ISAT 300

Spring Semester 2025, Section 1

Lab Instructor: Dr. Chris Bachman

Experiment #4: Ultrasonic Sensors

Date: February 26th, 2025

Authors: Joseph Gros, Alex Chizmadia, Sebastian Erb, Jaylon Taylor

Project Leader: Sebastian Erb

Honor Pledge: On our honor, we (Joseph Gros, Alex Chizmadia, Sebastian Erb, Jaylon Taylor) have neither given nor received unauthorized help on this assignment

Abstract:

Ultrasonic sensors are widely used in various technological applications, particularly in autonomous vehicle systems, where precise distance measurements are critical for navigation, obstacle avoidance, and overall safety. This lab was designed to investigate the performance, accuracy, and practical applications of the HC-SR04 ultrasonic sensor when integrated with a Raspberry Pi microcontroller, with a specific focus on its ability to measure distances, assess precision, and calculate velocity and acceleration based on detected motion. The experimental setup involved wiring the ultrasonic sensor to the Raspberry Pi, followed by the implementation of Python-based code to trigger the sensor, record data, and analyze results. Calibration tests were conducted to determine whether the sensor performed within the manufacturer's specifications, with repeated trials conducted at various fixed distances to establish consistency and accuracy in measurements. Additionally, data was analyzed to identify the sensor's minimum and maximum effective range, evaluate potential sources of uncertainty, and implement a secondary feature in which an LED warning system was activated when an object approached a specific threshold distance. The sensor's response time and data acquisition rate were also analyzed by modifying the code to determine the fastest possible sampling rate that still maintained reliable results, allowing for an evaluation of how efficiently the sensor could track rapid movement. Furthermore, motion-based trials were conducted where an object was moved at different speeds in front of the sensor, enabling calculations of velocity and acceleration using time-stamped distance readings, which provided insights into the sensor's capabilities for real-time motion tracking. The experimental results demonstrated that the ultrasonic sensor was able to produce reasonably accurate distance measurements within its expected operational range, with deviations observed at extreme distances that suggested limitations in either the sensor's precision or environmental factors affecting wave reflection. The data collected also indicated that while the sensor was effective for basic velocity and acceleration calculations, its inherent latency and sensitivity to environmental conditions, such as surface reflectivity and angle of incidence, introduced variability in motion-tracking accuracy, making it less suitable for high-speed applications requiring extremely precise measurements. Despite these limitations, the findings from this lab reinforce the practical applications of ultrasonic sensors in low-risk environments where moderate accuracy is sufficient, while also highlighting the necessity of careful calibration, statistical analysis of uncertainties, and consideration of hardware limitations when implementing such sensors in more advanced systems, such as robotics and autonomous navigation technologies.

Introduction:

The emergence of the Raspberry Pi has significantly impacted educational tools and hobbyist projects, offering an affordable and accessible platform for bridging technology and creativity. However, this lab places a particular emphasis on ultrasonic sensors and their application in real-world scenarios, especially in the context of autonomous vehicles. Ultrasonic sensors, renowned for their accuracy in distance measurement, play a pivotal role in autonomous systems, where their ability to detect objects and measure distances with precision is critical for navigation, obstacle avoidance, and safety features. This lab aims to explore the integration of ultrasonic sensors with the Raspberry Pi to measure distances and then apply these measurements to control external systems or devices, mimicking the functionality of sensors used in autonomous vehicles. [1]

Autonomous vehicles rely heavily on sensors such as ultrasonic devices to perceive their environment and make real-time decisions, including determining the proximity of obstacles, detecting parking spaces, and navigating tight spaces. The use of ultrasonic sensors allows these vehicles to measure distances accurately, enabling safe navigation in various conditions. By utilizing the Raspberry Pi as an interface, this lab demonstrates how students can engage with real-world technologies that are essential to autonomous systems. Through Python programming, participants can program the Raspberry Pi to process sensor inputs, such as triggering an alarm or activating a braking system when an object is detected too close, which mirrors functions in autonomous systems.[2]

In the context of autonomous vehicles, ultrasonic sensors offer a cost-effective solution for close-range object detection and are often used in tandem with other sensor technologies, such as cameras and LiDAR, to create a comprehensive perception system. This lab not only enhances students' understanding of ultrasonic sensors and Python programming but also provides insight into the practical applications of these sensors in modern technology. Students will learn how to integrate sensors with physical systems, analyze data, and apply it to control devices, bridging theoretical knowledge with practical, real-world applications in robotics and autonomous vehicle systems. The project's focus on the measurement precision and reliability of ultrasonic sensors reflects their critical role in ensuring the safety and effectiveness of autonomous systems, where accurate distance sensing is vital for vehicle navigation and interaction with the environment.

Methods and Materials:

The purpose of this lab was to gain familiarity with operating an ultrasonic sensor with a computer for data acquisition and processing, then determine if the sensor was within the manufacturer's specifications. With the advancement of technology in the 21st century, ultrasonic sensors have become a commonly used device in a wide array of applications with continued hopes of advances, highlighted in autonomous vehicles, climate control systems, industrial robotics, and much more.. In this experiment, a HC-SR04 5V ultrasonic sensor was wired to a Raspberry Pi 4B via a voltage divider constructed on an electronics breadboard with additional wires, one 1000 Ω resistor, and one 1500 Ω resistor. Then, python codes were inputted that instructed the Raspberry Pi and sensor to perform certain commands under specific conditions. Calibration tests were done prior to any experimentation to ensure precise and accurate recordings before any legitimate experiments were done. During experimentation, using an meter stick and a large enough object (like a box or white board), the maximum and minimum recordings were taken along with their mean and standard deviations from Microsoft Excel. From there, the uncertainty of the sensor was calculated with recordings of 100+ individual measurements, and the fastest speed of the sensor by manipulating various factors like the timing in the code and the object's distance from the sensor while considering the unchangeable factors like the speed of the processor and memory card. Then, a LED circuit that illuminated and alerted the system was installed to signal to the operator when an object got too close to the sensor. Finally, the velocity and acceleration were determined between two distances at two different times.

Ultrasonic Sensor and Raspberry Pi Setup

The ultrasonic sensor was electronically connected to the Raspberry Pi through a voltage divider. This was necessary because the Raspberry Pi operates at a maximum voltage of 3.3V, while the ultrasonic sensor requires 5V for accurate measurements. To bridge this voltage difference, the sensor was powered by the 5V supply from the Raspberry Pi's GPIO, and a voltage divider was built on a breadboard using additional wires and resistors. This setup split the sensor's output signal between the Raspberry Pi and the ground of the ultrasonic sensor, ensuring the voltage remained below 3.3V. The inputs and outputs of the ultrasonic sensor are illustrated in **Image 1: Ultrasonic Sensor Inputs/Outputs**, while the complete sensor and Raspberry Pi setup is depicted in **Image 2: Ultrasonic Sensor and Raspberry Pi Circuit Diagram**, both sourced from the digital laboratory manual [3]. According to these images, the ultrasonic sensor's +5V input was linked to the Raspberry Pi's 5V pin, the 'Trigger' pin was connected to 'GPIO 21 SCLK (SPI 1),' and the 'Echo' pin was split—leading to 'GPIO 20 MISO (SPI 1)' and 'Ground'—through a voltage divider consisting of a 1000 Ω resistor and a 1500 Ω resistor.

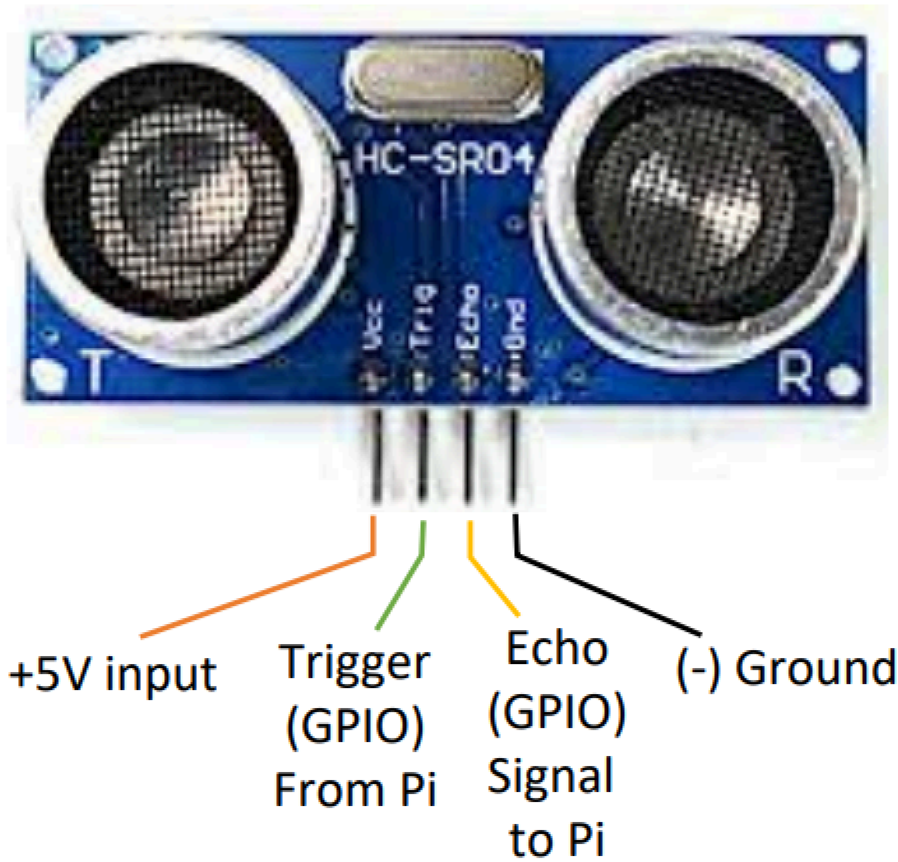


Image 1: Ultrasonic Sensor Inputs/Outputs

Image 1 *provides a clear visualization of the ultrasonic sensor's inputs and outputs, facilitating easier identification of its connections with the voltage divider and the Raspberry Pi.*

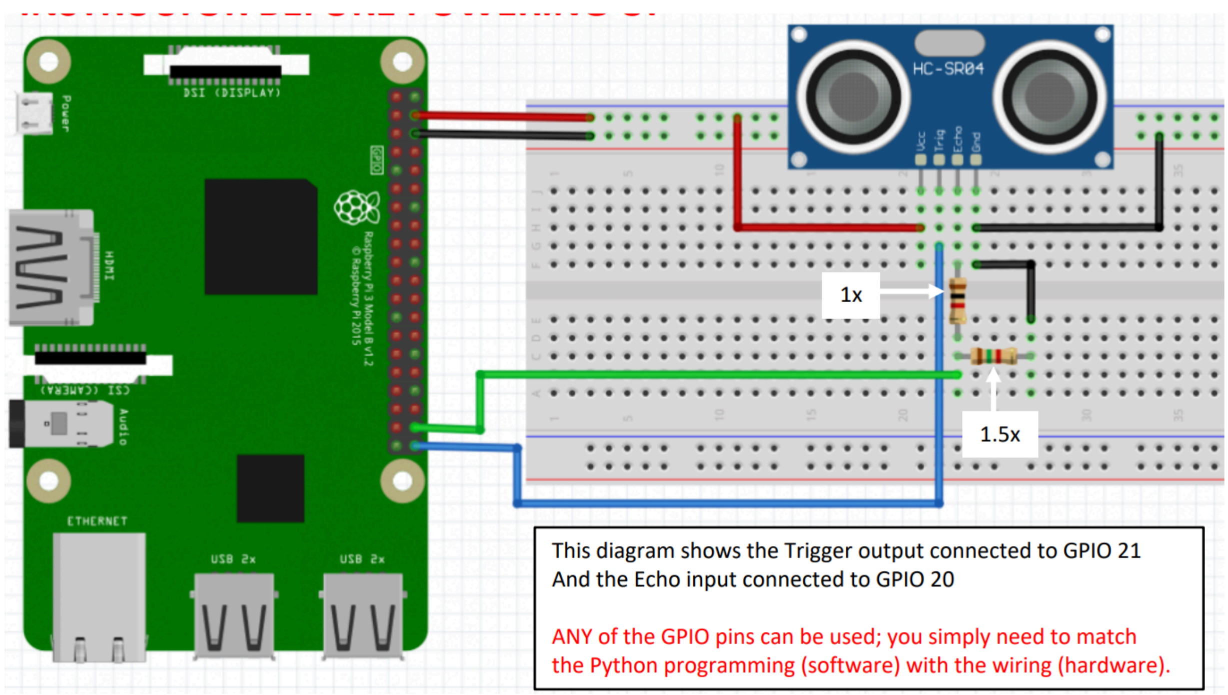


Image 2: Ultrasonic Sensor and Raspberry Pi Circuit Diagram

Image 2 *shows a circuit diagram of the final setup for the ultrasonic sensor connected to the voltage divider to the Raspberry Pi should look like.*

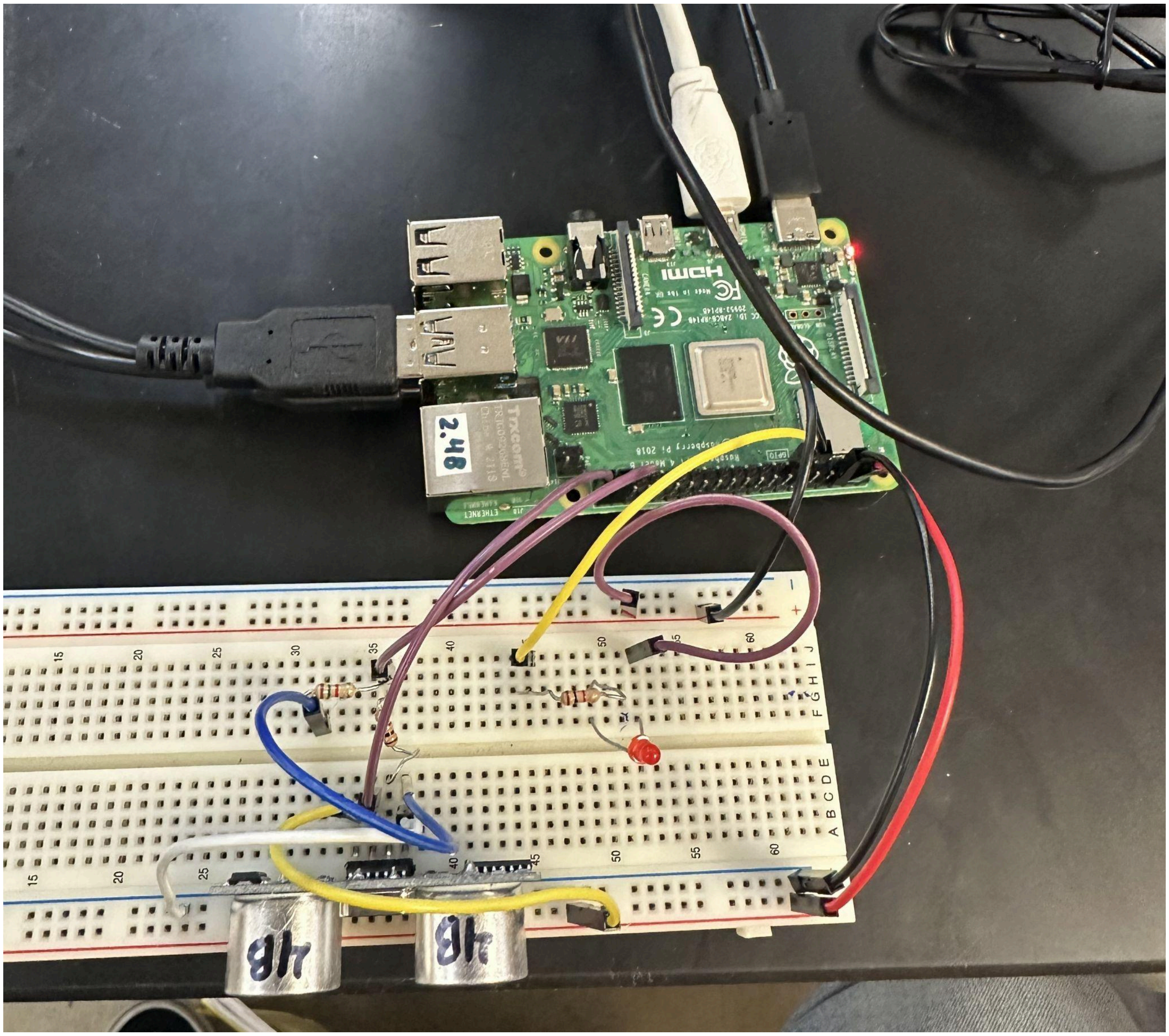


Image 3: Final Setup of Ultrasonic Sensor

Image 3 *showcases the final setup after assembling the ultrasonic sensor, voltage divider, and Raspberry Pi, clearly reflecting the design outlined in Image 2.*

Next, the cables were connected to the power outlet, computer monitor, and Raspberry Pi keyboard to input codes for operating the ultrasonic sensor.

Python Coding Setup

After assembling the hardware and logging into the Raspberry Pi, Python code was entered into the Thonny

Python Integrated Development Environment (IDE). The initial step involved defining the GPIO input and output pins. **Image 4: Python Codes to Establish GPIO Input/Output Pins** displays a screenshot of the first 14 lines of code from Step 1: Setting the GPIO input/output pins, as outlined in the digital laboratory manual [3]. Lines 2-3 instruct the Raspberry Pi to utilize GPIO, while Line 6 designates Broadcom Chip numbering for the GPIO. Lines 9-10 assign pin 21 to the Trigger and pin 20 to the Echo, while Lines 13-14 specify that the Trigger functions as an output from the Raspberry Pi and the Echo as an input to it.

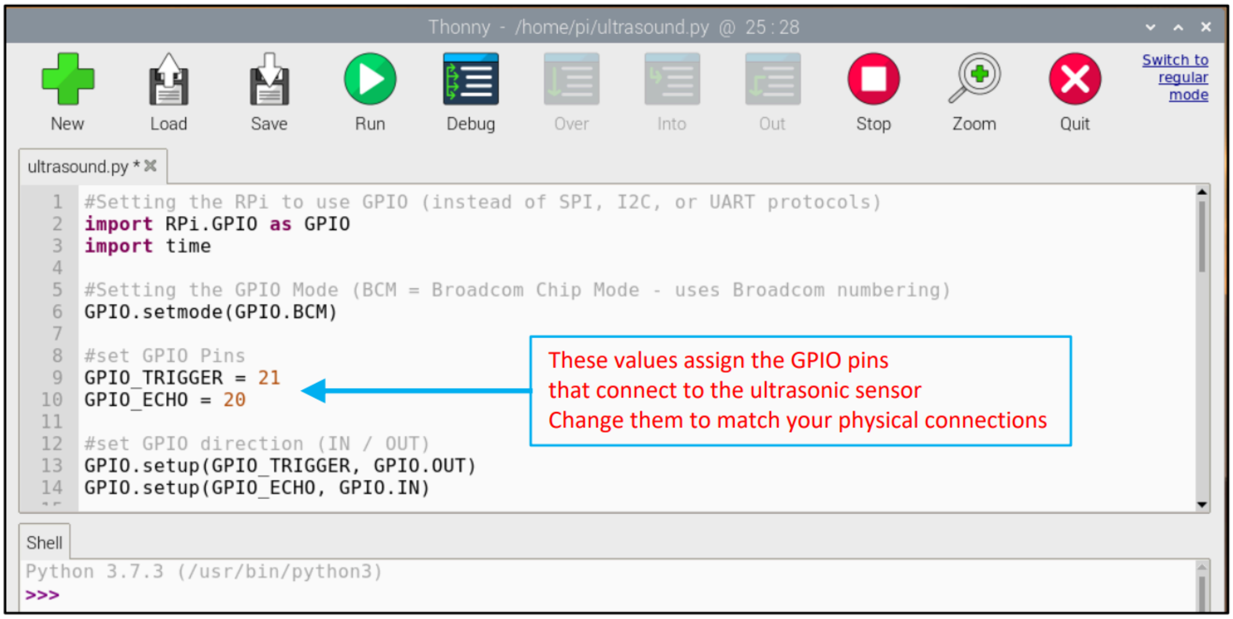


Image 4: Python Codes to Establish GPIO Input/Output Pins

Image 4 displays a screenshot from the digital laboratory manual, showcasing the first 14 lines of code designed to define the GPIO input and output pins.

The next step was to activate the ultrasonic sensor to take readings**. Image 5: Python Code to Activate the Ultrasonic Sensor** displays a screenshot of the next 25 lines of code (lines 15-40) from Step 2: Activating the Ultrasonic Sensor in the digital laboratory manual [3]. Line 15 defines the function named "distance." Line 17 sets GPIO_TRIGGER to "True," sending a voltage to the sensor to emit ultrasonic waves for measurement. Line 20 allows signaling for 0.01 milliseconds (0.00001 seconds), followed by Line 21, which sets GPIO_TRIGGER to "False," stopping the voltage and halting wave transmission. Lines 23-24 assign the StartTime and StopTime using the Raspberry Pi's internal clock. Lines 27-28 contain a while loop that records StartTime when no signal is detected from the Echo pin, while Lines 31-32 include another while loop that captures StopTime when a signal is received. Line 35 calculates TimeElapsed as the duration between the ultrasonic signal transmission and the returning Echo. Finally, Line 38 determines the object's distance from the sensor by multiplying TimeElapsed by 34,326 (the speed of sound in cm/s) and dividing by 2, accounting for the signal's round-trip journey.

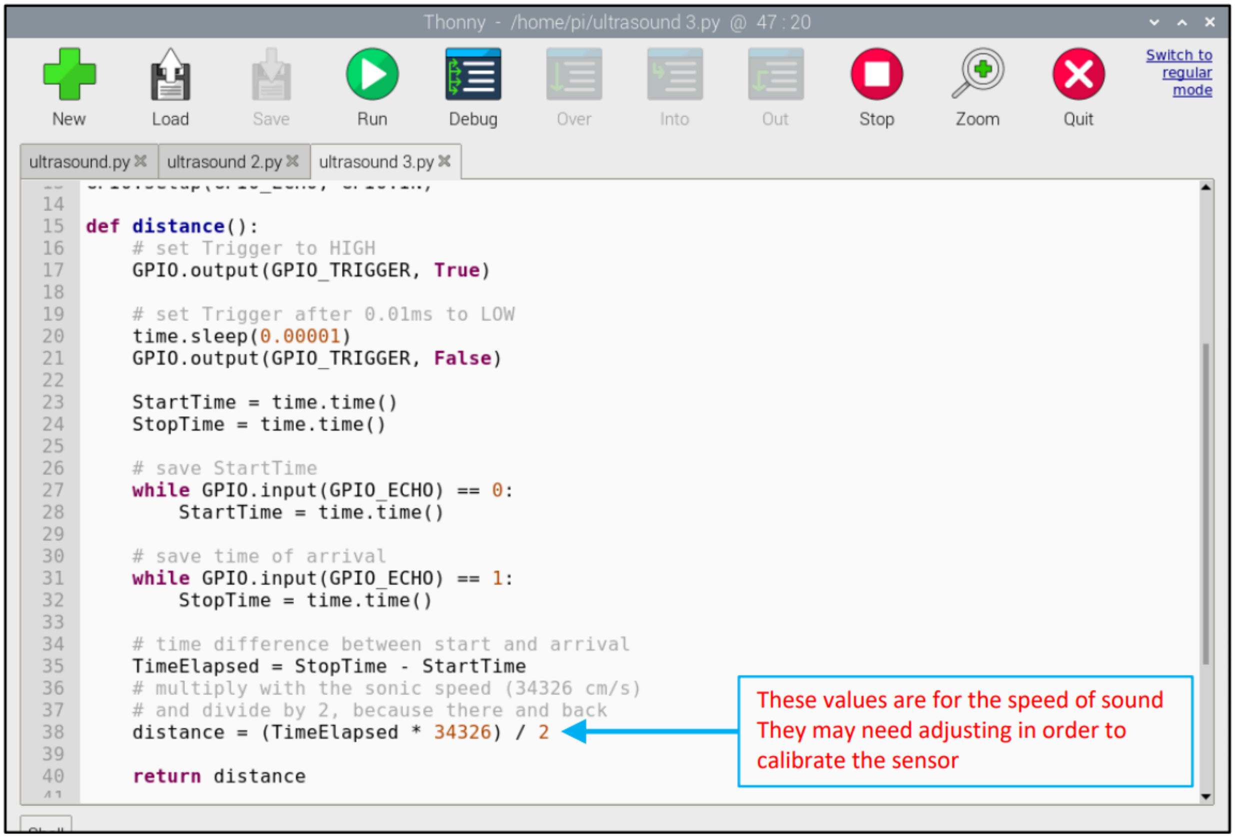


Image 5: Python Code to Activate the Ultrasonic Sensor

Image 5 *displays a screenshot of the next 25 lines of code (lines 15-40) from Step 2: Activating the Ultrasonic Sensor, as outlined in the digital laboratory manual.*

In the third step, code was inputted to allow the user to view the measurements while they were being recorded. **Image 6: Python Code to Print Measured Distances** shows the next three lines of code (lines 42-45) from Step 3: Printing the Measured Distance on the screen, as presented in the digital laboratory manual [3]. Line 42 initiates a for loop to collect a sample of 10 data points, which can be adjusted for a different sample size later. Line 43 assigns the variable "dist" to represent the distance, Line 44 prints the distance on the screen with "%.lf" later replaced by "%dist," and Line 45 ends the measurement.

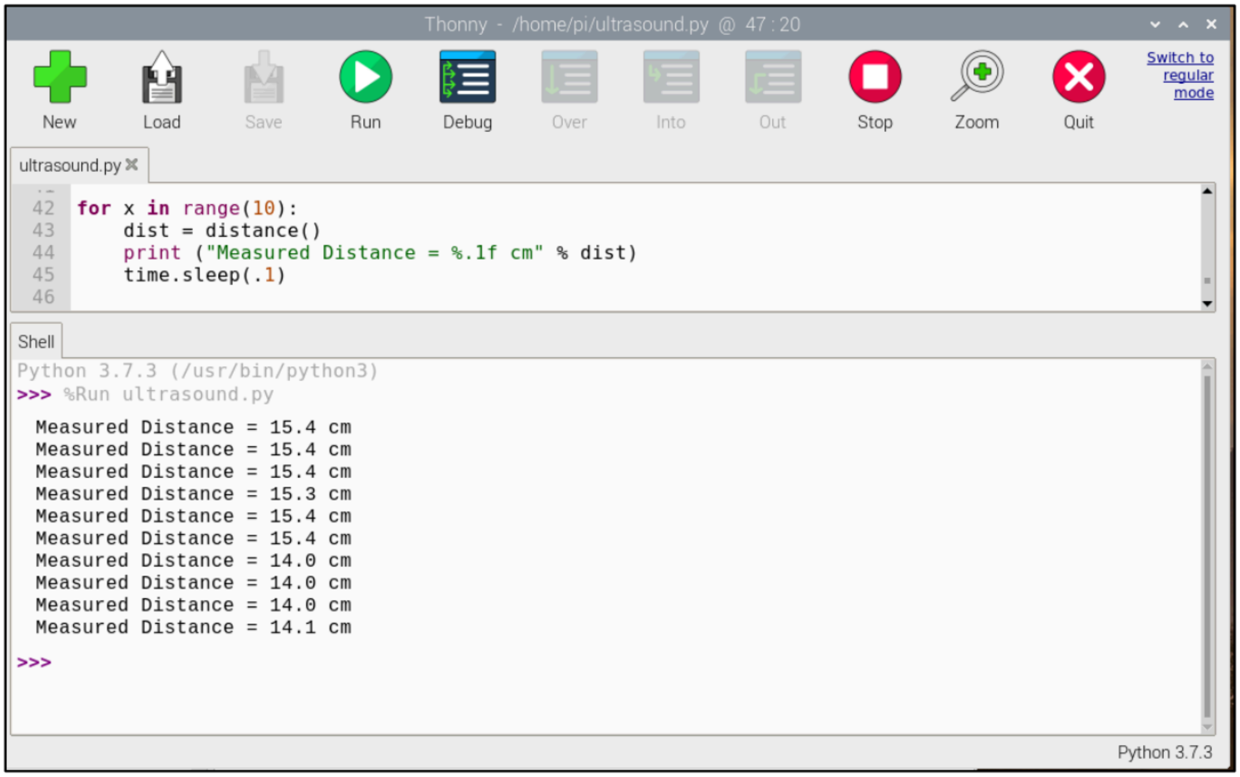


Image 6: Python Code to Print Measured Distances

Image 6 displays a screenshot of lines 42 to 45 from Step 3: Printing the Measured Distance on the screen, as outlined in the digital laboratory manual.

The final lines of code are used to write the data into a .CSV file. **Image 7: Python Code to Write Data to .CSV File** presents a screenshot of lines 42 to 47 from Step 4: Writing the Data to a .CSV file, as shown in the digital laboratory manual [3]. Line 42 initiates a for loop for a sample size of 100 data points. Line 43 defines "dist" to represent the distance. Line 44 prints the distance measurement on the screen. Line 45 creates a .csv file named "ultrasounddata.csv" (which can be renamed for better identification), with the "a" indicating that the file should append data rather than overwrite or restart data collection. Line 46 specifies how the measurements should be recorded in the .csv file, such as in centimeters, inches, or feet. Line 47 halts the measurement. During experimentation, the units will need to be adjusted as necessary.

```
42 for x in range(100):
43     dist = distance()
44     print ("Measured Distance = %.1f cm" % dist)
45     f = open("utrasounddata.csv", "a")
46     f.write('\n'+"Actual Distance =, 10cm, Measured Distance, %.2f" % dist)
47     time.sleep(0.01)
```

Image 7: Python Codes to Write Data to .CSV File

***Image** 7 displays a screenshot of the updated set of codes from lines 42 to 47 from Step 4: Writing the Data to a .CSV File.*

After setting up the software for the hardware, it was time to do the measurement and calculators, the first step being essential which is to establish the starting point for the measurements as 0 cm. After discussion, it was decided to set the starting point for all measurements at the edge of the sensor, as seen in **Image 8: Starting Point of Measurements**

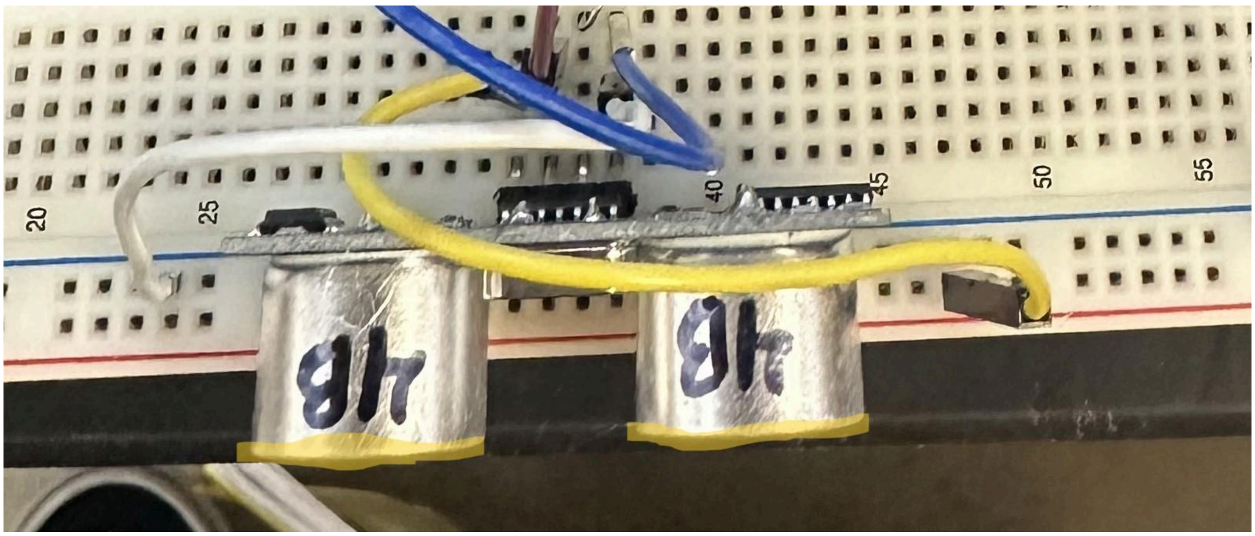


Image 8: Starting Points for Measurements

***Image 8** represents the starting point of measurements, 0cm, indicated by the highlighted yellow market at the end of the sensor.*

After setting a reference point, calibration tests were conducted to ensure the ultrasonic sensor provided accurate and precise readings. To assess the calibration, objects were placed at fixed distances from the sensor to create a baseline reference, and the sensor was activated. These distances included 20 cm, 50 cm, 100 cm, 200 cm, 300 cm, and 350 cm. Calibration runs were performed three times for each distance to ensure consistency. Following calibration, the maximum recordable distance of the ultrasonic sensor was tested by gradually increasing the distance of the object and measuring it with a meter stick. A large object, such as a whiteboard, was used to ensure the ultrasonic waves could reflect back to the sensor. After each run, the data was saved into a designated file. Once the maximum distance was determined, three trial runs were conducted to demonstrate consistent readings. The data was then transferred to a Microsoft Excel spreadsheet, where the mean and standard deviation of each data set were calculated. Next, the minimum recordable distance was assessed. Similar to the maximum distance, the distance was gradually reduced until the sensor could no longer provide accurate readings or produce errors. Measurements started at 20 cm and were shortened by 2-4 cm until the sensor could no longer generate precise readings. This phase also required three trial runs with consistent minimal distances, followed by calculations of the mean and standard deviation in Microsoft Excel.

The next experiment involved connecting a small LED to the Raspberry Pi and the already constructed circuit. **Image 9: LED Circuit Diagram** presents a screenshot from the digital laboratory manual [4], showing how the LED was to be attached to the preexisting circuit. It was important to understand that the LED could only operate in one direction: the shorter prong indicated negative (-), while the longer prong indicated positive (+). The negative terminal of the LED was wired to the common ground, while the positive terminal was connected to GPIO 17 on the 3.3V row, with a 1000 Ω resistor.

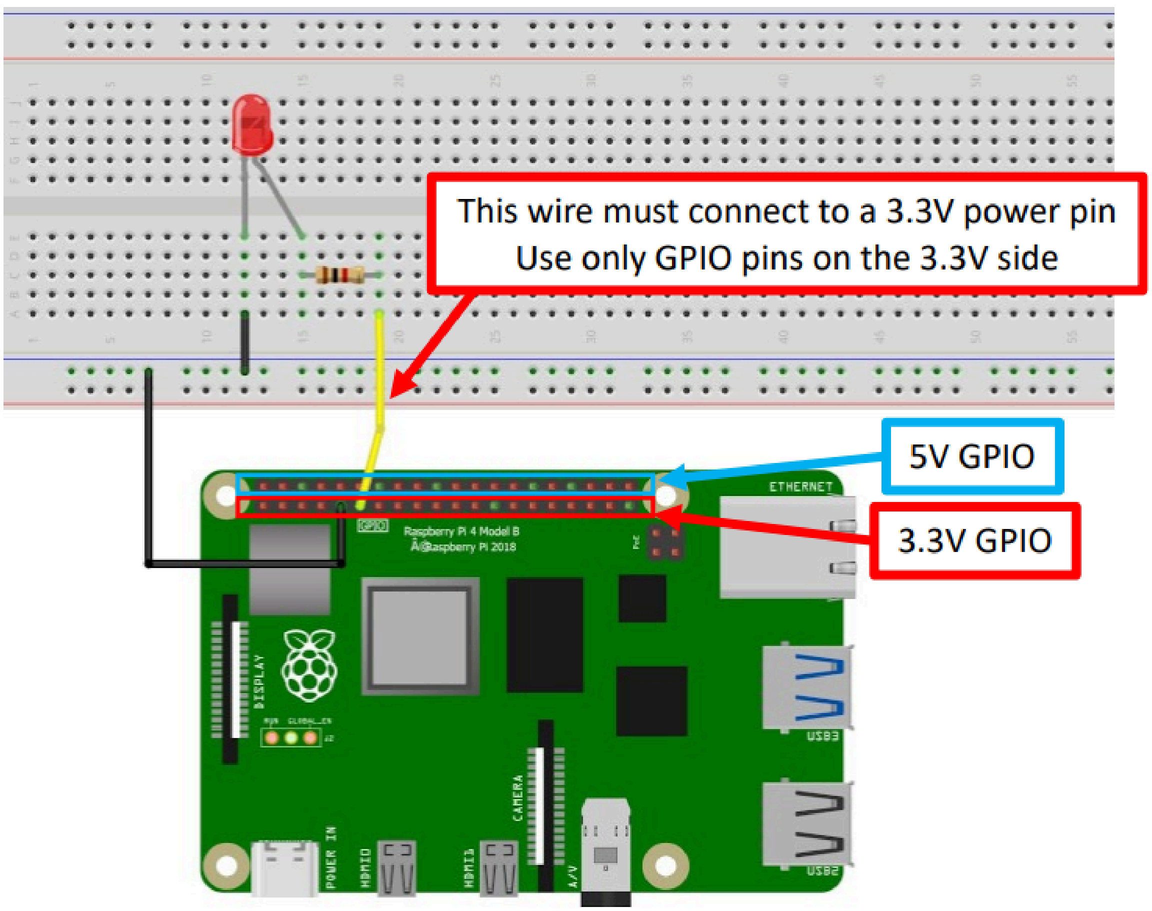


Image 9: LED Circuit Diagram

Image 9 *displays a screenshot from the digital laboratory manual, showing how the LED was to be connected to the preexisting circuit.*

Additional lines of code were then added to the existing Python script. **Image 10: Python Codes for Alert System** shows a screenshot from the digital laboratory manual, highlighting the newly added code [4]. These additions enabled the sensor to trigger the ultrasonic sensor and illuminate the LED when an object comes too close, which is a common feature in modern technology. The first line of code assigns GPIO 17 to the LED pin, in line with the wiring setup. The second line ensures the LED remains off when not activated. The rest of the code consists of an if-else statement, where the target distance is defined first. The following lines activate the LED and display the "WARNING!!!!" message on the software. The else statement turns the LED off, returning it to its initial state.

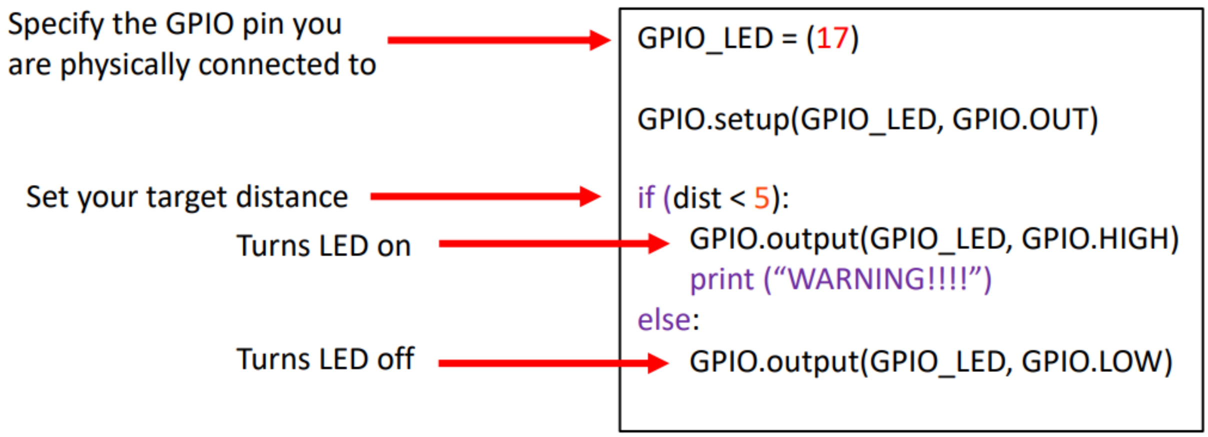


Image 10: Python Codes for Alert System

Image 10 *displays the collection of Python code that was added to the preexisting lines.*

The final experiment involved modifying the existing Python code or manually calculating the velocity of the object based on the ultrasonic sensor's measurements between two successive readings. The following formula was crucial for calculating velocity:

$$Velocity\ (m/s) = \frac{\Delta Distance}{\Delta Time} = \frac{(Distance\ 2 - Distance\ 1)}{(Time\ 2 - Time\ 1)}.$$

The data collected from evaluating the ultrasonic sensor's fastest processing speed (when the object was moved while the sensor was collecting measurements, as described in the "Uncertainty and Fastest Processing Speed" section) was used to calculate the velocity. Following this, the formula below was applied to calculate the object's acceleration during the same experiment:

$$Acceleration\ (m/s^2) = \frac{\Delta Velocity}{\Delta Time}$$

Results:

The purpose of this lab exercise is to provide hands-on experience in controlling sensors using a computer and utilizing the computer for data acquisition, processing, and control of physical systems. In this lab, an ultrasonic sensor is used to measure distance, controlled by a Raspberry Pi computer interfaced through the General Purpose Input and Output (GPIO) pins. The lab consisted of two 50 sampling rates which equates to 100 trials. The data collected is intended to investigate the optimal sampling rate, minimum distance, maximum distance, velocity, and acceleration using the ultrasonic sensor, which is stated to detect distances of up to 400 cm.

The optimal sampling rate came out to be in the range of 50. This is due to the sensor being able to collect data at a slower but more accurate rate compared to the given 100 in the lab. This could be also due to the processor of the raspberry pi as a better processor would make it run more smoothly giving more accurate data points over time.

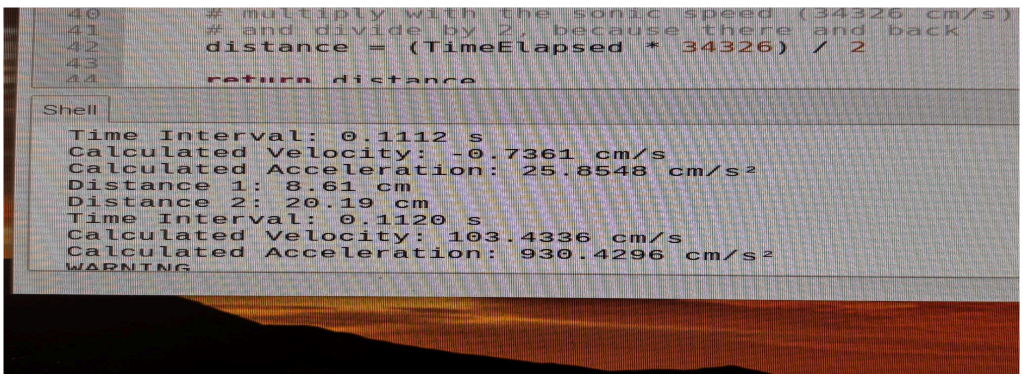


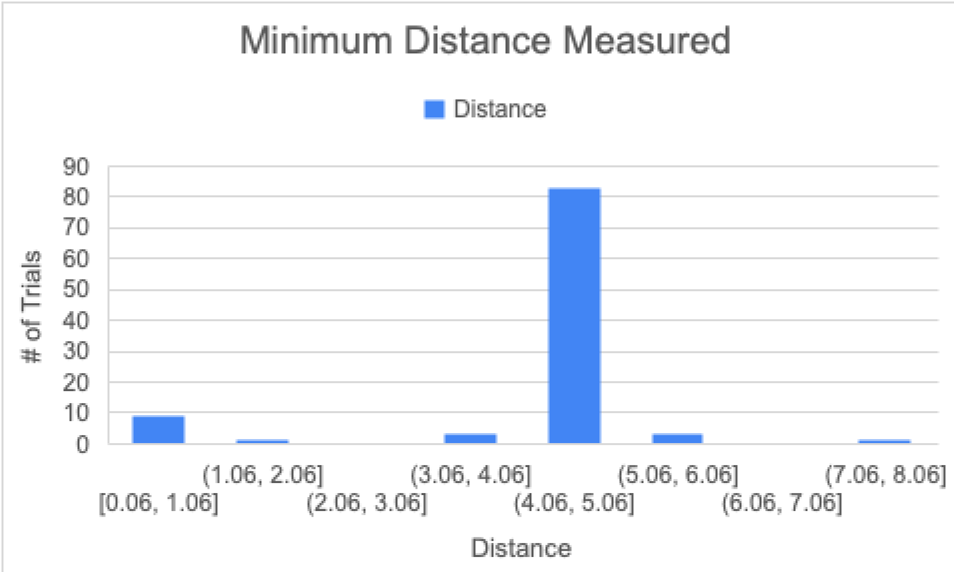
Image 11: Shows the calculated velocity and calculated acceleration of an object being dropped on the floor.

The minimum distance measured by the ultrasonic sensor came out to be 0.06cm during the two 50 sample rates in trials. This was done in 100 trials as the range for the data was from 0.06cm to 7.17cm. The uncertainty of the measured minimum distance for one standard deviation or 68% confidence interval is between 3.09cm to 5.75cm.

Minimum Distance

Minimum	**Maximum**	**Mean**	**STD**
0.06cm	7.17cm	4.42cm	1.33cm

Table 1: Table describes the min, max, mean, and standard deviation of the ultrasonic sensor measuring the **minimum distance**.



Graph 1: This graph shows over a 100 trials how the distance mostly stays around 4–5 cm range. There are multiple outliers where the distance drops to nearly 0 cm, indicating sudden reductions and measurements in the 7cm - 8cm range. The outliers suggest fluctuating measurements, possibly due to an object moving back and forth or sensor inconsistencies.

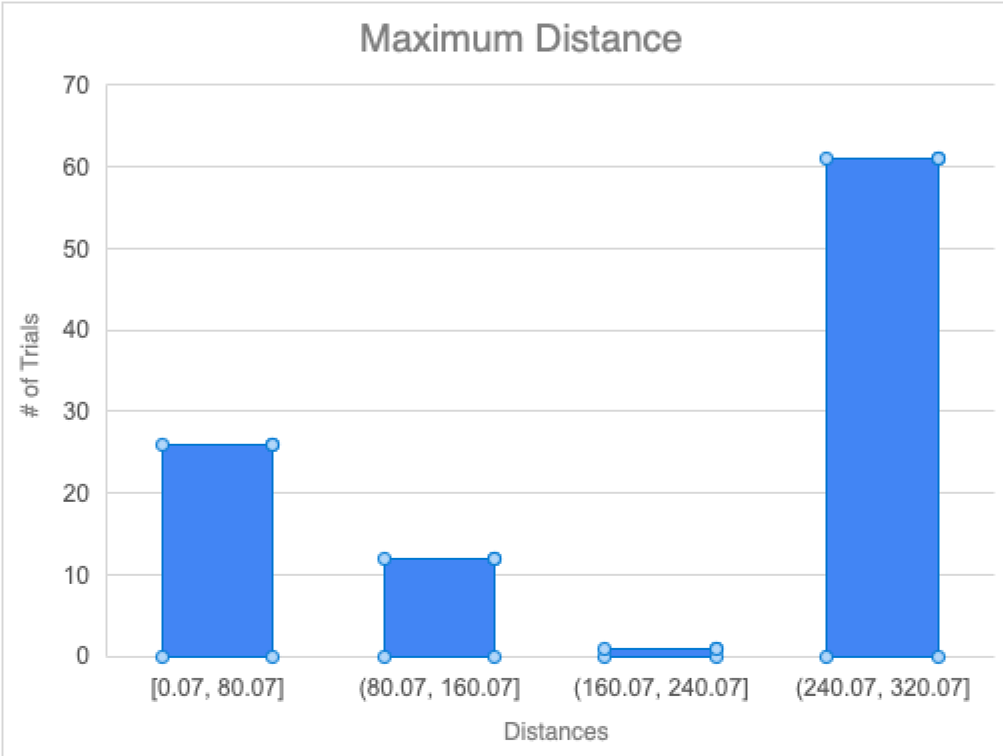
The maximum distance measured by the ultrasonic sensor came out to be 302.03 cm during the two 50 sample rates in trials. This was done in 100 trials as the range for the data was from 0.07cm to 302.03 . The uncertainty of the measured maximum distance for one standard deviation or 68% confidence interval is between 82.2cm to 294.9cm.

Maximum Distance

Minimum	**Maximum**	**Mean**	**STD**
.07cm	302.03cm	188.5cm	106.3cm

Table 2: Showing the min, max, mean, and STD for the maximum distance measured using the ultrasonic sensor.

The minimum distance measured by the ultrasonic sensor came out to be 0.06cm during the two 50 sample rates in trials. This was done in 100 trials as the range for the data was from 0.06cm to 7.17cm. The uncertainty of the measured minimum distance for one standard deviation or 68% confidence interval is between 3.09cm to 5.75cm.



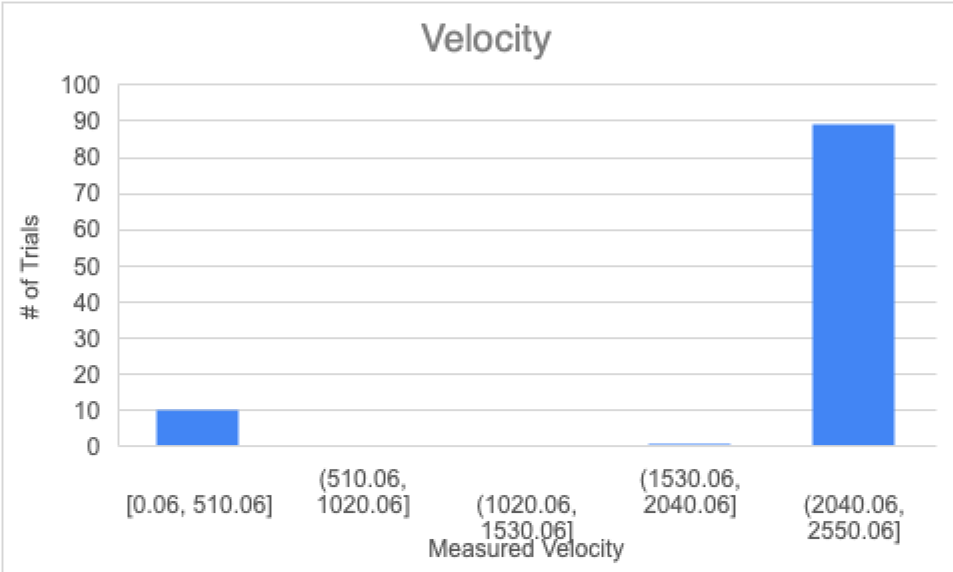
Graph 2: This graph shows over a 100 trials how the maximum distance was mostly in the 240.07-320.07 range. The average was 188cm even though the trials itself doesn't have many data points in that range indicating there were many outliers in the measurements. This can be due to the sensor and the calibration of the ultrasonic sensor.

The mean velocity measured by the ultrasonic sensor came out to be 202 m/s during the two 50 sample rates in trials. This was done in 100 trials as the range for the data was from 0.06m/s to 225 m/s. The uncertainty of the measured velocity for one standard deviation or 68% confidence interval is between 134.5m/s to 269.5m/s.

Measured Velocity

Mean	**Maximum**	**Minimum**	**STD**
202m/s	225m/s	0.06m/s	67.5m/s

Table 3: Showing the mean, max, min, and STD of the measured velocity using the ultrasonic sensor.



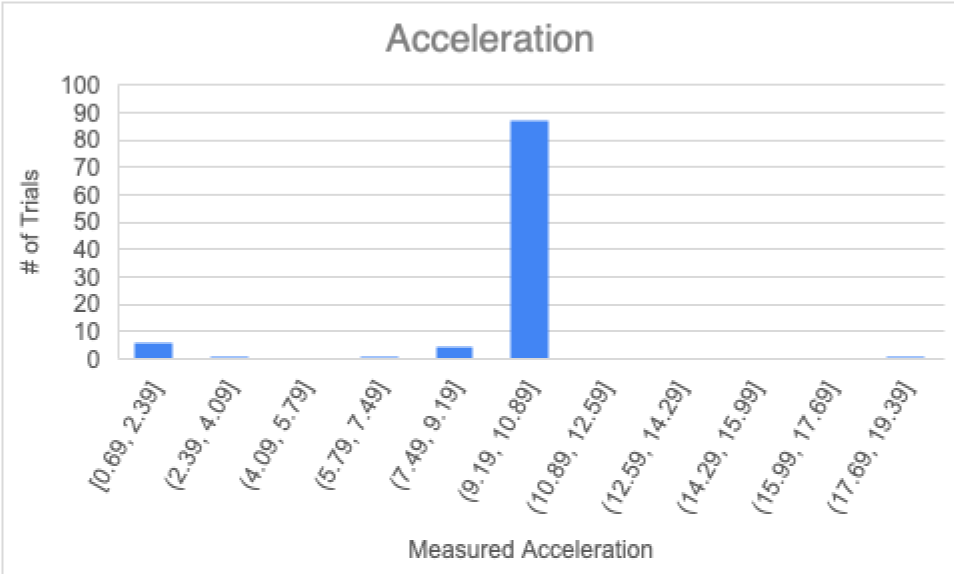
Graph 3: This graph shows over a 100 trials how the velocity measured was mostly in the 240.1-255.1 range. The average was 202m/s which validates the graph as most of the data points were in that range. There were many data points outside that range which were outliers. This can be due to the sensor not working properly for a few of the measurements and the calibration of the ultrasonic sensor being off.

The mean acceleration measured by the ultrasonic sensor came out to be 202 m/s during the two 50 sample rates in trials. This was done in 100 trials as the range for the data was from 0.69m/s^2 to 17.94m/s^2. The uncertainty of the measured acceleration for one standard deviation or 68% confidence interval is between 6.61m/s^2 to 11.15m/s^2.

Measured Acceleration

Mean	**Maximum**	**Minimum**	**STD**
8.88m/s^2	17.94m/s^2	0.69m/s^2	2.27m/s^2

Table 4: Showing the mean, max, min, and STD of the measured acceleration using the ultrasonic sensor.



Graph 4: This graph shows over a 100 trials how the acceleration measured was mostly in the 9.2m/s^2- 10.9m/s^2 range. This makes sense as the acceleration on Earth equates to 9.81m/s^2. The average was 8.88m/s^2 which validates the graph as most of the data points were near that range set. There were many data points outside that range which were outliers. This can be due to the sensor not working properly for a few of the measurements and the calibration of the ultrasonic sensor being off.

Two-Point Calibration

The ultrasonic sensor system measuring acceleration and velocity was calibrated using two reference points: 9.37 m/s^2 at 203 m/s and 7.7 m/s^2 at 224 m/s, with a true acceleration of 9.81 m/s^2. Measured values deviated by 0.44 m/s^2 and 2.11 m/s^2, respectively, indicating a scaling error in distance readings or calibration error in the ultrasonic sensor. This makes sense as the data wasn't giving both the acceleration and velocity at the same time during the measurements which can lead to inaccurate data. A correction factor of 1.16 was applied to raw distances, improving acceleration outputs toward 9.81 m/s^2. The calibration addresses systematic underestimation, though velocity-dependent errors.

Discussion:

The purpose of this lab was to familiarize oneself with the operation of an ultrasonic sensor for data acquisition and processing, as well as to determine if the sensor operated within the manufacturer's specifications. The primary goals were to perform calibration tests, evaluate the maximum and minimum measurable distances of the HCSR04

5V ultrasonic sensor, calculate the uncertainty, and assess its accuracy through the movement of an object. Additionally, the sensor was re-programmed to measure velocity and acceleration, which required adjustments to the software. This process involved multiple measurements to minimize errors and ensure the reliability of the acquired data. The average of these trials provided a linear relationship between the measured distance and the actual distance, demonstrating the sensor's capability to provide calibrated and accurate readings. The minimal uncertainty observed is crucial for applications requiring precise distance measurements, such as robotics or autonomous vehicles, where even small errors could lead to significant issues.

When calculating the uncertainty for the minimum distance, it was found to be approximately 0.11 cm. This value is a result of repeated measurements and reflects the sensor's reliability at short distances. For the maximum measurable distance, the sensor demonstrated an uncertainty of 298.58 cm, determined through similar methods. The maximum distance achieved in the lab was 2241.61 cm, well within the manufacturer's specification of a 0.9 cm range. This uncertainty for maximum distance emphasizes the sensor's limitations, particularly in long-range detection. While larger sensors may be necessary for applications such as satellite-based missile tracking, this ultrasonic sensor performs adequately for low-risk applications like basic robotics or home automation.

The sensor was also tested to determine the maximum sampling rate at which it could reliably capture data. The maximum sample rate achieved was 100 samples per second, with an associated uncertainty of 10 samples per second. This rate was determined by modifying the Python code to increase the frequency at which data was collected. The uncertainty in the sample rate indicates potential limitations of the hardware and software, as the Raspberry Pi may not process data fast enough at higher sampling rates. In real-world applications, this may impact the ability to track fast-moving objects, such as in autonomous vehicles or high-speed industrial robotics. However, this sample rate is sufficient for many low-risk applications like fitness trackers or simple proximity sensors for smart device

A major goal of this lab was to modify the software to measure velocity and acceleration in addition to distance. To measure velocity, the sensor was programmed to take two distance measurements at different timestamps and calculate the difference in distance divided by the time interval, using the formula:

$$Velocity\ (m/s) = \Delta distance/\Delta time = (Distance\ 2 - Distance\ 1)\ /\ (Time\ 2 – Time\ 1)$$

This method successfully calculated velocity by determining the rate of change in position over time, providing useful data for tracking moving objects. Similarly, acceleration was determined by calculating the change in velocity over time using the formula:

$$Acceleration = \Delta Velocity \Delta Time \backslash \{Acceleration\} = \{\{\Delta Velocity\}\{\Delta \{Time\}\} Acceleration = \Delta Time \Delta Velocity$$

In this case, velocity readings were calculated at regular intervals, and the differences in velocity were used to calculate acceleration, providing insight into the sensor's ability to track changes in motion.

To ensure the accuracy of the velocity and acceleration measurements, controlled experiments were conducted with a moving object. The object was moved a known distance from the sensor, and the velocity and acceleration were hand-calculated based on the sensor's readings. Despite some human error in the moving object experiment, which likely led to inaccuracies in distance measurement, the calculations yielded velocity and acceleration values of 103.43 cm/s and 930 cm/s^2, respectively. These calculations provided a baseline for evaluating the sensor’s capabilities in tracking moving objects.

The primary source of error in the moving object experiment likely stemmed from inconsistent object movement, as human error in manually moving the object could have introduced discrepancies. Additionally, the sensor may have lost its calibration during the experiment, further contributing to the inaccuracies. As a result, the velocity and acceleration calculations are not fully reliable and would require further refinement, particularly in more precise applications where high accuracy is necessary.

These results have real-world implications in several fields. For instance, in autonomous vehicles, precise velocity and acceleration measurements are critical for collision avoidance and navigation, and a high degree of accuracy is needed. While the ultrasonic sensor is not suitable for high-stakes applications such as these, its use in lower-risk applications like fitness trackers, home automation devices, and basic robotics is more appropriate. Sensors like this are ideal for tracking objects nearby, with minimal risk of catastrophic failure in the event of a miscalculation. However, sensors with greater accuracy and reliability would be required for more critical applications, such as missile tracking or industrial robots.

The ability to measure velocity and acceleration expands the potential applications of the ultrasonic sensor, particularly in scenarios where object movement is involved. However, the uncertainty in both distance and movement measurements suggests that this sensor is not a suitable choice for high-speed tracking or environments requiring pinpoint accuracy.

Conclusion:

The results of this experiment demonstrate that the ultrasonic sensor effectively measures distance, velocity, and acceleration, but with notable limitations in accuracy and precision. The minimum and maximum distances measured fell within the expected range, though inconsistencies and outliers suggest potential sensor calibration issues and environmental factors affecting performance. The measured velocity and acceleration values were reasonably accurate when compared to theoretical expectations, but some discrepancies highlight the limitations of using this sensor for high-precision motion tracking. The data further supports the conclusion that while the sensor is suitable for general proximity sensing and basic motion tracking, it may not be ideal for applications requiring highly accurate distance or speed measurements.

These findings largely align with the initial hypothesis that the ultrasonic sensor would provide reliable data within the manufacturer's specified range, though with some degree of uncertainty. The observed variability in measurements, particularly at extreme distances, suggests that while the sensor is generally accurate, its reliability diminishes in certain conditions. The calculation of uncertainty and standard deviation reinforces the importance of repeated trials and statistical analysis in evaluating sensor performance. Furthermore, the experiment confirms that software modifications can enable the sensor to measure velocity and acceleration effectively, but real-world applications may require more sophisticated calibration techniques to reduce errors.

This lab emphasizes the significance of understanding sensor limitations, calibration, and error analysis in data acquisition. The experiment highlights the fundamental trade-offs between measurement accuracy, precision, and environmental influences, demonstrating the necessity of statistical tools to quantify uncertainty. These concepts are critical in fields such as robotics, automation, and scientific research, where sensor reliability directly impacts decision-making and system performance. [5] While the ultrasonic sensor proved useful for basic measurements, its limitations emphasize the importance of selecting appropriate sensors for specific applications and implementing rigorous calibration procedures to ensure data integrity.

References:

[1] Raspberry Pi. (n.d.). https://www.raspberrypi.com/ date accessed 3/5/25

[2]Yeong J, Velasco-Hernandez G, Barry J, Walsh J. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. Sensors (Basel). 2021 Mar 18;21(6):2140. doi: 10.3390/s21062140. PMID: 33803889; PMCID: PMC8003231.

[3]. Bachmann, C., & Rudmin, J. (n.d.). *Measuring Distance with an Ultrasonic Sensor*. College of Integrated Science and Engineering James Madison University.

[4]. Bachmann, C., & Rudmin, J. (n.d.). *Measuring Distance with an Ultrasonic Sensor Physical Computing*. College of Integrated Science and Engineering James Madison University.

[5] Tripathy, D., R. Gottumukkala, and D. Kim. "Exploring the Nexus Between Sensor Reliability and System Performance: A Comprehensive Analysis". Annual Conference of the PHM Society, vol. 16, no. 1, Nov. 2024, doi:10.36001/phmconf.2024.v16i1.3888. Date accessed: Mar. 7th, 2025