- **Team Name:** RobotGirlz
- **Team Members:** Mira Sivaprasad, Catie Robinson, Lubha Churiwala
- **Github Repository URL:** https://github.com/upenn-embedded/final-project-f25-f25-final_project_robotgirlz.git
- **Github Pages Website URL:** https://mira10-git.github.io/robotgirlz.github.io/
- **Description of Hardware:** ATmega328PB, MLX90632 FIR temperature sensor, LCD screen, Stemma speaker with audio amplifier, High sensitivity water level sensor, Audio FX Mini Sound Board, Linear Voltage regulator, Red and Blue LED lights

1. Abstract

This page is a brief of the working behind your 'new hydration buddy' - HYDROBOT. The bottle reminds the user to drink water if the water level in the bottle has not reduced every hour. It consists of an LCD which gives information such as water temperature and water level. The LCD has 5 different personality modes which can be switched by pressing the button. Along with the LCD, the bottle also has a speaker which gives the user auditory reminder for drinking water. The 5 different modes are: Happy, Sad, Angry, Sassy and Bored. If the user does not wish to have auditory interruptions, the bottle can be set to mode 5 (bored) which is the 'DO NOT DISTURB (DND)' mode. When the water temperature is below 30 °C, a blue LED lights up and when the water temperature is above 30°C, the red LED lights up.

2. Motivation

Everyone is so busy and occupied in their daily lives. Sometimes even the basic things such as drinking water or eating food on time are forgotten or ignored due to the strenuous schedule. For example, the day of a university student in UPenn includes - attending multiple classes + labs, completing back to back deadlines, sometimes working part-time, perhaps working on additional projects in GRASP lab, looking for summer jobs, trying to maintain a social life, taking up extra-curricular activities and so on. The list of milestones to complete are endless and time constraining. Due to the hustle of life, human beings forget to drink water -> get dehydrated -> loose energy to complete tasks -> fall sick -> create back logs and this cycle is hopelessly repetitive. Here's introducing **Hydrobot**, a bottle that reminds you to hydrate yourself and stay healthy!!!

Don't we all also have a preferred temperature to have our drinks in? Sometimes a little too cold or a little too hot than normal temperature? 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. Here's why we are adding the feature of temperature readings to **Hydrobot**, so that everyone can enjoy their drinks in the temperature they'd like and avoid mishaps such as burning your toungue. Even when you don’t need a reminder, **Hydrobot** is a friendly face monitoring the temperature of your water using infrared technology, so you know the temperature is just right!!!

HYDROBOT takes you through your own personalised hydration journey:)))

3. Goals

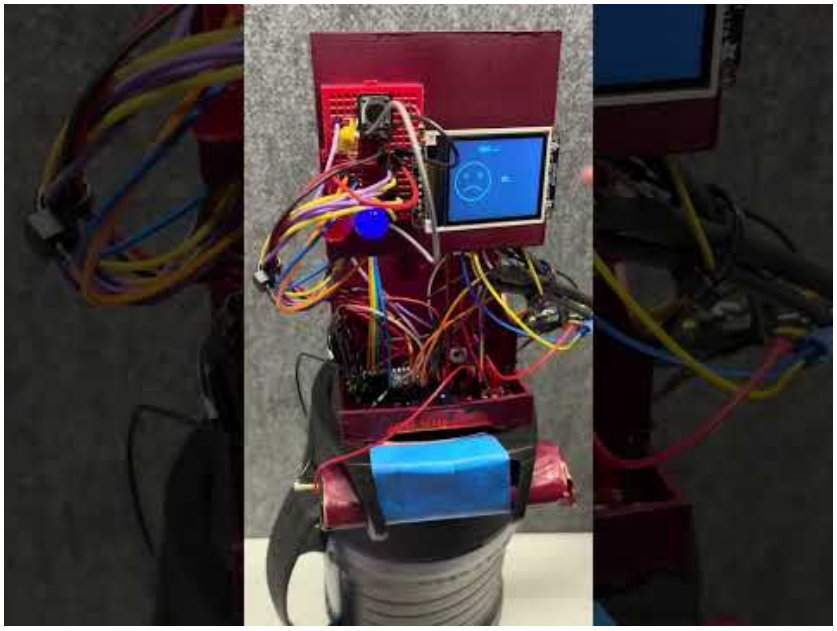
- Create an water bottle that would remind the user automatically to intake a significant amount of water every hour.
- The bottle gives the temperature read of the beverage inside, so that the consumers can have their preferred beverage in the exact temperature of their choice (hot/cold).
- Two LED lights to instantly know if the water is simply hot (red LED lights up) or cold (blue LED lights up).
- A speaker that talks in the voice of the user's personality choice and reminds them to hydrate themself.
- An LCD that tries to interact with the user by showing emojis, temperature value and how much water is left in the bottle.
- A switch that lets the user choose their preferred mood (happy, sad, angry, sassy or bored) and if they want the speaker to interact with them or not (so that the bottle does not create any inconvenience in public places).

4. Users

This water bottle can be used by users of all age and gender as drinking water is essential to existing and most people forget to hydrate themselves regularly. In addition to this, we have auditory aid from the speaker and visual aid from the LCD; this could further help differently abled people who are blind and deaf.

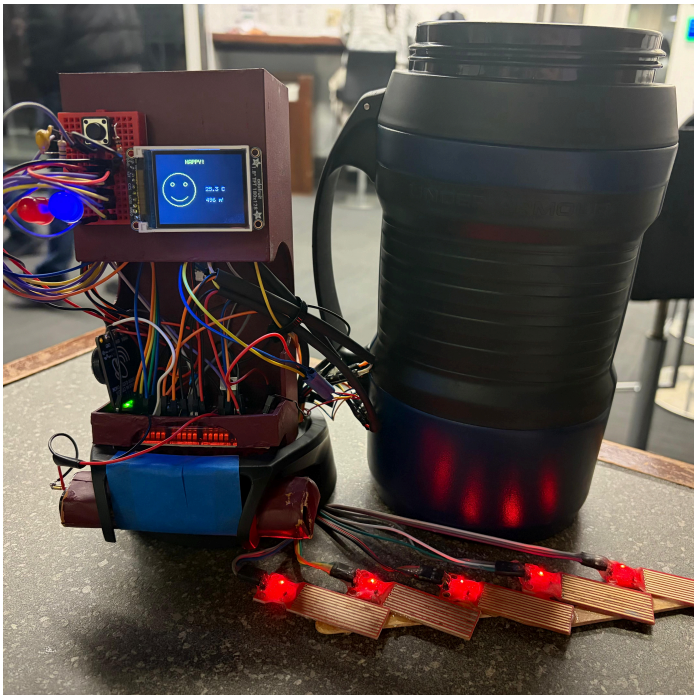
5. Video

Below is the video for the entire working of **HYDROBOT**

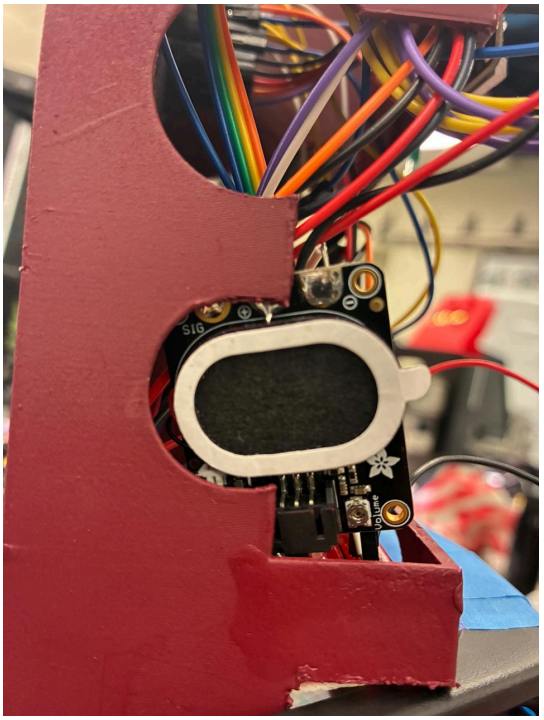


6. Images

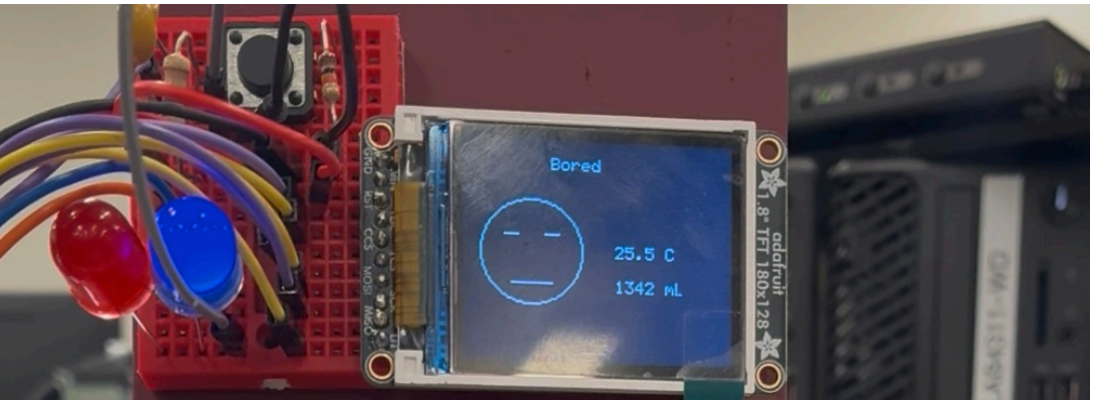
FINAL PRODUCT

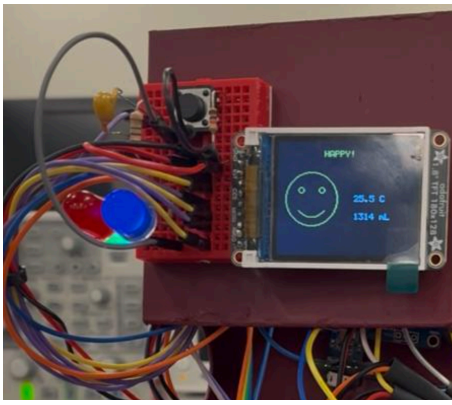


The speaker fit to the case can be seen below:

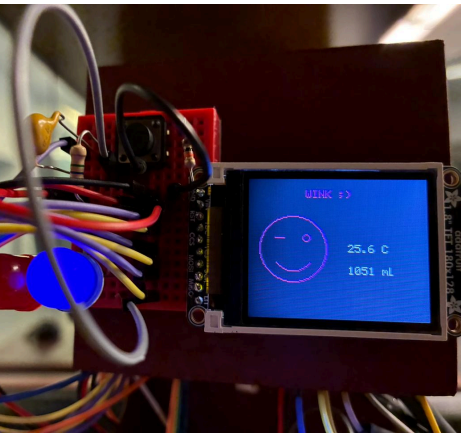


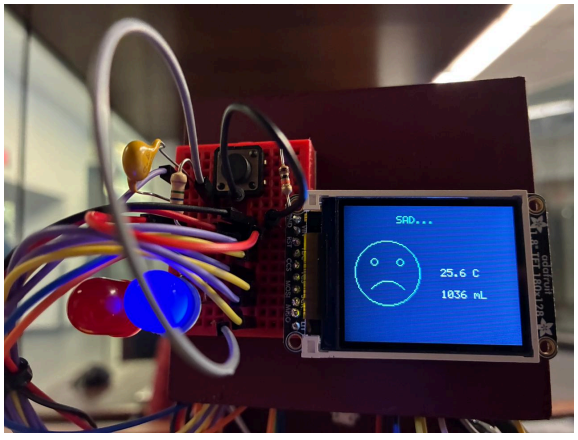
The LCD screen mounted on the case displaying 5 different modalities can be seen here: The below two images are bored (DND mode) and happy (default mode) -

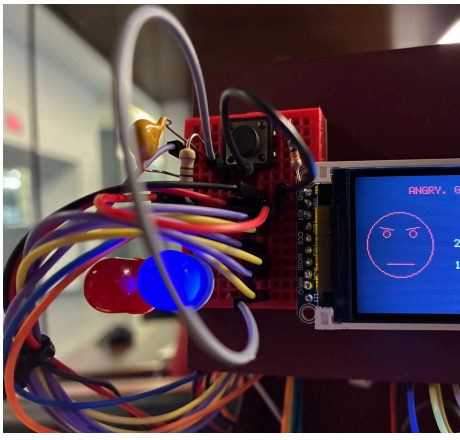




The below three images are the other addiitonal modalities -

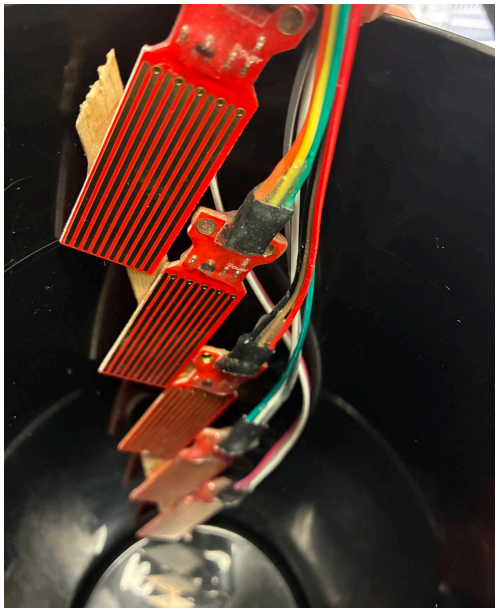


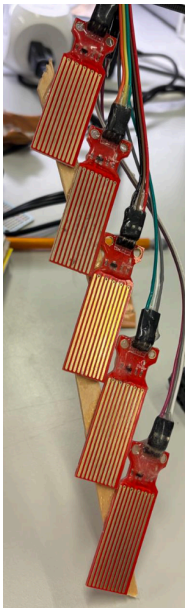




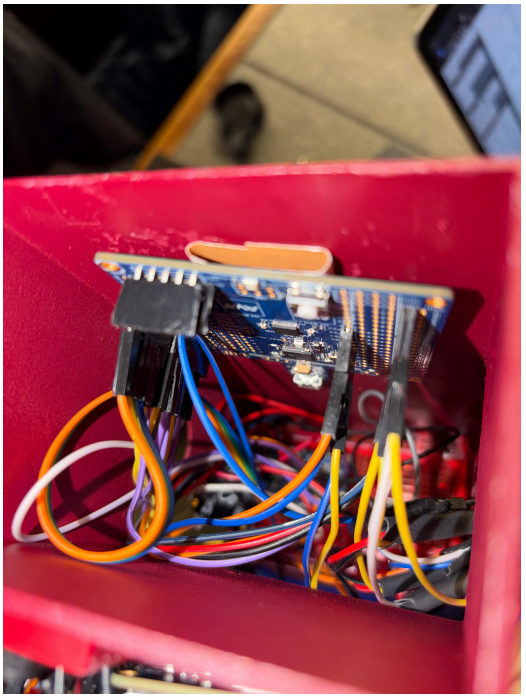
The temperature sensor placed inside the bottle cap can be viewed here:

The customised stacked up water level sensor can be seen below. The first image shows how it is placed inside the bottle and the second image gives a picture on the stacking of five different individual water level sensors to create a new long one:

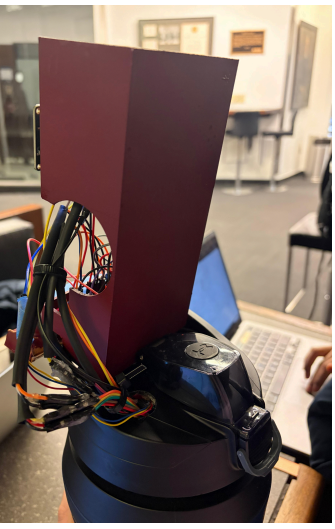
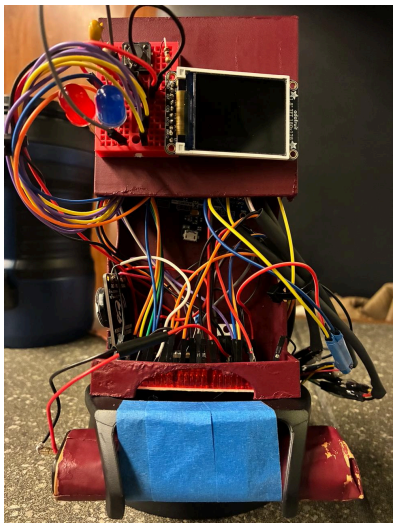
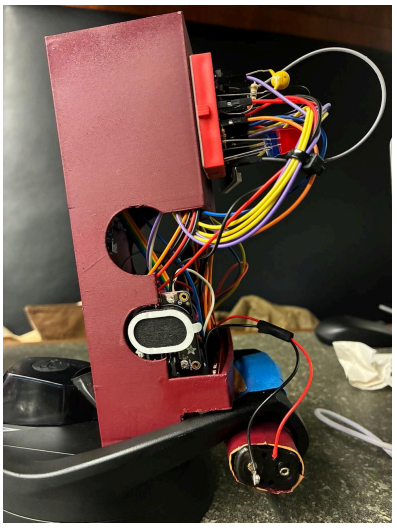
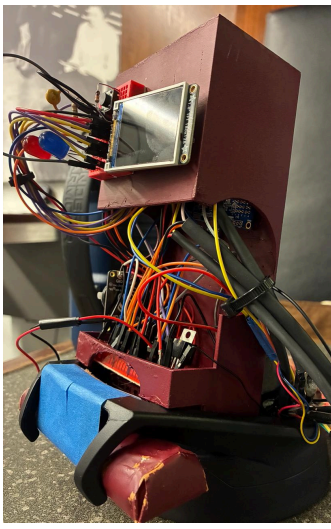




Below the interior of the mechanical casing can be seen. It has securely placed the ATmega328PB, the speaker and the breadboard along with all wirings. All the wires are color coded. They are also either tied together with heat shrink wraps, tape or zip ties:

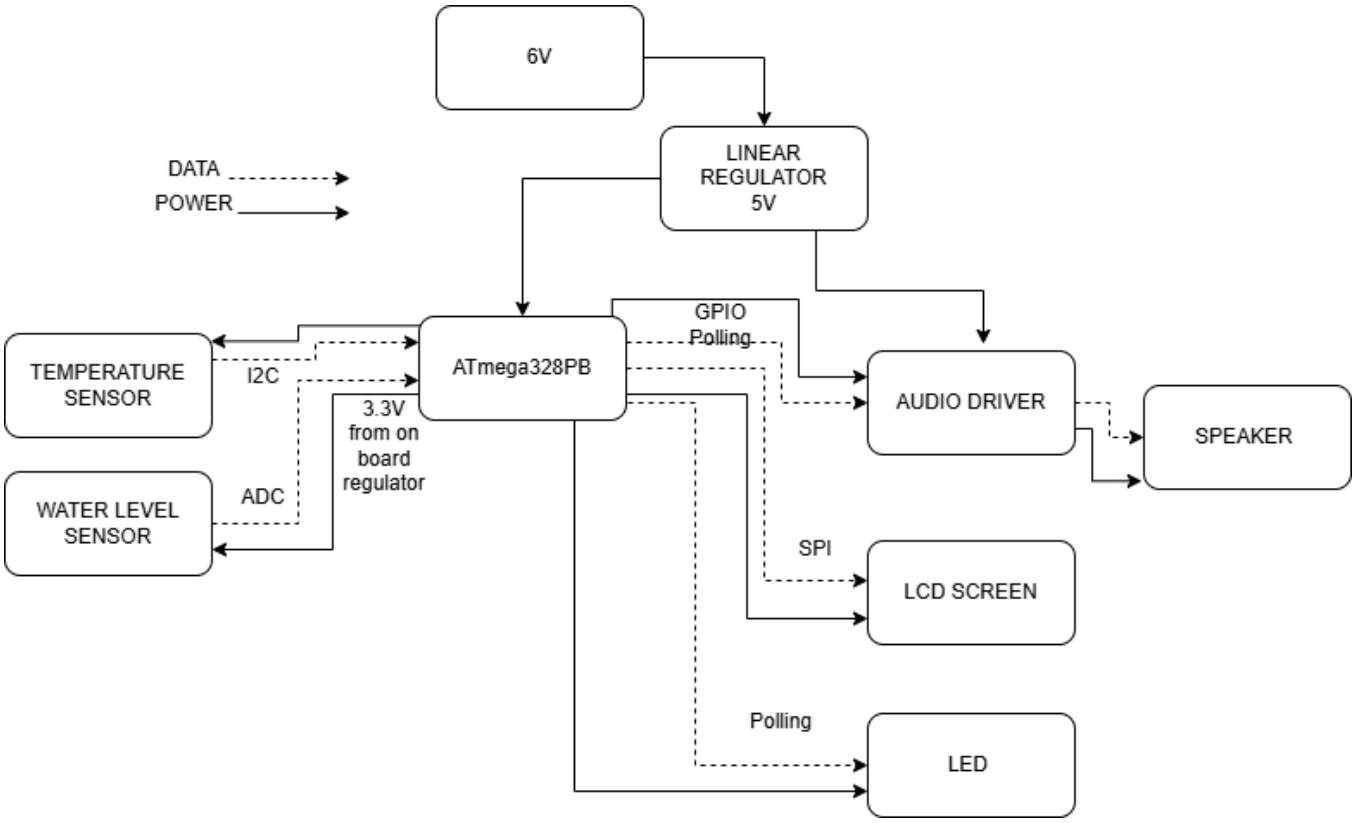

Below the exterior of the mechanical casing can be seen. The casing is spray painted with red color to match with the color of the breadboards used so that everything fits into a theme color:

Below the extended battery case taped up to the bottle can be viewed:

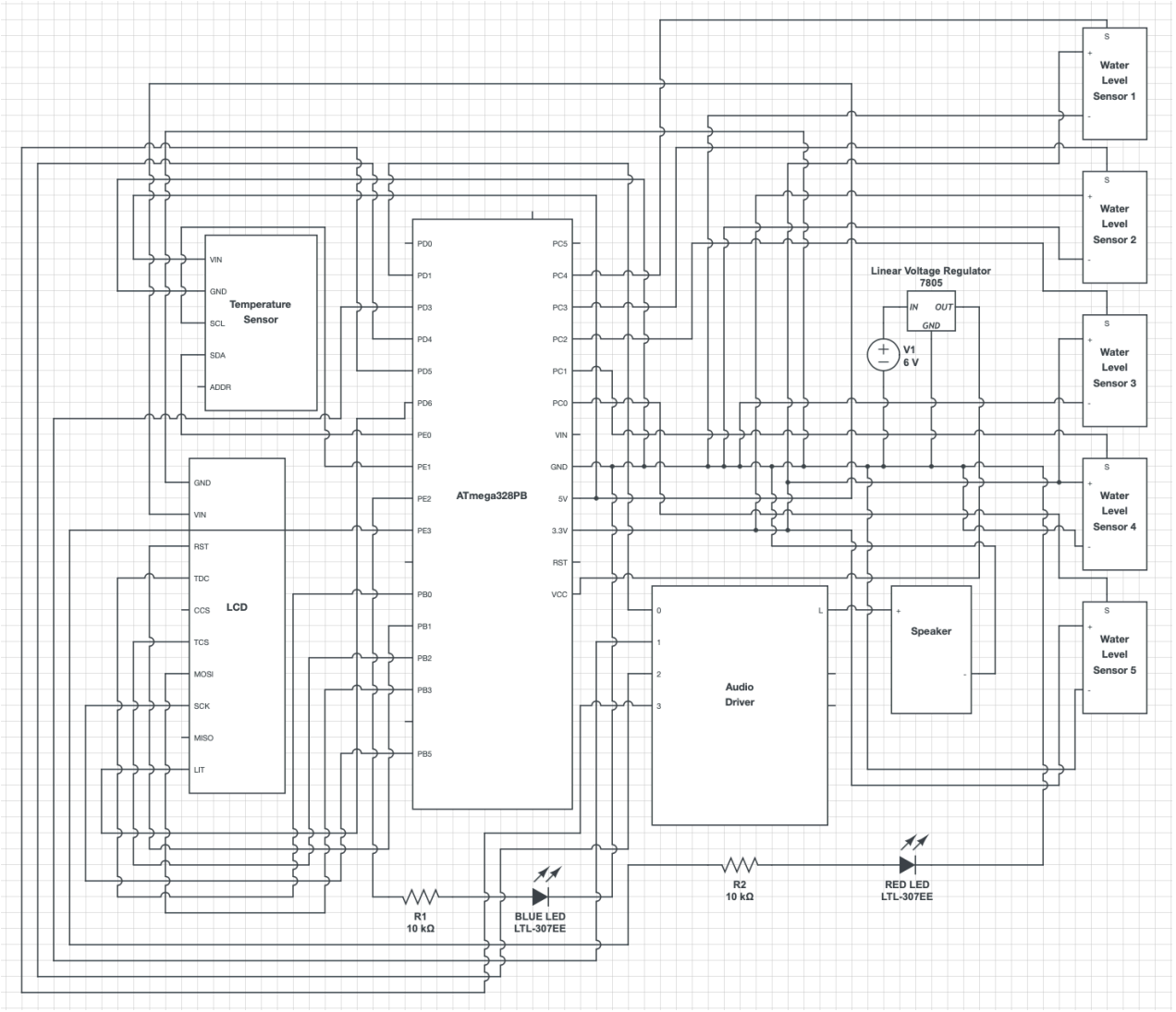
7. System Block Diagram

The image of the system block diagram can be seen below:



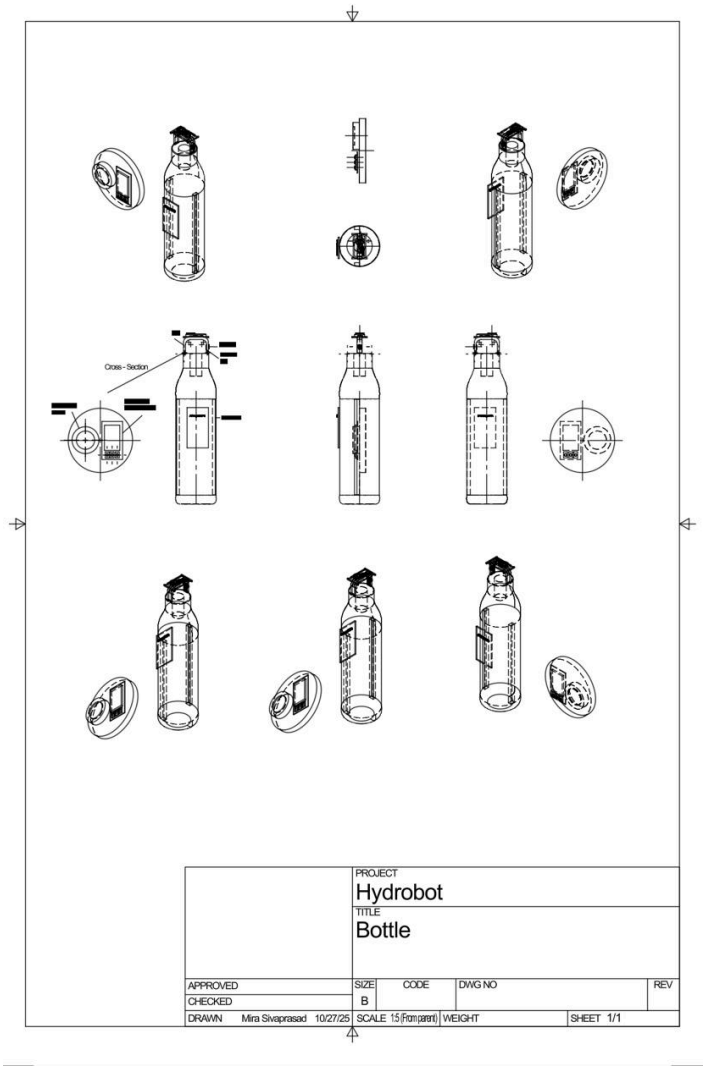
8. Electrical

Below is the diagram of the electrical schematic:

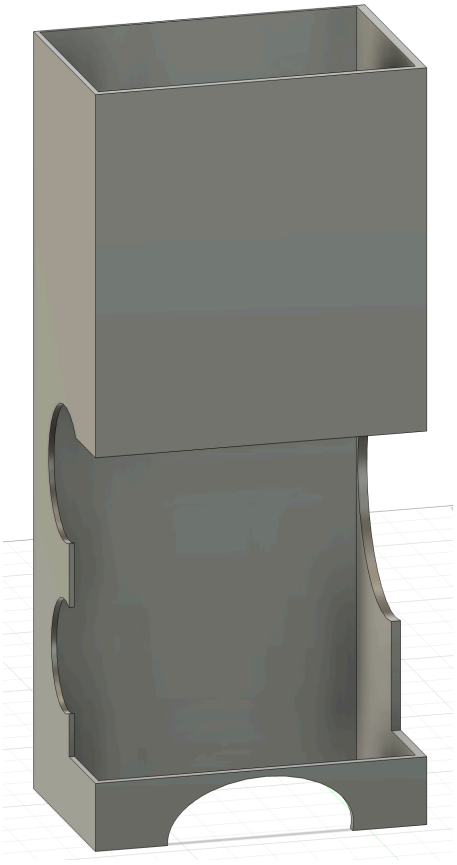


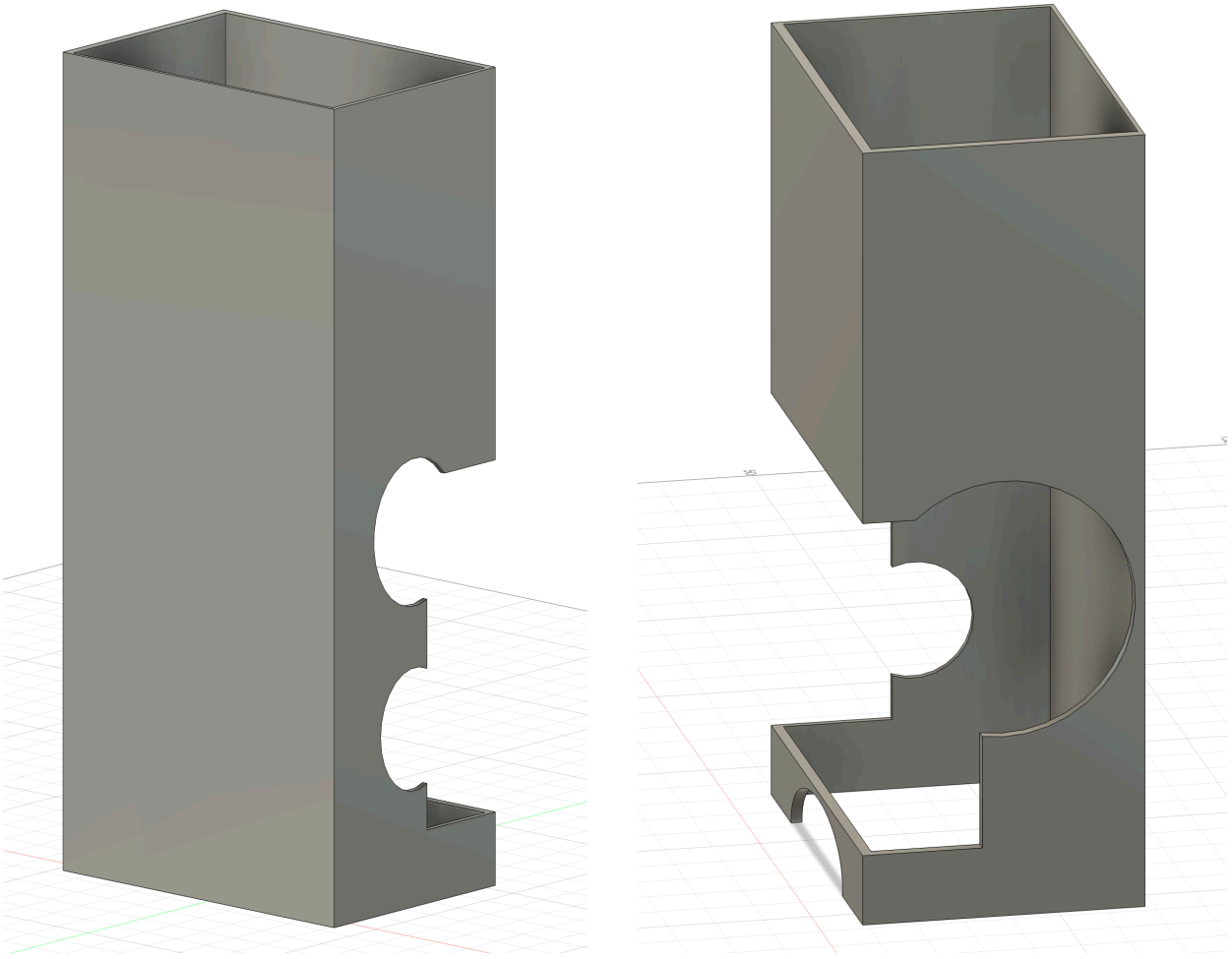
9. Mechanical

Below are the initial drawings of the bottle model. This initial design did undergo multiple changes before we finalized on the final design.

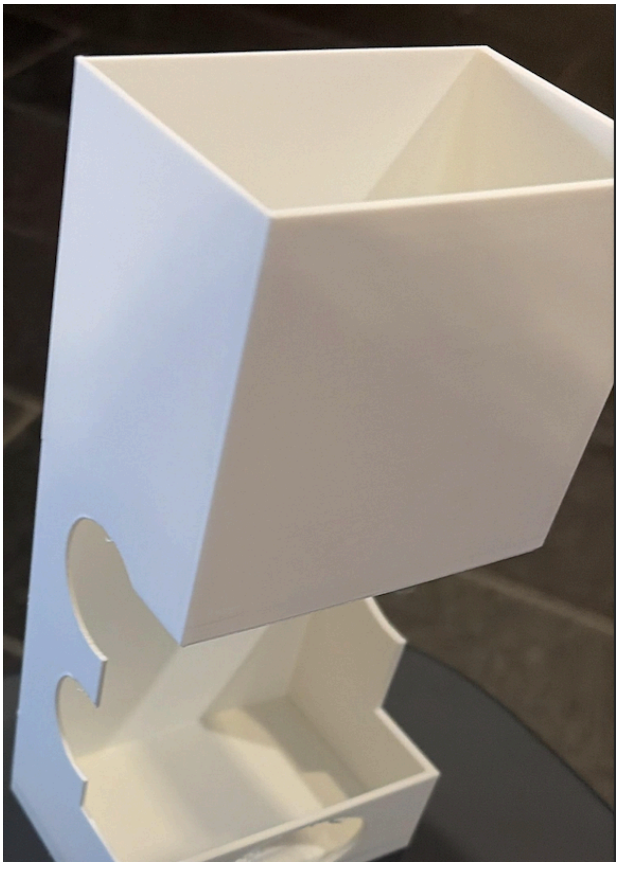


Below are the 3D designs created on Fusion 360 that were used for 3D printing from the RPL lab:

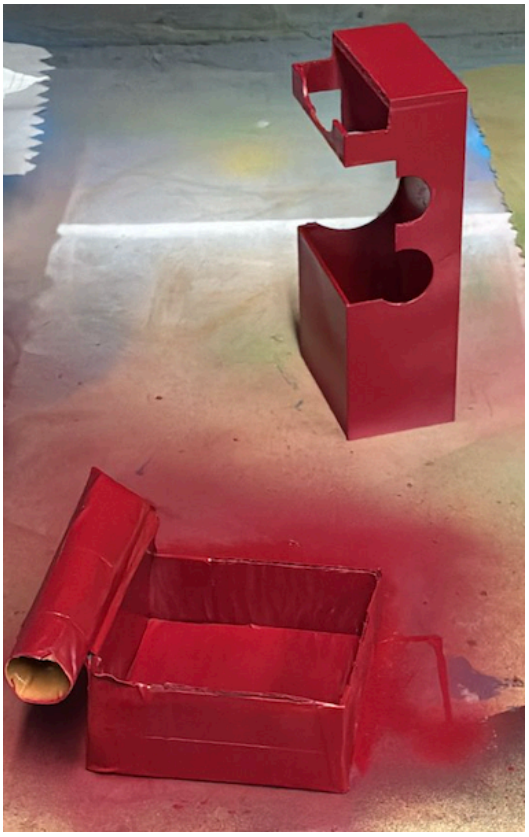




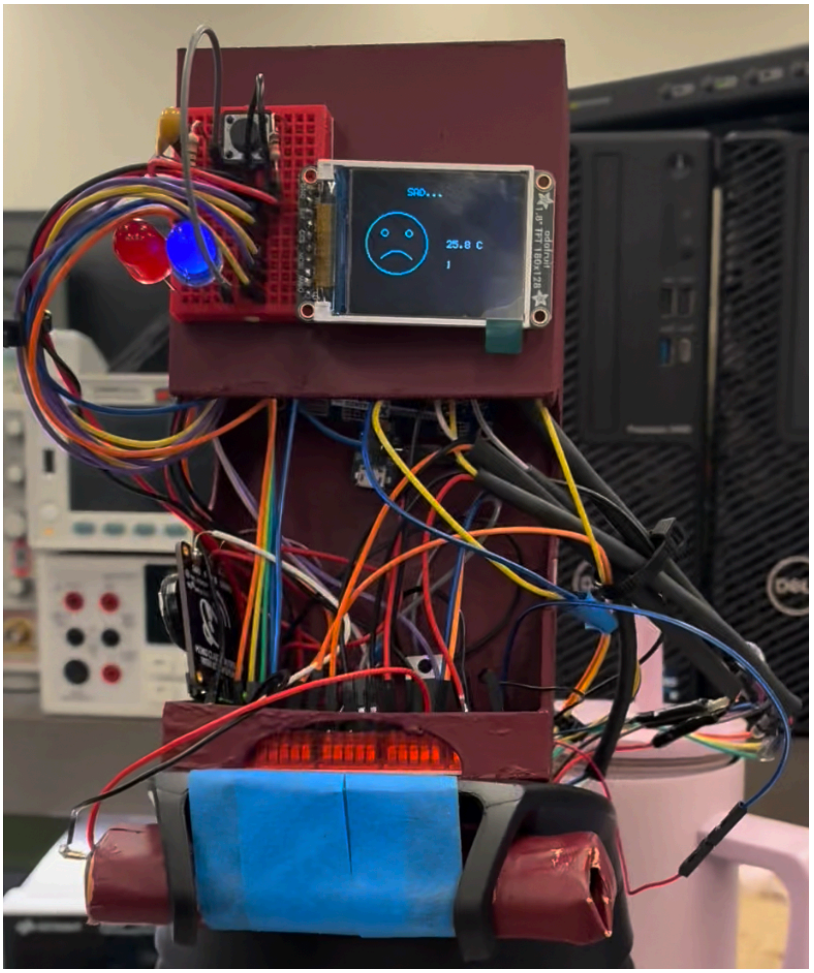
Below is an image of the 3D printed case before any changes were made to the model:



Before the case was spray painted, some design changes were made by cutting certain sections off the case (due to size constraints) from the Garage Lab using a Uni-point Radial Arm Saw. Below is an image of both the case for electrical components and battery case spray painted from the Venture Lab:



The below image shows the final 3D printed model mounted on top of the bottle comprising of all electrical components, breadboards and wirings inside of the print:



10. Project Complexity

Despite the seeming simplicity of a smart water bottle, this device was fairly complex to implement. It uses two inputs and two outputs, that all communicate with one another and the MCU in different ways. Two types of serial communication: I2C and SPI, were used along with the ATmega328PB's internal analog to digital converter. Additionally, an external audio processor to drive the speaker that communicates with the ATmega328PB via GPIO was used.

11. Challenges

The team encountered many obstacles that were not anticipated:

- Neopixel LED strips were not used for the final demo. This was due to the complexity of the LED strips communication proccess which would have taken too much time that was better spent improving other parts of the device.
- It was a challenge to integrate the temperature sensor with the rest of the electrical components. As it communicates through I2C, the lack of an acknowledgment from the temperature sensor would cause the entire code to hang causing difficulty in diagnosis .
- The speaker module used initially was not compatible with the audio driver.
- The buck-boost converter was faulty and the linear regulator heated up quickly due to the larger step down voltage value.
- The team went through three different set ups to measure water level sensor before a customised version started working. The individual sensors were glued together with epoxy. However, they kept coming off because of the different kinds of testing that were performed with it (as the prolonged exposure to warm/hot water softened the glue).
- There were multiple versions of the 3D design (that did not fit the purpose of the design) before the team finalised on a casing with apt form and function.

Due to the above challenges along with many other factors like time and feasibility, component compatability and information availability, the team did have to change the approach for multiple functionalities:

- The team changed from using neopixel LED strips to individual LEDs to indicate water temperature. The LEDs were much straightforward and easy to code copared to the LED strips.
- There was no particular fix for this other than debugging each component seperately.
- The speaker model was changed to one with an in-built amplifier to simplify the design instead of the two separate amplifiers required to boost the signal enough for the original 5W speaker.
- 6V battery were connected to a linear voltage converter to step down to 5V and this worked perfectly for the entire system.
- Most notably, our approach to sensing the water level completely changed from a distance based sensor to a stack of 2-inch resistive waterlevel sensors -
    1. This change was initialy due to our first chosen part (ToF distance sensor) having intentionally incomplete documentation, due to the part manufacturers wanting the user to only implement the device using their API library to guarantee quality. This made it nearly impossible to write the necessary I2C code for the device.

2. From here, the first back up plan was to use the ultrasonic sensor. This however did not work for the anticipated application; as the field of view of the ultrasonic sensor was too wide for the water bottle. It would sense the walls of the water bottle before the water.
3. After these, the team looked into how typical resistive water level sensors operated. These sensor options were avoided during the initial proposal because of their size (either being too big to fit in the initial bottle choice or too short for the length of the bottle). The multiple options looked into includes: a load sensor that communicated through I2C, two homemade water level sensor deisgns and two in-factory made water level sensor.
4. The final option was considered the best for this project since it had the most consistent behavior that could be implemented in the remaining time. It took more effort than anticipated, but eventually it worked such that it would return values reasonable for the set up.

12. Results

The final result is that a bottle that gives regular reminders to drink water every hour depending on the water level inside the bottle, while also displaying the temperature of the water inside was created. The firmware implmentation of these functionalities is available in the GitHub repository linked above. The details of device's features are as follows:

- Temperature sensor -
    1. The device consistently measures the temperature using an infrared object temperature sensor operating in its extended range, that communicates to the MCU through I2C.
    2. Additionally, when the temperature of the bottle is more than 30 degree celsius, a red LED would light up, indicating that the water is hot. Similarily, when the temperature of the water is less than or equal to 30 degree celcius, the blue LED would light up, signifying the water is colder in temperature.
- Water Level Sensor
    1. The device also consistently measures the water level through resistive analog waterlevel sensors that are read by the MCU using ADC with its internal 1.1 V reference voltage.
- LCD screen
    1. The raw data from each of the sensors was reinterpreted by the MCU to return values in degree celsius and mililiters respectively. These values are then displayed and periodically updated by an LCD screen that communicates to the MCU via SPI.
    2. Alongside the displayed values is a color coded emoji associated with a particular personality option.
- Personality modes and speaker
    1. Five different personality modes were created: Happy, Sad, Angry, Sassy and Bored.
    2. Additionally, a speaker with an in-built amplifier was used so that the reminder was not only visual but also auditory. These audios were customised in accordance to each personality modes and saved into the Audio FX Mini Sound Board.
    3. The last mode (bored) is a 'DO NOT DISTURB' mode which means the speaker will not be playing anything when in that mode. This mode can be taken advantage of when the user is

in public place and do not prefer to be disturbed with loud noices. These personality modes can be changed at a click of a button.

13. Specification Results

13.1 SOFTWARE REQUIREMENT SPECIFICATIONS

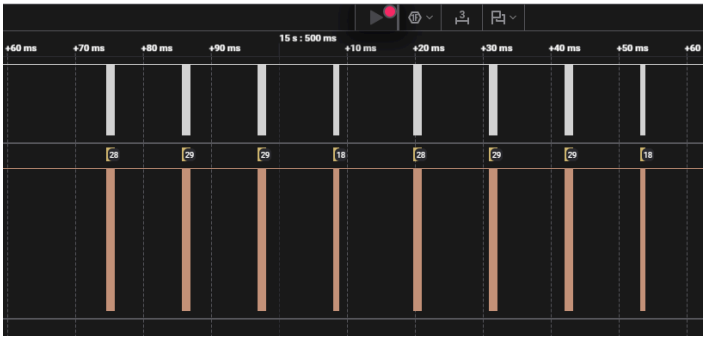
Most of the software requirements were achieved. Values were received from the waterlevel sensor and temperature sensor. The audio files were playing audible audio according to the display in the LCD which changed at every click of a button. There was an intial plan to diplay a string of color changing LEDs to show if the water was hot or cold. However, this was a tough software implementation as far as bare metal coding was concerned. It would have also added additional pressure on the memory use of ATmega328PB. So instead, a simplified logic of the same idea was deliberately used. Two individual blue and red LEDs were used that indicated temperature of the water. Only one of the software requirement was changed. From using a strip of LEDs that changes color to using two individual LEDs indicating red for hot water and blue for cold water. The reason this chnage was made was because it was much easier to implement code-wise and much efficient memory-wise.

ID	***Description***	**Validation Outcome**
SRS-01	The temperature sensor shall operate and report values every 1 second.	The IR temperature sensor starts and completes the 3 measurements necessary for an object temperature reading in < 100ms, as seen in the image below.
SRS-02	The water level sensor shall operate and report values no longer than 1 second after the opateration starts .	The analog water level sensors are continuously providing an input voltage to the ADC. Since the ATmega328PB's ADC is already enabled in single conversion mode after initialization mode, each ADC reading takes only takes 13 clock cycles, or 1.04ms from a 12.5kHz clock frequency (16Mhz with a 128 prescaler), making the enitre waterlevel reading of all 5 sensors procedure take 5.2ms
SRS-03	Upon non-trivial temperature change (≥ ±0.5°C), the display shall update within 1 second.	The LCD screen updates at least twice every time the processor cycles though our main while loop taking 3.17 seconds.
SRS-04	The audio processor shall store and play up to 4 pre-recorded 10-second audio files, with playback starting ≤1 second after trigger.	A signal comes through the speaker at 180ms after GPIO trigger, as seen in the image below.
SRS-05	When the received temperature enters a new	When the temperature changes the LED color updates on the next cylces of our while loop (the loop may take longer)

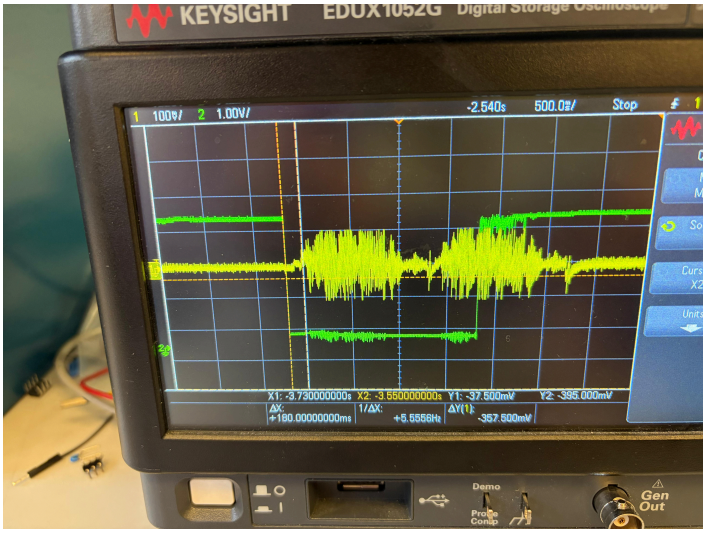
ID	Description	Validation Outcome
	range (cold/normal/hot), the LED color shall change within 1 second.	
SRS-06	The reminder timer shall be user-adjustable between 30–90 minutes through a simple switch or software parameter.	The timer period can be adjusted through the software parameter called ""PERIOD""
SRS-07	The system shall log daily total water intake (number of refills) and reset the count at midnight or on power-cycle.	The size of our water bottle made this functionality unreasonable. The prototype bottle was 1.9L and it seemed illogical to refill it.

SRS verifications:

SRS-01: Temperature timing image



SRS-04: Audio timing image



13.2 HARDWARE REQUIREMENT SPECIFICATIONS

Due to changes in our design we had to change a couple of our hardware requirements. Namely, the water level detection mechanism changed from a distance sensor configuration to a resistive waterlevel sensor, and so the hardware requirement was updated to reflect this. Most of the updated hardware requirements were achieved, with some inoptimalities from the sensors. The temperature sensor were wired, water level sensor customised, speaker connected to the audio diver, LCD screen connected to two different colored LEDs and a button. However, there were slight changes made in power regulation. Initially a single 9V battery was connected to a buck-boost convertor to step down voltage. However, the buck-boost convertor in the lab was faulty. This is when four 1.5V AA batteries that adds up to 6V were used. Them, a linear voltage regulator was used to step down the voltage to 5V (which was the highest voltage required to power all components). This was then connected to ATmega328PB from where 5V input was delivered to the temperature sensor, LCD screen and speaker attached to the audio driver. The 3.3V input was used to operate the water level sensor. This power setup was much more efficient and simpler than the earlier one. Another significant change made was to switch from ultrasonic/ToF distance sensor to water level sensor to measure the water level

ID	Description	Validation outcome
HRS-01	A water level sensor shall operate detect levels from 0 mL (empty) to 1900mL (full) with a resolution of 1mL.	A cascade of resistive waterlevel sensors (linked here: Adafruit water level sensor) were used and detected the water level from 0 mL to 1900 mL with an accuracy of +- 100mL, with higher quality reported values at higher volumes. This is suboptimal perfomance due to the varying floating charge that exists on the water level sensors that are not in use (wet) at a given time.
HRS-02	A temperature sensor shall measure water temperature in the range of 0 °C - 60 °C with an accuracy of +- 1 °C. It should be able to measure the temperature of water without being in contact with it.	The temperature sensor accurately measures the temeprature of the water from 0°C - 40 °C. At hot temperatures the steam in the bottle clouds the sensor, blocking it from reading the water itself.
HRS-03	A speaker shall emit an audible alert of at least 60 dB.	The reminder is audible! Video confirming this is linked below.
HRS-04	A bright display shall display water temperature and amount of water drank legibly under typical indoor lighting	The screen is bright and visible. The image confirming this can be seen below.
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 SPI.	The ATmega is able to operate this way via battery power

ID	Description	Validation outcome
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.	The system is left on for an afternoon and still operated normally throughout.
HRS-07	Water bottle should be able to withstand both hot water (up to 60°C) and cold water (till 0°C) without deformation or leakage.	The water bottle is plastic, but its still able to withstand the range of water temperature.

HRS verification

HRS-03 Reminder Audio The video showing the audio reminder is linked HERE

HRS-04 Display images:

14. Future Improvements

There could be multiple improvements to this project:

- The mechanical casing can be an enclosed box with a latch system. Such a design would encompass all components without anything been shown outside. This was not implemented in the prototype so that all internal setup could be clearly seen during the demo.
- We could soldered down the device on protoboards, to prevent loose wiring.
- A single water level sensor the length of the bottle could be used to give more accurate and consistent readings than the customised one that we have used in the prototype.
- The temperature readings works best for lower temperature than higher water temperature. Increasing the robustness of the code (taking and averaging entire measurement tables opposed to taking one reading at a time) and further waterproofing of the sensor set up itself would improve the performance of the temperature readings.
- The LED strips could be further implemented, as it does not only display two colors, but a range of red/blue could be shown as the temperature varied.

- A switch to power on the device could be implemented to avoid the delicate process of of plugging and unplugging wires every time the device is turned on.

15. Reflection and Conclusions

- **What did you learn from it?**
  From this project, the team got a wholistic view of the development process of an embedded system prototype. From planning out desired functions and building the bill of materials to testing individual components and final integration, this project expanded upon the course material of ESE 5190, giving one specific design where almost every lesson was implemented from scratch. Learning how to navigate multiple datasheets for desired information, writing code for I2C communication for a device, finding the necessary initializations for desired implementation, setting up a linear regulatior and putting all these pieces together affecting the entire system operation, were all necessary for the succesful operation of the final design. The team also gained experience with the arduous debugging process, isolating each component one by one. This is a very important skill for working in the industry as most of the time, components that work fine individually stop working during integration and so, finding the root cause of the issue is a very complex problem. The project gave us hands on experience with bare metal coding, in which some of our team members had little to no experience with before. The team learned individually about the intricacies of each main component: the temperature sensor, water level sensor, LCD screen, speaker, audio driver and how to integrate these together using ATmega328PB. The team also gained some experience in 3D designing, 3D printing and fabrication.

- **What went well?**
  Many aspects of our project went quite well. Most notably the speaker with an in-built amplifier attached to the audio driver worked very well. There were no complications during any of the testing sessions and the audio were quite audible, clearly differentiating between four different personality mode voices. Additionally, though it was the first part to be affected by integration bugs, the LCD screen consistently displayed the information reported from the ATmega328PB, giving a clear place to look to see if there was an issue. If the code was hung up, the LCD screen froze. If a sensor wasn't working right, the garbage value could be seen on display or if a button was not debouncing properly the effect could be seen directly. This really helped with the debugging process.

- **What accomplishments are you proud of?**
  The team was particularily proud of implementing the temperature sensor. Since it communicated through I2C, it required a lot of firmware implementation specific to the part itself. It took a lot of time to research, write code for as well as test the necessary initializations, calibrations and measurement start parameters, along with the firmware required to turn the raw data from the sensor into information that was interpretable. The temperature sensor took the most time to intergate with other parts, because the lack of an expected acknowledgement from the peripheral would hang up the entire application function. It was very rewarding when the team figured what was going on and got everything to work (code got hung up waiting for an ACK if the I2C peripherals were not plugged in). Aside from the temperature sensor, finally putting everything

together was very exciting even if it did take significant time and effort. The working of the entire project felt worth the effort!!!

- **What did you learn/gain from this experience?**
  In addition to the general practice, the team learned how executing a project like this takes a lot of time, trial and error, testing, adapting and overcoming the inevitable obstacles along the way. The team learned to approach the same problem in multiple ways, preservering through seemingly dead-end situations and improvising when needed. These skills can be applied not only to any embedded systems project, but also to any project the team may work on in the future.

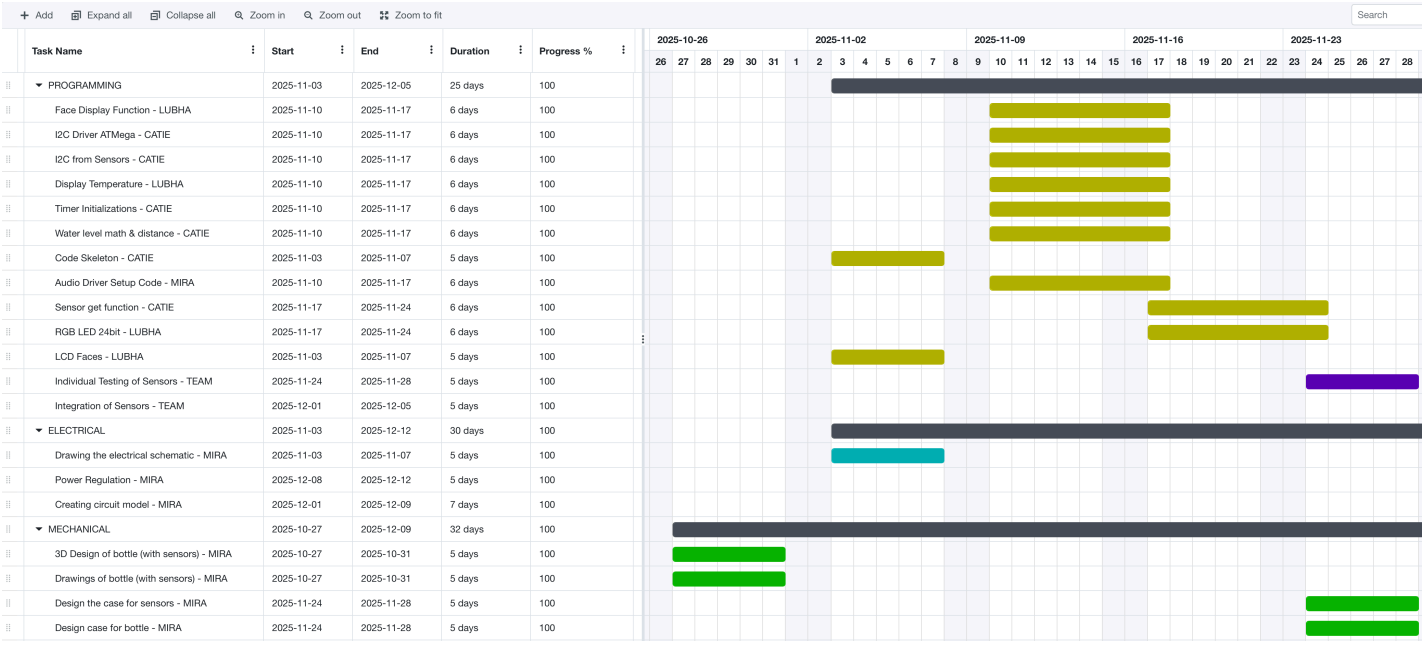
16. Project Proposal Presentation

Below is the link used for the Project Proposal Presentation:
https://docs.google.com/presentation/d/1BS_oT94HW1l447_tcnxJzq2WIQPKSj7Fw4KtnmdRAFw/edit?usp=sharing

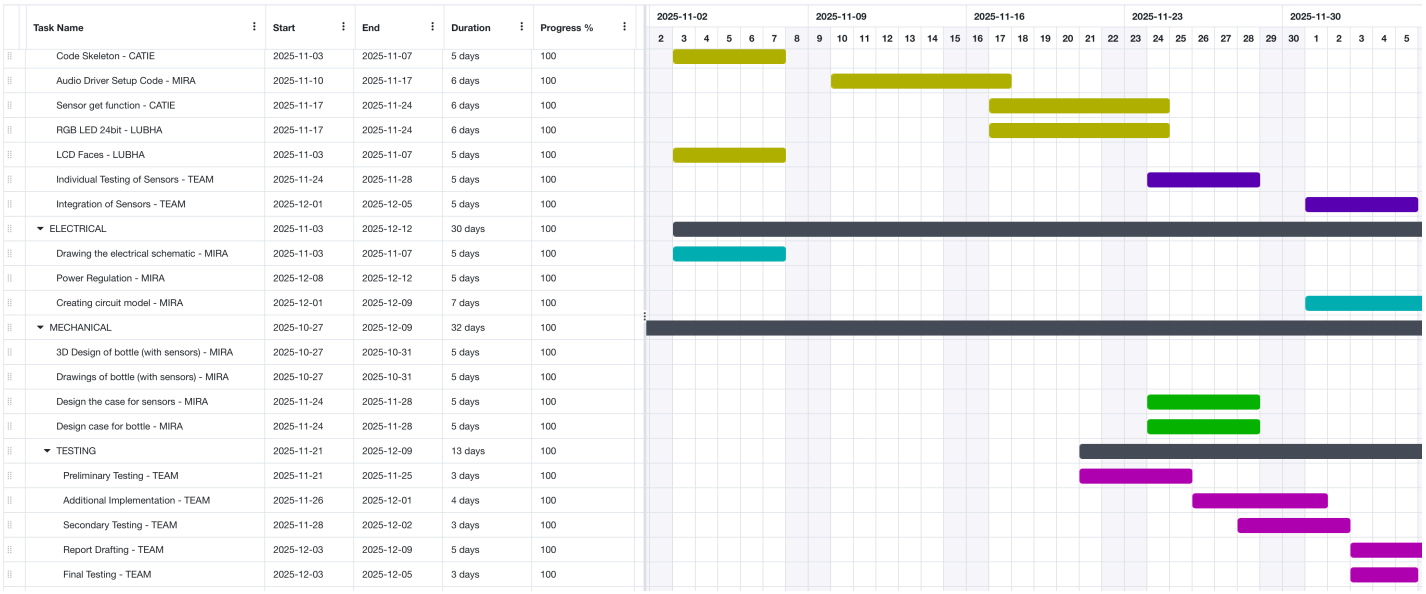
Below is the link used for the Final Project Presentation:
https://docs.google.com/presentation/d/1IQ7ptsauxRLAOfhBziD9jSM5K8ZaqktJYuk62H05T6E/edit?usp=share_link

17. Timeline

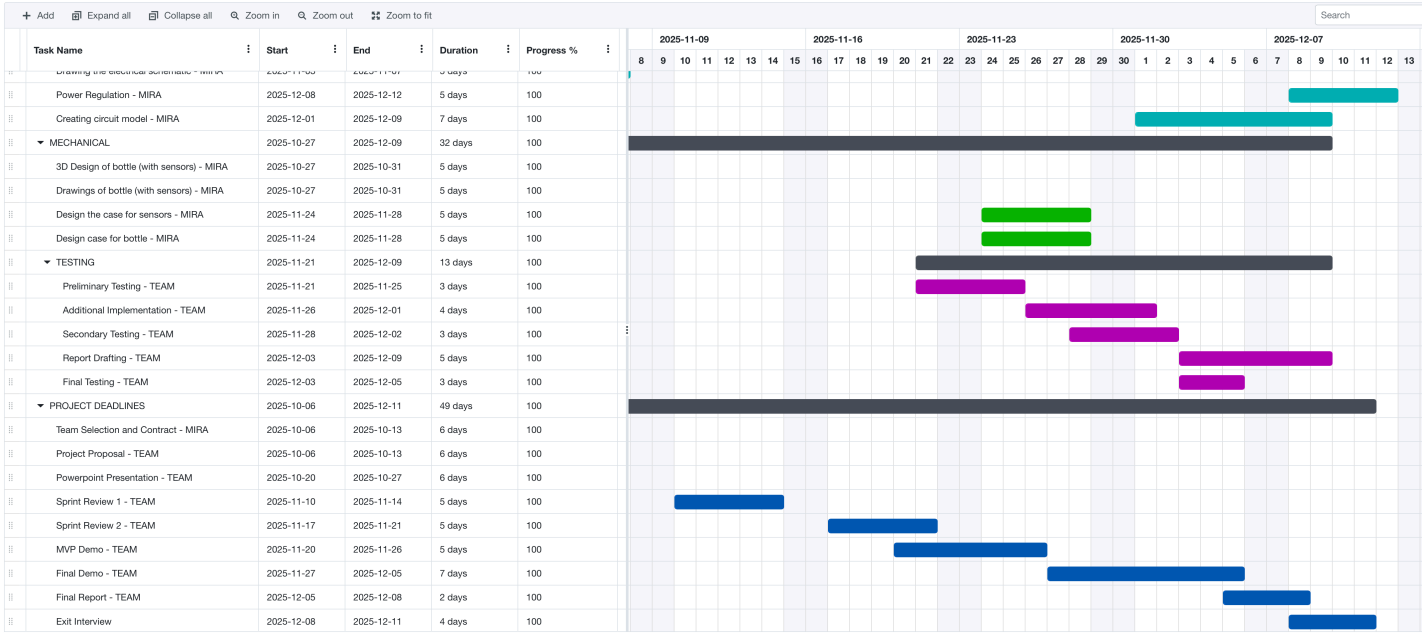
Below is a Gantt Chart that was followed throughout the project. All milestones were achieved as per deadlines by the team.



Task Name	Start	End	Duration	Progress %
PROGRAMMING	2025-11-03	2025-12-05	25 days	100
Face Display Function - LUBHA	2025-11-10	2025-11-17	6 days	100
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	100
Display Temperature - LUBHA	2025-11-10	2025-11-17	6 days	100
Timer Initializations - CATIE	2025-11-10	2025-11-17	6 days	100
Water level math & distance - CATIE	2025-11-10	2025-11-17	6 days	100
Code Skeleton - CATIE	2025-11-03	2025-11-07	5 days	100
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	100
RGB LED 24bit - LUBHA	2025-11-17	2025-11-24	6 days	100
LCD Faces - LUBHA	2025-11-03	2025-11-07	5 days	100
Individual Testing of Sensors - TEAM	2025-11-24	2025-11-28	5 days	100
Integration of Sensors - TEAM	2025-12-01	2025-12-05	5 days	100
ELECTRICAL	2025-11-03	2025-12-12	30 days	100
Drawing the electrical schematic - MIRA	2025-11-03	2025-11-07	5 days	100
Power Regulation - MIRA	2025-12-08	2025-12-12	5 days	100
Creating circuit model - MIRA	2025-12-01	2025-12-09	7 days	100
MECHANICAL	2025-10-27	2025-12-09	32 days	100
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawings 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	100
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	100



Task Name	Start	End	Duration	Progress %
Code Skeleton - CATIE	2025-11-03	2025-11-07	5 days	100
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	100
RGB LED 24bit - LUBHA	2025-11-17	2025-11-24	6 days	100
LCD Faces - LUBHA	2025-11-03	2025-11-07	5 days	100
Individual Testing of Sensors - TEAM	2025-11-24	2025-11-28	5 days	100
Integration of Sensors - TEAM	2025-12-01	2025-12-05	5 days	100
ELECTRICAL	2025-11-03	2025-12-12	30 days	100
Drawing the electrical schematic - MIRA	2025-11-03	2025-11-07	5 days	100
Power Regulation - MIRA	2025-12-08	2025-12-12	5 days	100
Creating circuit model - MIRA	2025-12-01	2025-12-09	7 days	100
MECHANICAL	2025-10-27	2025-12-09	32 days	100
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawings 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	100
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	100
TESTING	2025-11-21	2025-12-09	13 days	100
Preliminary Testing - TEAM	2025-11-21	2025-11-25	3 days	100
Additional Implementation - TEAM	2025-11-26	2025-12-01	4 days	100
Secondary Testing - TEAM	2025-11-28	2025-12-02	3 days	100
Report Drafting - TEAM	2025-12-03	2025-12-09	5 days	100
Final Testing - TEAM	2025-12-03	2025-12-05	3 days	100



Task Name	Start	End	Duration	Progress %
Power Regulation - MIRA	2025-12-08	2025-12-12	5 days	100
Creating circuit model - MIRA	2025-12-01	2025-12-09	7 days	100
MECHANICAL	2025-10-27	2025-12-09	32 days	100
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawings 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	100
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	100
TESTING	2025-11-21	2025-12-09	13 days	100
Preliminary Testing - TEAM	2025-11-21	2025-11-25	3 days	100
Additional Implementation - TEAM	2025-11-26	2025-12-01	4 days	100
Secondary Testing - TEAM	2025-11-28	2025-12-02	3 days	100
Report Drafting - TEAM	2025-12-03	2025-12-09	5 days	100
Final Testing - TEAM	2025-12-03	2025-12-05	3 days	100
PROJECT DEADLINES	2025-10-06	2025-12-11	49 days	100
Team Selection and Contract - MIRA	2025-10-06	2025-10-13	6 days	100
Project Proposal - TEAM	2025-10-06	2025-10-13	6 days	100
Powerpoint Presentation - TEAM	2025-10-20	2025-10-27	6 days	100
Sprint Review 1 - TEAM	2025-11-10	2025-11-14	5 days	100
Sprint Review 2 - TEAM	2025-11-17	2025-11-21	5 days	100
MVP Demo - TEAM	2025-11-20	2025-11-26	5 days	100
Final Demo - TEAM	2025-11-27	2025-12-05	7 days	100
Final Report - TEAM	2025-12-05	2025-12-08	2 days	100
Exit Interview	2025-12-08	2025-12-11	4 days	100

Task Name	Start	End	Duration	Progress %
Power Regulation - MIRA	2025-12-08	2025-12-12	5 days	100
Creating circuit model - MIRA	2025-12-01	2025-12-09	7 days	100
MECHANICAL	2025-10-27	2025-12-09	32 days	100
3D Design of bottle (with sensors) - MIRA	2025-10-27	2025-10-31	5 days	100
Drawings 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	100
Design case for bottle - MIRA	2025-11-24	2025-11-28	5 days	100
TESTING	2025-11-21	2025-12-09	13 days	100
Preliminary Testing - TEAM	2025-11-21	2025-11-25	3 days	100
Additional Implementation - TEAM	2025-11-26	2025-12-01	4 days	100
Secondary Testing - TEAM	2025-11-28	2025-12-02	3 days	100
Report Drafting - TEAM	2025-12-03	2025-12-09	5 days	100
Final Testing - TEAM	2025-12-03	2025-12-05	3 days	100
PROJECT DEADLINES	2025-10-06	2025-12-11	49 days	100
Team Selection and Contract - MIRA	2025-10-06	2025-10-13	6 days	100
Project Proposal - TEAM	2025-10-06	2025-10-13	6 days	100
Powerpoint Presentation - TEAM	2025-10-20	2025-10-27	6 days	100
Sprint Review 1 - TEAM	2025-11-10	2025-11-14	5 days	100
Sprint Review 2 - TEAM	2025-11-17	2025-11-21	5 days	100
MVP Demo - TEAM	2025-11-20	2025-11-26	5 days	100
Final Demo - TEAM	2025-11-27	2025-12-05	7 days	100
Final Report - TEAM	2025-12-05	2025-12-08	2 days	100
Exit Interview	2025-12-08	2025-12-11	4 days	100

18. References

Below are all the datasheets that were used as references for the working of the project.

Audio Driver: https://drive.google.com/file/d/1QlMNYdf2Et_ecabLgy2boJrilUhCZPAJ/view?usp=share_link

LCD: https://drive.google.com/file/d/1TSAaAhcV6VfpOCJ3L3C6UGDC89guZD3E/view?usp=sharing

Speaker with in-built amplifier: https://drive.google.com/file/d/1wQi7yzXNHPSx-Hg629KHsQ0qWIKia3_e/view?usp=sharing

Temperature Sensor: https://drive.google.com/file/d/1fPWtl4aOOyst-5PTwyC-3pgUk2JZRs-c/view?usp=sharing