Final-project-skeleton

Team Number: 15

Team Name: RobotGirlz

Team Member Name	Email Address
Mira Sivaprasad	mira10@seas.upenn.edu
Lubha Churiwala	lubha@seas.upenn.edu
Catie Robinson	corobin@seas.upenn.edu

GitHub Repository URL: https://github.com/upenn-embedded/final-project-f25-f25-final_project_robotgirlz.git

GitHub Pages Website URL: [for final submission]

Final Project Proposal

1. Abstract

The **Hydrobot** is a smart water bottle, which is an intelligent hydration assistant. It will use sensors to monitor the user's water consumption and encourages regular hydration. The Hydrobot continuously checks the water level in the bottle using an ultransonic distance sensor and counts how many times the user has refilled the bottle. If the user hasn't drunk enough water for a set time period, the speaker on the bottle will alert the user by making a loud sound. It also measures the water temperature using a temperature sensor. A screen is used to display the measured data, such as water temperature and water level. This project combines sensing, data logging, and feedback control using the ATmega328PB microcontroller. Additional features include showing red light when water is hot, and blue light when water is normal/cold temperature.

2. Motivation

Everyone is so busy in their daily lives that, sometimes the basic things such as drinking water or eating food on time are forgotten. Take the example of a university student in UPenn - attending multiple classes + labs related to them, back to back deadlines, sometimes working part-time,

perhaps working on projects joining labs, looking for summer jobs, trying to maintain a social life, extra-curricular activities and so on. The list of things to complete in a day is endless. Due to the hustle of life, we forget to drink water -> get dehydrated -> loose energy to complete tasks or survive -> fall sick -> create back logs and the cycle is never ending.
Here's where we are introducing **Hydrobot**, a bottle that reminds you to hydrate yourself.

Additionally, don't we all have a prefered temperature to have our drinks in? Sometimes a little too cold or a little too hot. For example, most university students fall sick, quite often catching a flu during certain months of the season. When you catch a cold, most of us prefer to drink warm water to ease our throat. Even as the weather gets more chilly in Philly, we start avoiding colder water. While getting hot beverages, how many of us have slightly burnt our tongue? I am sure quite a few of us have.
Here's why we are adding the feature of temperature detection, so that everyone can enjoy their drinks in the temperature they'd like and avoid mishaps such as burning your toungue.

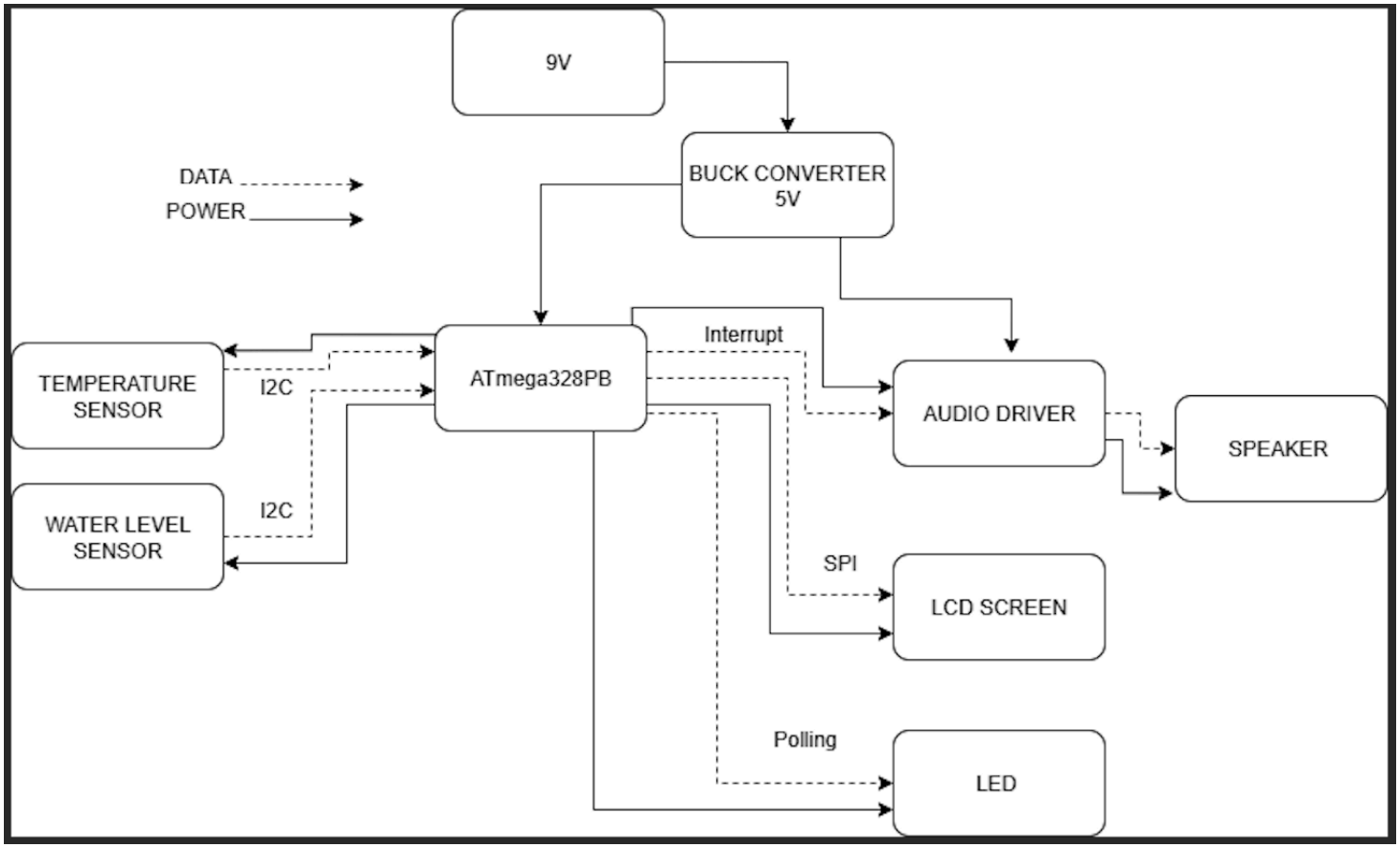
These are our motivations to building this project:

- Create an automatic water bottle that would remind us to intake a significant amount of water every hour.
- The bottle gives the temperature read of the beverage inside, so that the consumers can have their preferred beverage in the temperature of their choice (hot/cold).

What also makes our product more interesting are its following features:

- A speaker that talks in the voice of your choice and style that reminds you to hydrate yourself.
- An LCD that tries to interact with the user by showing emojis, temperature value and how much water was drank in that day.
- A switch that lets you choose if you would like the speaker or LCD to interact with you or not (so that the bottle does not create any inconvinience in public places)

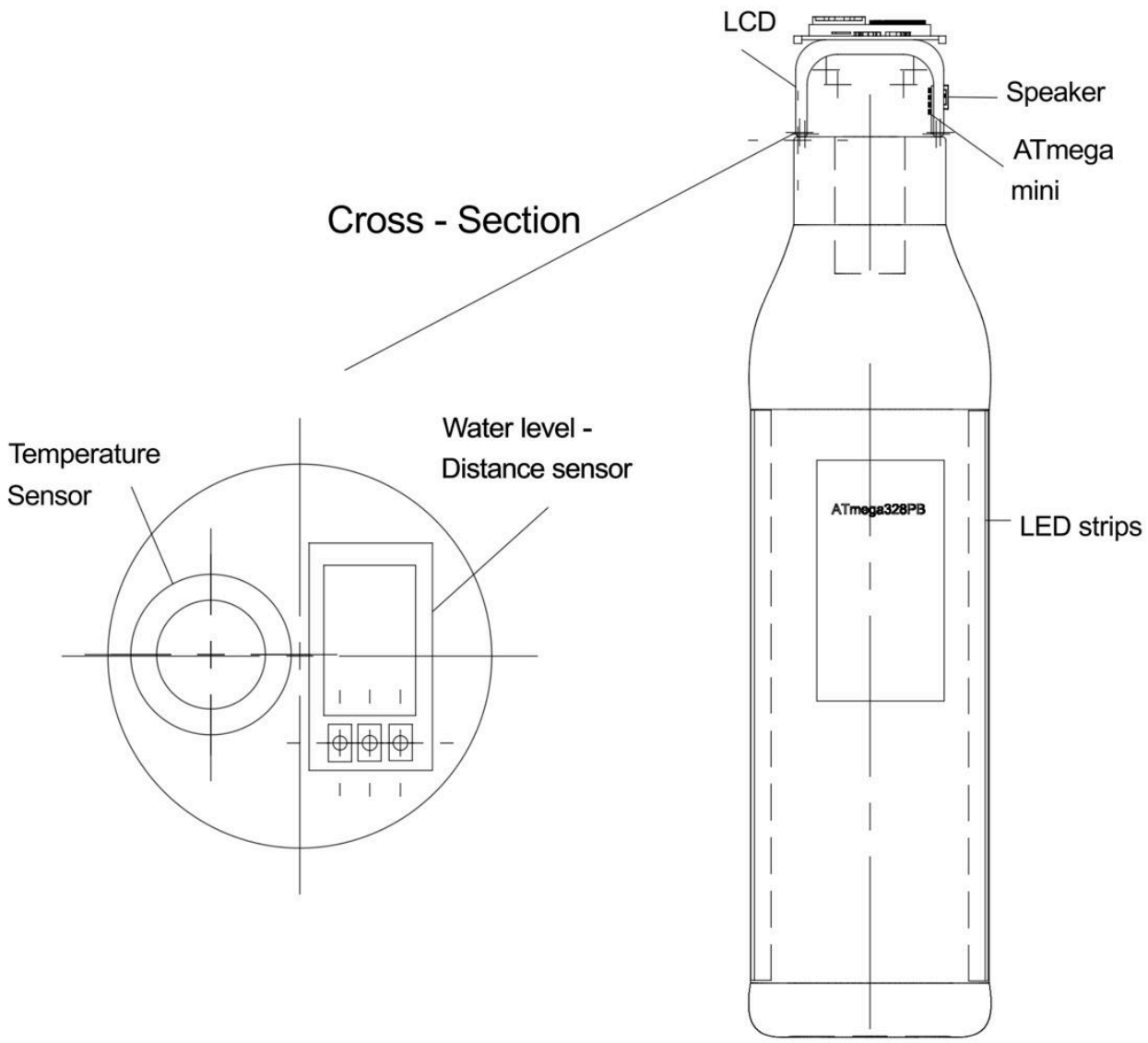
3. System Block Diagram

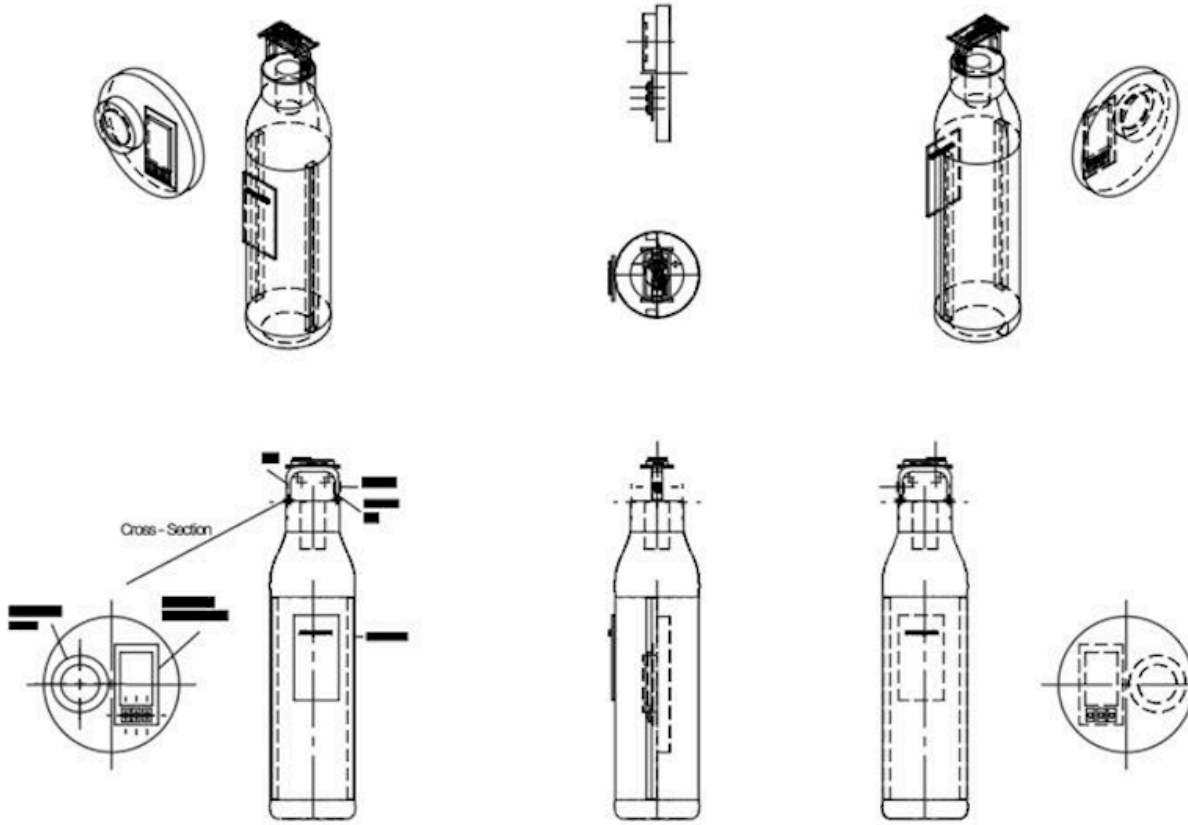


4. Design Sketches

What will your project look like? Do you have any critical design features? Will you need any special manufacturing techniques to achieve your vision, like power tools, laser cutting, or 3D printing? Submit drawings for this section.

This project consists of a water bottle with smart features acheived with the help of LED strips, speaker, LCD screen, speaker, water level distance sensor and temperature sensor. There will be a case attached to the bottle to encase all the wirings and for the LCD to sit well on the top of the cap. These would require a some 3D printing. The sensors inside the cap will have wires that will be taken out by drilling the cap. Drilling holes for the wiring would require the use of some power tools such as a drilling machine.


LCD
Speaker
ATmega
mini
Cross - Section
Water level -
Distance sensor
Temperature
Sensor
ATmega328PB
LED strips



PROJECT

Hydrobot

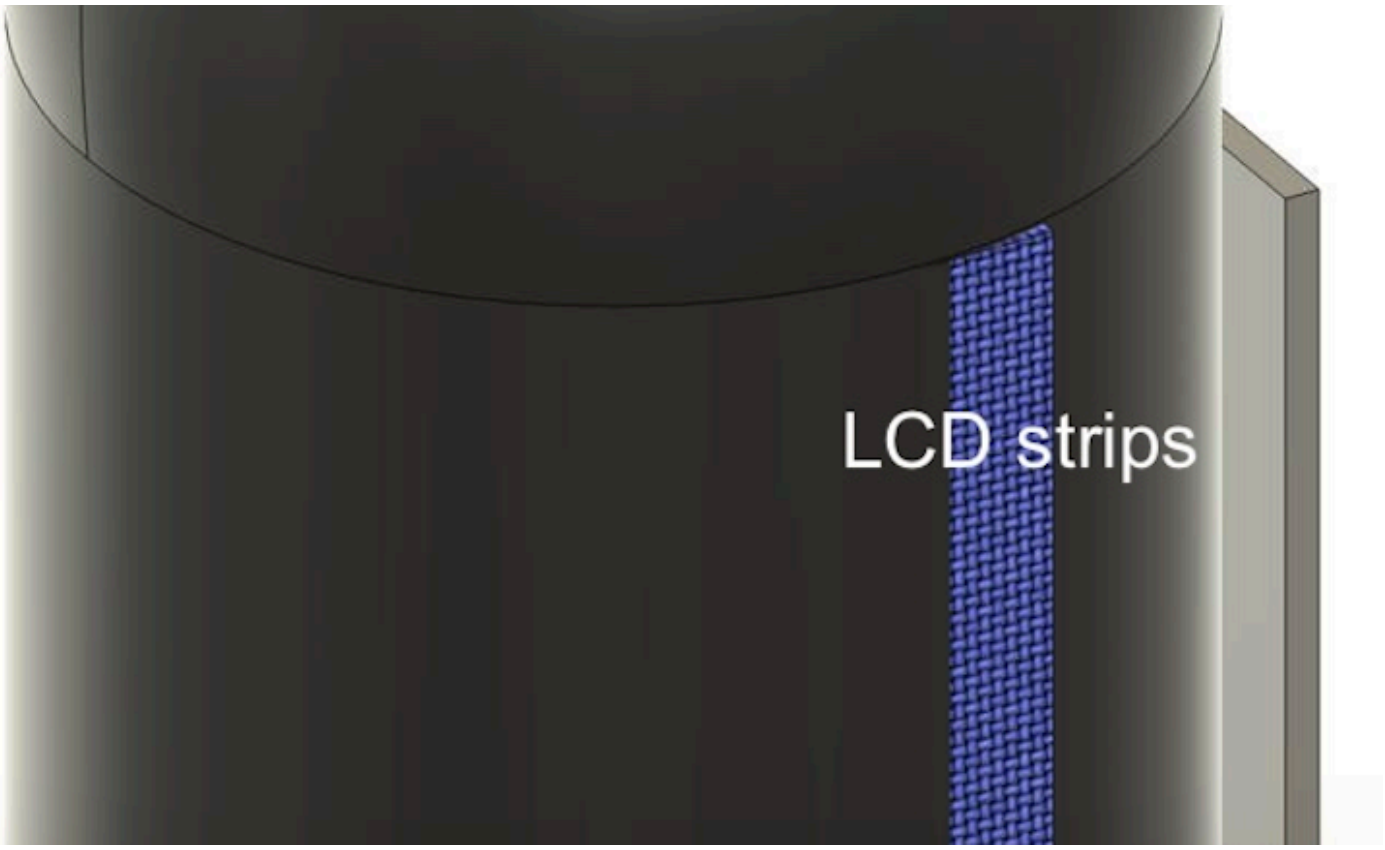
TITLE

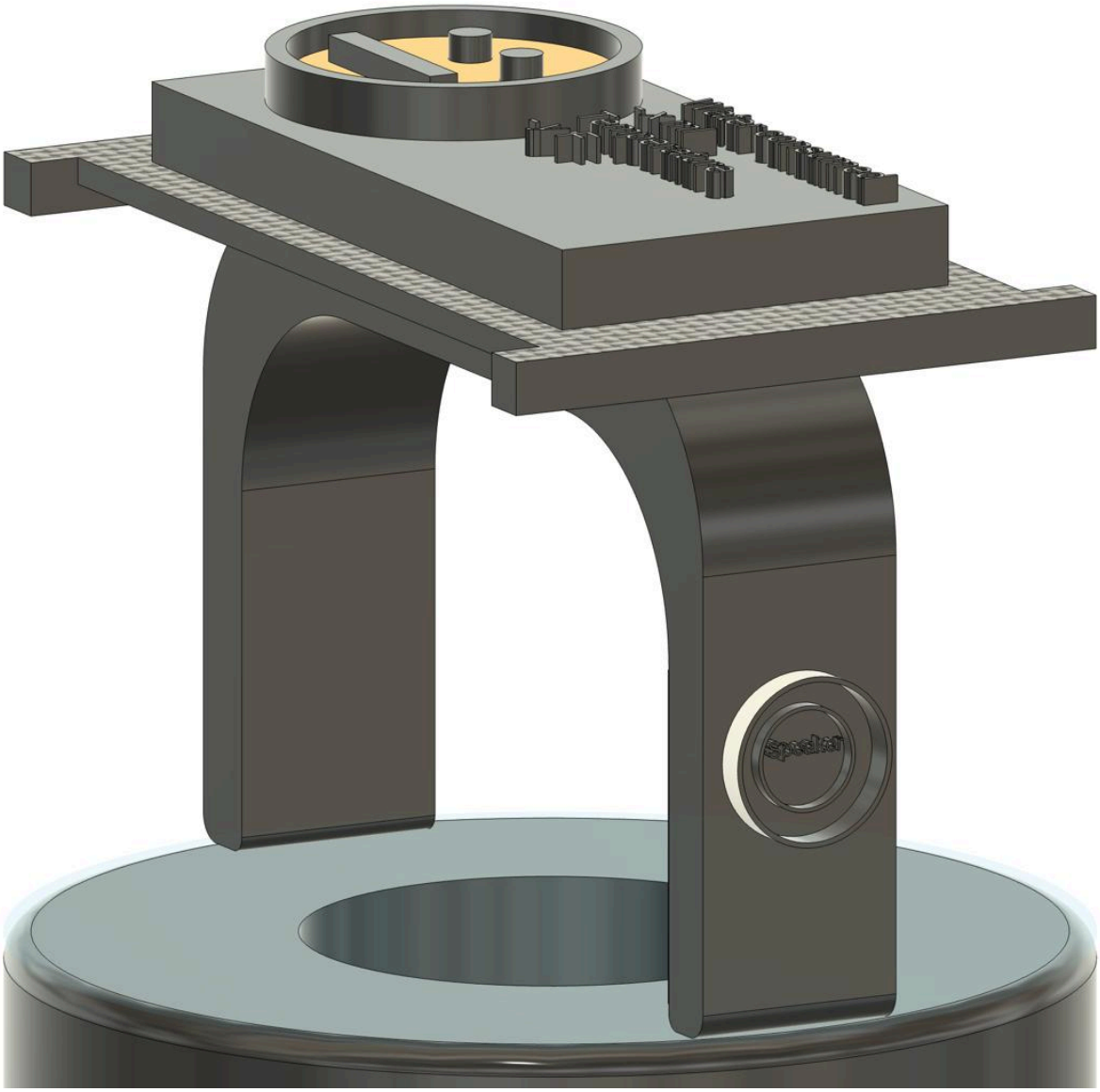
	Bottle				
APPROVED		SIZE	CODE	DWG NO	REV
CHECKED		B			
DRAWN	Mira Sivaprasad 10/27/25	SCALE 1:5 (From parent)	WEIGHT	SHEET 1/1	

Inside the bottle cap


Water consumed
- x ml
Temperature
- y F


Audio
FX Mini
Sound
Board


LCD strips



5. Software Requirements Specification (SRS)

5.1 Definitions, Abbreviations

Abbreviation	Definition
MCU	Microcontroller Unit (ATmega328PB)
IR	Infrared (temperature sensor)
ToF	Time-of-Flight or Ultrasonic sensor for distance measurement
LCD	Liquid Crystal Display used for feedback
LED	Light Emitting Diode for temperature indication
ISR	Interrupt Service Routine

Abbreviation	Definition
SRS	Software Requirements Specification

5.2 Functionality

ID	*Description*
SRS-01	The temperature sensor shall operate and report values every 1 second.
SRS-02	The distance sensor shall operate and report values no longer than 1 second after an operation interrupt is received.
SRS-03	Upon non-trivial temperature change (≥ ±0.5°C), the display shall update within 1 second.
SRS-04	The audio processor shall store and play up to 4 pre-recorded 10-second audio files, with playback starting ≤1 second after interrupt trigger.
SRS-05	When the received temperature enters a new range (cold/normal/hot), the LED color shall change within 1 second.
SRS-06	The reminder timer shall be user-adjustable between 30–90 minutes through a simple switch or software parameter.
SRS-07	The system shall log daily total water intake (number of refills) and reset the count at midnight or on power-cycle.

6. Hardware Requirements Specification (HRS)

6.1 Definitions, Abbreviations

Abbreviation	Definition
MCU	Microcontroller Unit
ToF	Time-of-Flight sensor which measures distance using emitted light pulses.
I2C	Inter-Integrated Circuit. It is a serial communication protocol.
SPI	Serial Peripheral Interface. It also is a serial communication protocol.
dB	Decibel

6.2 Functionality

ID	Description
HRS-01	A ToF or ultrasonic sensor shall be used to measure the water level in the bottle. The sensor shall detect levels from 0 cm (empty) to 30 cm (full) with a resolution of less than or equal to 1 cm.
HRS-02	A temperature sensor shall measure water temperature in the range of 0 °C - 60 °C with an accuracy of +- 1 °C. Validation: It should be able to measure the temperature of water without being in contact with it.
HRS-03	A speaker shall emit an audible alert of at least 60 dB.
HRS-04	A bright display shall display water temperature and amount of water drank legibly under typical indoor lighting (atleast 500 lux).
HRS-05	ATMega328PB should operate within its specified voltage range, that is 3.3 V-5V. It should be able to send display signals to the display screen via I2C/SPI.
HRS-06	Power supply should be suffient to run the peripherals (temperature sensor, water level sensor, display screen, ATMega328PB) continuously without interruption for atleast 4 hours.
HRS-07	Water bottle should be made of metal and it should be able to withstand both hot water (up to 60°C) and cold water (till 0°C) without deformation or leakage.

7. Bill of Materials (BOM)

Our design relies on 7 major components: 2 sensors to gather data, 3 output peripherals and 2 processors.

Component	Descrtiption
Temperature sensor	In order to detect the temperature from the cap of the water bottle we use the Melexis IR temperature sensor. We are choosing this tyoe of sensor over others so that we can keep all of the components inside the cap avoiding wires in the bottle itself and limiting the screw cap functionality of the lid.
Distance sensor	In order to measure the water level in the bottle we will use a distance sensor not unlike the sensor we used in our thermin lab. The distnace sensor we choose is physcially smaller and can measure smaller changes in distance

Component	Descrtiption
	appropriate for the dimensions of the water bottle. The unit we have chosen also has low power idle mode that can be interrupted to take data, perfect for our application where we only need to take in data every 30 mins - 1 hour.
LCD screen	In order to display the temperature and amount of water drank in a day along with character expression we will be using an LCD screen. (the small LCD screen used in LAB 4).
LED strip	To less aggresively indicate the temperature of the water we will use a trip of LEDs that will change color between red and blue depending on the temperature.
Speaker	In order to output an audible sound volume at an audible frequency we use an 4 ohm speaker as suggested by our audio driver data sheet.
ATmega328PB	For all of the main processing we will use the familiar (to us) ATmega328PB because it has all the necessary I2C, SPI, uart, and GPIO functionalities.
Audio driver	In order to store and play audio outside a simple PWM buzz we need an audio processing unit.

Here we have the link to our specific BOM: BOM LINK

8. Final Demo Goals

The demonstation of our device will be simple, not requiring much space or complicated parameters. We will use prepared hot and cold water to demonstrate the temperature measurement functionality on the LCD and the LED strip. When hot water is in the bottle we will see the temperature displayed change and the LED color change. Whether by drink or pouring our the water into a differnet vessel, we will show that the "amount of water drank today" has also changed. Additionally, to demonstrate the reminder mechanism, we will shorten the period for reminders to ensure that a reminder goes off during the demonstration.

9. Sprint Planning

Milestone	Functionality Achieved	Distribution of Work
Sprint #1	Testing the sensors and actuators to check their functionality.	Catie - Audio driver, LCD and LEDs, Mira - Temperature sensor, Lubha - Distance sensor

Milestone	Functionality Achieved	Distribution of Work
Sprint #2	Successfully triggering correct outputs based on the inputs.	Catie, Mira and Lubha - ATMega3208PB software development
MVP Demo	Hardware and software of all electronic components working as intended.	Catie - Integration of the components, Mira and Lubha - Testing and validation
Final Demo	Fixing issues and any deviation from original, 3D printed system enclosure if time permits.	Catie - Hardware issues, Lubha - Software issues, Mira - 3D Printing

This is the end of the Project Proposal section. The remaining sections will be filled out based on the milestone schedule.

Sprint Review #1 (11.14.2025)

Last week's progress

Most of the week was spent on formalizing the big picture of the device:

1. A more detailed schematic with specific pins connections was drawn.
2. A code skeleton link describing the necessary definitions, initializations, ISRs and functions along with a first draft of the main loop was written.

More specific steps toward the final integratation were also taken such as:

3. Emotive reminder display graphics were drawn HERE.
   Demo was shown with the test code where the faces were shown in a loop (the test code is included HERE).
   Tested in video linked HERE.
4. Code for audio driver was written linked HERE.
   UART files for the audio driver HERE and HERE.
5. I2C iniitlaization from the ATmega was written link here.

Current state of project

Generally we are about halfway through the neccesary code for the project which gives us plenty to work on as we continue our wait for parts.

We have tested the LCD Display functions and can confirm they are working as seen in the video HERE.

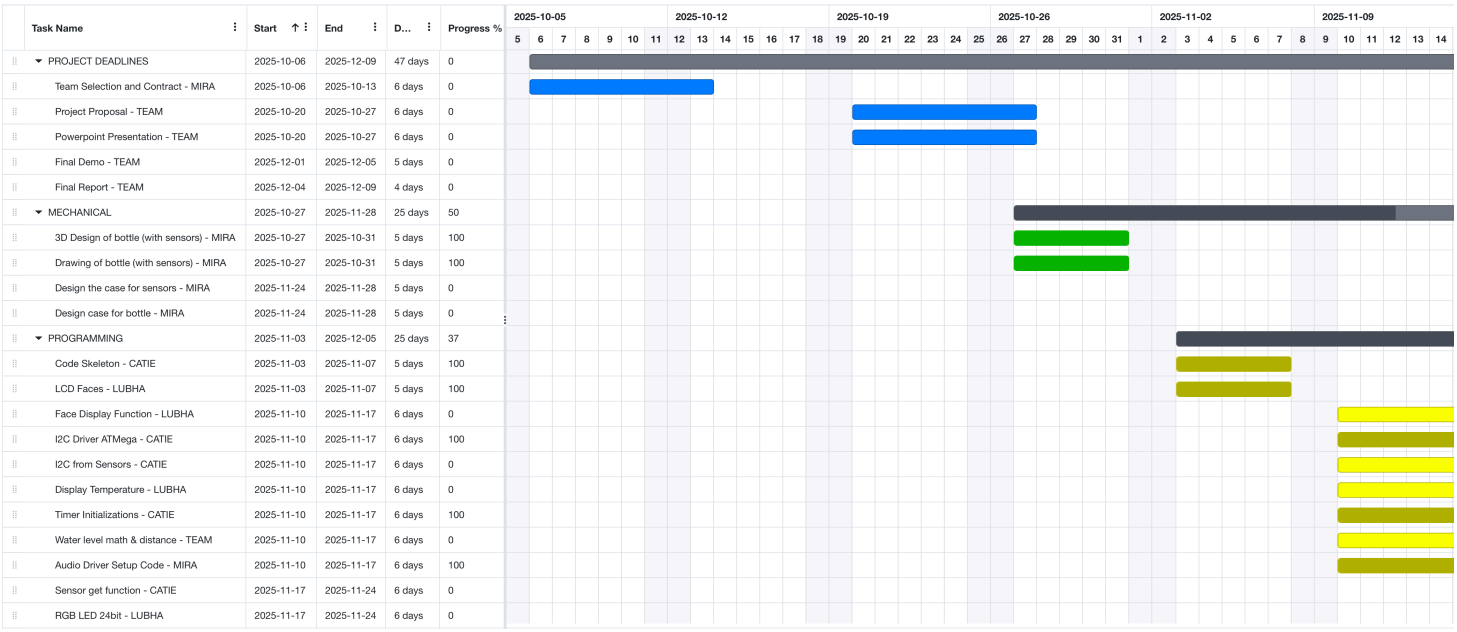
We have a good start on all the remaining code and are about to be ready to test the parts.

Next week's plan

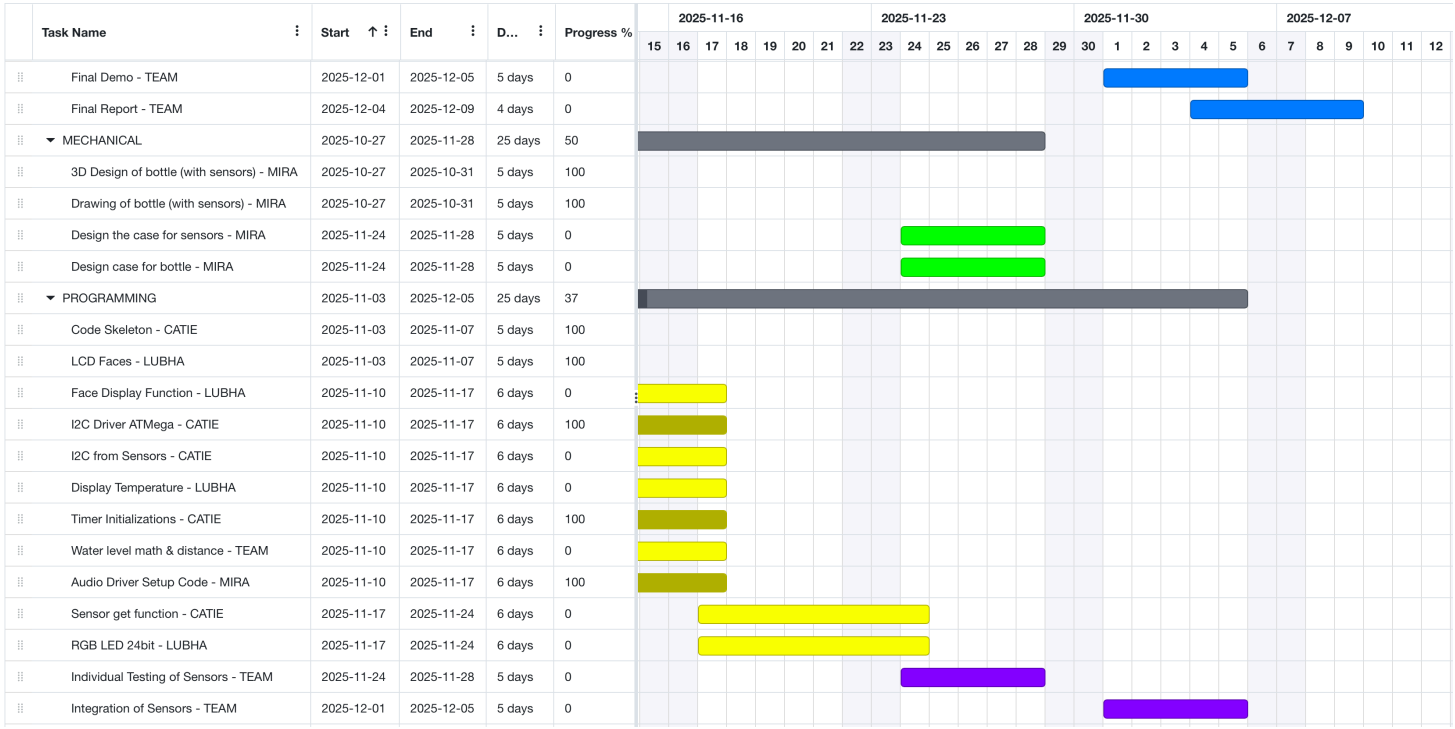
As we wait for our parts, our plans for next week involve finishing the code for the remaining parts, more specifically, writing the functions that will interpret the data from the sensors and the LED color changing output function (more complicated than it seemed at first).

While we do this (hopefully) our parts will come in and we will be able to test functionality with the freshly written code. As we start working with parts we'll also test and implement the power supply. To organize our plans we have created a Gantt chart documenting previous work and our plans for the necessary future work.

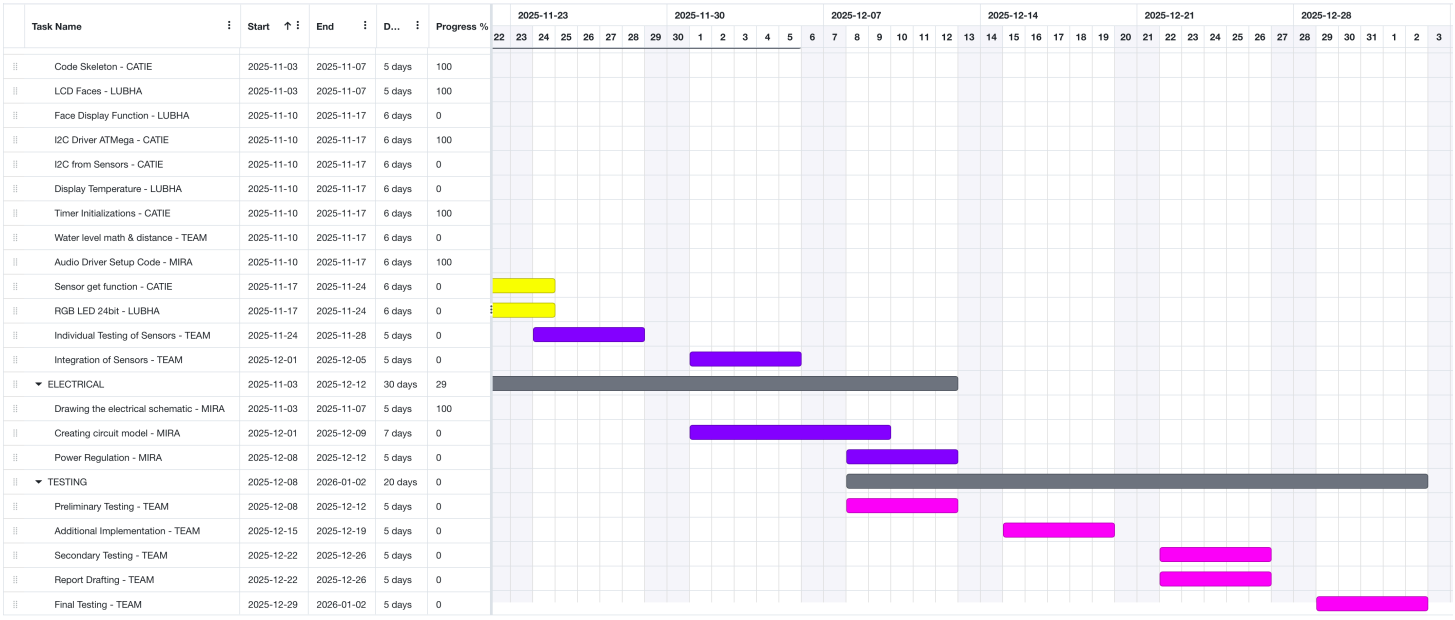
Created Gantt Chart



Task Name	Start	End	D...	Progress %
PROJECT DEADLINES	2025-10-06	2025-12-09	47 days	0
Team Selection and Contract - MIRA	2025-10-06	2025-10-13	6 days	0
Project Proposal - TEAM	2025-10-20	2025-10-27	6 days	0
Powerpoint Presentation - TEAM	2025-10-20	2025-10-27	6 days	0
Final Demo - TEAM	2025-12-01	2025-12-05	5 days	0
Final Report - TEAM	2025-12-04	2025-12-09	4 days	0
MECHANICAL	2025-10-27	2025-11-28	25 days	50
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawing of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Design the case for sensors - MIRA	2025-11-24	2025-11-28	5 days	0
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	0
PROGRAMMING	2025-11-03	2025-12-05	25 days	37
Code Skeleton - CATIE	2025-11-03	2025-11-07	5 days	100
LCD Faces - LUBHA	2025-11-03	2025-11-07	5 days	100
Face Display Function - LUBHA	2025-11-10	2025-11-17	6 days	0
I2C Driver ATMega - CATIE	2025-11-10	2025-11-17	6 days	100
I2C from Sensors - CATIE	2025-11-10	2025-11-17	6 days	0
Display Temperature - LUBHA	2025-11-10	2025-11-17	6 days	0
Timer Initializations - CATIE	2025-11-10	2025-11-17	6 days	100
Water level math & distance - TEAM	2025-11-10	2025-11-17	6 days	0
Audio Driver Setup Code - MIRA	2025-11-10	2025-11-17	6 days	100
Sensor get function - CATIE	2025-11-17	2025-11-24	6 days	0
RGB LED 24bit - LUBHA	2025-11-17	2025-11-24	6 days	0



Task Name	Start	End	D...	Progress %
Final Demo - TEAM	2025-12-01	2025-12-05	5 days	0
Final Report - TEAM	2025-12-04	2025-12-09	4 days	0
MECHANICAL	2025-10-27	2025-11-28	25 days	50
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawing of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Design the case for sensors - MIRA	2025-11-24	2025-11-28	5 days	0
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	0
PROGRAMMING	2025-11-03	2025-12-05	25 days	37
Code Skeleton - CATIE	2025-11-03	2025-11-07	5 days	100
LCD Faces - LUBHA	2025-11-03	2025-11-07	5 days	100
Face Display Function - LUBHA	2025-11-10	2025-11-17	6 days	0
I2C Driver ATMega - CATIE	2025-11-10	2025-11-17	6 days	100
I2C from Sensors - CATIE	2025-11-10	2025-11-17	6 days	0
Display Temperature - LUBHA	2025-11-10	2025-11-17	6 days	0
Timer Initializations - CATIE	2025-11-10	2025-11-17	6 days	100
Water level math & distance - TEAM	2025-11-10	2025-11-17	6 days	0
Audio Driver Setup Code - MIRA	2025-11-10	2025-11-17	6 days	100
Sensor get function - CATIE	2025-11-17	2025-11-24	6 days	0
RGB LED 24bit - LUBHA	2025-11-17	2025-11-24	6 days	0
Individual Testing of Sensors - TEAM	2025-11-24	2025-11-28	5 days	0
Integration of Sensors - TEAM	2025-12-01	2025-12-05	5 days	0



Task Name	Start	End	D...	Progress %
Code Skeleton - CATIE	2025-11-03	2025-11-07	5 days	100
LCD Faces - LUBHA	2025-11-03	2025-11-07	5 days	100
Face Display Function - LUBHA	2025-11-10	2025-11-17	6 days	0
I2C Driver ATMega - CATIE	2025-11-10	2025-11-17	6 days	100
I2C from Sensors - CATIE	2025-11-10	2025-11-17	6 days	0
Display Temperature - LUBHA	2025-11-10	2025-11-17	6 days	0
Timer Initializations - CATIE	2025-11-10	2025-11-17	6 days	100
Water level math & distance - TEAM	2025-11-10	2025-11-17	6 days	0
Audio Driver Setup Code - MIRA	2025-11-10	2025-11-17	6 days	100
Sensor get function - CATIE	2025-11-17	2025-11-24	6 days	0
RGB LED 24bit - LUBHA	2025-11-17	2025-11-24	6 days	0
Individual Testing of Sensors - TEAM	2025-11-24	2025-11-28	5 days	0
Integration of Sensors - TEAM	2025-12-01	2025-12-05	5 days	0
ELECTRICAL	2025-11-03	2025-12-12	30 days	29
Drawing the electrical schematic - MIRA	2025-11-03	2025-11-07	5 days	100
Creating circuit model - MIRA	2025-12-01	2025-12-09	7 days	0
Power Regulation - MIRA	2025-12-08	2025-12-12	5 days	0
TESTING	2025-12-08	2026-01-02	20 days	0
Preliminary Testing - TEAM	2025-12-08	2025-12-12	5 days	0
Additional Implementation - TEAM	2025-12-15	2025-12-19	5 days	0
Secondary Testing - TEAM	2025-12-22	2025-12-26	5 days	0
Report Drafting - TEAM	2025-12-22	2025-12-26	5 days	0
Final Testing - TEAM	2025-12-29	2026-01-02	5 days	0

Sprint Review #2 (11.21.2025)

Last week's progress

Temperature Sensor:

- We completed the code linked HERE and did first tests (using code linked HERE) on the I2C temperature sensor.
- Some timing details and temperature calculation adjustments are still needed to ensure accurate outputs every single time, but calibration values temperature measurements can all be read from

the sensor!
- Images of the correctly read out calibration constants are linked HERE.
- Image displaying expected delay between setting the start measurement bit and taking the measurements are linked HERE.
- Image of the ATMega328PB successfully reading all 18 registers necessary for calculating object temperture is linked HERE.

Audio Driver Implementation with Speaker and Amplifier:

- There were some road bumps we faced with integrating our intial speaker (Speakers & Transducers Speaker - 40mm Diameter - 4 Ohm 5 Watt) with the Mini audio driver. This was because our the MINI audio driver did not have an amplifier and this was not compatible with the speaker.
- The speaker model was changed to 'Adafruit STEMMA Speaker with the Audio Amplifier'. This was the right model compatible with the Mini audio driver.
- Initially the implementation was done with the UART setup so that lesser pins from the ATmega328PB is used. The code for this is linked HERE. This was getting a little heavy on time to debug which is when we realised GPIO setup would be much easier (and we did have enough spare pins on the ATmega too!).
- With the GPIO setup, we were able to successfully load and play four different files from the audio driver through a speaker. Each of these four files represented four different personalities (happy, angry, sad, sassy). The code used for this is linked HERE and the video displaying this implementation is linked HERE.

LCD display updates:

- Successfully updated the LCD display code to display temperature and water level values along with the personality expression.
- These faces are also included in a function that will change the face according to the input, ready for integration with timer.
- The temperature and water level values are hardcoded into the test code right now.
- The test code is attached HERE.
- A video of the demo is attached HERE.
- In the video, it can be seen that the temperature and total water drank values are being displayed beside the smiley faces.
- A push button is used to switch between the five faces. This is because the push button will be used as per the user input to change between the different modes in the final model also.

Water Level sensor updates:

- One less successful update is the reworking of how the water level implementation in our design will be.
- In starting the code for the I2C of the orginally planned on distance sensor, we found that the datasheet did not give the neccessary information for implementation (did not list the registers, necessary initializations, or what read data would look like).
- Our first backup plan was using the ultrasonic sensor, previously used for our theremin lab. Unfortunately it did not meet the necessary requirements either, as the wide field of sensing of the ultrasonic rays meant that the sensor would detect the sides of the water bottle before it detected the water.
- We then looked into the proximity sensor. However, that would not work either because it used infrared rays and it is meant to detect opaque or heavily coloured objects, while water is transparent.
- From here we started planning other methods of water level detection, both using ADC that we will test and decide on Saturday:

1. Homemade resistive water level sensor in the water. There are two options to implement this:

- The first method was to solder a series of multiple resistors (5-6 resistors) and the end of each resistor represents the water level.
- The second method was to connect each resistor to a transistor and arrange them parallel to each other. The wires from each transistor goes inside the bottle arranged in a line. The transistor gives signal out, providing the water level information.

2. Weight sensor outside the water bottle.

Current state of project

Currently, we have four major updates:

- Successful temperature sensor implementation.
- Successful audio driver implementation with speaker + amplifier.
- Improved LCD output functions with temperature and water level displayed, controlled by a pushbutton.
- Reworking of the waterlevel sensor model.

The updated electrical schematic can be seen below:
![alt text](Updated Schematic.png)

The updated Gantt Chart can be seen here:
![alt text](Updated gantt chart 1.png)

`![alt text](Updated gantt chart 2.png)

To sum up, we have 3/4 of our major components working, a plan for our final component of the water level sensor, and a start on the main function that integrates all of them (linked HERE).

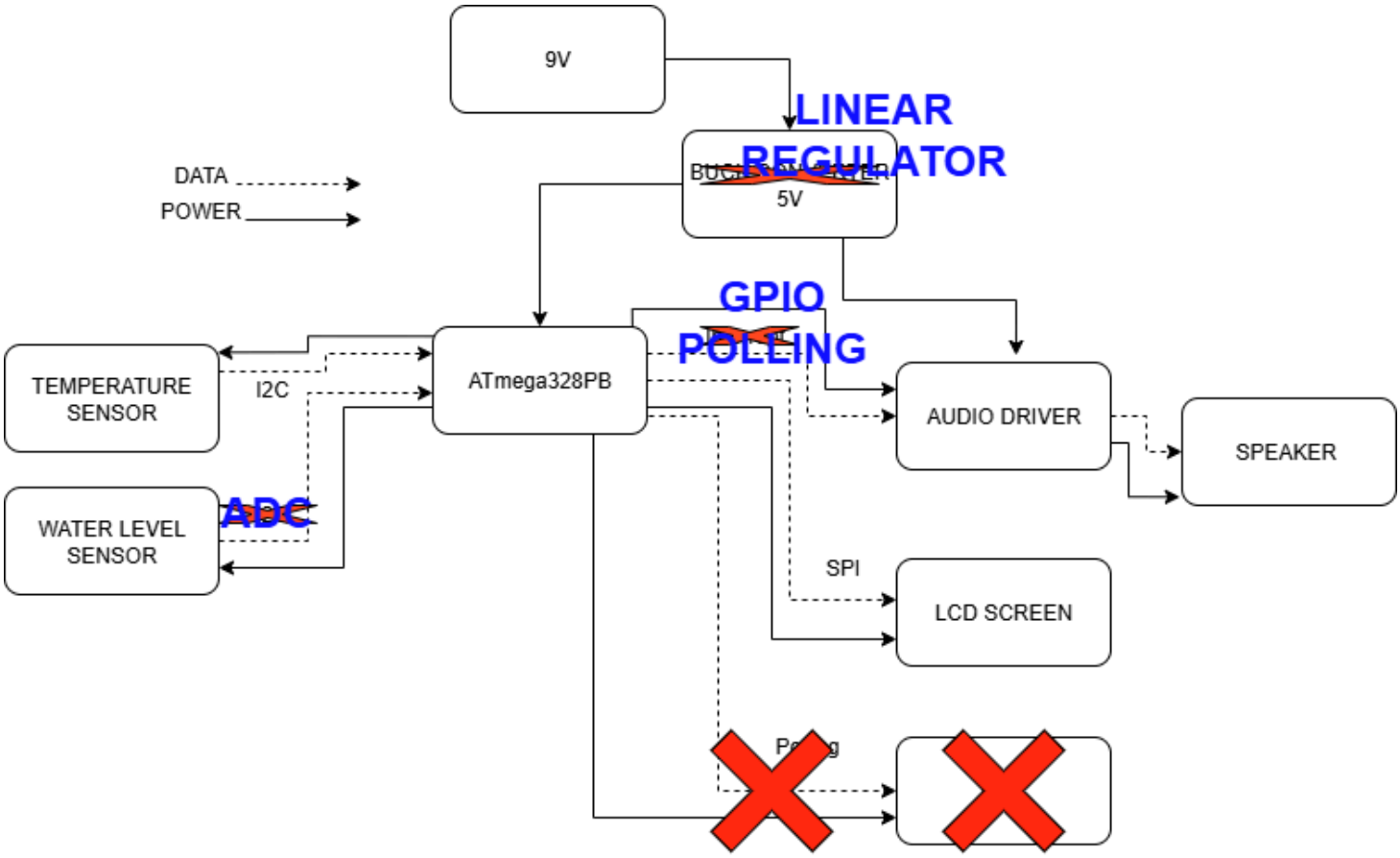
Next week's plan

For the next week, we will try to achieve the following milestones:

- Working on some improvements with certain individual components - Saturday (Catie)
- Working on power requirements - Saturday (Mira)
- Water level sensor implementation - Saturday (Catie and Mira)
- Integration of all sensors together - Saturday (Lubha)
- Testing all sensors together - Sunday (Catie, Mira and Lubha)
- Mechanical design specifications - Sunday (Mira)
- Getting ready for the MVP Demo!!

MVP Demo

1. Show a system block diagram & explain the hardware implementation.



From our proposal to now we had to adjust a few components.

- For water level sensor:
  Our original plan for our water level detection implementation (ToF distance sensor) turned out to be much more difficult to implement than not (no register description in its datasheet or user manual)
  Back up plan of the ultrasonic sensor from lab 3, was also unable to perform in our context.
  From here we tested multiple different options for sensing waterlevel before landing on stacking five 2 inch Analog output resisitive waterlevel sensors and making use of the on board ADC mux to sample the value from each of these sensors and adding up their returned ADC values.
- For audio driver:
  There was a change from our original plan is the comminication between the audio driver and the MCU.
  We had orginally planned on using the audio drivers UART capabilities to play the desired audio track but have sice switched to using GPIO signals for this functionality, for the sake of firmware simplicity.
- LEDs were eliminated for the sake of simplifying the design.
- Linear regulator is being used instead of a buck converter for power management.

Unchanged parts of our hardware include:

- An IR tmeperature sensor reading object temperature and communicating through I2C
- LCD screen displaying the input information received through I2C.
- Most of our components are able to run off of a 5V input that we get from the linear regulator. Some, however, require 3.3V, namely the temperature sensor and the analog water level sensors.
- However, separate power regualtion is not needed for these, since we are able to pull the 3.3V from the ATmega's 3.3V supply pins and the on board regulation included with the temperature sensor breakout board.

2. Explain your firmware implementation, including application logic and critical drivers you've written.

All our codes used for MVP demo are in the MVP_Demo folder.

- Our main function starts by initializing ATmega's methods of communication with its peripherals: initializing GPIO pins (for button input and audio dirver output), ADC functionality and I2C functions.
- Then, we initialize our peripherals themselves: the LCD through SPI and the temperature sensor through I2C (writing to its control register the value corresponding to our desired mode of operation).

- The application itself runs within the while loop, where the functions that capture and process data from our two inputs (gettemp and mlWL) runs and then the function that updates the screen to display these values is implemented.
- An "if statement" polls for a change from a button. It changes the personality displayed and audio played according to specified cases. The default state is a happy reminder. When the button is pushed the reminder becomes sad, then angry, then wink and then bored (idle state). There is no audio file in DND mode (bored).
- The trigger to play the audio file is by polling for a counter value that is updated by timer overflow interrupt. This last timer intergation is still to be done, but at the moment the other capabilities of our device our working with room for quality improvement through button debouncing, garbage rejection, and display update delay optimization. Right now our device is a water bottle that tells you the temperature and waterlevel of your water with an accommanying face of your choice while playing a corresponding hydration reminder once when the personality change occurs.

3. Demo your device.

Demo was done in person with our Account manager. All the components were integrated (LCD screen, push button switch, speaker and audio driver, temperature sensor and water level sensor), and the information was displayed on the LCD screen. This video, Demo video, shows the LCD screen displaying the temperature readings taken from the temperature sensor and the water level readings taken from the water level sensor. The speaker plays the audio file in the background and the switch is used to change the personalities.

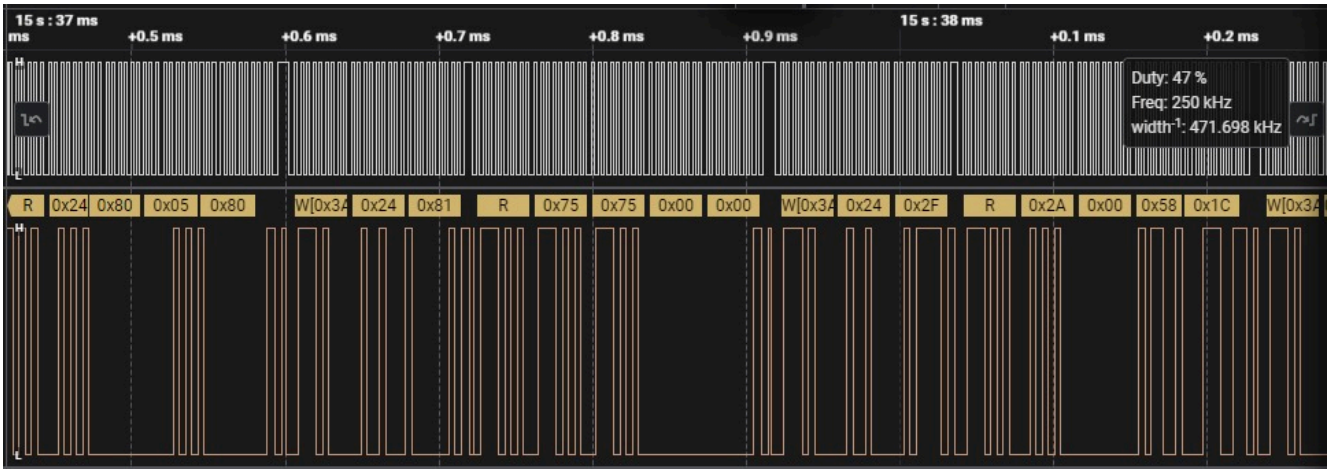
4. Have you achieved some or all of your Software Requirements Specification (SRS)?

We have acheived most of our Software Requirements Specifcations:

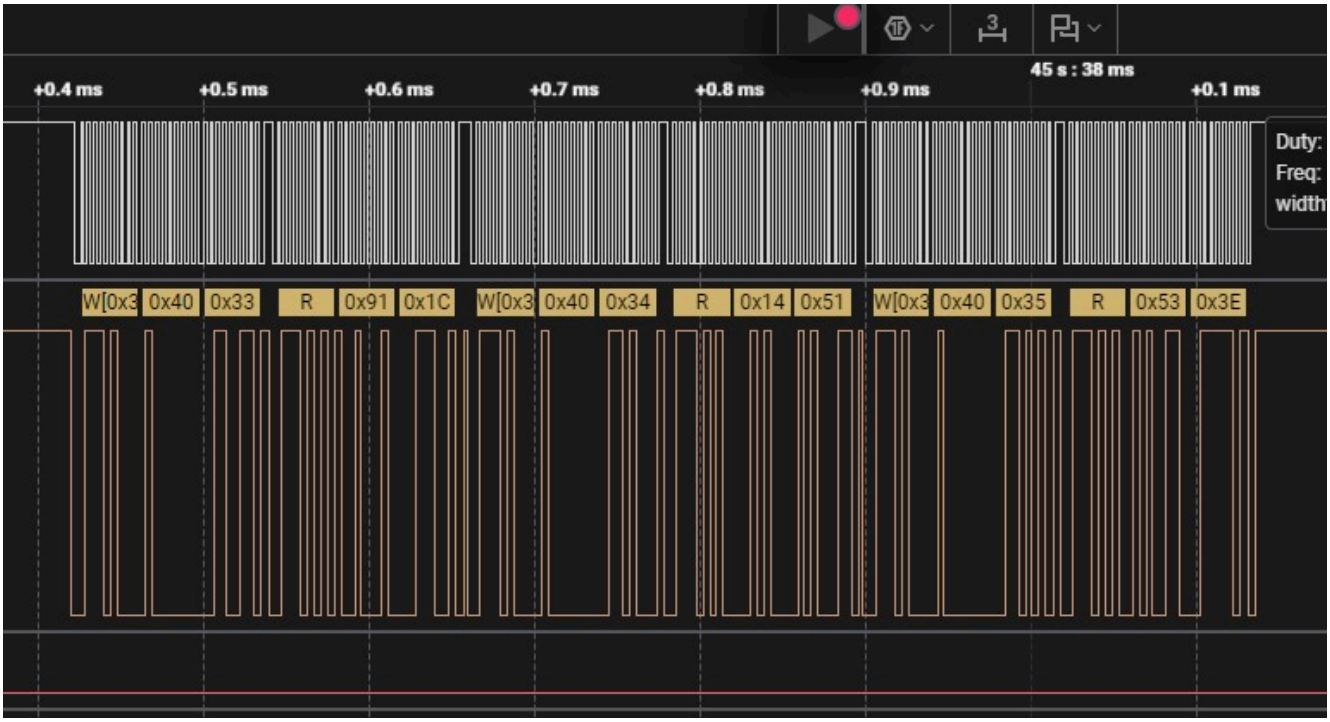
- SRS-01 -> The temperature sensor shall operate and report values every 1 second. It reports values in degrees celsius every 100ms. Every few measurements the sensor gives a bad reading, and thus returns garbage values. This worsens when the device is integrated with all the otehr components. This problem can be fixed by rejecting these bad reads through software implementation.
- SRS-02 -> The distance sensor shall operate and report values no longer than 1 second after an operation interrupt is received. A water level sensor was used to implement the same functionality of the distance sensor and reports values within 1 second due to the timing requirements for the ADC.
- SRS-03 -> Upon non-trivial temperature change (≥ ±0.5°C), the display shall update within 1 second. The display updates the screen within 1 second after the temperature or water level is reported. Delays longer than this can occur when pesonality is reset and the entire display must update.

- SRS-04 -> The audio processor shall store and play up to 4 pre-recorded, 10-second audio files, with playback starting ≤1 second after interrupt trigger. The audio processor stores 4 prerecorded audio files, but plays them back up to 2 seconds after the pin is set.
- SRS-05 -> Color changing LEDs have been removed from the scope of the project.
- SRS-06 -> The timer period is modifiable through software parameter.
- SRS-07 -> The system was supposed to log daily total water intake (number of refills) and reset the count at midnight or on power-cycle. It does not do this.

  a. Show how you collected data and the outcomes.
  - Readings of Temperature sensor (SRS-01):
    The code that is used to test the temperature sensor is linked Link Text
  - Below the reading of the caliberation constants can be seen:



  - Below the starting measurement and the reading from the appropriate registers can be seen:



  - And here we have the readings for one temperature measurement all happening in a timely manner:

- Readings from the LCD (SRS-03):

```
temp is 25.28
temp is 25.29
temp is 25.35
temp is 25.33
temp is 25.32
temp is 25.35
temp is 25.36
temp is 25.34
temp is 25.45
temp is 28.97
temp is 30.01
temp is 29.90
temp is 29.83
temp is 29.79
temp is 29.75
```

5. Have you achieved some or all of your Hardware Requirements Specification (HRS)?

- HRS-01 -> We used a water level sensor from Adafruit, linked here: Adafruit water level sensor. It detects the specified range of water levels. returning values with a precision of 1 ml and an accuracy of +-10 ml. This is reasonable for our applications because our "amount of water you should have drank in the last hour " is on the order of 100 ml
- HRS-02 -> Temperature sensor is measuring the temperature of the water and detects the specified temperature range, with precision of .1 degrees celsius, and accuracy varying depending on the temperature of the object.
- HRS-03 -> A speaker connected to an audio driver gives audio alert in four different personalities. The sounds are audible.
- HRS-04 -> The LCD display shows the water temperature, the current water level and the mode in which the bottle is in at the moment legibly under typical indoor lighting.

- HRS-05 -> ATmega328PB is working within its specified range. It is used to integrate all the sensors. I2C and SPI is implemented for serial communication.
- HRS-06 -> A battery of 9V is used with a linear regulator to give an output of 5V sufficient enough to run four peripherals continuously without interruption for hours.
- HRS-07 -> Since we couldn't find a metal water bottle with a big enough lid to fit all the sensors, we chose a plastic bottle. It is able to withstand both hot and cold water without deformation.

  a. Show how you collected data and the outcomes.

  To ensure the temperature sensor reported data form one temperature measurment in a timely manner we used the logic analyzer. Here we see the sets of register readings for temperature measurement come through in <100ms:

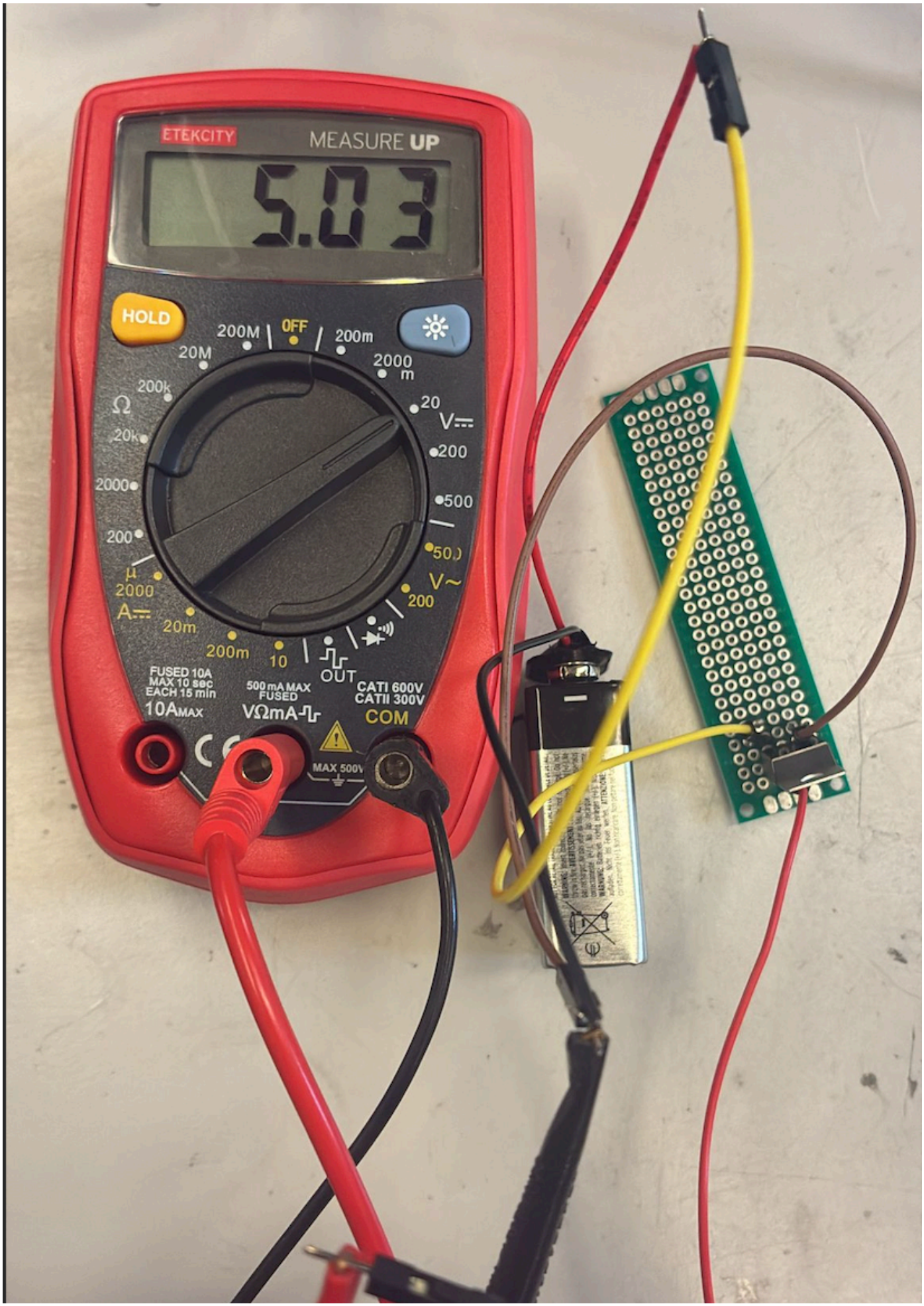
  To check that the system was able to receive and process information from both sensors, we used the code linked here LINK

  From this we found that the waterlevel and temperature were able to update as expected, the temperature rises when a warm object is palced in front of it while still reporting the current water level:

```
temp is 26.07
waterlevel in ml 43
temp is 27.33
waterlevel in ml 44
temp is 29.28
waterlevel in ml 44
temp is 30.15
waterlevel in ml 44
temp is 29.21
waterlevel in ml 44
temp is 28.24
waterlevel in ml 44
temp is 28.42
waterlevel in ml 45
temp is 29.59
waterlevel in ml 45
temp is 30.65
waterlevel in ml 45
temp is 26.74
```

Reading the voltage output from the linear regulator can be seen below (HRS-06):

Below a voltmeter is used to measure the voltage output caliberating power requirements:


ETEKCITY
MEASURE UP
5.03
HOLD
OFF
200M
200m
20M
2000
m
200k
20
Ω
20k
200
2000
500
200
50
μ
2000
V~
200
A
20m
200m
10
OUT
FUSED 10A
MAX 10 sec
EACH 15 min
10AMAX
500 mA MAX
FUSED
VΩmA
CATI 600V
CATII 300V
COM
MAX 500V

6. Show off the remaining elements that will make your project whole:

The final step for our project will be afixing all the parts to the waterbottle itself. Do to this we've design mechanical casework to enclose the parts.

The mechanical case work can be seen below:

Figure 1:

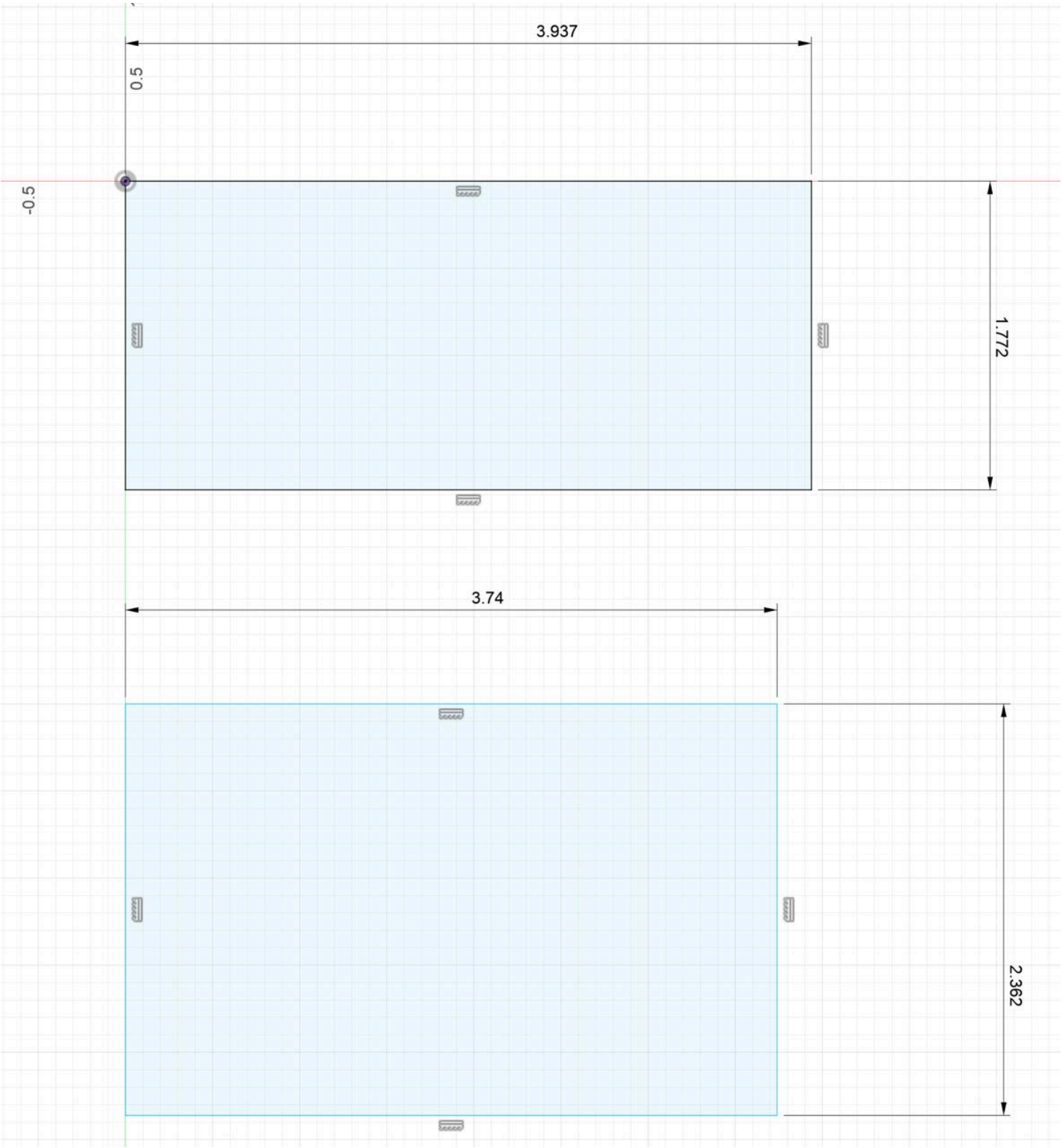


Figure 2:

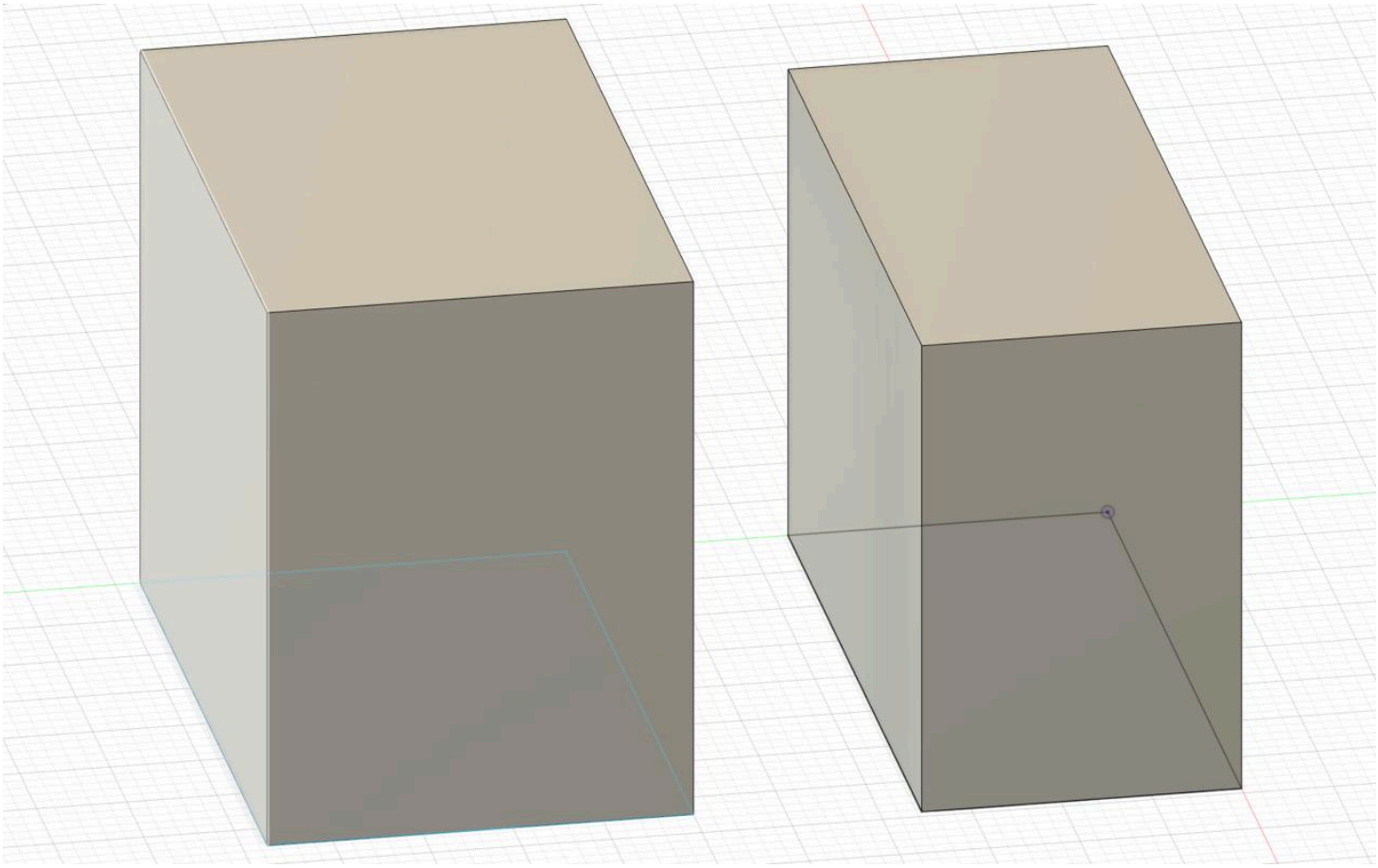
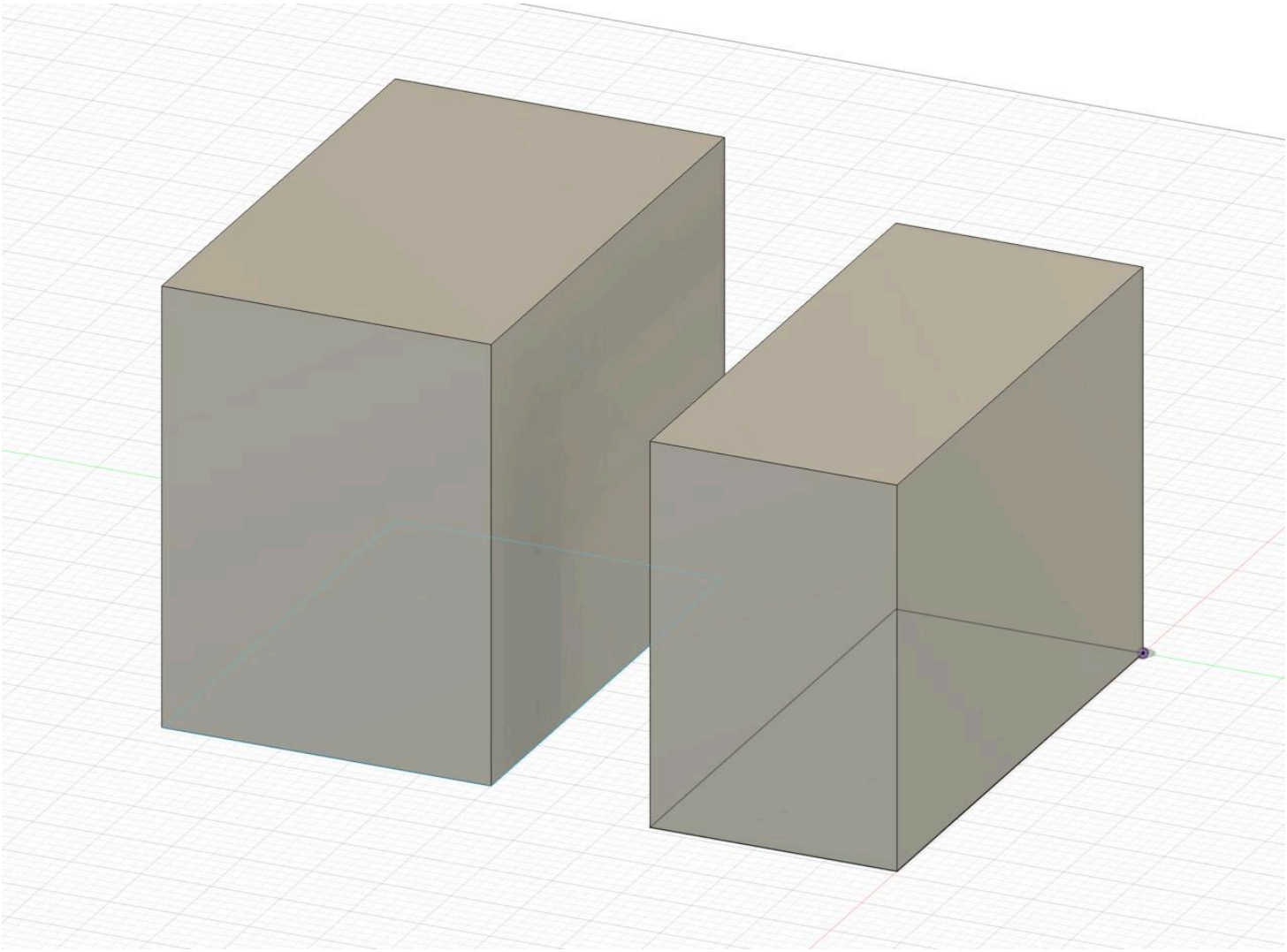


Figure 3:



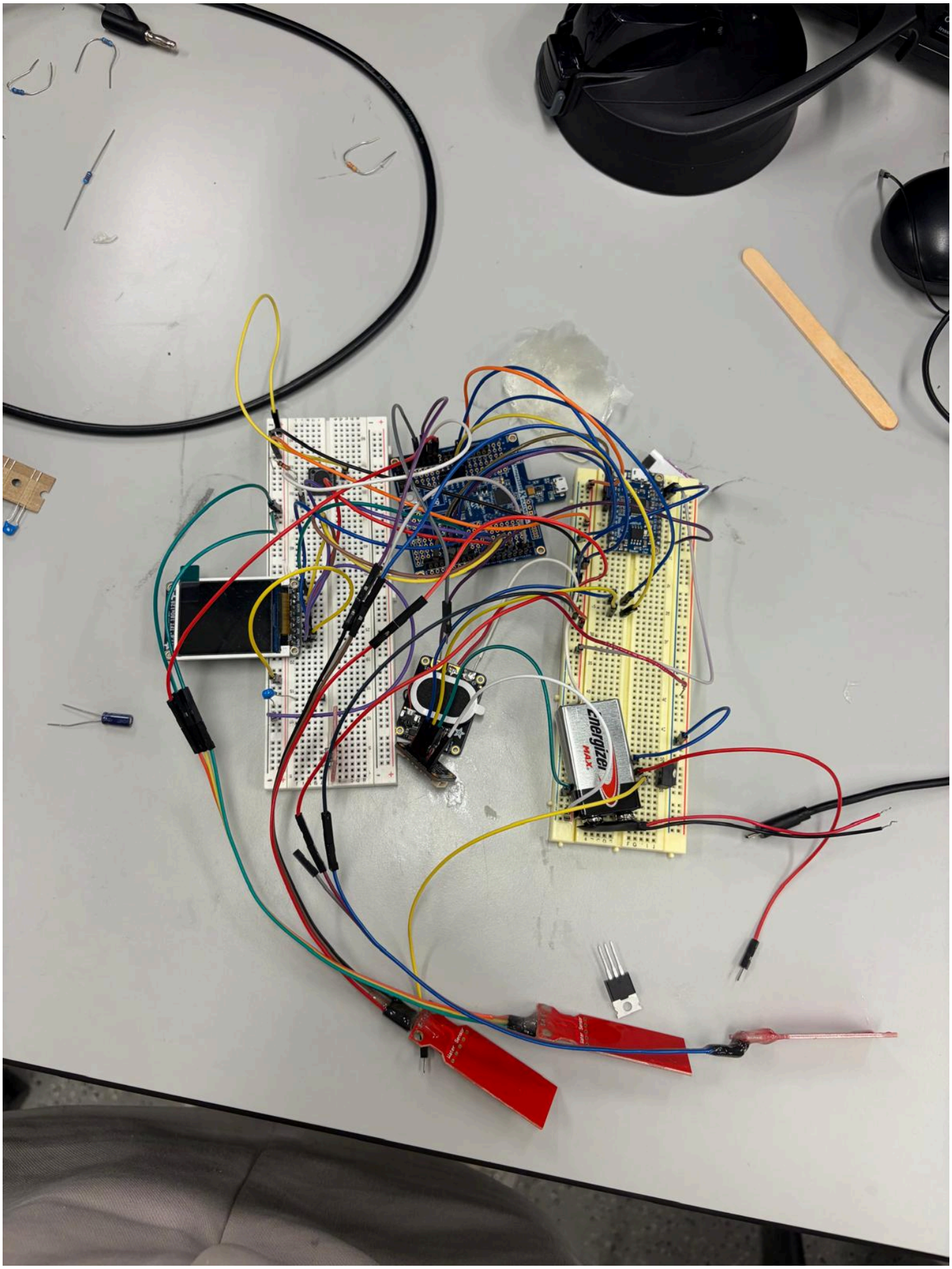
There are two versions for the mechanical design. Under Figure 1, the top design is expected to go above the lid, so that all the different components in the breadboard and microcontroller is neatly placed on top. However, if this does not work out then the design below it (in Figure 1), with slightly different measurements, will be placed along the handle. The second design is a back-up design plan. The 3D version from different angles can be seen in Figure 2 and Figure 3.

Once the other parts have been put together we could potentially go further integrating graphical user interface (GUI), web portal, etc.

7. What is the riskiest part remaining of your project?

Intergation has proved to be a tall mountain to cross. All of our different component require many wired connections, and so parasitics from using all of these wires at once has had a larger affect than anticipated (e.g. sensors that return accurate data when printed from the serial moniter, have more garbage readings when integrated). Additionally the quantity of wires leads to not-easily-noticed loose connections and physical limitations on the spacing of components.

Wires in circuit set up here:


energizer
MAX

Aside form the wires, our use of a linear regualtor to go from 9->5 volts leads to a lot of dissapated poer and hot compenents, risking damage to our power integration set up.

Enabling the necessary interrupts for the timing aspect of our device, seems to cause (not unexpected) timing issues for the rest of the components (blocking during the middle of an I2C read or an SPI transfer). So, intergating this seemingly simple final aspect could risk ruining the processes for other components.

a. How do you plan to de-risk this?
Rewiring with more intentional color-coding and placement, along with further securing connections between wires (with glue or electrical tape), should help limit the influence of parasitics and improve connections. Keeping the hot regulator as far form the other components as possible will also help lower the risk of damage.

8. What questions or help do you need from the teaching team?
How do interrupts affect serial communication?

Final Project Report

Don't forget to make the GitHub pages public website!
If you've never made a GitHub pages website before, you can follow this webpage (though, substitute your final project repository for the GitHub username one in the quickstart guide): https://docs.github.com/en/pages/quickstart

1. Video

[Insert final project video here]

- The video must demonstrate your key functionality.
- The video must be 5 minutes or less.
- Ensure your video link is accessible to the teaching team. Unlisted YouTube videos or Google Drive uploads with SEAS account access work well.
- Points will be removed if the audio quality is poor - say, if you filmed your video in a noisy electrical engineering lab.

2. Images

[Insert final project images here]

Include photos of your device from a few angles. If you have a casework, show both the exterior and interior (where the good EE bits are!).

3. Results

What were your results? Namely, what was the final solution/design to your problem?

3.1 Software Requirements Specification (SRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
SRS-01	The IMU 3-axis acceleration will be measured with 16-bit depth every 100 milliseconds +/-10 milliseconds.	Confirmed, logged output from the MCU is saved to "validation" folder in GitHub repository.

3.2 Hardware Requirements Specification (HRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
HRS-01	A distance sensor shall be used for obstacle detection. The sensor shall detect obstacles at a maximum distance of at least 10 cm.	Confirmed, sensed obstacles up to 15cm. Video in "validation" folder, shows tape measure and logged output to terminal.

4. Conclusion

Reflect on your project. Some questions to address:

- What did you learn from it?
- What went well?
- What accomplishments are you proud of?
- What did you learn/gain from this experience?
- Did you have to change your approach?
- What could have been done differently?
- Did you encounter obstacles that you didn't anticipate?
- What could be a next step for this project?

References

Fill in your references here as you work on your final project. Describe any libraries used here.