ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Capacitance Sensors for Mason Bee Activity Measurement

Team #27

JYOTSNA JOSHI
(jyotsna3@illinois.edu)
KEERAT SINGH
(keerats2@illinois.edu)
PIYUSH SUD
(psud2@illinois.edu)

TA: Amr Ghoname

May 4, 2022

Abstract

A serious problem in U.S. agriculture is the declining honey bee population. While the honey bee's hypothesized affliction, Colony Collapse Disorder, is still being researched, we are proposing that native bee species can help support pollination activities in North America. Mason bees are one such bee population, a solitary bee species that lives in tunnels and pollinates native plants more efficiently than the honey bee. This paper outlines our design for a sensor-equipped mason bee house, which enables beekeepers to collect data regarding their kept mason bees. Using capacitive sensors, we can infer when bees enter and exit the tunnels. To simulate bee activity, we moved 6 mm spherical glass beads through the tunnels and confirmed that our system can differentiate between bees and parasites. The data generated by our device is logged onto an SD card for further analysis by the beekeeper. We hope with access to this information, beekeepers are empowered to make the best decisions for their bees and boost local pollinator populations.

Contents

1 Introduction

1.1 Objective and Background

The well-known loss of pollinators around the world has been a concern for ecologists, agriculturalists, farmers, and everyday citizens for several years. While the exact reason for the infamous Colony Collapse Disorder affecting honeybees is unknown, there is interest in boosting their populations so that honey bees can be the thriving pollinators they were once. But what if we are focusing too squarely on the honey bees? What if there is an effective way to support native plant pollination with a bee population that is native to North America? That would be the mason bee, *megachilidae osmia*. Mason bees, solitary bees that do not live in hives, do not make honey, and usually do not sting, are a species of bee that can pollinate an area of native plants almost 100 times more efficiently than the western honeybee [1]. They are a good species for even amateur apiarists to keep near their homes to boost local native bee populations and participate in solving global ecological issues.

1.2 Problem

There are some key hurdles to keeping mason bees. In nature, they live in dark crevices found in trees, rocks, or in the ground, but kept mason bees live in tunnel homes. Please refer to Figure 1 to see an example of these tunnel nests.

Female bees will populate these tunnels with eggs for next year, and seal the tunnel when they are done laying their eggs. Beekeepers must clean these tunnels, especially at the end of every season, and harvest the bee cocoons for the next year. It is often challenging to know when to clean a tunnel, especially if it is not possible to determine if the tunnel is occupied by an alive adult bee. This difficulty arises from the fact that these tunnels are long and dark, and so bee activity deep inside the tunnels is hard to gauge [2].

1.3 Solution

Our device is a sensor-equipped mason bee house so that beekeepers can be confident their bees are active and healthy. Beekeepers can view this collected bee movement data in aggregate and can infer when unwanted visitors like parasites are entering the bee home; with this information, beekeepers can be proactive about removing tunnels or cleaning the structure to prevent the spread of infections or parasites.

Implementation: At the core of our solution are capacitive sensors that non-intrusively detect bee behavior in tunnels. These sensor readings can be interpreted to determine when bees are entering and exiting, how much of the tunnels are being actively used, and when unwanted intrusions occur. This data is communicated to the beekeeper, who can view the collected data at their leisure. Please see Figure 2 to view a diagram detailing how a bee's activity is captured onto the user's SD card.

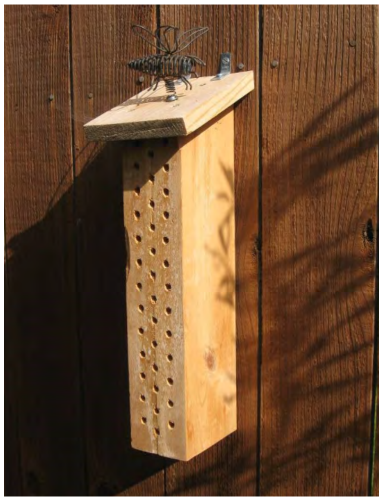


Figure 1: Wood block with holes drilled in to make tunnels for mason bee nesting.

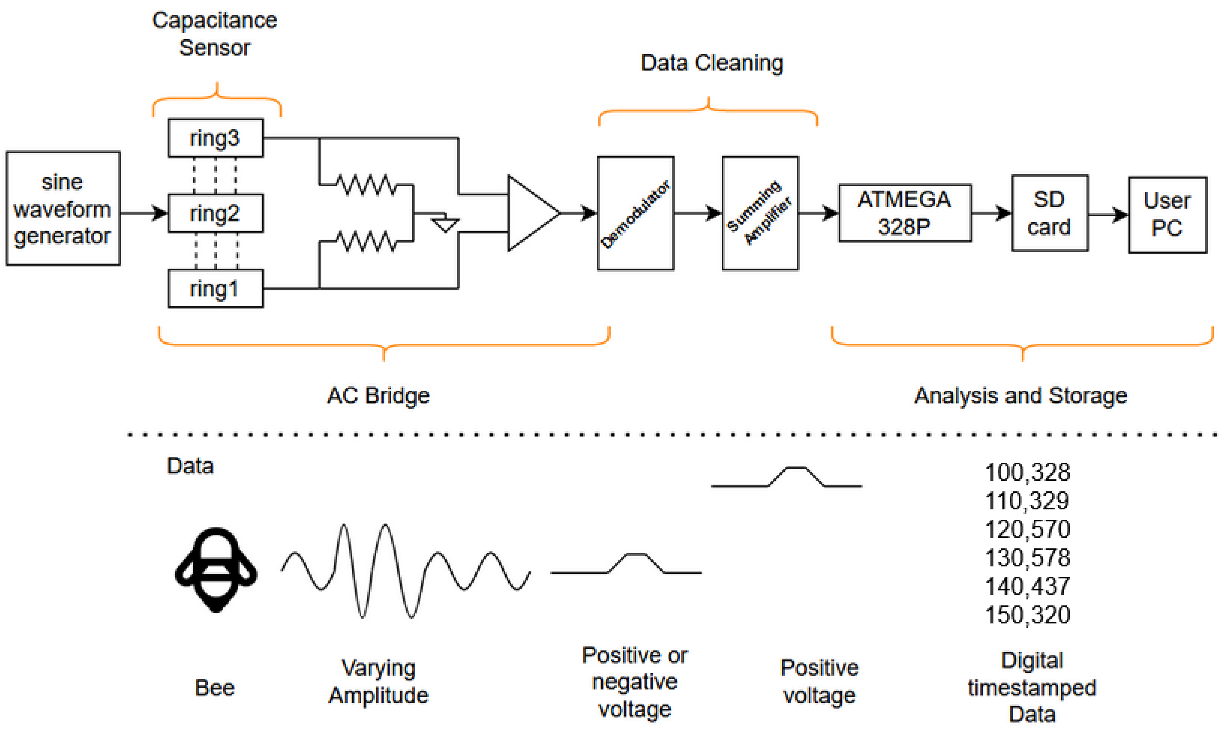
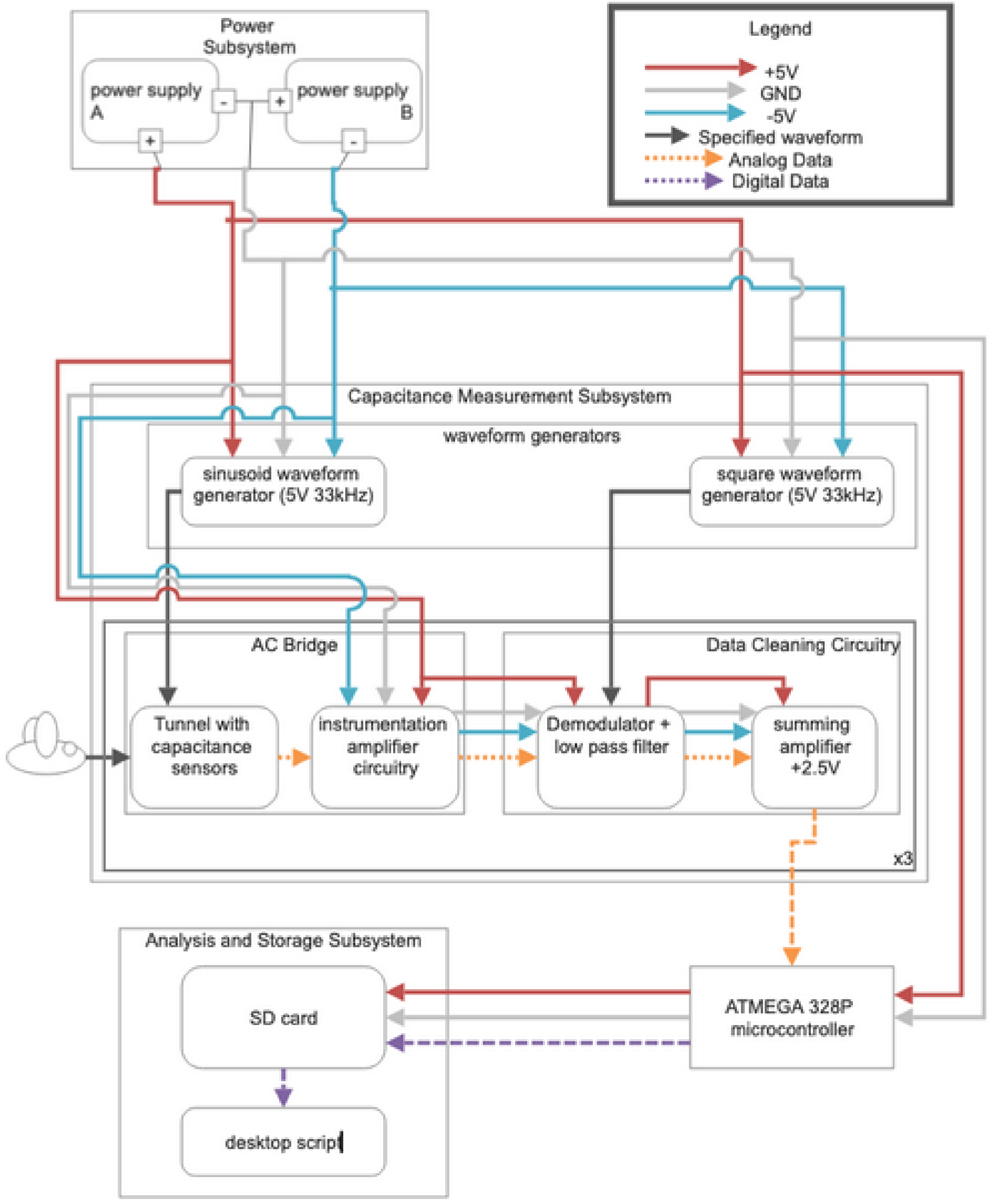


Figure 2: This diagram shows the process of turning a bee's movement into data that can be viewed on a user's PC.

1.4 Block Diagram



Power Subsystem
power supply A
power supply B
+
-
+
-
Legend
+5V
GND
-5V
Specified waveform
Analog Data
Digital Data
Capacitance Measurement Subsystem
waveform generators
sinusoid waveform generator (5V 33kHz)
square waveform generator (5V 33kHz)
AC Bridge
Data Cleaning Circuitry
Tunnel with capacitance sensors
instrumentation amplifier circuitry
Demodulator + low pass filter
summing amplifier +2.5V
x3
Analysis and Storage Subsystem
SD card
desktop script
ATMEGA 328P microcontroller

Figure 3: Block diagram of circuit diagram.

The blocks in Figure 3 are described as follows:

1. The Capacitance Measurement Subsystem, which is responsible for measuring the change in capacitance from the entry/exit of the bee as a sinusoidal signal of amplitude proportional to ΔC.
2. The Power Subsystem, which is responsible for generating a +5 V, GND, and -5 V power supply for the board.
3. The Data Cleaning Subsystem, which is responsible for extracting the bee data from the sinusoidal signal outputted by the Capacitance Measurement Subsystem.
4. The Analysis and Storage Subsystem, which stores data from the device in a readable format on an SD card, and does visualization and initial processing of the data.

1.5 High Level Requirements List

1. Capacitor sensor setup shall differentiate between entering and exiting simulated bees (6mm ball) in each tunnel.
2. Sensor setup shall differentiate between simulated bee and simulated parasite (3mm ball to approximate the size of the Houdini fruit fly, a parasite that eats mason bee larvae).
3. An integrated SD card shall store data about behavior of bees in the tunnel, which can be read and displayed by plugging it into a separate computer.

1.6 Demonstration

In our demonstration, we simulated the bee behavior using glass beads. It was important to select an electrically accurate bee simulation material to generate the most realistic change in capacitance during testing and demonstration. We found research documenting the dielectric constant of the western honey bee [3]. Since mason bees have roughly the same size and weight as western honeybees, we estimated that a mason bee's body has a dielectric constant between 10.32[4] and 10.64 [3]. For our demo, we used 6 mm radius glass beads as glass has a dielectric constant of about 10 and mason bee bodies are approximately 6 mm in size.

To test whether the device can differentiate between parasites and bees, we moved a 3 mm glass bead through the tunnel and compared the system's response to the response during a 6 mm size bee simulation. We confirmed that these created measurably different signals on the oscilloscope; the parasite created a smaller response and the bee created a larger response. This difference was reflected in the data recorded on the SD card.

2 Design

2.1 Design Procedure

2.1.1 Power Subsystem

For the power subsystem, we needed to provide +5 V, GND, and -5 V to all four of the boards. Our initial idea was to use a 10 V power supply and reference the supplied 5 V as ground, which would make the supplied 0 V function as -5 V. An alternate idea that came up in discussion was to use two separate 5 V supplies and connect the 5 V of one supply to the GND of the other. We selected the second option based on conclusions from our bee safety research. With outputs of 5 V, GND, and -5 V, the magnitude of change when compared to earth ground is never more than 5 V.

2.1.2 Mason Bee Housing Subsystem

For our mason bee house, we used a wooden frame structure with an angled roof, three tunnel slots, and one electronics storage slot. Each tunnel slot contains an acrylic tunnel. We mounted sensors onto only one tunnel to reduce the cost of the bee house prototype. The angled roof forms an attic above the highest tunnel so that eggs can be placed in this sheltered nook when it is time for young mason bees to hatch in the new season.

The dimensions of the tunnel were chosen to foster healthy mason bee populations. Research from environmental groups [2] has shown that male mason bees hatch from eggs laid near the mouth of the tunnel, and female mason bees hatch from eggs laid deep within the tunnel. Therefore, tunnels that are too short or too long can cause an unhealthy imbalance in mason bee populations. Additionally, tunnels that are too wide or too narrow will be ignored by mason bees. With this in mind, we initially designed our tunnels to have an 8 mm inner diameter and a length of 5.5 inches. After receiving feedback from the Machine Shop, we decided on tunnels with a 9 mm inner diameter due to material availability. The sensor design and shielding design was done with 8 mm tunnels in mind, but the increase in tunnel diameter meant that the sensor was not as sensitive as it could have been. We determined through breadboard prototyping that the sensor still had enough sensitivity to meet our requirements.

We chose acrylic for the tunnel material because it is non-conductive, resistant to breakage, and suitable for outdoor use. Our original plans included MOLEX connectors to connect the sensors on the tunnel to the boards mounted on the side of the house. However, the MOLEX connectors were too big to fit in the space between the tunnel and the outer copper shielding, so we used a VGA connector instead. With the connector in place, users can easily detach the wires leading to the sensors to remove and store their tunnels.

2.1.3 Capacitance Measurement Subsystem

The capacitor sensors that form the heart of our project are based on research conducted at the University of Prince Edward Island in 2005 [5]. In their paper "Capacitance-based sensor for monitoring bees passing through a tunnel", they describe a setup where they directed bumblebees to enter their hives through tunnels. In these tunnels, they had set up a "two-capacitor set-up along with an AC bridge and phase-sensitive detection" which "produced an asymmetric double pulse for each bee passage". We had two choices for the capacitor setup as specified in the paper - we could either use parallel plates or ring-shaped sensors. We chose to use the ring sensors because it was easier to fit those around the tunnels in a small package. Previous research also indicated that the ring design was more sensitive.

The spacing and measurements of the copper rings that form the capacitive sensor were initially chosen based on recommendations from previous research in tunnel-based insect monitoring [5], and fine-tuned with trial-and-error prototyping in the lab. We originally planned to make the sensor rings out of thick, 9 gauge copper wire, but our machinist, Mr. Glen Hedin, advised that this would be too difficult to bend into rings of our desired radius. Instead, we used cut sections of copper tubing to meet most of our requirements for the rings while also allowing for practical machining methods.

The circuitry associated with the sensor also went through many design iterations. At first, we constructed an instrumentation amplifier ourselves using multiple op-amps. However, after extensive prototyping, we found that our constructed instrumentation amplifier did not produce repeatable, reliable results. As a solution, we purchased the AD620 instrumentation amplifier because of its matched resistor pairs and quality control testing. The AD620 allowed our system to be very precise and enabled our AC bridge to be reactive to even small imbalances between its two sides.

Some of our first prototyping was centered around selecting the AC bridge's driving voltage. Based on lab tests, we knew a higher amplitude sinusoidal waveform would make the sensor more responsive to insect movement, so we decided to use a 5 V waveform since it provided the necessary sensitivity without requiring extra hardware to implement.

2.1.4 Data Cleaning Subsystem

The data cleaning subsystem takes the waveform output of the capacitance measurement subsystem and turns it into a pulse signal that can easily be read by the analysis and storage subsystem, as shown in Figure 2. To accomplish this, we use phase-sensitive detection, which is able to extract the small amplitude changes of the waveform amidst noise. This output is then fed through a low-pass filter to remove noise.

2.1.5 Analysis and Storage Subsystem

This subsystem takes continuous measurement from the sensors, stores the data, and does initial processing to determine potential entry and exit times. We chose an SD card for data storage because it is possible to log data directly onto an SD card from an ATMega328P, whereas it would require additional hardware to transmit the data wirelessly to another computer. Additionally, a layperson can read data from an SD card using a standard laptop without having to learn new technical skills or acquire specific hardware.

We chose to use the ATMega328P because it is a popular microcontroller that can be easily programmed using the Arduino IDE, is compatible with the USBasp programmers available in the lab, and can use readily available libraries to write data directly to an SD card.

The analysis portion of this subsystem takes place within a Python script that would run on the user's computer. We planned to do the analysis on the microcontroller, but we decided to perform the analysis on the user's computer instead. This is because our microcontroller could not analyze the collected data while maintaining the desired data collection rates.

2.2 Design Details

2.2.1 Power Subsystem

This subsystem had simple goals but a difficult implementation - we had to supply 5 V, GND, and -5 V to the board, but accomplish this in a way that would not disturb the bees. To do this, we connected the positive output of one power supply to the ground of the other, creating a three output power supply. We chose the ALP002 adjustable voltage power supply because it was relatively inexpensive and had a max current output of 1.5 A, which was far above the total current load from all of our chips. Crucially, it is designed with a floating ground, allowing us to force one of the power supplies to be negative.

The total current draw of all of the chips was calculated as follows:

$$I_{total} = I_{microcontrollerboard} + 3 * I_{sensorboard} \tag{1}$$

$$I_{total} = I_{ATmega328p} + 2 * I_{MAX038} + 2 * I_{uA741} + 3 * (I_{AD620} + I_{AD630} + 2 * I_{uA741}) \tag{2}$$

$$I_{total} = 1.5mA + 2 * 89mA + 3 * (300mA) \tag{3}$$

$$I_{total} = 1.0795A < I_{max} = 1.5A \tag{4}$$

The microcontroller board contains the ATMega328P, which draws 1.5 mA according to its datasheet [6]. The board also contains two MAX038 waveform generator ICs, which each have a maximum power dissipation of 889 mW [7]. Since we are using a 10 V (peak to peak) supply, and assuming the waveform generator follows Ohm's law, the maximum output current of each of the waveform generators is 889 mW/10 V = 89 mA. Lastly, the microcontroller board contains two HA2-2515-5 hi-slew op-amps. The input bias current is 80 nA, which is negligible, so the only significant current drawn is from the output load of the op-amp, which is the same as the current coming from the waveform generators.

The sensor board contains the resistors for the AC-bridge, an instrumentation amplifier (AD620), a balanced demodulator (AD630), and two op-amps. According to the ATMega328P's datasheet, the input impedance to the Analog Digital Converter (ADC) is 100 MOhms; for a voltage of 5 V, this corresponds to a current draw of 50 nA, which is negligible [6]. The balanced demodulator has a maximum bias current of 300 mA. While our use case uses a bias current of 100 mA, calculations were done with the 300 mA value to account for unexpected behavior. Lastly, the AD620 has an input bias current of 1 nA, which is also negligible.

We confirmed the current draw calculations above by monitoring the current draw displayed on the power supplies in lab while powering the fully connected circuitry.

2.2.2 Mason Bee Housing Subsystem

The Mason Bee Housing dimensions take into account mason bees' living needs along with the physical implications of our electrical design. Previous research [5] has determined that a tunnel-based capacitive sensor for tracking insect behavior is most effective when the shield radius is at least 1.5 times the diameter of the rings, and when the width of the rings is as thick as the design allows. Additionally, the distance between ring pairs should be between half to one radius apart. With these restrictions in mind, we created a design of 3 copper rings with a 9 mm radius, separated by 8 mm, as seen in figure 5. This makes up one sensor. A tube has three sensors: one at the mouth of the tube, one in the middle, and one in the deepest portion of the tube, as seen in figure 4.

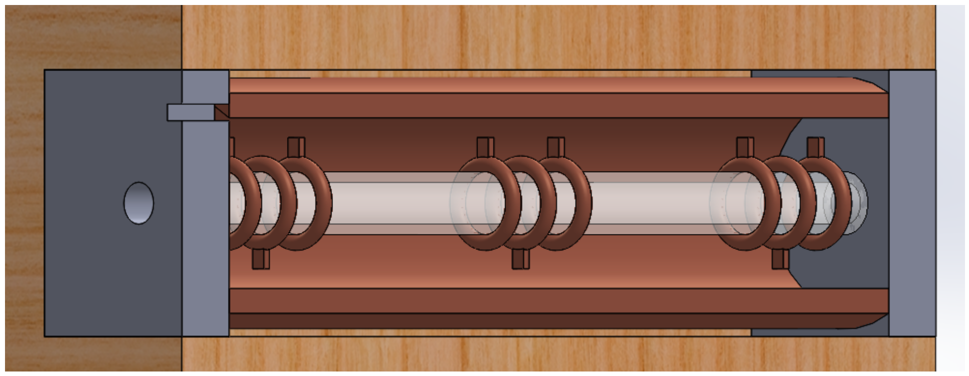


Figure 4: Cutaway visual of tunnel with 3 sensors and copper shielding.

2.2.3 Capacitance Measurement Subsystem

This subsystem is the most integral part of our mason bee house. It contains the AC bridge which consists of 2 resistor-capacitor pairs and an instrumentation amplifier that indicates whether the capacitance values are equal. As seen in Figure 2, the two outer rings of each capacitor sensor are connected to the resistors of the AC bridge, while the

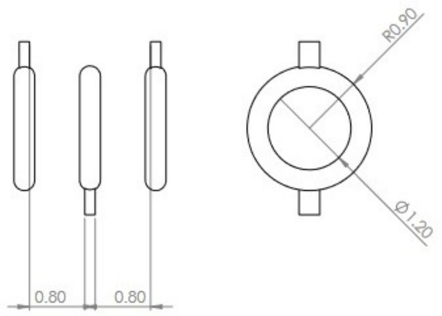


Figure 5: Dimensions of capacitor pair sensor

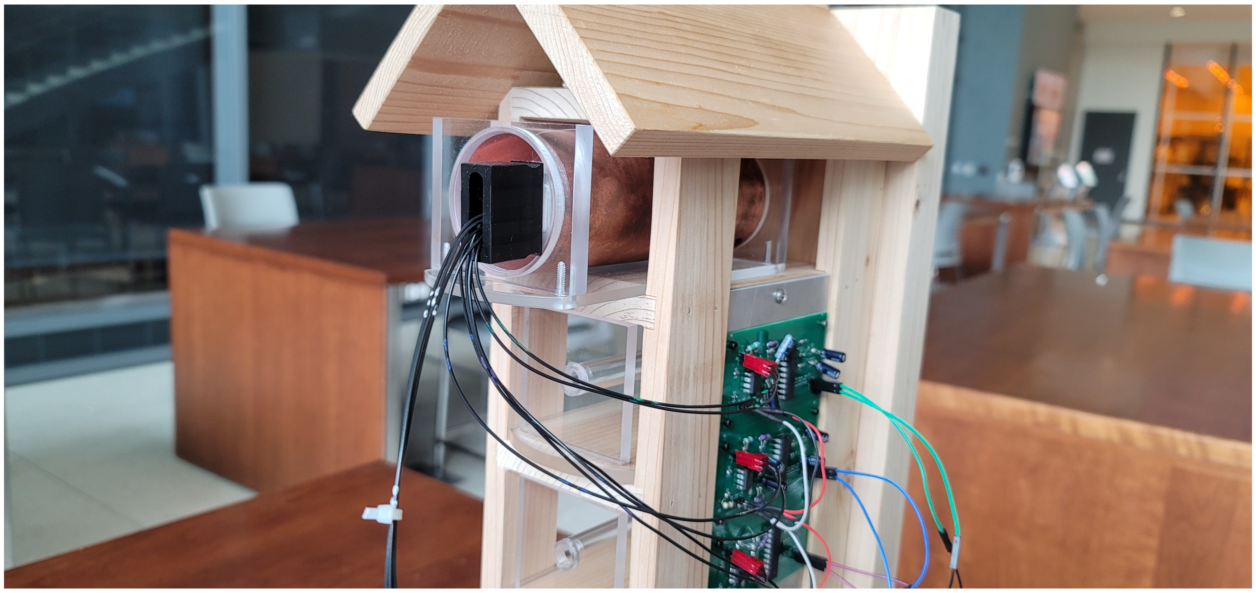


Figure 6: Image of the completed house, with VGA connector, acrylic tubing, sensor PCBs, and copper shielding visible

middle ring is connected to a sinusoidal voltage source. This configuration results in the outer two rings behaving like capacitors with the middle ring, which changes capacitance when a dielectric material (the bee) passes through it.

The capacitors are made of sections of thick copper, whose dimension and spacing considerations can be seen in Figure 5.

The sinusoidal waveform that drives the AC bridge is generated using a MAX038 waveform generator chip. It is set up using a 3300 pF capacitor and a 47K Ohm resistor to generate a frequency of approximately 33 kHz. The chip can only create waveforms with an amplitude of approximately 2.25 V, so we used an op-amp with a gain of 2 to create a final amplitude that meets our requirements. The resulting waveform can be seen in Figure 9.

Since each capacitor in a capacitive sensor set is connected to an arm of the AC bridge, the change in capacitance that occurs when a bee passes through causes the two arms of the AC bridge to become unbalanced. This results in an amplitude change from the instrumentation amplifier's output waveform; the direction of change depends on which capacitor the bee is going through. This information can be used to determine whether a bee is entering or exiting the tunnel.

The general equation for the capacitance measurement subsystem output is given by

$$V_{out} = \frac{GV_0\Delta C}{C\sqrt{(\frac{1}{\omega RC} - \omega RC)^2 + 4}} sin(\omega t + \frac{\pi}{2} + \arctan \frac{2}{\frac{1}{\omega RC} - \omega RC}) \tag{5}$$

The circuitry associated with each sensor was placed on the sensor PCB. We used three sensor PCBs: one for each sensor in our electrically outfitted tube. This PCB contains an AD620 instrumentation amplifier, an AD-630 balanced demodulator, a low-pass filter, and resistors to form the resistor-capacitor pair of the AC bridge, as well as an input for the square reference waveform necessary for the demodulator and inputs for the sensor from the tunnel. Each sensor board has one output, which is connected to the microcontroller board. The PCB design for this board can be seen in Figure 7.

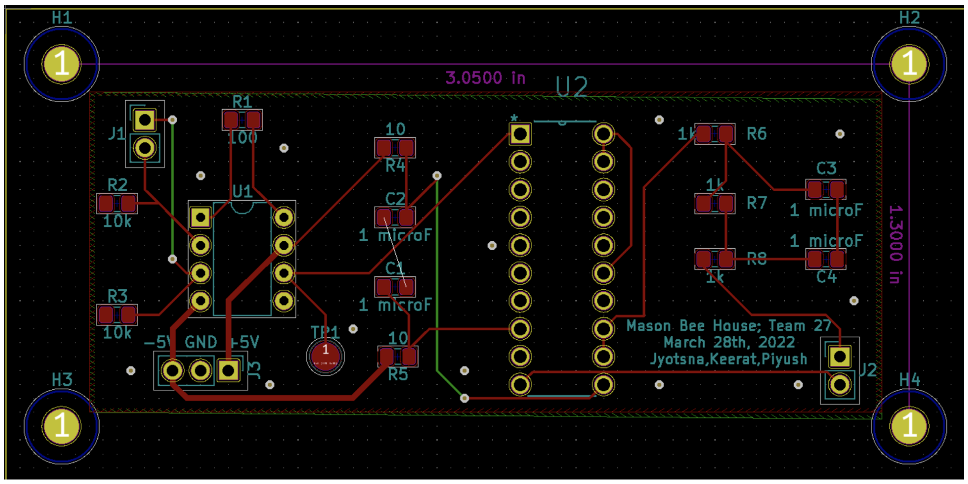


Figure 7: PCB layout of sensor board, including AC bridge, balanced demodulator, and low pass filters

2.2.4 Data Cleaning Subsystem

The data cleaning subsystem uses a technique known as phase-sensitive detection to extract the change in amplitude in the presence of noise. This is done as follows: the input to the balanced demodulator is inverted at a frequency equal to frequency of the sine wave. Since the noise is at a different frequency than the desired signal and is equally likely to be negative and positive, it eventually goes to zero, leaving the desired signal. This signal is a DC signal, which contains spikes and some high frequency noise. To get rid of the high frequency noise, the output is fed through a low-pass filter. This leaves a DC signal containing spikes whose amplitude is proportional to the change in capacitance of the sensor.

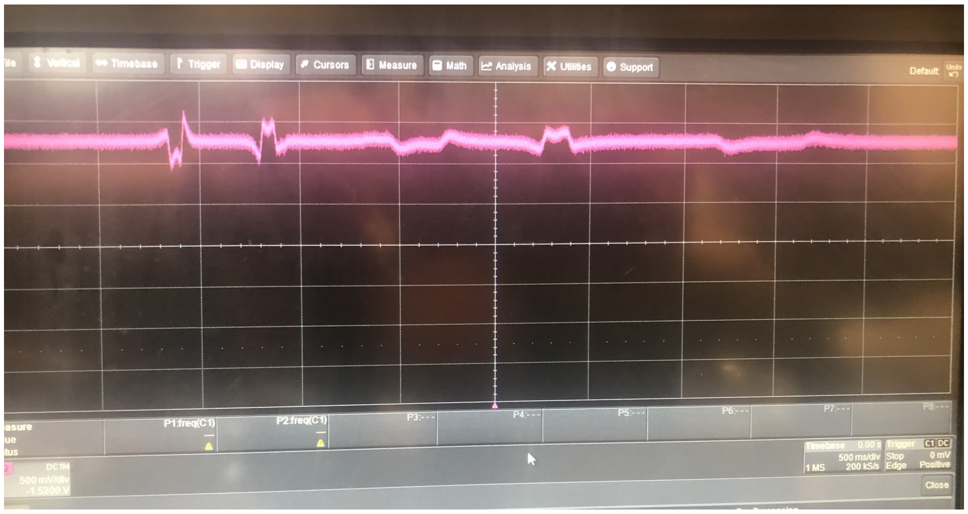


Figure 8: Negative and positive spike created when simulated bee enters and exits tunnel. Vertical scale of 500 mV/div and time scale of 500ms/div. First pair shows fast bee movement speed, second pair shows medium speed, and third pair shows slow speed.

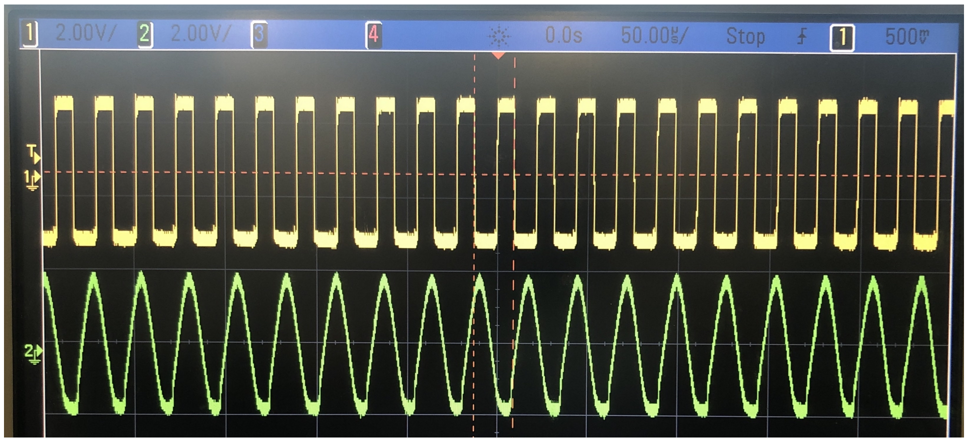


Figure 9: Square and sinusoidal waveform generated by MAX038 and hi-freq opamps

For the modulation frequency, we used the same 33 KHz frequency as the sinusoidal signal that drives the AC Bridge, because it is much higher than the frequency of the signal generated from the bee movement, so the phase-sensitive detection does not damage the signal of interest. We chose to use a 5 V waveform because a larger voltage results in a larger change in amplitude of the signal, which yields better results.

2.2.5 Analysis and Storage Subsystem

The analysis and storage subsystem consists of the ATMega328P microcontroller and the SD card used to store the data. The microcontroller PCB also includes an external 16 MHz oscillator which connects to the microcontroller.

The microcontroller board design incorporates breakout connections from the ATmega328P to an SD card module. This module is placed at the front of the bee house structure, so that the beekeeper can easily access the card. The data is written onto the SD card in a comma separated format (CSV). The CSV data is structured as (time, voltage_shallow, voltage_mid, voltage_deep) to reflect the voltage value collected at each of the three sensors within the tunnel.

The data on the SD card is then processed and visualized in a Python script on the user's laptop. The script displays the bee data to an interactive webpage on the user's computer, an example of which can be seen in Figure 11. It uses the plotly Python library to create the interactive graphic, which allows the user to mouse over traces for exact data points, highlight data from a specific sensor, and even zoom in on specific portions of the graph. In addition to visualizing the raw data, the script also highlights data points that could show insect activity in the tunnel. This is done through statistical analysis.

First, a rolling average of each sensor's data is taken, using overlapping windows of four data points each. This smooths out but does not entirely remove spikes in the data, and preserves more of the patterns that are present for longer time periods. In other words, a spike that lasted for five data points would be less smoothed out than a spike that lasted for only two data points. These numbers were chosen because our oscilloscope prototyping showed that simulated insect activity in the tubes created pulses of at least 100 ms, so spikes of only 30 ms (the approximate separation between each data point collected on the SD card) would not be significant. After the rolling average was applied, the median of relevant data from each sensor was computed. Data points where this median was crossed were identified as potential insect movement. Focusing on these data points, the data immediately before and immediately after the crossing was analyzed. If it was consistently above and then consistently below the median, or vice versa, it was a contender for the asymmetric pulse pattern that is generated when an insect passes a sensor. In this case, the data point is marked with a circle in the visualization.

The code running on the ATMega328P to write collected data to the SD card can be found in Appendix C. The Python code written to visualize the results are in Appendix B.

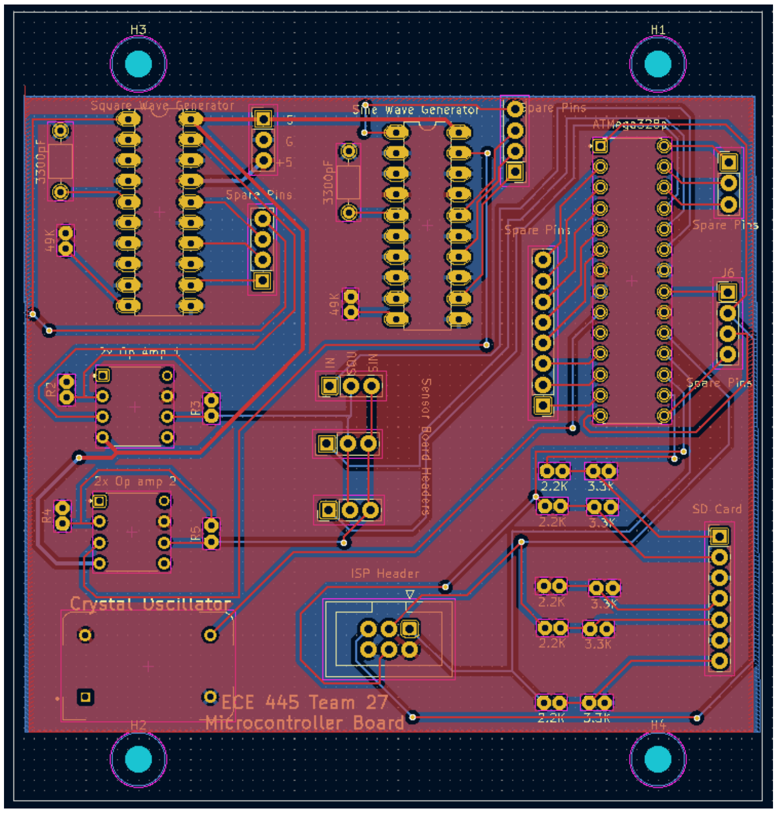


Figure 10: PCB layout of microcontroller board

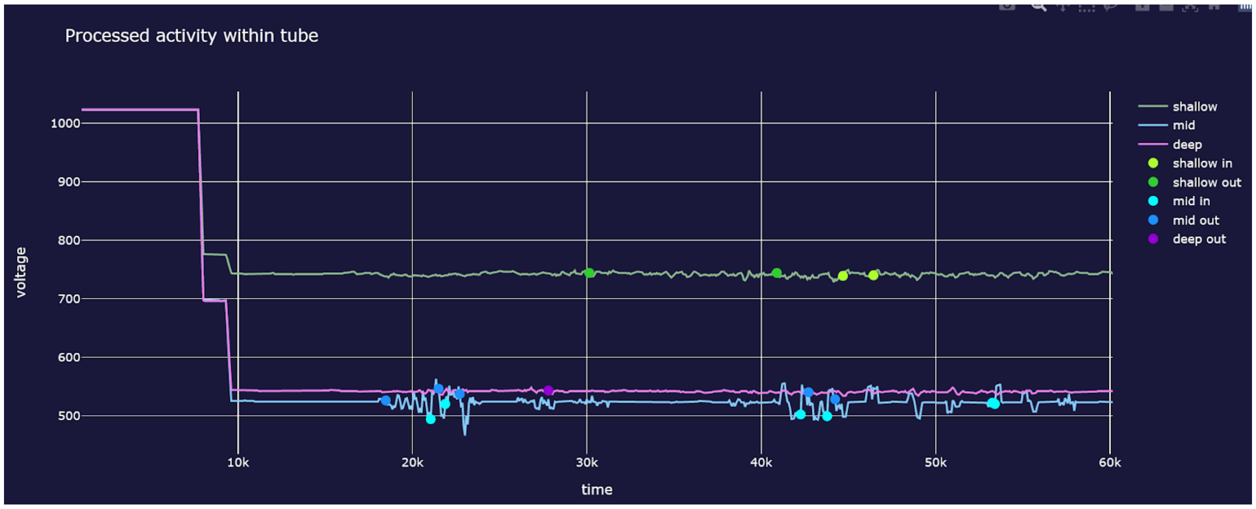


Figure 11: Output of display script showing data collected and initial analysis

3 Verification

As our subsystems grew to be quite electrically complex, prototyping and verification consumed a large portion of the semester's work.

We went down our data pipeline in order (as outlined in Figure 3), and built the system from the most core component, the AC bridge measuring the capacitance inputs, to the outermost component, the user-facing data visualizing script. This modular design allowed us to take measured, methodical steps towards completion, and allowed us to build our system incrementally. It also allowed us to roll back to a "working version" quickly as we could undo our progress to the last known functional state in an easy and reliable manner.

One of the early verifications we had trouble with was the AC bridge functionality. Originally we planned to create an instrumentation amplifier out of op-amps to save on cost, since they were readily available in the lab. However, we found that our original attempt at simply replacing the instrumentation amplifier with an op-amp did not reproduce the same functionality. Upon further research, we used a connection of three opamps and various resistors to replicate the behavior of an instrumentation amplifier, but found that mismatches in the resistor behavior caused irregular behavior in the AC bridge output. That is, it would not respond the same way to the same input at different times. In order to meet the requirements necessary for this subsystem, we ordered an AD620 instrumentation amplifier that used matched resistors.

The AD620 instrumentation amplifier's functionality was verified through breadboard testing, as pictured in Figure 13. Here, we have a prototype of our capacitance sensor on a short segment of tunnel functioning as the two capacitors in the AC Bridge. We were able to confirm that passing a simulated bee through the tunnel creates a difference in the waveforms generated by the two resistor capacitor pairs that make up the AC Bridge arms. These are the green and blue waveforms on the lower portion of the oscilloscope. Then, when these waveforms are fed into the AD620 instrumentation amplifier, the amplifier's output is a very small amplitude waveform when the sensor is undisturbed. The amplitude of the output waveform increased when a simulated bee caused an imbalance in the AC bridge. The results of our testing are shown in Table 1.

Our tunnel verification was simpler: we soldered the connecting wires onto the ring ourselves, and so tested the continuity from the ring to the end of the wire to confirm that the connection was solid and strong.

We then moved onto testing the demodulator. The demodulator's main function is to extract the amplitude changes in the waveform outputted by the instrumentation amplifier; in this way, it is able to take the waveform data and produce a cleaner signal that contains the same data.

This works because the output of the instrumentation amplifier is the sum of a modulated DC signal containing the change in capacitance and large amounts of noise. The demodulator extracts this modulated DC signal from the noise and demodulates it, resulting in the desired signal. We confirmed this behavior by connecting the instrumentation amplifer's output to the demodulator input, and could see that negative and positive spikes could be seen on the oscilloscope when a simulated bee entered or exited the tunnel, as seen in Figure 12 .

The demodulator also requires low pass filters at its output to reduce noise; without these filters, the signal response we were seeking was difficult to identify among the noise. At this point, our capacitor measurement circuitry was complete.

This was an eventful stage of testing, as many key observations were made at this time. Firstly, we noticed that the sensor is sensitive to changes in the speed of the bee moving through the tunnel. Faster movements generate taller, sharper spikes while slower movements generate smaller, more rounded spikes, as seen in Figure 8. At a bee's natural pace of movement, our circuit outputs a gentle response to the activity, which is more difficult to detect as the response can be lost in noise.

We hypothesize that a larger supplied input voltage, different materials for our machined sensor rings, or better insulation around the sensors would lead to a more sensitive system that could better reject noise and highlight bee movement; however, we were unable to investigate these ideas this semester due to time and material constraints. Nevertheless, we found that fast movements created easily identifiable spikes in the signal. In order to compare different bead size movements, we set a testing standard: in comparisons, all bead sizes are moved at top speed with maximum force by the same person, which keeps as many variables constant as possible. With this standard in place, we confirmed that a 3 mm bead representing a parasite generates smaller signal spikes when passing through the tunnel than when a 6 mm bead representing a mason bee moves through the tunnel.

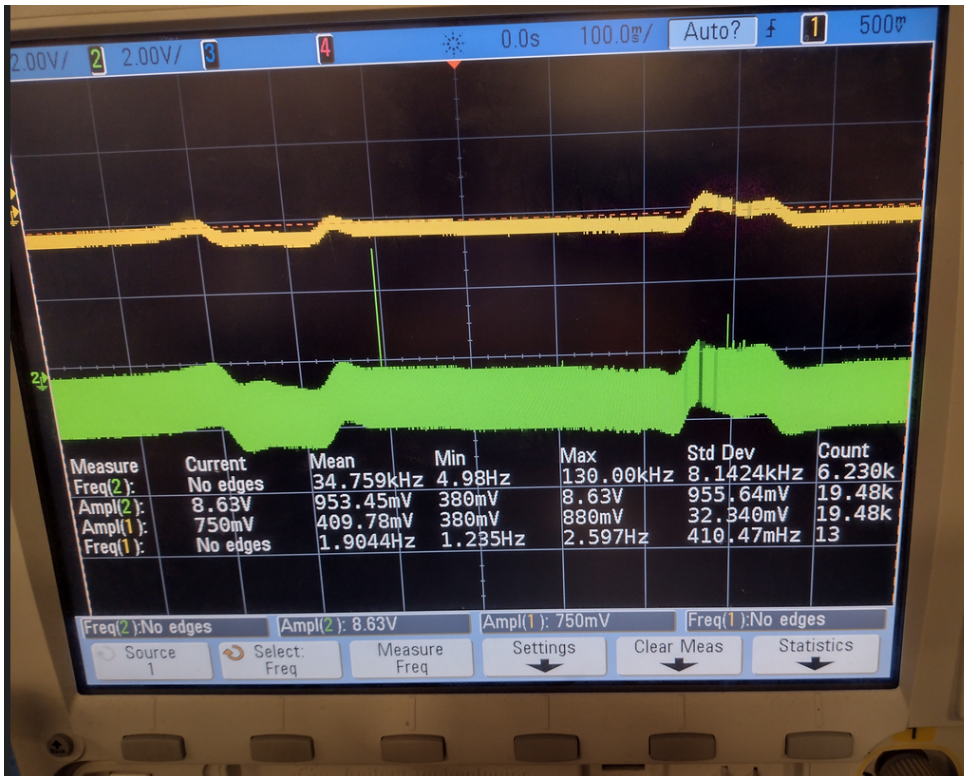
Following the demodulator verification, we worked on generating the required waveforms from the MAX038 waveform generator chips. The hurdle here was interpreting equations from the datasheet correctly to select resistors and capacitors that are used with the chip to set the frequency and shape of the output. Once we performed the calculations and went through some trial-and-error in determining the required resistance and capacitance values, we verified that the generated signal met our specifications by observing the output on an oscilloscope, as seen in Figure 9.

Confirming that our power subsystem gave us the required -5 V, GND, and +5 V was simple; we used a voltmeter to measure the voltage output.

We verified that our SD card could be reliably used to store continuously collected data by continuously collecting data for two minutes and observing how much space this took up on the card. From this we were able to calculate that a day's worth of data would take up 0.3 GB on the device. With the SD card we have selected, that means the device can continuously collect data for 107 days before running out of storage space, which is long enough to last a whole mason bee season.

Finally we wrote and tested our Python desktop-side script that displays the data collected on the SD card. From observing the visualization created by the script, we realized we needed to collect data more frequently on the ATMega328P. We decreased the delay in collecting the data and saw more continuous results on the visualizations. Our intent was to model the graphic after the oscilloscope readings we captured in out testing. Please refer to Figure 11, where ”spikes” are marked by entry and exit time markers; these spikes look like the oscilloscope readings.

These verification activities brought us to the end of performing active prototyping and hardware verification. From here, we moved onto soldering, system integration, and building the rest of the house structure and mounting.



Measure	Current	Mean	Min	Max	Std Dev	Count
Freq(2):	No edges	34.759kHz	4.98Hz	130.00kHz	8.1424kHz	6.230k
Ampl(2):	8.63V	953.45mV	380mV	8.63V	955.64mV	19.48k
Ampl(1):	750mV	409.78mV	380mV	880mV	32.340mV	19.48k
Freq(1):	No edges	1.9044Hz	1.235Hz	2.597Hz	410.47mHz	13

Figure 12: Output of the instrumentation amplifier (green) and the balanced demodulator (yellow).

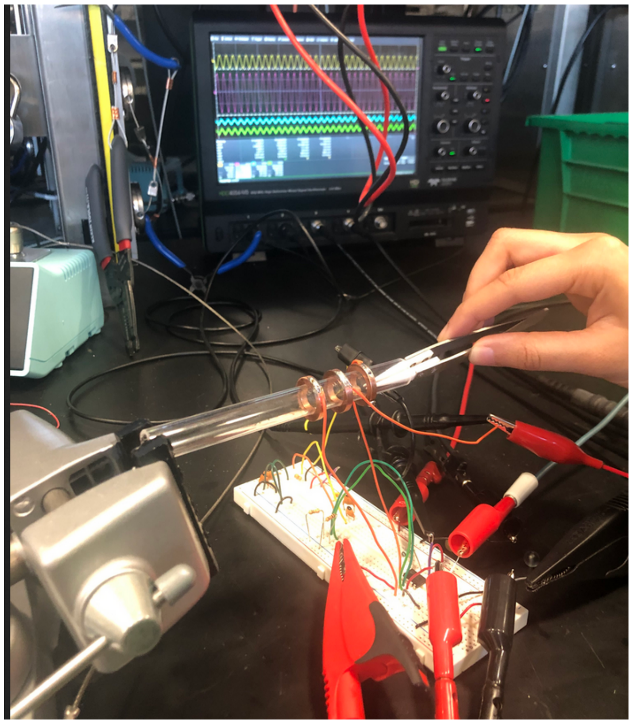


Figure 13: First time test of the instrumentation amplifier working together with the tunnel and sensor to produce a change in amplitude in response to bee movement.

Table 1: Table showing verification results of instrumentation amplifier circuit when glass beads of various sizes and treatments are inserted into tube

Bead Size	**Output voltage waveform amplitude no bead in tube**	**Output waveform amplitude bead between cap rings**
8 mm gray synthetic opal	703 mV	834 mV
8 mm champagne bead	702 mV	820 mV
6 mm unbuffed synthetic opal	701 mV	754 mV
6 mm champagne bead	704 mV	760 mV
3 mm champagne bead	703 mV	725 mV

4 Costs

Table 2 shows the cost breakdown of a single unit of the bee house. This is as we have manufactured it, i.e. there are only capacitance sensors in one out of the four tunnels in the house. For the labor cost, we assume each person in the group spent a total of 60 hours prototyping, debugging, and designing, with a salary is $35 per hour. Similarly for the machine shop, we assume that they spent a total of 30 hours designing and machining our product, with a salary of $35 per hour.

Table 2: Table showing cost breakdown of project

Item	Retail Cost/ Item	Student Cost/ Item	Manufacturer	Quantity
Hour of Member Labor	$35	$0	N/A	640
Hour of Machine Shop Labor	$35	$0	N/A	40
Instrumentation Amplifier	$20.58	$20.58	Analog Devices	3
Balanced Demodulator	$39.04	$39.04	Analog Devices	3
Operational Amplifier	$1.31	$0.00	Texas Instruments	3
High-slew Operational Amplifier	$7.00	$0.00	Intersil	2
MAX 038 Waveform Generator	$5.00	$0.00	Maxim Integrated	2
MicroSD Card and Adapter	$8.54	$8.54	Sandisk	1
ATMega 328P	$5.00	$5.00	Atmel	1
Power Supply	$11.23	$11.23	Allover Power	2

In total, the parts would cost us $214.86, which, in combination with labor, brings the total to $24,022.79 for the entire bee house with only one tunnel containing capacitive sensors. However, as the student labor was unpaid and many parts and machine shop time were available as course resources, the bee house's actual costs are closer to $200 for a house with one fully functional tunnel. Please note that this base cost excludes standard parts such as wood, copper, and acrylic, and common resistors and capacitors.

5 Conclusion

In conclusion, our bee house serves as a proof-of-concept for a consumer-oriented mason bee house. In its current form, this bee house is a ready-to-use tool for research applications, as the device is able to collect data accurately and precisely. However, additional work needs to be done in order to make the bee house a viable consumer product.

Future work can be done in the following areas:

1. Software to provide a less technical, more user-friendly interface that displays the behavior of the bees in a way that can be understood by someone who has no knowledge of bees
2. Make the device weather resistant
3. Visual indicators of urgent sensor readings like parasite detection

We have also put our utmost effort into preventing accidental harm to mason bees that would eventually inhabit the device. Our literature search led us to infer that mason bees do not seem to mind the electric fields generated by the capacitor rings, as their close genetic cousin, the leafcutter bee, was studied to not mind the capacitor rings at all [4]. We have also sized the mason bee tunnels in line with best practices informed by current research on healthy mason bee populations. Research published in the journal Apidologie in 2013 [1] concluded that tunnels with a length of at least 15 cm produce the healthiest offspring and a suitable male-female larva ratio. This aligns with Section 1.2 of the ACM Code of Ethics: avoid harm [8].

This mason bee house has been designed to be as approachable to amateur beekeepers as possible, since our objective is to empower beekeepers to make the best decision possible for their bees. Even though the technical functionality can be complex, the parts directly interfacing with a user - such as an SD card, Python scripts, and data stored in .txt or .csv - are common technologies in consumer electronics. This design is in keeping with the ACM Code of Ethics Section 1.4, where it is noted that technology should be as accessible as possible [8].

The beekeeper's physical safety is also a high priority concern; we did not create developer safety procedures as our device cannot cause harm to the people who use it, even by accident; it has no moving parts or batteries, and does not presently involve real bees. We have avoided using batteries to protect the user, the bees, and the environment; by using DC power supplied from wall outlet connections, the device overall does not generate much heat or produce toxic waste. Our electronics are also all low-voltage and low current, so our device poses no risk of injury due to electrical shock.

By bringing transparency to mason bee houses, we have provided beekeepers a way to stay vigilant against today's most pressing mason bee home concerns. We hope that from our efforts ,we have created a tool to boost native pollinator populations and have contributed to the betterment of the environment, for ourselves, for our posterity, and of course, for the bees.

References

[1] Sedivy, C., Dorn, S. Towards a sustainable management of bees of the subgenus Osmia (Megachilidae; Osmia) as fruit tree pollinators. Apidologie 45, 88–105 (2014). https://doi.org/10.1007/s13592-013-0231-8

[2] Mader, E., Shepard, M., Vaughan, M., & Guisse, J. (2018, May). Tunnel Nests for Native Bees. https://xerces.org. Retrieved January 25, 2022, from https://xerces.org/sites/default/files/2018-05/13-054_02_XercesSoc_Tunnel-Nests-for-Native-Bees_web.pdf

[3] Alzaabi, O. (2019). Airborne Insect Radar Scattering Characterization Using Electromagnetic Modeling (dissertation).

[4] Alzaabi, O., Lanagan, M., Breakall, J., &amp; Urbina, J. (2019). Dielectric Properties of Honey Bee Body Tissue for Insect Tracking Applications.

[5] Campbell, J. M., Dahn, D. C., & Ryan, D. A. (2005). Capacitance-based sensor for monitoring bees passing through a tunnel. Measurement Science and Technology, 16(12), 2503–2510. https://doi.org/10.1088/0957-0233/16/12/015

[6] "Atmega 328 P Datasheet," 2015. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Accessed: 04-May-2022].

[7] "Max038 High-frequency waveform generator: Maxim integrated," High-Frequency Waveform Generator, 20-Sep-2007. [Online]. Available: https://www.maximintegrated.com/en/products/analog/clock-generation-distribution/silicon-crystal-oscillators/MAX038.html. [Accessed: 04-May-2022].

[8] "ACM Code of Ethics and Professional Conduct." ACM.org. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Feb 10, 2022].

[9] http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/opampvar5.html

Appendix A Requirements and Verifications Table

Mason Bee Housing Subsystem:

Requirement	Verification
1. Provide secure housing for tunnels and circuitry.	1A. House will be shake tested to make sure no parts are unsecured and to ensure fit of connections. 1B. A successful test will have none of the electronic components disconnect, ie. after the shake, the device is still functional.
2. Shall be weather resistant.	2A. Wooden structure of outside housing will be exposed to 100 ml of water from the top to simulate rain. 2B. Insert three glass balls into tunnels to simulate bee activity. 2C. Manually check [pull SD card out from housing, load into desktop, open file that the data has been written to] the data written to the SD Card and verify that activity has been recorded as expected. In particular, three spikes indicating bee entry and exit should be visible.

Capacitance Measurement Subsystem:

Requirement	Verification
2. Square wave and sinusoidal wave are generated from waveform generator and amplifier with characteristics at least 4.75V and no more than 5.25V amplitude and 33kHz frequency (at least 25kHz and no more than 50 kHz).	2A. Connect an oscilloscope probe to the labelled square and sinusoidal wave outputs on microcontroller PCB. Make sure to ground the probe by connecting the ground clip. 2B. Press 'default' on the oscilloscope to return it to its default setup. Zoom in on the oscilloscope vertically and horizontally to 2V vertical scale (for both waveforms) and 20 micro seconds horizontal scale. 2C. Supply +5V, -5V, and ground to the microcontroller board and wait 5 seconds. 2D.Then click measure on the oscilloscope and set up measurements for amplitude and frequency of the two waveforms. Confirm that the readings are within acceptable range.

3. Output of instrumentation amplifier (proto-bee data) is a waveform that increases in amplitude in response to a bead passing through the capacitive sensor and is sensitive to the bead's direction of motion.	3A. Set up oscilloscope probe on test point of sensor board circuit. Make sure to ground the probe's clip. 3B. Press 'default' on the oscilloscope to return it to default setup. Zoom in on the oscilloscope vertically and horizontally to 500mV vertical scale and 100ms horizontal scale. 3C. Supply the sensor board and the microcontroller board with +5V, -5V, and ground. Make sure the microcontroller board's sin output is connected to the center rings of the capacitor sensor and that the microcontroller board's square output is connected to the demodulator's reference (lower right header on sensor board). Wait five seconds. 3D. Insert a bead into the tunnel, moving it in and out, making sure to pass the capacitive sensor whose circuit it being measured. Watch the oscilloscope and confirm that when a bead is not inserted, the waveform is consistent, and when it is inserted, the waveform changes, and that when the bead is removed, the waveform changes in the opposite way.
4. Sensor board output follows reaction of instrumentation amplifier test point to bead entry and exit, but waveform has less amplitude (no more than 0.25 V) and clearly shows the entry and exit pattern of increase and decrease corresponding to direction of bead travel.	4A. Set up test described above in section 3. 4B. Add a probe on the output of the sensor board (top pin on the bottom right header of the sensor board), making sure to ground the probe's reference clip. 4C. Turn on the channel for the additional probe and set vertical zoom to 1V per division. 4D. Insert and remove the bead, and visually confirm that the sensor board output follows the instrumentation amplifier signal's envelope. Confirm the entry and exit pattern of increase and decrease corresponding to direction of bead travel, noting that a lower velocity of travel will make a pulse with less amplitude but longer duration. A higher velocity of travel will make a pulse with higher amplitude and shorter duration.

Analysis and Storage Subsystem:

Requirement	Verification
1. SD card contains readings from capacitance sensor system in CSV format.	1A. Verify that the outputs of the 3 sensor boards (the top pin on the bottom right header of the sensor board) are connected to A0, A1, A2 on the ATMEGA in the correct order (topmost sensor board to A0, etc.) 1B. Verify that the microcontroller board sin wave output is connected to capacitive sensor center rings and the microcontroller board square wave output is connected to the demodulator reference (bottom pin on the bottom right header of the sensor board). 1C. Supply +5V, -5V, and ground to the boards. 1D. Simulate bee behavior in the sensor tunnel by inserting a bead, making sure to pass through all capacitive sensors that are involved in the test. Do this for approximately half a minute. 1E. Stop supplying power. Remove the SD card and view its contents on a computer. Confirm that there exists a file (probably titled sen3.txt) that contains four columns of data and many rows, of which the first column increases each row and the other three columns reflect readings of the sensors.
2. SD card will be able to contain the data resulting from continuous readings of 1 day.	2A. Simulate 2 minutes of bee behavior. 2B. Note the amount continuous data generated from the 2 minutes by inspecting the size of the text file. Multiply this size by 720. (24 hours/ 2 minutes = 720) 2C. Confirm that the SD card is at least this size by checking its reported storage capacity on a computer.
3.Computer-side program (display.py) should interpret csv data in a way that does not require technical expertise on the part of the user.	3A. Run display.py after installing the necessary libraries (pandas and plotly). 3B. Visually confirm that the bead motion is identifiable on the generated picture.

Power Subsystem:

Requirement	Verification
1. Power subsystem should be able to supply voltage at +5V(from 4.8 to 5.8), ground, and -5V (from -4.8 to -5.8)	1A. Take the two variable power supplies and connect the breadboard output by attaching it to the barrel. Unscrew the terminals and use a wire to connect the positive terminal of one power supply to the negative terminal of the other. Attach a wire to the two other terminals. 1B. Use an oscilloscope to probe each terminal. Confirm that the positively marked, unconnected terminal is within the correct range for +5V, the joined terminal is at ground, and the negative, unconnected terminal is within the correct range for -5V.
2. Current draw of electronics should not exceed 1.5A.	2A. Connect all the electronics as described in Test 1 of the Analysis and Storage Subsystem. 2B.Supply power using the power supply in lab. Use the +25v, com, and -25v outputs, turn on the output, and set them to +5V and -5V. 2C. Conduct the test described in Test 1 of the Analysis and Storage Subsystem, but monitor the current draw displayed on the power supply to confirm it does not exceed 1.5 Amps.

Appendix B   Python Display Code

```
# this script displays data from the bee tube in a time vs voltage
# graph. To use this you will need to install plotly and pandas
# pip install plotly
# pip install pandas
# the script does not currently deal with timestamp overflow, because
# millis starts at zero when the arduino is powered and does not roll over
# until apporx 50 days have passed, which is like a whole bee season
# the graph will display on your default browser
# the script also clears the file in which the arduino writes data, and make
# copy of the data it cleared in a file named with the data and time

import csv
import plotly.express as px
import shutil
import datetime
import numpy as np

fontcolor='#ffffff'

# lists to store data from the text file
time=[]
sen1=[]
sen2=[]
sen3=[]

avg_time=[]
avg_sen1=[]
avg_sen2=[]
avg_sen3=[]

# populate lists with data from csv file
# make sure data is being read as numbers and not strings

# use below line for use with device
with open("D:\SEN3.TXT") as File:

# use below line for use during debugging
# with open("SEN3v2.TXT") as File:
  beedata = csv.reader(File, delimiter = ',')

  for row in beedata:
    time.append(int(row[0]))
    sen1.append(int(row[1]))
    sen2.append(int(row[2]))
    sen3.append(int(row[3]))

# make picture of continuous data
'''
fig = px.line(title='Raw measured activity within tube')
fig.add_scatter(x=time, y=sen1, name='shallow')
fig.add_scatter(x=time, y=sen2, name='mid')
fig.add_scatter(x=time, y=sen3, name='deep')

fig.update_layout(
    xaxis_title="time",
    yaxis_title="voltage",
    )
'''
# do analysis to identify interesting bee behavior patterns

# average out the data
# uses slightly overlapping averaging windows for a smoother graph
for i in range(1,(len(time)-3),1):

    avg_time.append(round((time[i]+time[i+1]+time[i+2]+time[i+3])/4))
    avg_sen1.append(round((sen1[i]+sen1[i+1]+sen1[i+2]+sen1[i+3])/4))
    avg_sen2.append(round((sen2[i]+sen2[i+1]+sen2[i+2]+sen2[i+3])/4))
    avg_sen3.append(round((sen3[i]+sen3[i+1]+sen3[i+2]+sen3[i+3])/4))

fig1 = px.line(title='Processed activity within tube')

fig1.add_scatter(x=avg_time, y=avg_sen1, name='shallow', line=dict(color='Da
fig1.add_scatter(x=avg_time, y=avg_sen2, name='mid', line=dict(color='Light
fig1.add_scatter(x=avg_time, y=avg_sen3, name='deep', line=dict(color='Viol

fig1.update_layout(
    xaxis_title="time",
    yaxis_title="voltage",
    plot_bgcolor='#19183a',
    paper_bgcolor='#19183a',
    title_font_color=fontcolor,
    legend_font_color=fontcolor,
    legend_grouptitlefont_color=fontcolor,
    legend_title_font_color=fontcolor,

    )

fig1.update_xaxes(color=fontcolor)
```

```
fig1.update_yaxes(color=fontcolor)

# find interesting things in the data

# find "normal" to be able to make conclusions about abnormal
med1=np.median(avg_sen1[75:])
med2=np.median(avg_sen2[75:])
med3=np.median(avg_sen3[75:])

# put the reference lines on the graph
#fig1.add_hline(ref1)
#fig1.add_hline(med2)
#fig1.add_hline(med3)

# mark potential bee crossing points, stage 1: cross the median
in1=[]
out1=[]
in2=[]
out2=[]
in3=[]
out3=[]

sensitivity = 2 #how much of a deviation from normal is significant (in same

for i in range(1,(len(avg_time)-10)):

    #identify potential bee crossings on shallow data
    #ins
    if ((avg_sen1[i]<med1) and (avg_sen1[i+1]>med1)):
        if ((sum(avg_sen1[i-5:i])/5)< med1-sensitivity and ( sum(avg_sen1[i
            in1.append((avg_time[i], avg_sen1[i]))
    #outs
    if ((avg_sen1[i]>med1) and (avg_sen1[i+1]<med1)):
        if ((sum(avg_sen1[i-5:i])/5)> med1+sensitivity and ( sum(avg_sen1[i
            out1.append((avg_time[i], avg_sen1[i]))

    #identify potential bee crossings on mid data
    #ins
    if ((avg_sen2[i]<med2) and (avg_sen2[i+1]>med2)):
        if ((sum(avg_sen2[i-5:i])/5)< med2-sensitivity and ( sum(avg_sen2[i
            in2.append((avg_time[i], avg_sen2[i]))
    #outs
    if ((avg_sen2[i]>med2) and (avg_sen2[i+1]<med2)):
        if ((sum(avg_sen2[i-5:i])/5)> med2+sensitivity and ( sum(avg_sen2[i
                out2.append((avg_time[i], avg_sen2[i]))
    #identify potential bee crossings on deep data
    #ins
    if ((avg_sen3[i]<med3) and (avg_sen3[i+1]>med3)):
        if ((sum(avg_sen3[i-5:i])/5)< med3-sensitivity and ( sum(avg_sen3[i
            in3.append((avg_time[i], avg_sen3[i]))
    #outs
    if ((avg_sen3[i]>med3) and (avg_sen3[i+1]<med3)):
        if ((sum(avg_sen3[i-5:i])/5)> med3+sensitivity and ( sum(avg_sen3[i
                out3.append((avg_time[i], avg_sen3[i]))


#if any entry or exits are found, label them on the graph
if (in1):
    fig1.add_scatter(x=list(zip(*in1))[0], y=list(zip(*in1))[1],
    mode='markers', name='shallow in', marker=dict(size=10, color='GreenYel
if (out1):
    fig1.add_scatter(x=list(zip(*out1))[0], y=list(zip(*out1))[1],
    mode='markers', name='shallow out', marker=dict(size=10, color='LimeGre
if (in2):
    fig1.add_scatter(x=list(zip(*in2))[0], y=list(zip(*in2))[1],
    mode='markers', name='mid in', marker=dict(size=10, color='Aqua'))
if (out2):
    fig1.add_scatter(x=list(zip(*out2))[0], y=list(zip(*out2))[1],
    mode='markers', name='mid out', marker=dict(size=10, color='DodgerBlue'
if (in3):
    fig1.add_scatter(x=list(zip(*in3))[0], y=list(zip(*in3))[1],
    mode='markers', name='deep in', marker=dict(size=10, color='Fuchsia'))
if (out3):
    fig1.add_scatter(x=list(zip(*out3))[0], y=list(zip(*out3))[1],
    mode='markers', name='deep out', marker=dict(size=10, color='DarkViolet

# add pictures of reference value for visual confirmation
fig1.show()



# uncomment the lower code when using it on the device
# make a correctly dated copy of data

now = str(datetime.datetime.now())[5:19]
now = now.replace(":","_")

src="D:\SEN3.TXT"
dst="D:data"+str(now)+".txt"
```

```
shutil.copyfile(src,dst)

#clear the original file to make space for next reading
with open("D:\SEN3.TXT", ’w’):
    pass
```

Appendix C ATMega328P Data Logging Code

```
#include <SPI.h>
#include <SD.h>

File myFile;

int chipsel = 4;

//variables to read data from tubes
int val1, val2, val3;
unsigned long timestamp;

int sen1 = A0;
int sen2 = A1;
int sen3 = A2;

void setup() {
  // initalize ss pin , 10
  pinMode(chipsel, OUTPUT);

  //set up input pins for 3 capacitor pairs
  pinMode(sen1, INPUT);
  pinMode(sen2, INPUT);
  pinMode(sen3, INPUT);

  digitalWrite(chipsel, HIGH);

  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);

  Serial.begin(9600);

  if (!SD.begin(chipsel)) {
      while(1);
      Serial.println("not initialized SD");
  }
}

void loop() {
  //debugging message
  Serial.print("within loop ");

  //take reading from each capacitor pair
  val1 = analogRead(sen1);
  val2 = analogRead(sen2);
  val3 = analogRead(sen3);

  //open file for writing
  File dataFile = SD.open("sen3.txt", FILE_WRITE);

  //debugging message to see measured values
  Serial.print(val1);
  Serial.print(",");
  Serial.print(val2);
  Serial.print(",");
  Serial.println(val3);

 //write measured values to file in csv format
  if (dataFile) {
    timestamp = millis();
    dataFile.print(timestamp);
    dataFile.print(",");
    dataFile.print(val1);
    dataFile.print(",");
    dataFile.print(val2);
    dataFile.print(",");
    dataFile.println(val3);
    dataFile.close();
    Serial.println("w");
  }

  //short delay to allow SD card library to finish its tasks
  //50ms was selected based on the oscilloscope readings we took
  //it should be frequent enough to take more than one reading during the u
  //making it recognizable in the readings from the csv
  delay (25);

}
```