Ultrasonic Range Finder

Progress Report II

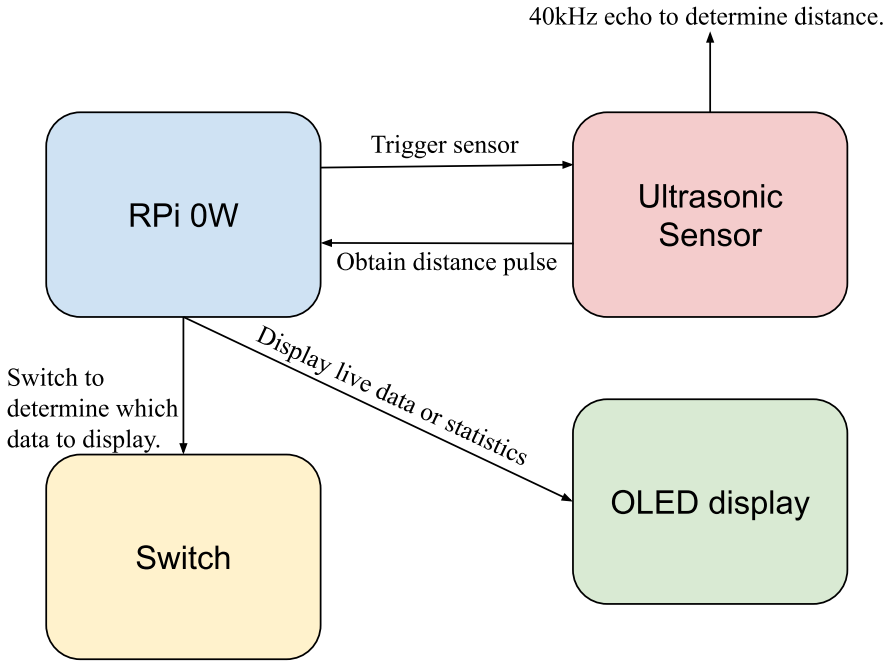
Team: Siddhartha Bajracharya, Joshua Wilbur

The team is building an embedded system device that measures distance using an ultrasonic sensor. The distance from the ultrasonic sensor is presented on an OLED display. A Raspberry Pi microcontroller interprets and outputs data from the ultrasonic sensor. This project demonstrates the team's embedded system skills through the implementation of hardware and software. Due to the nature of this sensor, real-time considerations have influenced the design. The ultrasonic range finder displays the team's ability to optimize software and debug issues.

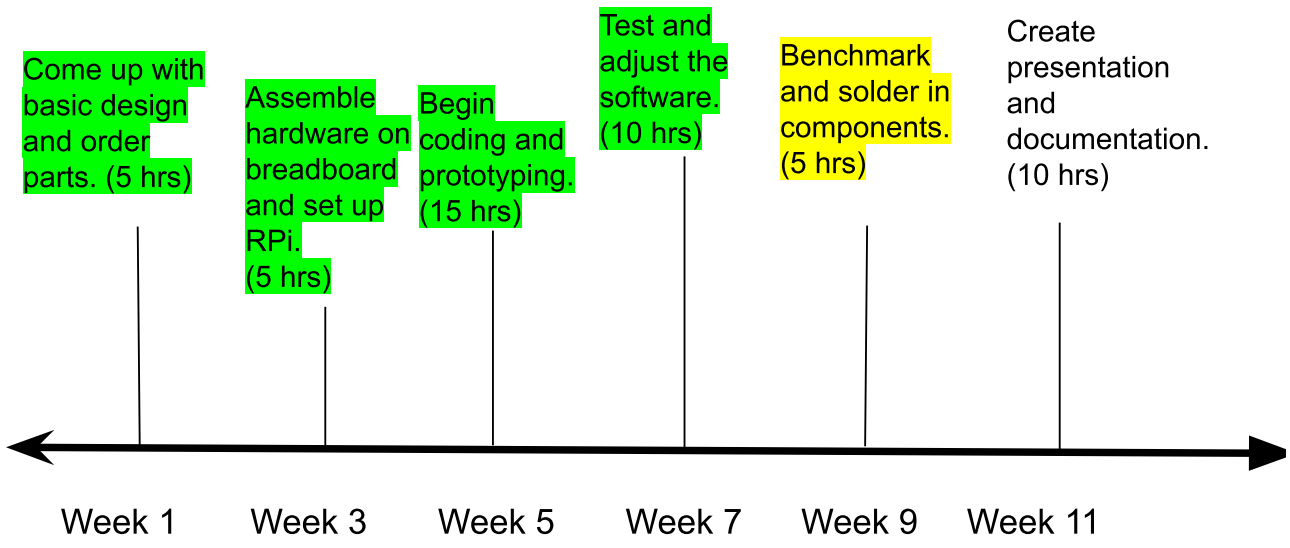
The hardware design for this project is simple. After prototyping is complete, the components will be connected to a solder board. Jumper wires will link these components to the microcontroller. The ultrasonic sensor must be placed at the edge of the board to avoid obstructions from other components.

The software design is more involved. GPIO pins are used to trigger the ultrasonic sensor and input echo pulses. The sensor is triggered by a 100uS wide pulse. Statistics such as mean and standard deviation of the inputted values are taken over a short time period to ensure data accuracy. A slide switch is included so the user can view live readings or statistics. The OLED display is driven by the I2C communication bus. The C programming language is used for communication between the hardware. A block diagram of the design is presented below along with additional information regarding the project.

Block Diagram



Project Timeline



Project Progress by October 9th

The project has progressed well so far. PWM to drive the trigger pin on the ultrasonic sensor has been generated. The OLED display is functional and ready to be built upon. Pulse reading code has also been implemented. The team is ready to begin merging these individual parts together. The team is abiding to the timeline above, no modifications have been required so far. Currently, the project is on track to be completed before December.

Project Progress by November 6th

Currently, the project is functioning and close to completion. Code for the individual components has been merged together to create a prototype. Code has been written to output distance statistics, such as mean and standard deviation, to the display. Some minor changes have been made to the design to add additional functionality. A slide switch has been added to allow users to switch between live readings and statistics. The timeline above hasn’t needed modifications thus far. The project is on track to be completed by Thanksgiving break.

Work Logs

Work logs are being used to track team progress and hours. These will be continuously updated throughout the semester. Each team member has their own log, some work will appear on both logs if the team is working concurrently.

Siddhartha's Work Log

Date	Hours Worked	Progress Notes
9/12	2	Created project proposal. Also researched parts and design.
9/17	1.5	Created slideshow for presentation 1, further refined design.
10/6	2	Wired up the Ultrasonic Sensor onto a breadboard and researched on how to make it work with a Raspberry Pi
10/6	5	Wrote code for the Ultrasonic Sensor using C. Found some code for it in Python and translated it into C. (Source of the website: https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi). Also added a while loop to make the readings continuous. The trigger input for the TRIG pin is a HIGH signal for 100 us. I have not implemented PWM onto this code yet but it seems to be working without it. Used the "pigpio" library to initialize and set the GPIO pins.
10/7	2	Got code to compile but was only outputting 0.93cm. Tried to debug any issues with the code or the hardware but could not find any.
10/7	1	Got a new sensor from Josh and it worked. Got distance values from 1cm till 400cm.
10/8	2	Work done on slides and progress report
10/26	5	Wrote code to calculate the mean and standard deviation values. The initial code displayed value every second. The new code takes an average of 3 data readings, i.e. three seconds, and then uses standard formulas to get the mean and standard deviation values. For standard deviation, the differences between the readings and the mean are squared and summed up. This value was then divided by the number of readings, which was just 3. Then the square root of that value is taken.
10/26	2	Worked on implementing a switch case so that if the switch is closed, the standard deviation and mean are displayed. Otherwise, the normal readings are shown. Implemented a high-level method to control the GPIO pins for the switch. This was changed by Josh to implement a more low-level approach.
11/3	2	Worked on the progress report and progress presentation.

Joshua's Work Log

Date	Hours Worked	Progress Notes
9/12	2	Created project proposal. Also researched parts and design.
9/17	1.5	Created slideshow for presentation 1, further refined design.
9/24	2	I worked on adding PWM to allow for continuous triggering of the sensor. PWM on the Pi is straightforward and only requires writing to configuration files. Here is a breakdown of what I did. ● I set up the hardware for this project. Pin connections are in the repository README and are subject to change. ● I did some research to figure out how to enable/control PWM. The link below helped me get started. https://developer.technexion.com/docs/using-pwm-from-a-linux-shell ● I found those files within my Pi and wrote two functions. The first writes values to the PWM config files. The second utilizes the first function to do a full PWM setup. ● I made header files to allow these functions to be used in other programs. ● I set the PWM frequency to 20 Hz at 20% duty cycle. Tested and worked with an oscilloscope. Image is in the repository at docs/RPi0_PWM_Result.png ● Developed a Makefile for easy compilation. Issues encountered: ● Had to add header file guards to avoid multiple import errors. ● Have to run the program as the root user. Fixed by adding a conditional to check this. ● Errors with file writing, ended up having to use const *char type paths to fix this. ● Typical Makefile pain.
9/28	1.5	I set up and tested the OLED display. I used a library linked in the repository due to the complexity of display drivers. I've used this library in past projects and it has worked well. Step by step: ● I imported the "ssd1306" display directory from another project on my Pi. ● This library displays text/shapes on the SSD1306 via an executable binary. Passing command line arguments to this will generate outputs. This library is a bit odd, but it's one of the only ones written entirely in C for the Raspberry Pi. ● Basic testing showed this program worked, it will be built upon in the near future. Issues encountered: None

10/1	1	Didn't do too much, just set up a display function. Step by step: ● I wrote a function to fork a new process and run the executable on that process. Homework two from my COS 331 course was referenced. ● Basic testing showed this program worked, it will be built upon in the near future. Issues encountered: ● OLED Binary wasn't being executed but no errors were given. After some testing, I found I had to change my working directory.
10/6	3	Work: ● Started working on progress report I and slides, both due 10/9. ● Refactored display program so it was easier to operate. Issues encountered: ● The SD card decided to corrupt when I rebooted. Took some time to reinstall Raspbian.
10/8	0.5	Work: ● Adjusted PWM based on Siddhartha's recommendation. ● Finalized slides and report No issues encountered.
10/12	1.5	Work: ● Scrapped PWM code as isn't necessary for this device to work. ● Integrated range-finding, triggering and display code. ● Added a conversion to output the distance in inches. Issues encountered: ● Program doesn't work if pigpio daemon is already running.
10/26	2	New display library is being used, the prior one was confusing to use. This one is provided by Adafruit with minor modifications from Ilia Penev. Work: ● Set up new library, source is here: https://github.com/iliapenev/ssd1306_i2c ● Used demo.c as a reference to build code in main.c ● No major issues encountered
10/27	1.5	Work: ● Cleaned up displayed text formatting ● Made user input code lower level ● Modified README
11/2	2	Worked on progress report and presentation slides for Wednesday

Bill of Materials

Component	Part Number	Unit Price	Supplier
Raspberry Pi Zero W	3400	$15.00	Adafruit
OLED Display	SSD1306	$5.50	Digikey
Ultrasonic sensor module	HC-SR04	$4.50	Sparkfun
SPDT switch	805	$0.95	Adafruit
USB Power Cable	N/A	$0.00	Hackerspace
Jumper Wires	N/A	$0.00	The team
Solder Board	N/A	$0.00	The team

Additional Information

The team is using a library from GitHub to drive the OLED display, this is linked below. The below datasheet contains information regarding the ultrasonic sensor and will be referred to by the team. Images of the working prototype and measurements are found in the images directory of our repository.

- Our GitHub Repository: https://github.com/JoshWilbur/Ultrasonic-Rangefinder
- HC-SR04 Datasheet: https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf
- Display Library (Ilia Penev): https://github.com/iliapenev/ssd1306_i2c
- Reference: https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi