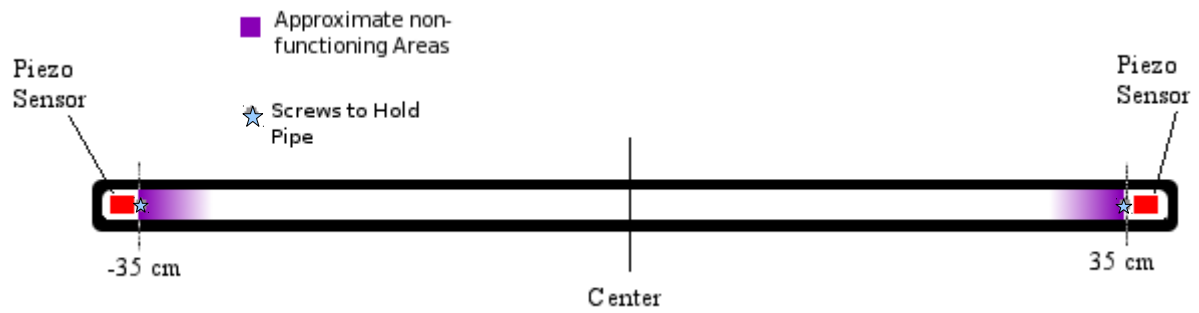


Figure 8: Approximate Non-Functioning Areas

A major revision to the project was the addition of the op-amps. These allowed the AD converters to read the whole signal rather than the half-rectified signal where the negative voltages were cut off. An example of what the operational amplifiers changed in the project can be seen in this image:

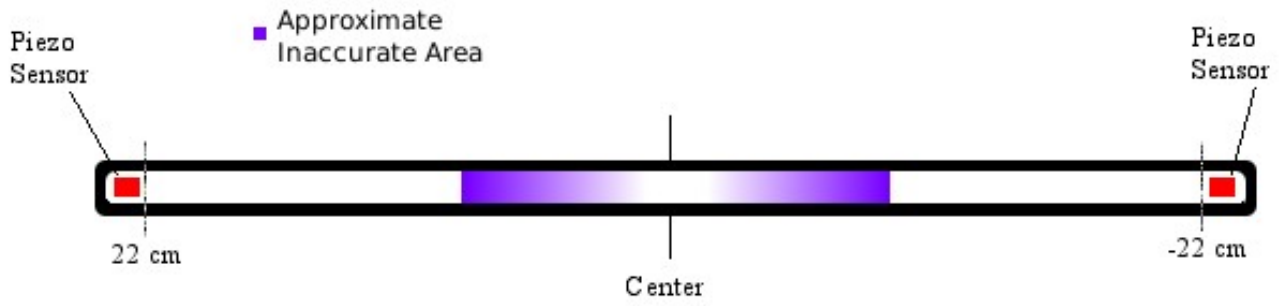


Figure 9: Previously Inaccurate Areas - without Op-amps

The op-amps dramatically increased the accuracy of the system in the center regions, where results previously inaccurate by more than 100%.

Besides making the idea work, this project was also a learning experience, if not primarily so. Having had no experience in electronics, microcontrollers, and interfaces before, this project was a direct and immediate immersion into all these topics. As an example, I had not known that all chips need some sort of timing clock pulse to keep them working properly, and that these are often created by ceramic or quartz resonators. In fact, after finding out this fact, I identified the quartz and ceramic resonators used on the Arduino (one for the USB interface – quartz, and one for the Arduino itself – ceramic). Naturally, finding something new leads to curiosity and then reading about it, making this type project a particularly effective way of learning about new concepts.

Other Remaining Problems

Before the problem of negative voltages was solved, there was another problem. It had to do with electrical interference. When the project was in its beginning stages, this was a major problem, and it took a while to even find it existed. I realized that it was there after moving my project to a downstairs room, where I was no longer sitting on a light fixture in the ceiling of the floor below me, producing amazingly large amounts of 60 Hz household interference on the oscilloscope I was using for testing. This is evidence toward how much was learned in this project; I didn't even realize electrical interference was a problem, nor did I realize that there were entire fields of research and work dedicated to studying such phenomenon and minimizing them. Even though I eventually converted the long cables connecting the piezo film sensors to the ADC's to coaxial (shielded cables that don't get affected by interference), there could be much more work in this area to reduce noise. Something interesting to find out about interference would also be to see if rewiring the electronics to be more compact would affect results via crosstalk. Electrical interference is one of those problems which is always there (radio, analog TV, etc.) in plain sight, but is almost never questioned. Especially now that most things are digitally encoded this problem is ignored even more.

Future & Applications

Besides fixing the problem listed above, this project could be expanded into a planar version in the

future (2D instead of the current 1D). By implementing solutions to the problems above, the system could be made more accurate and more sensitive, greatly expanding the range of possibilities. In fact, if this can be made very reliable and portable, it could lead to small piezoelectric sensors connected to wireless units, which can be attached to any surface to transform it into a touchscreen!

More realistically and within the scope of this project (and not future iterations), here are some applications:

- Cheap impact detection and localization such as:
  - Impacts on structures such as antennae
  - Impacts on aircraft
  - Impacts on spacecraft and general localization
  - Damage on pipes (like hitting them with a shovel while digging)
  - Damage on bridges (such as when pieces crumble and fall)
- Use-anywhere linear musical instruments (just attach to surface) such as xylophones
- Using several linear localizers:
  - keyboard (one for each row)
  - In-home wall knock-screens

Wrap Up

While the original conceptual idea of scaling down earthquake-style detection could not be implemented, this simpler linear version is a definite step in that direction. Future projects could expand into that domain, opening whole new potential applications. Indeed, this project's function could primarily be viewed as a springboard into next generation of piezoelectric localization.

Background Research

This project started out based on the idea behind earthquakes location - using triangulation to locate the epicenter. After reading into how earthquake detecting systems worked, time was spent figuring out which methods could be translated into something useful, and which had to be discarded since they could not suit application. The idea was then simplified to account for time constraints and lack of previous knowledge, leaving the need for some sort of vibration sensing equipment and a microcontroller as a basic start.

Earthquake detection goes back to the ancient Chinese. However, nowadays, not only are there machines that notify that an earthquake happened somewhere, but also pinpoint exactly where that incident occurred (Pennington). The exact workings of a modern seismograph are not critical at this point, since were are not used in this project. However, they are still the underlying technology that make the earthquake detection systems work, and so worth discussing. Some useful clues that can be used in the project appear in this overview.

Before digital seismometers (sometimes called seismographs), recordings were created using a pen attached to a mass, which acted like a pendulum, with the tip of the pen resting on a barrel of paper that continually rotated. The resulting paper is called a seismogram. During an earthquake, the structure supporting the mass and the pen sways, but the mass prevents the pen from moving. Spikes on drawn on the paper then corresponded in time and amplitude to the time and force of the wave that moved the support. A scientist could then identify the fast P waves and the slower S waves (Pennington). With the speed of each type of wave known (from previous measurements), the time difference is translated into a distance. Once three different seismometers have such readings, it is a simple matter to triangulate the epicenter of the earthquake.

While it would be nearly impossible to apply the idea of the pendulum and the roll of paper to a project

in which vibrations didn't even cause visible distortions, the idea of using time delays is definitely applicable. Also, the triangulation method sounded usable, so it was kept it in my mind while doing further research, which included research on measuring small-scale vibrations, something directly applicable to the project.

When detecting movement, large or small, it comes down to measuring a change from the normal state. The most commonly used device to do that is the accelerometer, which measures acceleration, and therefore also movement/vibrations. Accelerometers are found from hard drives, to planes, to phones, and due to their abundance seemed like an obvious choice for a project such as this one, with plenty of documentation and support (Guide to Accelerometers).

Accelerometers are very useful devices. However, they also seemed somewhat overkill for this project. They can provide specific data for the direction and amount of acceleration, both of which are not necessary. It was only necessary to have a device that produced a signal when moved, nothing more. Also, while trying to decide if accelerometers would be useful to the project, basic calculations showed a fast vibration with relatively high amplitude would not be detectable by some standard accelerometers found on the hobbyist market.

The word “piezoelectric” came up several times during research about accelerometers and how some of them worked internally. At the time, the word didn't appear to be something that could be useful, although by its definition it should have. The piezoelectric effect was discovered by the Curie brothers in the early 1900's. It can be defined as: the production of electricity or electric polarity by applying a mechanical stress to certain crystals (Merriam-Webster). In other words, materials that have this effect can turn movement into electrical signals, and vice versa. By definition then, piezoelectric materials are in the category of transducers.

There are two types of piezoelectric transducers, and they do opposite things. Piezoelectric actuators turn electrical signals into movement, and are commonly used in inchworm motors in medical instruments that need to be able to make extremely fine movements. On the other hand, piezoelectric sensors turn mechanical stress into electricity. This mechanical stress includes bending in any direction. Modern piezo films have many useful properties as transducers. Among these, a frequency range from one to one mega-hertz, operating temperatures up to 100 degrees Celsius, high voltage output, and more (Piezo Film Sensors 5). The fact that they can be shaped into almost any size or form, along with the ability to glue them with commercial adhesives make them very flexible in application (7).

Therefore, piezoelectric sensors are very well suited for this application. Their high voltage output from relatively little bending negates the need for any sort of amplifier. Also, since the vibrations are oscillating at unknown speeds, it's much safer to use a device which is rated for a higher frequency (compared to many accelerometers). The ease of application onto a surface would simplify the testing process, since adhesives like tape and glue are easy to remove and reapply onto a new surface or area for further testing.

Something eventually has to be at the core of this system, controlling everything happens. This is the job of the microcontroller, analogous to a tiny computer without operating system that can be made to do a variety of jobs. Sometimes microcontrollers are wrongly labeled microprocessors. The difference lies in the fact that microprocessors are meant to be more stand-alone in application, such as acting as the heart of small personal computers. Microcontrollers are part of the embedded systems engineering category. An embedded system can be defined as such: any software system that must be designed on a platform different from the platform on which the system is intended to be deployed (Bessin). This definition comes from the fact that an embedded system is optimized for simplicity and/or performance.

In the professional world, embedded systems are often designed and built specifically for a certain task, in order to maximize efficiency and minimize chance of failure (Bessin). The “one size fits all” idea does not work very well in the embedded world. The problem with such a system is that it caters to too many people, providing much more function than is often necessary. For instance, to make a system that turns on all the lights when a door is opened, it is not necessary to have an embedded system with fifty input/output pins, an accelerometer, and a microcontroller running at 100 mHz. All one might need is two input pin, two outputs, and a much slower controller. In the first system, which would definitely be able to do the same job, there is much more waste: the price would be much higher, unnecessary parts, etc.

However, for an inexperienced hobbyist, the “one size fits all” is the most desirable configuration. Embedded systems professionals have the knowledge and time to create their own printed circuit boards with customized layouts and configurations. With the present lack of skill and time to acquire the necessary expertise, creating a board from scratch is not a viable way to go about doing this project. Hence, a simple, pre-made microcontroller is the best fit. And the best place to look for these kinds of materials is in established hobbyist communities.

The two main microcontrollers that came up during research were the BASIC Stamp and the Arduino, with the former being the older one. According to some articles, the Arduino has been touted as “The BASIC Stamp killer” (Kurt). An overview of each microcontroller makes it clear which is more beneficial for this project.

The BASIC Stamp is a controller board developed by Parallax, Inc. It uses a version of the user-friendly BASIC programming language called PBASIC to run (BASIC Stamp software 2). One important fact is that when it runs using a small interpreter rather than running compiled machine code (more on this later). The BASIC Stamp has been popular since the 1990's, and since has developed a

huge user base of amateurs that use this package for their projects (Leclerc 1). Because of this, there are many well-documented tutorials, sample projects, and forums dedicated to this chip. However, some things that make the BASIC Stamp less practical in some cases include the relatively high cost, low memory, slow execution speed, and to some enthusiasts, the lack of openness and ability to hack the package (it is proprietary) (Leclerc 2). Of all these, the first two (cost and memory) are probably the biggest obstacles preventing more people from using the BASIC Stamp.

The Arduino, for the most part, fills these holes in the BASIC Stamp. Most noticeable is the price. The BASIC Stamp 2 costs around fifty US dollars. On top of that, to properly use the BASIC Stamp, another board is required (unless one directly connects to the Stamp with an RS232 serial cable), again costing $50, and bringing the total up to about 100 dollars. On the other hand, the Arduino Uno comes in a package currently costing thirty eight dollars, a very significant decrease. Apart from cost, the Arduino software is better suited for applications with higher speed requirements. Firstly, this is due to the fact that the Arduino requires compiled programs to be uploaded to the chip, which runs faster since they are already machine code. On the other hand, the BASIC Stamp has to re-interpret the code every time it runs, limiting speed (Kurt 2). Secondly, there are some additional libraries provided with the Arduino. If connecting to peripherals like external memory, analog-digital or digital-analog converters, etc. the Arduino has a built-in SPI protocol, which is faster than using high-level software to bit-bang a connection, which could be done on a BASIC Stamp. The Arduino also has some extra hardware (not related to speed), most notably the inclusion of an internal analog-digital converter, and therefore not only digital input pins, but also analog input pins (Kurt 3).

This project can be generally categorized as a high-speed application. Technically, this on its own is enough to rule out the BASIC Stamp, but further comparisons can't hurt. In the project, the other main component being used is the piezoelectric sensor, which is an analog device. Since the Arduino has a

built-in analog-digital converter, it would simplify the task a little. Even if an external AD converter is necessary for any reason (or some other peripheral), the Arduino has the SPI protocol which could be used to communicate with the device.

Overall, it would seem like this project would benefit from the Arduino more than the BASIC Stamp. Besides the software and hardware implications of each, the Arduino also tends to be viewed as a more "accessible" device, attracting many people. Still, the BASIC Stamp has had a long time to attract a larger user base. To some, this may or may not outweigh the fact that the BASIC Stamp is closed source and the Arduino is open source, so support for the latter is more widespread. For example Parallax (makers of the Stamp) only supports Windows software, with 'unofficial' software for Mac and Linux written by independent users (BASIC Stamp Software). The Arduino has supported software for all of these systems, and available source code for advanced users. Since this project is mostly made on the Ubuntu operating system (a Linux derivative), it also makes sense here to use the Arduino (Kurt 2).

Piezoelectric sensors are really the centerpiece of this project, which warrants a further look into how they are used today, and where they stand in relation to this project. This project has possible future applications in damage and impact detection and localization. However, in the field, many companies including NASA, General Motors, the US Army, General Electric, Westinghouse and more, generally use piezoelectric sensors for dynamic pressure and acoustic measurements, and not too much more (except for specialized designs). In the PCB Piezotronics Inc. (Pressure and Force Sensors Division) product catalog, piezoelectric sensors are advertised with many capabilities: machinery vibration testing, integrated systems, calibration equipment, microphones, and much more (2-5).

Another document gave a range of applications for piezo films. However, some seem extremely specialized, and probably required custom piezoelectric sensors. Among this list are ultrasound

applications (especially scientific and medical), fetal phonocardiographic transducers, hail sensors, and more. None of them really relate to this project, though some deal with damage detection using complex mathematical models, physics, and a combination of other sensors, as is normally used in detecting liquid leaking out of major piping systems (Piezo Film Technical Manual 60-70). These applications are far beyond the scope of this idea, though perhaps in future evolutions of the project a combination of physics, mathematics, and piezoelectric sensors can be combined to provide more accurate and consistent results.

However, besides the very different applications above, there have been many uses for piezoelectric sensors in the field of basic impact detection, though without localization. An example is the collision of two cars, in which one car hits another car from the side. Normally, accelerometers are used to detect sudden changes in velocity, useful when hit from the front or back, but when hit from the side there isn't always a change in velocity. Therefore, 'crush sensors' (really piezoelectric sensors) are sometimes placed in the door to activate airbags for additional safety (Hubbard). If anything, this kind of real-world use demonstrates that piezoelectric sensors are a good choice.

In the case that the analog inputs provided by the Arduino cannot handle the required speeds and voltage ranges, the use of an external analog-digital converter (ADC) will have to be explored, as mentioned before. With this need also comes the necessity of communication between Arduino and the ADC, which, also as mentioned before, can be accomplished with the Serial Peripheral Interface library in the Arduino software.

The most difficult part about choosing the right analog-digital converter is knowing what specifications to look for. Here's a list of the most basic and important characteristics of analog-digital converters: resolution, sampling speed, interfaces, package type. The first three are set part by part, while the last is more of an option (as in, the same part can be obtained in a variety of different packages).

Resolution, measured in bits, is a measurement of how finely an input be quantified by the AD converter (Analog Digital Conversion 2). For example, assuming an analog-digital converter accepts a range of 0-5 V, and it is an 8 bit ADC. Eight bits means that there can be $2^8 = 256$ different values for the whole range. 5V/256 = 19.5 mV/value. Now assume the same range of 0-5V but with a 16 bit ADC. 16 bits means that there are $2^{16} = 65536$ different values that can be identified, resulting in 5V/65536 = 0.0076 mV/value. As is shown here, a larger number of bits implies the ability to more accurately measure analog input. For this project, with an estimated voltage scale of -50 mV to 50 mV, an 8 bit AD converter would not have the accuracy, while a 16 bit converter would be too fine-grained. Somewhere in between is probably a good compromise.

Sampling speed is an indicator of how fast and therefore how often the ADC can sample in a certain time. However, the communication interface is more important in the speed characteristics of certain application. This stems from the fact that sampling is an almost instant process, while transmitting data takes huge amounts of time in comparison. Common interfaces used for ADC's are SPI and I^2C among others.

The last thing to consider is the package (or configuration) that the part comes in. A common mistake for first-time hobbyists is to buy a part that is actually meant for surface-mount application, meaning the part is really tiny and for machines to attach. Manual soldering and protoboard applications generally want to keep an eye out for dual-in-line packaging or something else that isn't a surface-mount part.

Overall, the research found simply provides a background from which to work. Some borrowed ideas, such as the idea of localizing vibrations can be transported into the project. Others, such as the old way of measuring earthquakes using a pendulum, are not so transmutable. The current normal uses of piezoelectric sensors help solidify the choice of a piezo film instead of an accelerometer. In the BASIC

Stamp vs Arduino choice, the Arduino was chosen due to its faster speed, higher functionality, and higher cross-platform usability. Finally, in the case that an external ADC is needed, general guidance was found and compiled into knowledge useful for acquiring the correct part.

Bibliography

"A Beginner's Guide to Accelerometers." *Dimension Engineering*.Web. Nov.-Dec. 2011. <http://www.dimensionengineering.com/accelerometers.htm>.

"Accelerometers and Accelerometer Info." *Omega*. Web. 05 Feb. 2012. <http://www.omega.com/prodinfo/accelerometers.html>.

“Analog to Digital Conversion”. Measurement Computing. PDF. <http://www.mccdaq.com/PDFs/specs/Analog-to-Digital.pdf>.

"BASIC Stamp Software." *Parallax.com*. Web. Nov.-Dec. 2011. <http://www.parallax.com/tabid/441/Default.aspx>.

Bessin, Geoffrey. "Embedded Systems: A Primer." *IBM.com*. 23 Nov. 2003. Web. 10 Dec. 2011. <http://www.ibm.com/developerworks/rational/library/806.html>.

Hubbard, James E. Smart Skin Sensor for Real-time Side Impact Detection and Off-line Diagnostics. Trustees of Boston University, assignee. Patent 5797623. 25 Aug. 1998. Print.

Karki, James. "Signal Conditioning Piezoelectric Sensors." Ti.com. 2000. Web. 14 Dec. 2011. <http://www.ti.com/lit/an/sloa033a/sloa033a.pdf>.

Kurt, Tod E. "Arduino, the Basic Stamp Killer « Todbot Blog." *Todbot.com*. Web. 10 Dec. 2011. <http://todbot.com/blog/2006/09/25/arduino-the-basic-stamp-killer/comment-page-1/>.

Leclerc, Vincent. "Basic Stamp 2 Workshop." 23 Nov. 2003. Web. 05 Feb. 2012. <http://uttermatter.com/bs2/>.

Pennington, Wayne. "How Are Earthquakes Studied?" *Geological Engineering & Sciences Michigan Tech*. Web. Nov.-Dec. 2011. <http://www.geo.mtu.edu/UPSeis/studying.html>.

Piezo Film Sensors Technical Manual. Measurement Specialities. <http://www.sparkfun.com/datasheets/Sensors/Flex/MSI-techman.pdf>

“Piezoelectric.” Merriam-Webster Dictionary. 11th ed. 2003.

“Pressure catalog”. PCB Piezotronics Pressure and Force Sensors Division.

<http://www.pcb.com/Linked_Documents/Pressure/PFScat.pdf>

Appendix A: Software

Microcontroller (Arduino – C++)
Data-collecting (Linux – Python)

1) Microcontroller Code – Arduino, C++ variant

```
#include <SPI.h>


const int ADSS1= 10;   //first AD is pin 10
const int ADSS2 = 9;   //second AD is pin
const byte TMP = B00111111;
const float multiplier = (22/715);

//adjustable constants
const long SPEED = 57600;   //Serial connection speed
const int RUNS = 3;         //number of times to refill array while (re)setting vars
const long LENGTH = 20;     //CAN'T BE BIGGER THAN 256 (byte)
const unsigned int MinThreshHold = (30) * LENGTH;   //modify the () only
//MinThresHold can't exceed 65536/LENGTH (for LENGTH = 50 that would be 1310)
//just treat the () as if it were the comparison #


//first impact variables (continously checking/evaluating)
 int dataAD1[LENGTH];
 int dataAD2[LENGTH];
 unsigned long dataTimes[LENGTH];
 long totAD1 = 0;
 long totAD2 = 0;
 int valAD1 = 0;    //long?
 int valAD2 = 0;
 byte counter = 0; //max 256
 unsigned long impactTime1;


//second impact variables (continously sampling, then evaluating)
 int fastAD1[100];
 int fastAD2[100];
 unsigned long timeStamp[100];
 unsigned long t;     //for filling timestamp array
 unsigned long impactTime2;

//success variable!
byte success = 0;


//--------------------------------------------------------------------------------

void setup() {
//configure pins as outputs
  pinMode(ADSS1, OUTPUT);
  pinMode(ADSS2, OUTPUT);
```

```
//set up SPI interface
  SPI.setBitOrder(MSBFIRST);
  SPI.setClockDivider(SPI_CLOCK_DIV2);
  SPI.setDataMode(SPI_MODE0);
  SPI.begin();

  setVars();

}


//-------------------------------------------------------------------------------------


void loop() {

//acquire data and assemble
  PORTB &= B11111011; // pull SS (pin 10) of arduino LOW (ADC sample)
  byte MSB1 = SPI.transfer(0);
  byte LSB1 = SPI.transfer(0);
  PORTB |= B00000100; // pull pin 10 of arduino to HIGH (ADC off) = PORT MANIPULATION
  valAD1 = ((MSB1 & TMP) << 6) | (LSB1 >> 2);

  PORTB &= B11111101; // pull SS (pin 9) of arduino LOW (ADC sample)
  byte MSB2 = SPI.transfer(0);
  byte LSB2 = SPI.transfer(0);
  PORTB |= B00000010; // pull SS (pin 9) of arduino HIGH (ADC off)
  valAD2 = ((MSB2 & TMP) << 6) | (LSB2 >> 2);

  impactTime1 = micros();

  //if it was the first ADC that registered the tap
  if (abs(totAD1-((long)valAD1*LENGTH)) >= MinThreshHold) {
    collectFast();  //fill data arrays fastAD1, fastAD2, timeStamp
    success = dataCrunch(fastAD2, dataAD2, totAD2); //crunch second data array
    if (success == 1) {
      long timeDelay = impactTime1 - impactTime2;  //unsigned long - another ul => int  seems to work
from test runs //Time2-Time1 is positive.       transmit(timeDelay); //transmit over Serial all the info
along with timeDelay
      delay(50); //estimated amount of time needed for return to normal behavior
      setVars(); //reset the data arrays and totals
    }
  }

  //if it was the second ADC that registered the tap
  else if (abs(totAD2-((long)valAD2*LENGTH)) >= MinThreshHold) {
    collectFast();  //fill data arrays cont AD1, cont AD2, timeStamp
    success = dataCrunch(fastAD1, dataAD1, totAD1); //crunch first data array
```

```
   if (success == 1) {
     long timeDelay = impactTime2 - impactTime1; //unsigned long - unsigned long => int  seems to
work from test runs //Time1 - Time2 is +
     transmit(timeDelay); //transmit over Serial all the info along with timeDelay
     delay(50); //estimated amount of time needed for return to normal behavior
     setVars(); //reset the data arrays and totals
   }
  }


//recording and math
  dataAD1[counter] = valAD1;                    //write into the current spot
  dataAD2[counter] = valAD2;
  dataTimes[counter] = impactTime1;


  totAD1 -= dataAD1[((counter+1)%LENGTH)];  //subtract the next spot, which is the oldest value
  totAD1 += valAD1;                     //add the new value


  totAD2 -= dataAD2[((counter+1)%LENGTH)];
  totAD2 += valAD2;

  counter++;                                //increase counter
  if (counter == LENGTH) {                  //reset counter ever LENGTH, faster than doing it every largest
multiple of LENGTH that fits in an int
     counter = 0;                           //reset to 0; all this is equivalent to counter%LENGTH but then
counter would overflow after 65536
  }

}
//end of loop()
//-----------------------------------------------------------------------------------------


void setVars() {
  int nums = LENGTH*RUNS; //to account for spike or drop at beginning

  //reset arrays of data and time
  for (int x = 0; x < LENGTH; x++) {
    dataAD1[x] = 0;
    dataAD2[x] = 0;
    dataTimes[x] = 0;
  }
  //to initialize totAD1 and totAD2 correctly, needs to start at the right value
  totAD1 = 0;
  totAD1 = 0;
  counter = 0;
```

```
   PORTB &= B11111011; // faster PORT MANIPULATION vs digitalWrite() (pull pin 10 LOW)
(ADC sample)
   byte MSB1 = SPI.transfer(0); //faster to use byte MSB/SLSBhere instead of declaring
   byte LSB1 = SPI.transfer(0); //it at the beginning for some reason...
   PORTB |= B00000100; //change pin 10 of arduino to HIGH (=PORT MANIPULATION) (ADC off)
   totAD1 = ((MSB1 & TMP) << 6) | (LSB1 >> 2); //THIS IS THE DIFFERENCE!!!!!!!

   PORTB &= B11111101; // faster PORT MANIPULATION vs digitalWrite() (pull pin 9 LOW) (ADC
sample)
   byte MSB2 = SPI.transfer(0); //faster to use byte MSB/SLSBhere instead of declaring
   byte LSB2 = SPI.transfer(0); //it at the beginning for some reason...
   PORTB |= B00000010; //change pin 9 of arduino to HIGH (=PORT MANIPULATION) (ADC off)
   totAD2 = ((MSB2 & TMP) << 6) | (LSB2 >> 2);//THIS IS THE DIFFERENCE!!!!!!!

  for (int x = 0; x < nums; x++) {
   PORTB &= B11111011; // faster PORT MANIPULATION vs digitalWrite() (pull pin 10 LOW)
(ADC sample)
   byte MSB1 = SPI.transfer(0); //faster to use byte MSB/SLSBhere instead of declaring
   byte LSB1 = SPI.transfer(0); //it at the beginning for some reason...
   PORTB |= B00000100; //change pin 10 of arduino to HIGH (=PORT MANIPULATION) (ADC off)
   valAD1 = ((MSB1 & TMP) << 6) | (LSB1 >> 2);

   PORTB &= B11111101; // faster PORT MANIPULATION vs digitalWrite() (pull pin 9 LOW) (ADC
sample)
   byte MSB2 = SPI.transfer(0); //faster to use byte MSB/SLSBhere instead of declaring
   byte LSB2 = SPI.transfer(0); //it at the beginning for some reason...
   PORTB |= B00000010; //change pin 9 of arduino to HIGH (=PORT MANIPULATION) (ADC off)
   valAD2 = ((MSB2 & TMP) << 6) | (LSB2 >> 2);

   dataAD1[counter] = valAD1;                    //write into the current spot
   totAD1 -= dataAD1[((counter+1)%LENGTH)];  //subtract the next spot, which is the oldest value
   totAD1 += valAD1;                             //add the new value

   dataAD2[counter] = valAD2;
   totAD2 -= dataAD2[((counter+1)%LENGTH)];
   totAD2 += valAD2;

   dataTimes[counter] = micros();

   counter++;                          //increase counter
   if (counter == LENGTH) {                  //reset counter ever LENGTH, faster than doing it every
largest multiple of LENGTH that fits in an int
     counter = 0;                      //reset to 0; all this is equivalent to counter%LENGTH but then
counter would overflow after 65536
   }
   //counter = 0 when exits loop and function
  }
  success = 0; //reset success byte variable
```

```
}

//-------------------------------------------------------------------------------------

//function to fill fast data arrays with data
void collectFast() {
   //variables for this are declared in beginning ("second impact variables")

   //collect 100 data from each quickly
   for (int x = 0; x < 100; x++) {
     PORTB &= B11111011; // pull SS (pin 10) of arduino LOW (ADC sample) = PORT
MANIPULATION
     byte MSB1 = SPI.transfer(0);
     byte LSB1 = SPI.transfer(0);
     PORTB |= B00000100; // pull pin 10 of arduino to HIGH (ADC off)
     valAD1 = ((MSB1 & TMP) << 6) | (LSB1 >> 2);

     PORTB &= B11111101; // pull SS (pin 9) of arduino LOW (ADC sample)
     byte MSB2 = SPI.transfer(0);
     byte LSB2 = SPI.transfer(0);
     PORTB |= B00000010; // pull SS (pin 9) of arduino HIGH (ADC off)
     valAD2 = ((MSB2 & TMP) << 6) | (LSB2 >> 2);
     t = micros();

     //add data to arrays
     fastAD1[x] = valAD1;
     fastAD2[x] = valAD2;
     timeStamp[x] = t;
   }
}

//-------------------------------------------------------------------------------------

int dataCrunch(int fastDataArray[], int origDataArray[], long total) {
  //use previous data/total
  //PLAN: hook together old LENGTH number of data from before the first sensor gets triggered to
new faster data
  //old data and newer fastDataArray are sampled at different speeds
  long combData[100+LENGTH]; //for hooking together old data and new fast data

  //copy older and faster new data into single longer array
  for (int x = 0; x < LENGTH; x++) {
    combData[x] = origDataArray[(counter+1+x)%LENGTH]; //assign oldest value first: counter is
newest value, so (counter+1)[oldest] + (x)[increment] = oldest to newest
  }

  for (int x = 0; x < 100; x++) {
    combData[x+LENGTH] = fastDataArray[x];
```

```
  }

  //copy total into new long so total doesn't get changed
  long cpTotal = total;

  for (int x = LENGTH; x < (100+LENGTH); x++) {
    if (abs(cpTotal-(combData[x]*LENGTH)) >= MinThreshHold) { //if the current total-val*LENGTH
is greater than the set threshold
      impactTime2 = timeStamp[x-LENGTH]; //set global var impactTime2 to the timeStampe
associated with the current data being checked.
      return 1;           //found! escape the function :)
    }
    cpTotal -= combData[x-LENGTH]; //remove oldest data in list
    cpTotal += combData[x];         //add newer data
  }
  return 0; //failed to find anything :(
}

//--------------------------------------------------------------------------------

int transmit(long Delay) {
   if (abs(Delay) > 750) {
     return 0;
   }
   Serial.begin(SPEED);
   Serial.println("\nTap!");
   for (int x = 0; x < LENGTH; x++) {
     Serial.print(dataAD1[(counter+x)%LENGTH]); //rearrage into correct order oldest to newest while
printing
     Serial.print("  ");
     Serial.print(dataAD2[(counter+x)%LENGTH]);
     Serial.print("  ");
     Serial.println(dataTimes[(counter+x)%LENGTH]);
   }
   Serial.println("Fast data!");
   for (int x = 0; x < 100; x++) {
     Serial.print(fastAD1[x]);
     Serial.print("  ");
     Serial.print(fastAD2[x]);
     Serial.print("  ");
     Serial.println(timeStamp[x]);
   }
   Serial.println("times\n");
   Serial.println(impactTime1);
   Serial.println(impactTime2);
   Serial.println(Delay);
   //Serial.println("Distance from middle: ");
   //float timeD = multiplier * Delay;
```

```
  //Serial.println(timeD,1);
  Serial.println("END");
  Serial.end();
 return 1;
}
```

2) Data Collecting Code – Linux, Python

A. Single Point Data Collection

```
#! /usr/bin/python
#for getting and saving the repetitive single point testing data
import serial
ser = serial.Serial("/dev/ttyACM0", 57600, timeout=1)
folders = ['left','halfLeft','center','halfRight','right']
cur = 0
info = ''
for y in folders:
        cur = 0
        for x in range(16): #change the range # for number of trials
                curFile = open("/home/joshua/Desktop/trials/%s" %x,"w")
                while "END" not in info:
                        info = ser.readline()
                        curFile.write(info)
                curFile.close()
                info = ''
                cur += 1
```

B. Whole Range Data Collection

```
#! /usr/bin/python
#for getting and saving the data from tapping up the pipe from one side to the other
import serial
ser = serial.Serial("/dev/ttyACM0", 57600, timeout=1)
cur = 0
info = ''
for x in range(31): #change the range # for number of recordings
        curFile = open("/home/joshua/Desktop/trials/CM/%s" %cur,"w")
        while "END" not in info:
                info = ser.readline()
                curFile.write(info)
        curFile.close()
        info = ''
        cur += 1
```

Appendix B: Raw Data

Data Collected Using Programs in Appendix A and Data collected By Hand

1. Left (15 Trials)	**2. Half Left (15 Trials)**	**3. Center (15 Trials)**	**4. Half Right (15 Trials)**	**5. Right (15 Trials)**
-25.9726027397	-17.2602739726	-3.45205479452	15.9452054795	30.5753424658
-27.2876712329	-17.7534246575	-3.45205479452	16.602739726	29.5890410959
-26.4657534247	-17.9178082192	-1.31506849315	16.9315068493	29.9178082192
-25.8082191781	-17.7534246575	-1.31506849315	16.602739726	29.095890411
-28.9315068493	-26.7945205479	-1.31506849315	16.9315068493	29.7534246575
-27.2876712329	-17.5890410959	-1.31506849315	17.4246575342	30.0821917808
-27.6164383562	-18.7397260274	-1.15068493151	15.9452054795	29.9178082192
-25.9726027397	-16.9315068493	-1.97260273973	17.2602739726	28.4383561644
-25.1506849315	-18.5753424658	-3.45205479452	15.9452054795	29.2602739726
-26.7945205479	-20.7123287671	-1.15068493151	17.5890410959	28.9315068493
-27.4520547945	-17.2602739726	-1.31506849315	15.1232876712	30.7397260274
-26.7945205479	-26.6301369863	-1.15068493151	16.602739726	30.0821917808
-27.4520547945	-17.5890410959	-2.95890410959	16.602739726	29.7534246575
-27.9452054795	-14.301369863	-2.79452054795	15.7808219178	30.5753424658
-26.4657534247	-14.6301369863	-1.31506849315	16.1095890411	29.2602739726

6. 2 cm Intervals (Left to Right)	***(cont)***
-29.7534246575	-1.1095890411
-28.9315068493	1.64383561644
-24.3287671233	1.80821917808
-22.8493150685	6.08219178082
-22.1917808219	5.42465753425
-24.6575342466	9.3698630137
-18.2465753425	14.4657534247
-19.2328767123	13.6438356164
-15.4520547945	16.1095890411
-14.1369863014	21.0410958904
-11.0136986301	20.8767123288
-8.21917808219	19.5616438356
-5.09589041096	22.3561643836
-3.12328767123	26.301369863
-2.13698630137	30.0821917808

7. Data Used to Make Graphs in Results Section

Left

Tap!

1759 1727 6068716
1759 1722 6068776
1759 1727 6068832
1759 1727 6068892
1770 1725 6068956
1755 1722 6069016
1755 1727 6069072
1755 1725 6069132
1759 1727 6069188
1755 1727 6069256
1755 1723 6069316
1757 1727 6069372
1755 1725 6069432
1755 1722 6069488
1755 1725 6069544
1755 1725 6069600
1755 1725 6069656
1755 1725 6069712
1755 1725 6069768
1770 1723 6069824

Fast data!

1643 1722 6069912
1581 1717 6069932
1527 1717 6069952
1493 1719 6069968
1482 1719 6069988
1495 1722 6070004
1517 1722 6070024
1503 1722 6070040
1463 1722 6070060
1455 1719 6070076
1442 1717 6070096
1410 1722 6070112
1413 1727 6070132
1450 1727 6070148
1495 1749 6070168
1568 1727 6070184
1629 1727 6070204
1665 1722 6070224
1666 1722 6070240
1706 1725 6070260
1771 1727 6070280
1834 1725 6070300
1814 1719 6070320
1805 1722 6070336
1834 1725 6070356
1877 1727 6070372
1952 1727 6070392

1963 1725 6070408
1952 1719 6070428
1898 1725 6070444
1858 1727 6070464
1824 1749 6070480
1759 1749 6070500
1707 1749 6070516
1669 1749 6070536
1622 1749 6070552
1578 1751 6070572
1527 1770 6070588
1519 1771 6070608
1527 1771 6070628
1546 1771 6070644
1517 1770 6070664
1455 1749 6070680
1397 1717 6070700
1349 1706 6070716
1327 1706 6070736
1344 1706 6070752
1386 1696 6070772
1450 1696 6070788
1514 1674 6070808
1578 1669 6070824
1611 1666 6070844
1622 1664 6070860
1642 1664 6070880
1666 1664 6070900
1685 1664 6070916
1706 1664 6070936
1707 1642 6070952
1738 1631 6070972
1775 1642 6070988
1856 1664 6071008
1901 1666 6071024
1952 1674 6071044
1962 1685 6071060
1952 1696 6071080
1930 1706 6071096
1910 1711 6071116
1903 1749 6071132
1903 1771 6071152
1903 1824 6071168
1901 1856 6071188
1898 1877 6071208
1898 1898 6071224
1877 1911 6071244
1856 1930 6071260
1814 1930 6071280
1771 1930 6071304
1738 1952 6071320
1730 1952 6071340
1733 1962 6071356
1749 1962 6071376
1751 1963 6071392
1770 1967 6071412
1755 1973 6071428
1751 1973 6071448

1749 1963 6071468
1719 1962 6071484
1711 1962 6071504
1717 1952 6071520
1749 1909 6071540
1773 1898 6071556
1814 1877 6071576
1856 1866 6071592
1879 1856 6071612
1911 1856 6071628
1952 1857 6071648
1962 1866 6071664
1962 1866 6071684
1962 1866 6071700
1962 1866 6071720

Half Left

Tap!
1770 1727 22887236
1770 1727 22887292
1770 1727 22887352
1770 1727 22887416
1770 1727 22887488
1770 1727 22887548
1770 1727 22887612
1770 1727 22887676
1770 1727 22887740
1770 1727 22887800
1770 1727 22887864
1770 1727 22887928
1770 1727 22887992
1770 1727 22888056
1770 1727 22888116
1770 1727 22888180
1770 1727 22888244
1771 1727 22888308
1770 1727 22888368
1783 1727 22888432
Fast data!
1802 1727 22888536
1770 1727 22888552
1749 1725 22888572
1750 1722 22888588
1749 1725 22888608
1707 1725 22888624
1669 1722 22888644
1623 1722 22888660
1602 1722 22888680
1578 1725 22888696
1557 1727 22888716
1515 1727 22888732
1461 1738 22888752
1440 1738 22888772
1410 1749 22888788
1391 1749 22888808
1386 1749 22888824
1365 1749 22888844

1349 1727 22888860
1346 1711 22888880
1354 1706 22888896
1367 1706 22888916
1397 1696 22888932
1419 1696 22888952
1440 1696 22888968
1450 1706 22888988
1455 1706 22889004
1467 1706 22889024
1493 1706 22889040
1519 1706 22889060
1581 1706 22889080
1643 1707 22889096
1696 1707 22889116
1706 1711 22889132
1707 1717 22889152
1707 1749 22889168
1709 1749 22889188
1730 1771 22889204
1749 1783 22889224
1749 1802 22889240
1749 1824 22889260
1739 1834 22889276
1738 1856 22889296
1719 1866 22889312
1707 1877 22889332
1696 1887 22889352
1645 1899 22889368
1610 1911 22889388
1578 1952 22889404
1519 1952 22889424
1495 1962 22889440
1463 1952 22889460
1450 1952 22889480
1429 1911 22889500
1429 1899 22889520
1429 1898 22889536
1419 1898 22889556
1418 1887 22889572
1399 1887 22889592
1391 1879 22889608
1387 1877 22889628
1397 1877 22889644
1419 1856 22889664
1451 1834 22889680
1494 1813 22889700
1547 1783 22889716
1583 1770 22889736
1623 1749 22889752
1664 1749 22889772
1696 1749 22889788
1706 1770 22889808
1730 1771 22889828
1759 1771 22889844
1802 1770 22889864
1834 1770 22889880
1861 1751 22889900

1898 1770 22889916
1899 1771 22889936
1901 1802 22889952
1899 1834 22889972
1879 1834 22889988
1877 1856 22890008
1861 1861 22890024
1858 1866 22890044
1856 1861 22890060
1857 1857 22890080
1858 1856 22890100
1861 1856 22890116
1858 1856 22890136
1857 1856 22890152
1837 1861 22890172
1834 1866 22890188
1829 1866 22890208
1824 1858 22890224
1824 1856 22890244
1834 1834 22890260
1856 1802 22890280
1866 1771 22890296
1879 1749 22890316
1899 1722 22890332
times

22888504
22888932
-428
END

times

6069884
6070588
-704

Center

Tap!
1770 1727 49996616
1770 1727 49996676
1770 1727 49996732
1770 1727 49996788
1770 1727 49996852
1770 1727 49996908
1770 1727 49996964
1770 1727 49997020
1770 1727 49997076
1770 1727 49997132
1770 1727 49997188
1770 1727 49997244
1770 1749 49997300
1770 1727 49997360
1771 1749 49997416
1755 1727 49997480
1773 1727 49997536
1751 1723 49997596

1749 1717 49997652
1749 1717 49997708
Fast data!
1867 1813 49997796
1877 1824 49997816
1879 1834 49997840
1899 1834 49997856
1911 1834 49997876
1962 1834 49997892
1983 1856 49997912
2006 1861 49997928
2006 1861 49997948
1979 1856 49997964
1962 1856 49997984
1901 1857 49998000
1867 1866 49998020
1856 1877 49998036
1834 1877 49998056
1813 1866 49998076
1771 1856 49998092
1711 1856 49998112
1645 1834 49998128
1578 1813 49998148
1517 1781 49998164
1477 1770 49998184
1440 1749 49998200
1387 1717 49998220
1365 1706 49998236
1366 1674 49998256
1386 1664 49998272
1381 1621 49998292
1365 1605 49998308
1327 1583 49998328
1323 1578 49998348
1349 1583 49998364
1397 1601 49998384
1455 1602 49998400
1519 1602 49998420
1579 1583 49998436
1622 1578 49998456
1645 1557 49998472
1665 1527 49998492
1669 1515 49998508
1706 1515 49998528
1749 1525 49998544
1813 1557 49998564
1858 1578 49998580
1878 1578 49998600
1878 1600 49998616
1866 1600 49998636
1837 1583 49998656
1824 1600 49998672
1802 1600 49998692
1773 1583 49998708
1755 1578 49998728
1739 1557 49998744
1707 1519 49998764
1669 1495 49998780

1623 1493 49998800
1581 1466 49998816
1541 1451 49998836
1493 1450 49998860
1440 1440 49998876
1399 1418 49998896
1378 1397 49998916
1349 1386 49998932
1322 1367 49998952
1281 1365 49998968
1258 1365 49998988
1246 1354 49999004
1259 1354 49999024
1282 1349 49999040
1312 1344 49999060
1344 1344 49999076
1367 1344 49999096
1391 1323 49999112
1418 1344 49999132
1429 1344 49999148
1440 1345 49999168
1450 1354 49999184
1461 1365 49999204
1493 1365 49999224
1515 1375 49999240
1547 1386 49999260
1578 1386 49999276
1583 1387 49999296
1610 1387 49999312
1621 1397 49999332
1642 1402 49999348
1646 1418 49999368
1666 1418 49999384
1675 1418 49999404
1697 1418 49999420
1711 1418 49999440
1749 1418 49999456
1771 1418 49999476
1783 1429 49999496
1813 1440 49999512
1834 1440 49999532
1837 1429 49999548
1861 1418 49999568
1877 1407 49999584
1879 1397 49999604
times

49997764
49997796
-32
END

Half Right

Tap!
1770 1738 94300480
1770 1727 94300540
1770 1727 94300600
1770 1727 94300660
1771 1727 94300720
1770 1727 94300780
1770 1727 94300840
1770 1727 94300900
1770 1738 94300960
1771 1738 94301024
1770 1727 94301088
1771 1738 94301144
1771 1749 94301216
1771 1727 94301280
1770 1727 94301336
1771 1738 94301396
1770 1738 94301460
1770 1749 94301524
1771 1727 94301588
1771 1749 94301648
Fast data!
1771 1771 94301752
1771 1749 94301768
1771 1722 94301788
1771 1707 94301804
1771 1706 94301824
1759 1666 94301840
1755 1631 94301860
1770 1623 94301880
1771 1621 94301896
1771 1621 94301916
1755 1621 94301932
1759 1601 94301952
1771 1578 94301968
1771 1546 94301988
1773 1514 94302004
1773 1493 94302024
1773 1451 94302040
1770 1450 94302060
1749 1429 94302076
1730 1399 94302096
1706 1387 94302112
1696 1386 94302132
1674 1386 94302148
1685 1386 94302168
1696 1386 94302188
1696 1367 94302204
1701 1365 94302228
1706 1365 94302248
1706 1365 94302264
1707 1365 94302284
1719 1365 94302300
1739 1365 94302320
1771 1365 94302336
1813 1365 94302356
1861 1365 94302372
1903 1365 94302392
1962 1375 94302408
1994 1386 94302428
2011 1397 94302448
2048 1397 94302464
2059 1397 94302484
2090 1387 94302500
2095 1386 94302520
2122 1375 94302536
2135 1375 94302556
2143 1386 94302572
2134 1386 94302592
2122 1386 94302608
2093 1386 94302628
2061 1386 94302644
2029 1367 94302664
2005 1365 94302680
1962 1354 94302700
1911 1349 94302716
1879 1345 94302736
1857 1344 94302756
1814 1344 94302772
1770 1344 94302792
1707 1344 94302808
1664 1344 94302828
1611 1344 94302844
1578 1354 94302864
1557 1365 94302880
1519 1365 94302900
1514 1375 94302916
1503 1386 94302936
1495 1387 94302952
1495 1397 94302972
1493 1403 94302988
1467 1418 94303008
1461 1440 94303028
1461 1450 94303044
1463 1451 94303064
1482 1470 94303080
1499 1493 94303100
1541 1495 94303116
1578 1514 94303136
1611 1527 94303152
1643 1546 94303172
1674 1578 94303188
1706 1578 94303208
1733 1583 94303224
1771 1605 94303248
1803 1610 94303268
1834 1621 94303284
1858 1621 94303304
1877 1623 94303324
1898 1631 94303340
1898 1664 94303360
1898 1664 94303376
1879 1666 94303396
1877 1674 94303412
1867 1674 94303432
1856 1674 94303448
1834 1674 94303468
1813 1674 94303484
1771 1696 94303504
1733 1696 94303520
1696 1696 94303540
1643 1706 94303556
times

94301708
94302096
388
END

Right

Tap!
1755 1725 138601512
1755 1725 138601576
1755 1727 138601640
1755 1725 138601700
1755 1725 138601764
1755 1727 138601828
1759 1727 138601892
1755 1727 138601956
1755 1727 138602016
1759 1727 138602080
1755 1727 138602144
1755 1727 138602204
1755 1727 138602264
1759 1727 138602324
1755 1727 138602388
1755 1727 138602444
1755 1727 138602512
1759 1727 138602568
1759 1727 138602628
1759 1738 138602684
Fast data!
1751 1600 138602784
1755 1578 138602800
1755 1546 138602820
1755 1527 138602840
1770 1546 138602856
1759 1557 138602876
1755 1546 138602892
1751 1514 138602912
1751 1493 138602928
1755 1451 138602948
1755 1450 138602964
1755 1418 138602984
1759 1391 138603000
1771 1386 138603020
1771 1386 138603036
1771 1386 138603056
1770 1387 138603072
1755 1386 138603092
1759 1386 138603108
1771 1386 138603128
1771 1397 138603148
1770 1418 138603164
1751 1450 138603184
1759 1451 138603200
1771 1482 138603220
1759 1493 138603236
1755 1495 138603256
1751 1493 138603272
1751 1482 138603292
1755 1450 138603308
1770 1450 138603328
1771 1450 138603344
1770 1461 138603364
1770 1493 138603380
1759 1514 138603400
1755 1519 138603420
1770 1514 138603436
1771 1493 138603456
1775 1461 138603472
1802 1450 138603492
1813 1429 138603508
1813 1405 138603532
1783 1402 138603552
1771 1397 138603568
1750 1386 138603588
1749 1375 138603604
1738 1367 138603624
1719 1365 138603640
1707 1365 138603660
1706 1365 138603676
1696 1365 138603696
1669 1365 138603716
1642 1367 138603732
1610 1386 138603752
1581 1387 138603768
1578 1397 138603788
1578 1418 138603804
1578 1450 138603824
1578 1466 138603840
1581 1493 138603860
1583 1519 138603876
1610 1546 138603896
1623 1578 138603912
1664 1578 138603932
1696 1602 138603948
1719 1621 138603968
1773 1623 138603988
1835 1664 138604004
1898 1674 138604024
1963 1706 138604040
2015 1706 138604060
2061 1707 138604076
2091 1717 138604096
2117 1711 138604112
2135 1707 138604132
2177 1707 138604148
2208 1717 138604168
2218 1749 138604184
2221 1749 138604204
2223 1770 138604220
2221 1771 138604240
2218 1781 138604256
2187 1781 138604276

2154 1775 138604296 1706 1877 138604456
2113 1771 138604312 1643 1877 138604476 138602748
2061 1770 138604332 1583 1877 138604492 138603492
2006 1770 138604348 1527 1877 138604512 744
1962 1770 138604368 1483 1877 138604528 END
1879 1775 138604384 1451 1877 138604552
1834 1813 138604404 1451 1879 138604572
1775 1834 138604420 1463 1898 138604592
1739 1857 138604440 times