SWE4211 Lab 9: Distance Measurement

1 Introduction

In the previous weeks, we started to control our robot. Now, with that being completed, we want to do a bit more. We want our robot to be able to measure how far away it is from other things. We can use this, for example, to design a robotics system which follows another robot a fixed distance behind. Or we can use this to automatically slow a robot as it approaches a wall to avoid a collision with the wall.

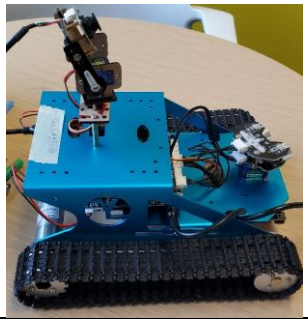
Our robot will not do these advanced things this week, but we will be able to measure distances. And, in doing so, you will need to design and perform an experiment related to measuring latency based upon distance measurements. Your experiment will verify that your robot code is working properly as well as compare latency results with those that you saw in week #2 and week #6. You also will practice analyzing your source code for required synchronization using mutexes.

Overall, this lab will be two weeks in duration. In the first week, you will focus on implementing the program. In the second week, you will focus on designing and performing an experiment with the robot.

2 Lab Objectives

- Implement software to control an ultrasonic distance sensor.
- Design and perform an experiment to analyze latency
- Construct software which uses timing to determine distances
- Design and integrate an algorithm using publish-subscribe related concepts to control a robot

3 Laboratory Equipment Needed

Measuring Device 	Yahboom Robot for Raspberry Pi Smart Car AI Starter Kit 
Robert Charger 	1 Micro SD Card with Rasperbby PI OS Installed 

4 Deliverables / Submission

This is a two-week lab. In the first week, the deliverable is simple. Demonstrate a brief video showing that the system can make measurements and display them to the console. Also commit the final code and push it to the repository.

In the second week, you should submit the full report and the second video.

In the second video, explain how your experiment works and demonstrate part of the experiment being executed. In your video be sure to be clear to explain what you are doing as you do it.

Finally, each team will be responsible for submitting one report (in Canvas) with the following contents:

1. Title Page
   a. Assignment, Date, Name of your github classroom repository, Video link
2. Introduction
   a. What are you trying to accomplish with this lab? This section shall be written IN YOUR OWN WORDS. DO NOT copy directly from the assignment. This section should be several grammatically correct sentences.
3. Prescribed results (from previous labs)
   a. What are the 2 plots of cyclictest (pi4 in powersave mode, pi4 in performance mode after RTE is installed)
4. Experimental Procedure
   a. Include the detailed writeup of the experimental procedure to assess the performance of the distance measurement code. The procedure should be detailed enough so that you can give it to a sophomore or junior student, and they can run your experiment without requiring your intervention. Make sure each step is clear and deliberate and identify any values that are to be recorded. Units and specific instructions are important.
5. Experimental Results
   a. Include the raw results of running your experiment. Be sure to record the units for all measurements that you take. Your raw results should be printed to a table.
   b. You ideally should have 2 sets of data, one for regular mode and one for performance mode.
6. Experimental Analysis
   a. Include a table showing any derived values used for the analysis of your experimental data. Be certain to explain how the measured data was transformed.
   b. Include an appropriate analysis of your experimental results. This should include anything you calculate from your data set and may include tables as well.
   c. This should include at least one to two appropriate graph(s), as well as any relevant equations or mathematical derivations relevant to the problem.
7. Experimental Conclusions
   a. Document any relevant conclusions from your experiment.
   b. In writing up your conclusions, can you map the results of the cyclic test to the results you obtained in previous labs? Be thorough in your discussion, explaining any differences that may be present.
8. Discussion
   a. How did you decide to place mutex locks and unlocks or lock guards in your code
   b. What is the purpose for the std_atomic data types where they are used?
   c. How did you approach design of the experiment?
   d. Based on the data recorded here, and the stopping data from the previous lab, will you need to limit maximum speed in order to guarantee that the robot can stop before hitting a given obstruction. Provide appropriate stopping distance data and coast data to justify your decision. For right now, assume that the distance sensor runs 10 times per second.
   e. Based on your robot design and implementation, what REPORT message will you need to add to your system to be able to add a class that will monitor the distance sensor and stop robot motion in time to prevent a collision?
9. Things gone right / Things gone wrong.
   a. This section shall discuss the things which went correctly with this lab. Include key things that came simply in completing this lab and experiences from other labs which helped to complete this exercise.
   b. This section shall also discuss the problems that occurred completing this lab. Include issues with C++ programming that occurred, issues with networking, or other problems encountered when successfully completing the lab.
10. Conclusions
    a. What have you learned with this experience?

b. What improvements can be made in this experience in the future?

This material should be submitted as a single pdf file.
If you have any questions, consult your instructor.

5 Measuring Distances

There are several methods for measuring distance, and robotic systems employ different ones depending on the specific requirements of a task. Advanced technologies like radar use radio frequency (RF) pulses and calculate the time it takes for the signal to reflect off an object. While radar is effective and reliable, it tends to be costly. LiDAR, commonly found in autonomous vehicles, uses pulsed laser beams to determine distances. It can offer better resolution than radar and perform well in certain scenarios, but it also comes with a high price tag. Additionally, LiDAR can struggle in foggy conditions, where the laser signal may not return properly, leading to inaccurate measurements.

For short-range distance sensing, infrared (IR) sensors are often used. These sensors emit an infrared signal via an LED, which reflects off nearby surfaces. The amount of reflected signal helps estimate the distance. However, IR sensors have limitations: they can be easily blocked by bright sunlight or fog, and they are only effective at short distances. The robot in question is equipped with infrared sensors, which will be utilized in upcoming labs for various applications.

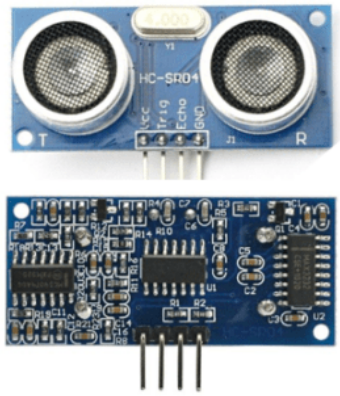


Figure 1 HC-SR04 Ultrasonic Sensor[1]

For simple distance measurements, one of the most commonly used devices is the ultrasonic sensor. As the name suggests, this sensor uses ultrasonic sound waves to determine the distance to a surface. Figure 1 illustrates the ultrasonic sensor, which is mounted on the front of the robot.

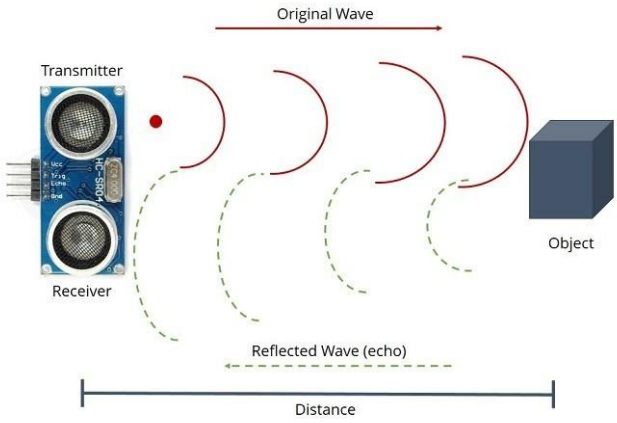


Figure 2 The HC-SR04 Ultrasonic Sensor setup.[2]

T

[1] https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
[2] https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

The operation of the ultrasonic sensor is illustrated in Figure 2. The transmitter emits a brief burst of ultrasonic sound waves, which travel through the air until they encounter an object. These waves then reflect off the surface and return to the receiver. By measuring the time it takes for the sound to travel to the object and back, and knowing the speed of sound—344 meters per second at 20°C (68°F) [3]—we can calculate the distance using the following formula:

$$\boldsymbol{Distance = S_{Sound} \times \frac{Time}{2}}$$ **Equation 1 Distance Measurement**

Despite its usefulness, the ultrasonic sensor has certain limitations, as shown in Figure 3. If the object is too far away, the sound waves may not reflect effectively. Although the sensor is rated for distances between 2 cm and 400 cm, reliable performance is typically observed only up to about 250 cm in real-world conditions. The sensor also performs best when the object is directly in front of it—at a 90-degree angle. Accuracy decreases significantly if the angle deviates more than 15 degrees from perpendicular. Additionally, small or soft objects can pose challenges, as they may absorb or scatter the sound waves rather than reflect them.

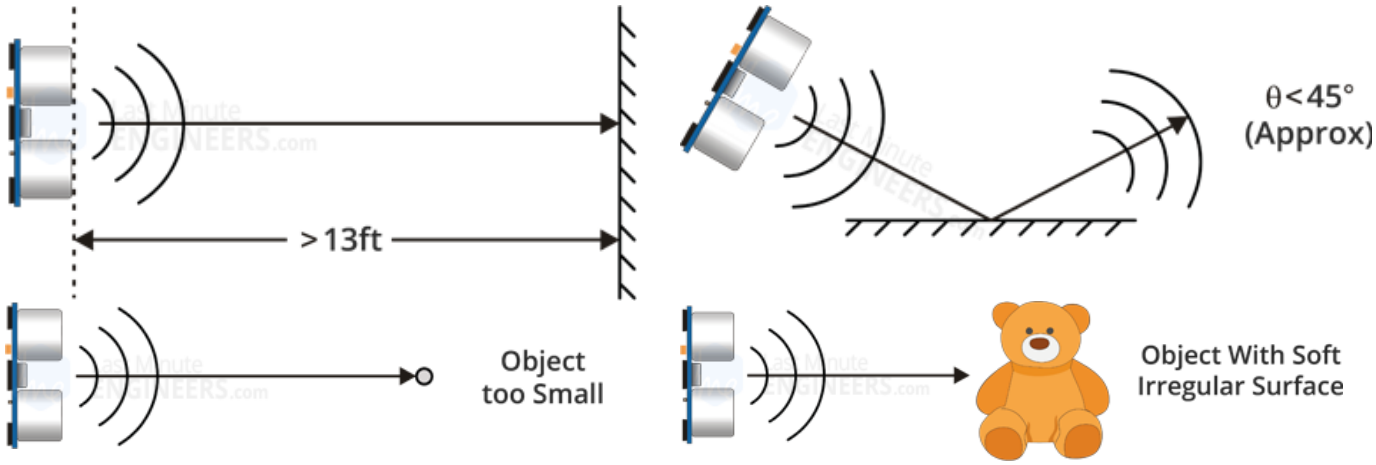


Figure 3 Limitations of Ultrasonic sensing[4]

To measure distance using an ultrasonic sensor, the software must control two pins: the Trigger pin and the Echo pin. The Trigger pin is configured as an output, while the Echo pin is an input. The measurement process begins with the Trigger pin held low. To initiate a reading, the software sets the Trigger pin high for at least 10 microseconds. This brief high signal causes the sensor to emit a burst of ultrasonic sound.

This burst contains a pseudo-random pattern, which helps distinguish signals when multiple sensors are active. However, for optimal accuracy, only one ultrasonic sensor should perform range finding at a time on a single robot.

Once the sound burst is transmitted, the Echo pin goes high, indicating that the sensor is waiting for the reflected signal. When the echo returns and the reception is complete, the Echo pin goes low, marking the end of the measurement cycle. This timing sequence is illustrated in Figure 4.

On many microcontrollers, such as the Arduino, this process can be handled automatically using a feature called input capture, which measures the duration of the Echo pulse without requiring continuous software intervention. However, the Raspberry Pi lacks built-in input capture hardware, so a separate software thread must monitor the pins to measure the pulse duration accurately.

[3] https://www.sfu.ca/sonic-studio-webdav/handbook/Speed__Of_Sound.html#:~:text=At%2020%C2%B0C%20(68,given%20in%20the%20chart%20below.
[4] https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

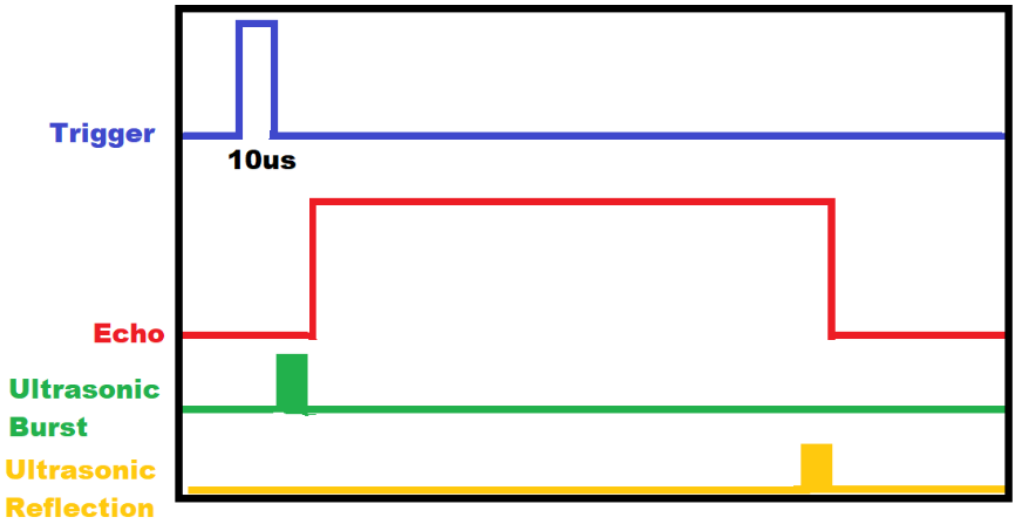


Figure 4 Ultrasonic sensing sequence for the device.

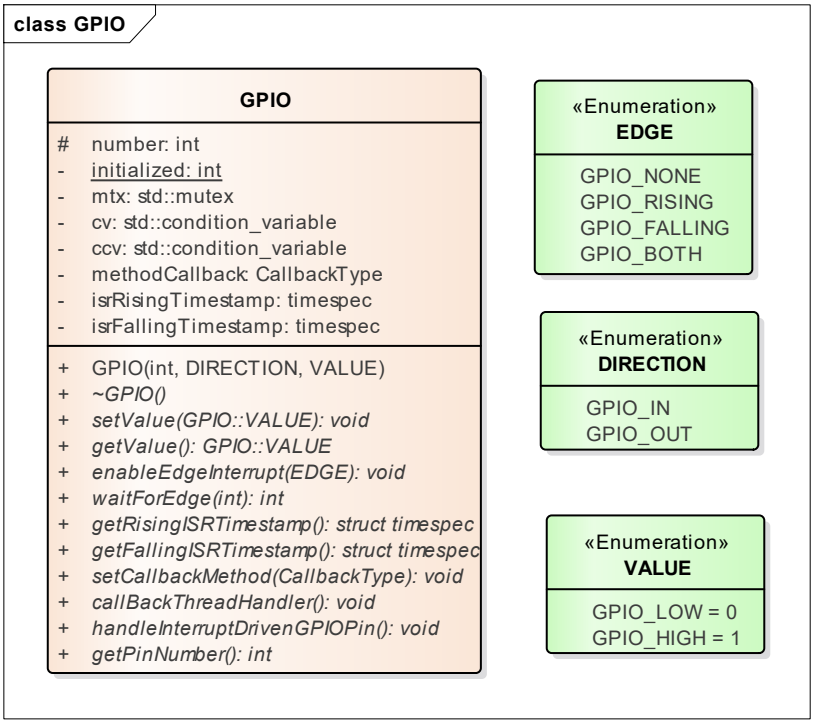


Figure 5 The GPIO Class

The provided GPIO class, shown in Figure 5, is designed to facilitate input capture operations. It includes a method called enableEdgeInterrupt, which allows the system to trigger an interrupt whenever a rising edge, falling edge, or both occur on a specified pin. When these edges are detected, the system records timestamps into two dedicated variables.

To measure distance using the ultrasonic sensor, interrupts are enabled on the Echo pin. The software then waits for a rising edge (indicating the start of the echo pulse) followed by a falling edge (marking the end of the pulse). Once both timestamps are captured, the time duration during which the Echo pin was high can be calculated. This duration directly corresponds to the time it took for the ultrasonic signal to travel to the object and back, allowing the distance to be computed.

On platforms like the Arduino, this process is typically handled by built-in input capture hardware. However, since the Raspberry Pi lacks such hardware, a separate software thread must monitor the pin states and handle timing manually.

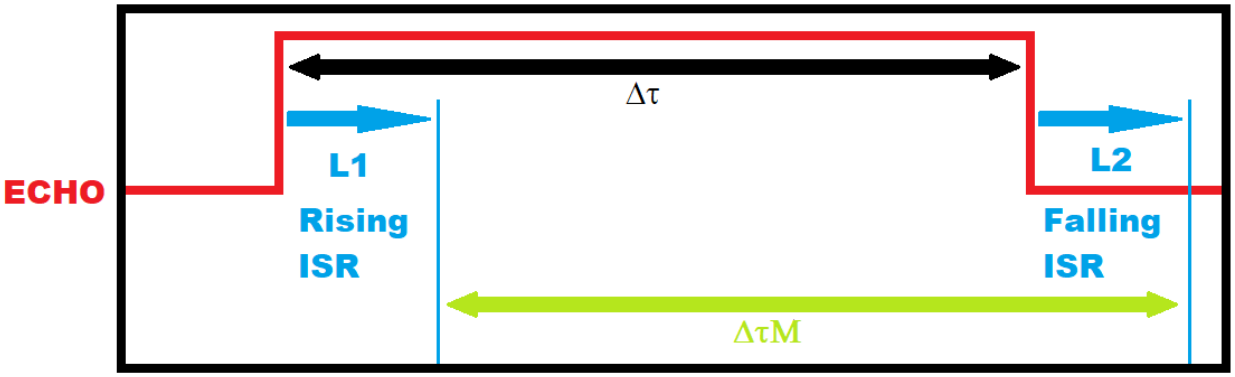


Figure 6 The measurements we can take.

5.1 The Impact of Latency on Calculations

While measuring the duration of the Echo signal may seem straightforward, the task becomes more complex when latency is considered. Figure 6 presents a timing diagram for this system. Ideally, we want to measure Δt, the actual time between the transmission of the ultrasonic pulse and the reception of its echo. However, due to software latency—specifically the time required for a context switch to execute the interrupt service routine (ISR) and record the timestamp—the recorded times are offset.

The timestamp for the rising edge of the Echo pin is delayed by L_1, and the timestamp for the falling edge is delayed by L_2. As a result, we measure Δt_m, which is the time between these two delayed timestamps.

- If $L_1 = L_2$, then $\Delta t_m = \Delta t$, and the measurement is accurate.
- If L_1 and L_2 are both small relative to Δt, then Δt_m is a good approximation.
- However, if L_1 and L_2 differ significantly, the measurement becomes skewed:
  - If $L_1 \approx 0$ and $L_2 \approx \bar{L}$ (average latency), then Δt_m will be longer than Δt by $\bar{L}$.
  - If $L_1 \approx \bar{L}$ and $L_2 \approx 0$, then Δt_m will be shorter than Δt by $\bar{L}$.

This variability means that latency can introduce measurable error. To estimate the extent of this error, we can analyze the **maximum and minimum distances** recorded by the sensor. Assuming latency follows a **continuous uniform distribution** with an average value of $\bar{L}$, and knowing the lower bound is 0 (i.e., no latency), the upper bound must be $2\bar{L}$. This assumption allows us to model and potentially correct for latency-induced error in distance measurements.

From this, we can show that the max distance recorded, DL is

$$D_L = S \times \frac{t + 2\bar{L}}{2} = \frac{St}{2} + \bar{L}S$$

From this, we can show that the min distance recorded, DS is

$$D_S = S \times \frac{t - 2\bar{L}}{2} = \frac{St}{2} - \bar{L}S$$

We can algebraically manipulate these formulas to show that

$$t = 2(\frac{D_L}{S} - \bar{L})$$

and

$$t = 2(\frac{D_S}{S} + \bar{L})$$

Resulting in

$$\frac{D_L}{S} - \frac{D_S}{S} = 2\bar{L}$$

Thus, if the distance is fixed and we measure the max and min times at that distance, we can estimate the latency. This is only an estimate – it may not match latency exactly, as there is no guarantee that by L_1 or L_2 has truly reached zero.

6 Implementation

The first part of this lab focuses on completing the DistanceSensor code and working with a templated main file. The UML for this is shown in Figure 7. You will construct the class for the distance Sensor. In constructing it, follow the documentation in the baseline docs folder, but also make sure you add synchronization to the code where it is necessary due to multiple threads potentially interacting. When finished, the program, when operating on the Raspberry Pi, will report back the current distance reading, the average distance reading, and the max and min readings. Pressing the Q button followed by enter will quit the program. Pressing the R button will reset the max, min, and average values. Pressing the P key will print information about the threads. This is all shown in Figure 8.

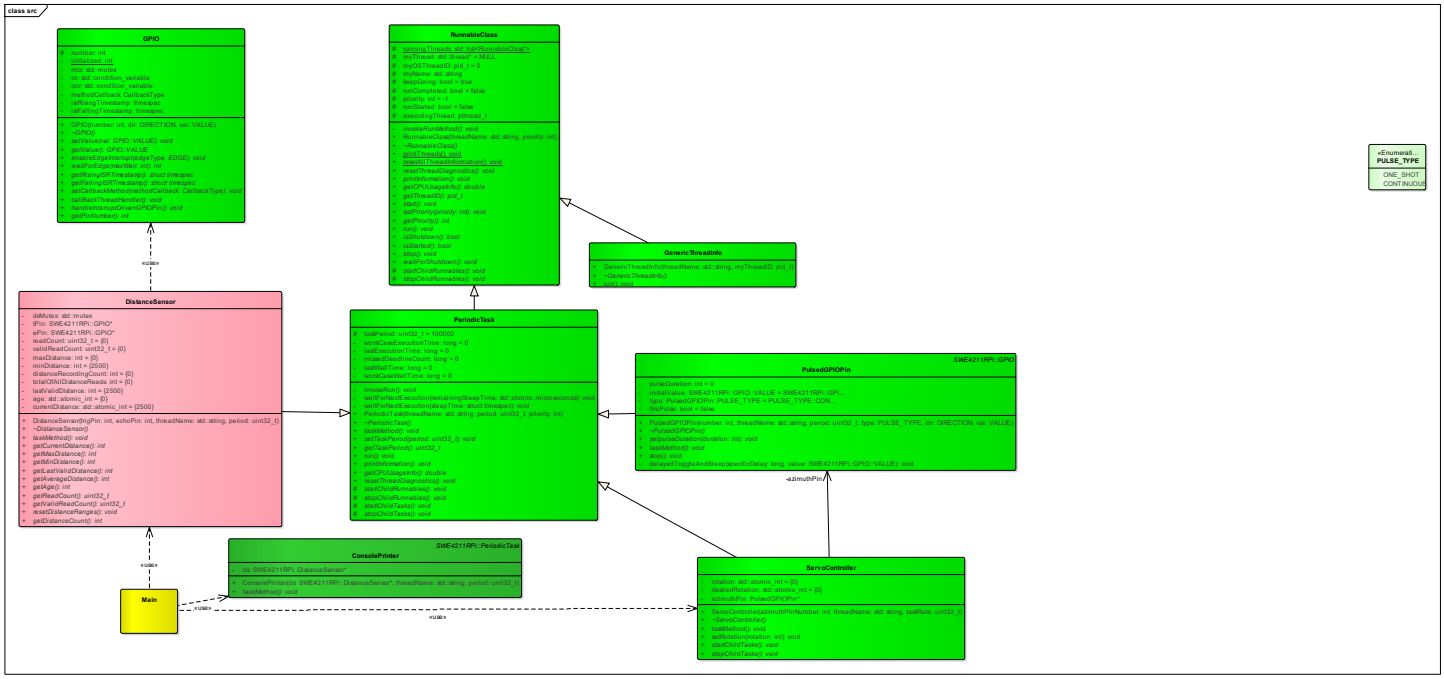


Figure 7 UML for the project

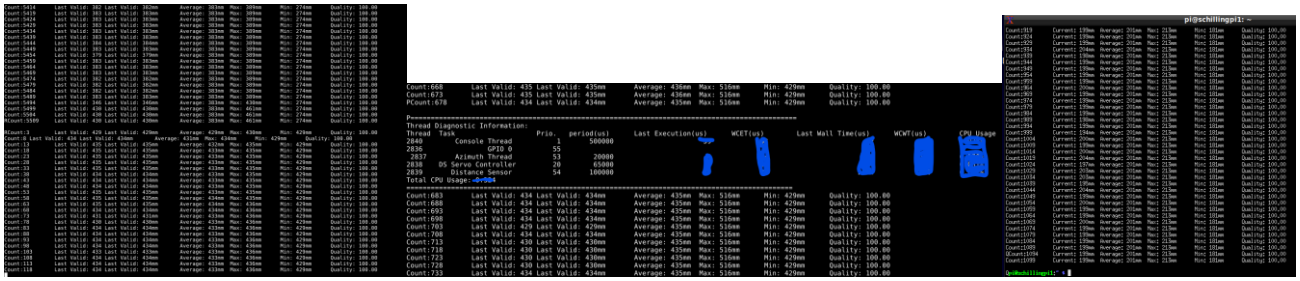


Figure 8 Sample program execution. Left: This shows the impact of resetting the values. Note that the average, max, and min all change here. Center: Printing out the thread information. This shows details about the thread. We won't worry too much about this right now. Right: Quitting the program with Q.

7 Your Experiment

Not that you have a working distance measurement system, we want to perform an experiment with the robot and the software you have developed. There are two objectives for the experiment. First and foremost, using various measurements of distance, including short, long, and medium distances, you are to attempt to estimate the latency present when measuring distances with the ultrasonic range finder. Second, based on experimental analysis, you are to determine the overall accuracy of the ultrasonic range finder over the range of 3 to 24 inches in distance. Your experiment must show, using the distance sensor, the difference when the processor is in performance mode and when it is in powersave mode, as this may impact accuracy.

To begin, you are to draft a procedure for performing this experiment. The procedure should be written so that someone who is not as technically savvy as you are could follow the steps and perform the steps. Make sure your procedure includes distances to test at, items to record in a data table, and specific calculations that you want the user to perform. You are provided with an experimental harness[5], shown in Figure 9, which will help with this experiment.[6]



Figure 9: The experimental measurement harness. The harness includes a yardstick which measures the distance in inches from an 18-inch square reflective surface of wood. The robot can be placed anywhere on the yardstick and distance measurements can be made.

In the lab, you then are to conduct the experiment, following the procedure you developed to experimentally determine the latency in the robot and the accuracy of the distance measuring code. Record all the data and generate a set of plots, one or more plots which explain your results.

[5] These devices will be available in the cyber security lab for you to use.
[6] Note: While this segment is included here, it is advisable to draft the process prior to lab in order to save time in lab.