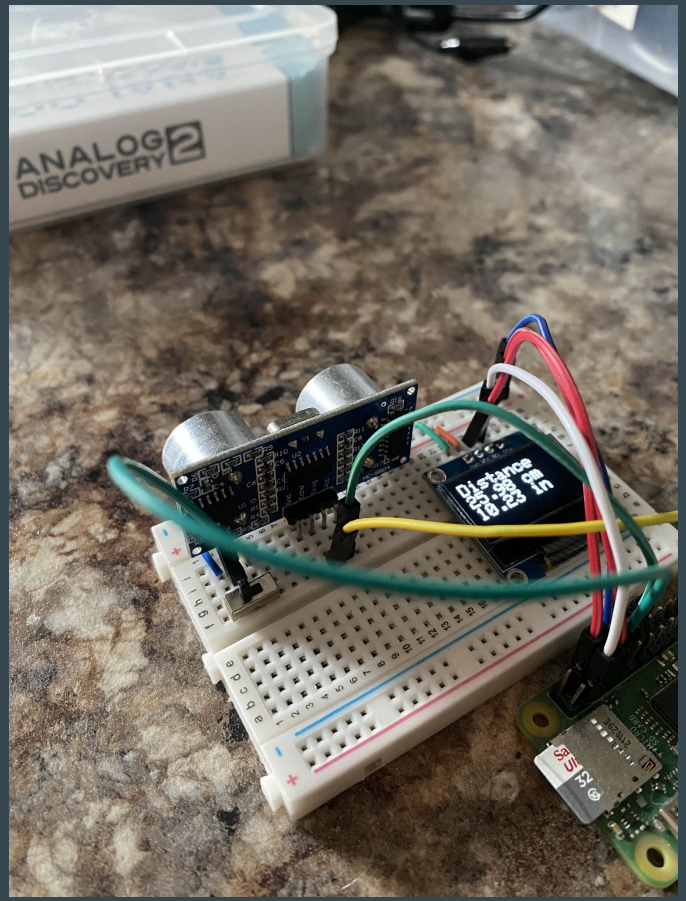
Ultrasonic Range Finder

Presentation III

Siddhartha Bajracharya & Joshua Wilbur

Project Overview

- A short-distance (<4m) rangefinder.
- Ultrasonic sensor uses 40KHz sound waves to detect distance.
- An OLED display outputs the live distance along with statistics.
- A switch allows user to toggle displayed data.



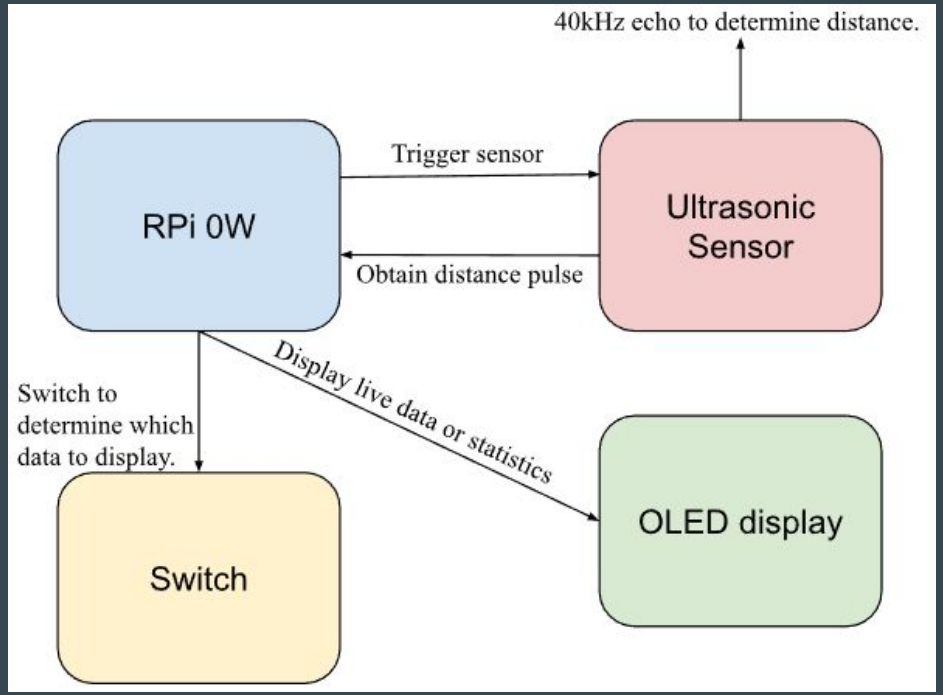
Design & Block Diagram

Hardware:

- Raspberry Pi 0W
- SSD1306 OLED Display
- HC-SR04 ultrasonic sensor module
- SPDT switch*
- USB power supply

Software :

- Using the C language to talk with hardware.
- Raspbian OS is running on the Pi.
- Display library used, more on slide 8.

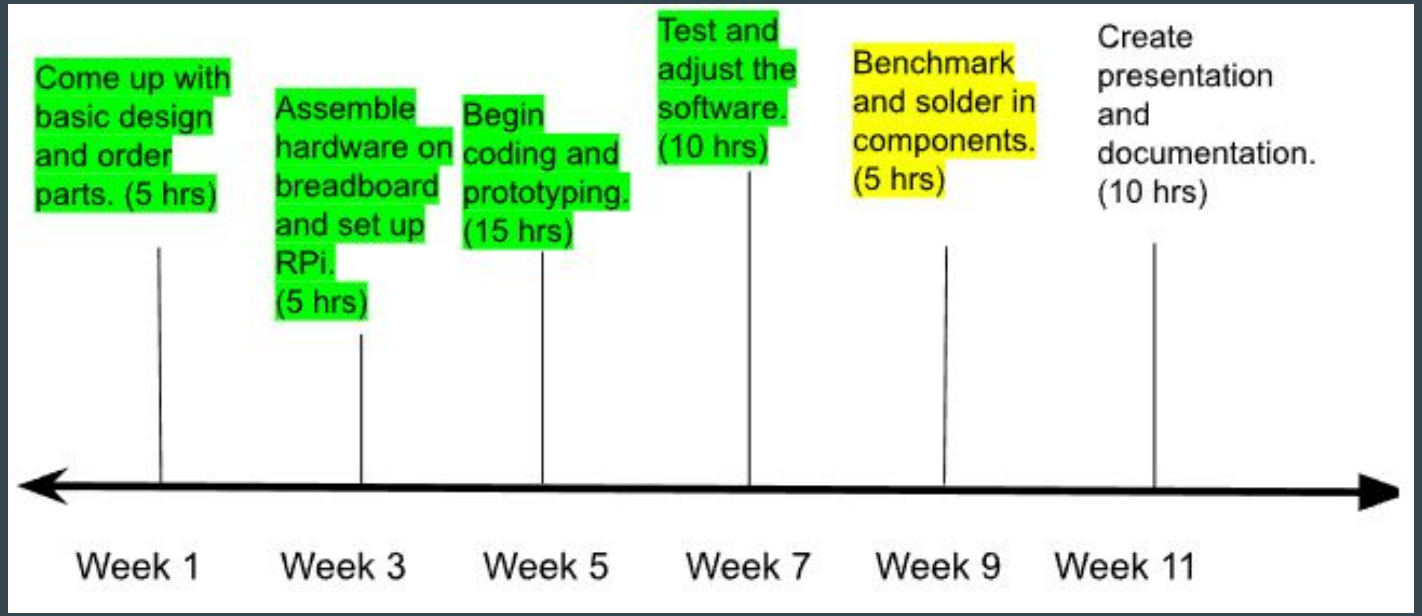


Bill of Materials

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

Current Project Timeline

- Note: It is week 10 of the semester.



Project Progress

What has been completed?

- Basic design and proposal
- Acquired and attached hardware.
- OLED display code.
- Initialize the Ultrasonic Sensor.
- Basic distance/pulse reading.
- Functional prototype.
- Testing to fine-tune software.
- Statistics program

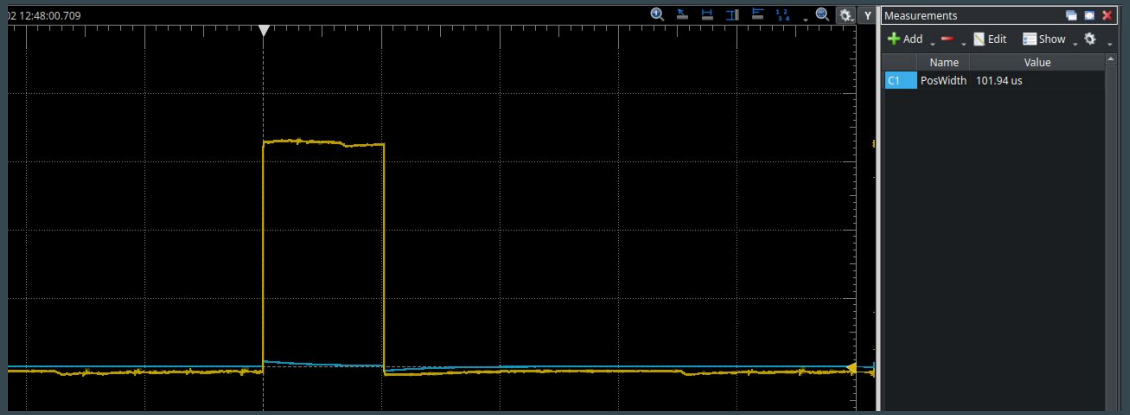
What is left?

- Solder the components to perfboard.
- Benchmark device
- 3D print a case, if time avails.

Triggering the Ultrasonic Sensor

- 100uS pulse to trigger a reading.
- PWM was inaccurate with the Pi.
- Turn GPIO pin on/off directly.
- Using pigpio library.

```
// Send a 100 microsecond pulse to TRIG
gpioWrite(TRIG, PI_ON);
usleep(100);  // Send pulse for 100 microseconds
gpioWrite(TRIG, PI_OFF);
```



New Display Library

- New display library is from Ilia Penev on GitHub.
- Much simpler than the previous library.
- No mysterious binaries!
- Lots of functions to interact with display.


```
void ssd1306_dim(unsigned int dim);

void ssd1306_drawPixel(int x, int y, unsigned int color);

void ssd1306_drawFastVLine(int x, int y, int h, unsigned int color);
void ssd1306_drawFastHLine(int x, int y, int w, unsigned int color);

void ssd1306_fillRect(int x, int y, int w, int h, int fillcolor);

void ssd1306_setTextSize(int s);
void ssd1306_drawString(char *str);
void ssd1306_drawChar(int x, int y, unsigned char c, int color, int size);
```

Statistics Program

- Takes the mean and standard deviation of the readings.
- Takes 3 readings to provide statistics (3 Seconds)
- Mean is pretty straightforward.
- For Standard Deviation, calculated the variance and divided it by the number of readings.
- Switch between the statistics or the live readings.

```
void display_statistics(double *distances, int count) {
    double sum = 0, mean, stddev = 0;
    for (int i = 0; i < count; i++) {  // For loop to add all the measured distances in 3 seconds
        sum += distances[i]; // Sum for the mean
    }
    mean = sum / count; // Calcaute the mean

    for (int i = 0; i < count; i++) {
        stddev += pow(distances[i] - mean, 2); // Sum of the squared deviations
    }
    stddev = sqrt(stddev / count);  // Square root of the variance
```

User Input via Switch

- Switch lets users change what data is being displayed (live/stats).
- Switch hooked up to 3V3, GND and GPIO4.
- Setup function is very similar to code provided in ECE 471.
- An ioctl request is run to check switch state.

```
int read_input(void){
        // Run ioctl when called
        memset(&data, 0, sizeof(data));
        rv = ioctl(req.fd, GPIOHANDLE_GET_LINE_VALUES_IOCTL, &data);

        // Error checking to make sure input was read in correctly
        if (rv == -1) {
                perror("GPIO value read error: ");
                exit(1);
        }

        // GPIO read result in data.values[0]
        int result = data.values[0];
        return result;
}
```

Problems Encountered

- Program won’t run if pigpio daemon is already running.
  - Resolved by disabling so it doesn’t run on startup.

Thank you! Questions?