Review the assignment due date

final-project-skeleton

- Team Number: Team 9
- Team Name: Oranges
- Team Members: Izzy, Helena, and Vidhu

- GitHub Repository URL: https://github.com/upenn-embedded/final-project-s25-oranges
- GitHub Pages Website URL: [for final submission]

Final Project Proposal

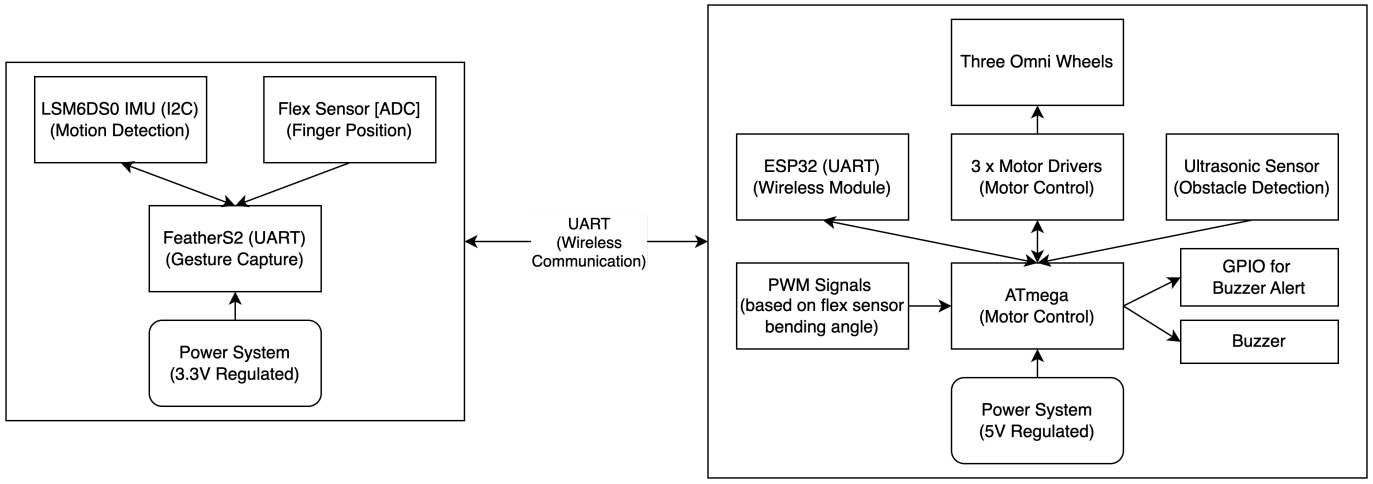
1. Abstract

Our final project is a gesture controlled rover robot designed to enable users to control a mobile robot's movement using intuitive hand gestures. The robot will use omni wheels for enhanced maneuverability, allowing for movement in multiple directions (forward, backward, lateral, and diagonal). Additionally, the rover will include a collision detection system using an ultrasonic sensor that triggers a buzzer/LED when obstacles are detected.

2. Motivation

Controlling robots using traditional joysticks or remotes can often be challenging and unintuitive, especially for new users. To address this, our project introduces a gesture-based control system that simplifies robot navigation while enhancing flexibility and precision through the use of omni wheels. The robot has the ability to move smoothly in multiple directions, including lateral and diagonal motion. Additionally, the inclusion of a collision detection system enhances the project's safety features by enabling the robot to identify and respond to nearby obstacles. These features make the project particularly valuable in practical applications like exploration and assistive robots for hazardous. The combination of intuitive gesture control and omni wheel movement offers a unique blend of precision, responsiveness, and user-friendly control.

3. System Block Diagram



Our system consists of two primary subsystems: the **WearableGesture Capture System** and the **Motor Control System** for the rover robot. They utilize various communication protocols including I2C, ADC, UART, PWM, and GPIO. Additionally, regulated power supplies are integrated into both subsystems to deliver stable voltage levels.

The **Wearable Gesture Capture System** is centered around FeatherS2, which manages both sensor inputs and wireless data transmission. It will handle gesture capture by interfacing with the LSM6DS0 IMU and Flex Sensor, while simultaneously transmitting control data to the rover's ESP32 module via UART.

LSM6DS0 IMU is responsible for detecting hand motion and orientation. It uses the I2C to communicate with FeatherS2. The Flex Sensor detects the degree of bending in the user's fingers, then it generates an analog voltage that is read directly by the FeatherS2's ADC. The ADC values are mapped to speed control commands, allowing different modes of motor speed with intuitive control.

The **Rover Robot Motor Control System** is built around the ATmega328PB microcontroller, which handles motor control, obstacle detection, and alert systems. FeatherS2 here acts as a wireless receiver, communicating with the FeatherS2 in the wearable controller via UART to receive gesture and speed control data.

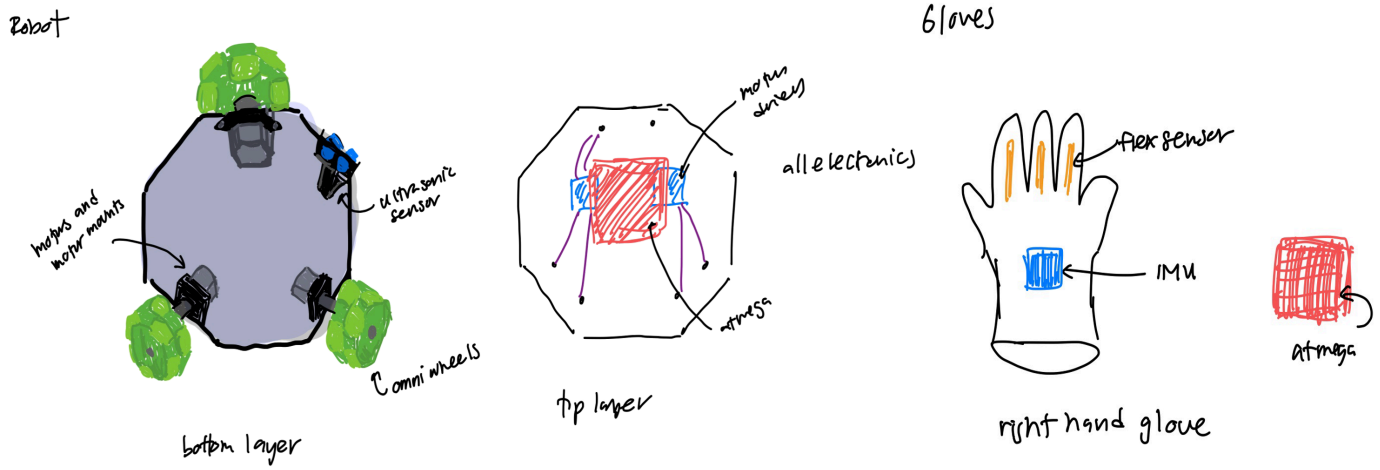
Motor control is achieved using three Motor Drivers that interface with ATmega. The motor drivers then manage the three Omni Wheels respectively, enabling movement in multiple directions. The ATmega generates PWM signals that control the speed and direction of the motors, with the PWM duty cycle dynamically adjusted based on the bending angle detected from the Flex Sensor and the hand motion detected from the IMU Module.

The rover also features an Ultrasonic Sensor for obstacle detection. The sensor communicates with ATmega using GPIO pins for both the Trigger and Echo signals. When an obstacle is detected within a specified range, the ATmega will activate the Buzzer via a dedicated GPIO output, alerting the user of the presence of an obstacle.

For the communication protocol, I2C links LSM6DS0 IMU to FeatherS2. The ADC on FeatherS2 is used to read variable voltage signals from the Flex Sensor. UART is employed for wireless communication between the two FeatherS2 boards. ATmega generates PWM signals to modulate motor speed and direction. GPIO connections are used for the ultrasonic sensor and the buzzer.

For power regulation, FeatherS2 requires a 3.3V regulated power supply. Meanwhile, the 5V regulated power supply in the rover system delivers stable power to the ATmega328PB, motor drivers, ultrasonic sensor, and buzzer. This separation of power systems ensures reliable performance, minimizes electrical noise, and protects components from voltage instability.

4. Design Sketches



For our design, the robot's base uses omni wheels for enhanced motion in multiple directions. The use of an ultrasonic sensor for obstacle detection ensures improved environmental awareness. The top layer of the robot houses all essential electronics, including the ATmega board, motor drivers, and power distribution system. On the glove controller, the IMU is positioned centrally on the back of the hand to accurately detect wrist movements, and the flex sensors are embedded along the fingers to track bending angles for speed control. We also plan to employ 3D printing and laser cutting for the motor mounts, structural supports, and the robot’s chassis.

5. Software Requirements Specification (SRS)

The software for the gesture-controlled rover robot will be able to process real-time motion and detect different gestures, wirelessly transmit commands, and be able to precisely control the rover. This can be broken down into three different subsystems: wearble gesture controller, rover control system, and collision detection system. The wearable gesture controller detects user gestures and transmits these movemements wirelessly. The rover control system will be able to interrept these commands and control the rover's movement using predefined gesture mappings. The collision detection system monitors the rover/s surroundings and alerts the users of any obstacles. Key requirements include gesture recognition accuracy, reliable wireless communication, precise motor control, and responsive collision detection

5.1 Definitions, Abbreviations

IMU (Inertial Measurement Unit): A sensor that combines an accelerometer and a gyroscope to track orientation, acceleration, and angular velocity. In this project, it is used to detect hand gestures such as tilts and wrist rolls.

Flex sensor: A sensor that changes its resistance based on the amount of bending or flexing. It is used here to detect finger position and aid in gesture control.

Feather ESP32: The wireless communication module using SPI to transmit data between the wearable controller and the robot.

Ultrasonic Sensor: A distance sensor that uses sound waves to detect objects. It is used here for obstacle detection on the robot.

5.2 Functionality

ID	Description
SRS-01	The IMU and flex sensors shall track predefined hand gestures (forward, backward, left, and right tilt, wrist roll, and open palm) within 200ms.
SRS-02	The ATmega328PB shall process the IMU data and classify gestures based off of certain threshold values correctly.
SRS-03	The ESP32 will transmit gesture data from the user to the robot with latency < 200ms.

ID	Description
SRS-04	The robot shall have three speed modes controlled based on the angular velocity detected from wrist rolling.
SRS-05	The ultrasonic sensor shall detect obstacles within 5–100 cm. If an obstacle is within 20 cm, the robot will stop, ignore commands, and the LED will turn red. Once cleared, the robot resumes movement, and the LED turns green.
SRS-06	The entire system will run independently on the ATmega328PB without the need of an external computer.
SRS-07	If wireless connection is lost, the robot shall automatically stop within 500ms.

6. Hardware Requirements Specification (HRS)

The hardware for the gesture controller rover will allow it to move freely in many directions as well as sense the motion of the controller. Additionally, it will be able to sense obstacles ahead using an ultrasonic sensor.

6.1 Definitions, Abbreviations

Here, you will define any special terms, acronyms, or abbreviations you plan to use for hardware

Feather ESP32: Board for wireless communication

ATMega328pb: Main microcontroller on the rover

IMU: Inertial Measurement Unit: Sensor able to detect rotations and accelerations

6.2 Functionality

ID	Description
HRS-01	The rover must be able to run for at least 15 minutes continuously
HRS-02	The ultrasonic sensor must be able to accurately detect obstacles within 5-100 cm

ID	Description
HRS-03	The motors must be able to move the rover at varying speeds
HRS-04	The rover must be able to move in perpendicular directions without needing to turn using the omni wheels
HRS-05	The flex sensor must be able to control the speed of the rover using an ADC
HRS-06	Motion of the IMU must translate into motion of the rover
HRS-07	The two ESP32 boards must be able to wirelessly communicate with each other directly
HRS-08	The ESP32 and ATMega328pb must be able to communicate using SPI

7. Bill of Materials (BOM)

We need the ATMega328pb, 2 FeatherS2, the LSM6ds0 IMU, the US-10 ultrasonic sensor, 3 motors with encoders, 3 omni wheels, 3 motor drivers, flex sensor, battery pack

https://docs.google.com/spreadsheets/d/1VVz9bghLVpC_rwVTz9ZApd166kRQwFx1JgdSo3LERfQ/edit?gid=253149064#gid=253149064

8. Final Demo Goals

On Demo Day, we will build an obstacle course for our rover to test in. Someone will wear the controller and manuver the rover around obstacles. Part of the test will include getting close to an obstacle and stopping before a collision. The test will conclude when the rover is successfully able to get to the end of the obstacle course.

9. Sprint Planning

You've got limited time to get this project done! How will you plan your sprint milestones? How will you distribute the work within your team? Review the schedule in the final project manual for exact dates.

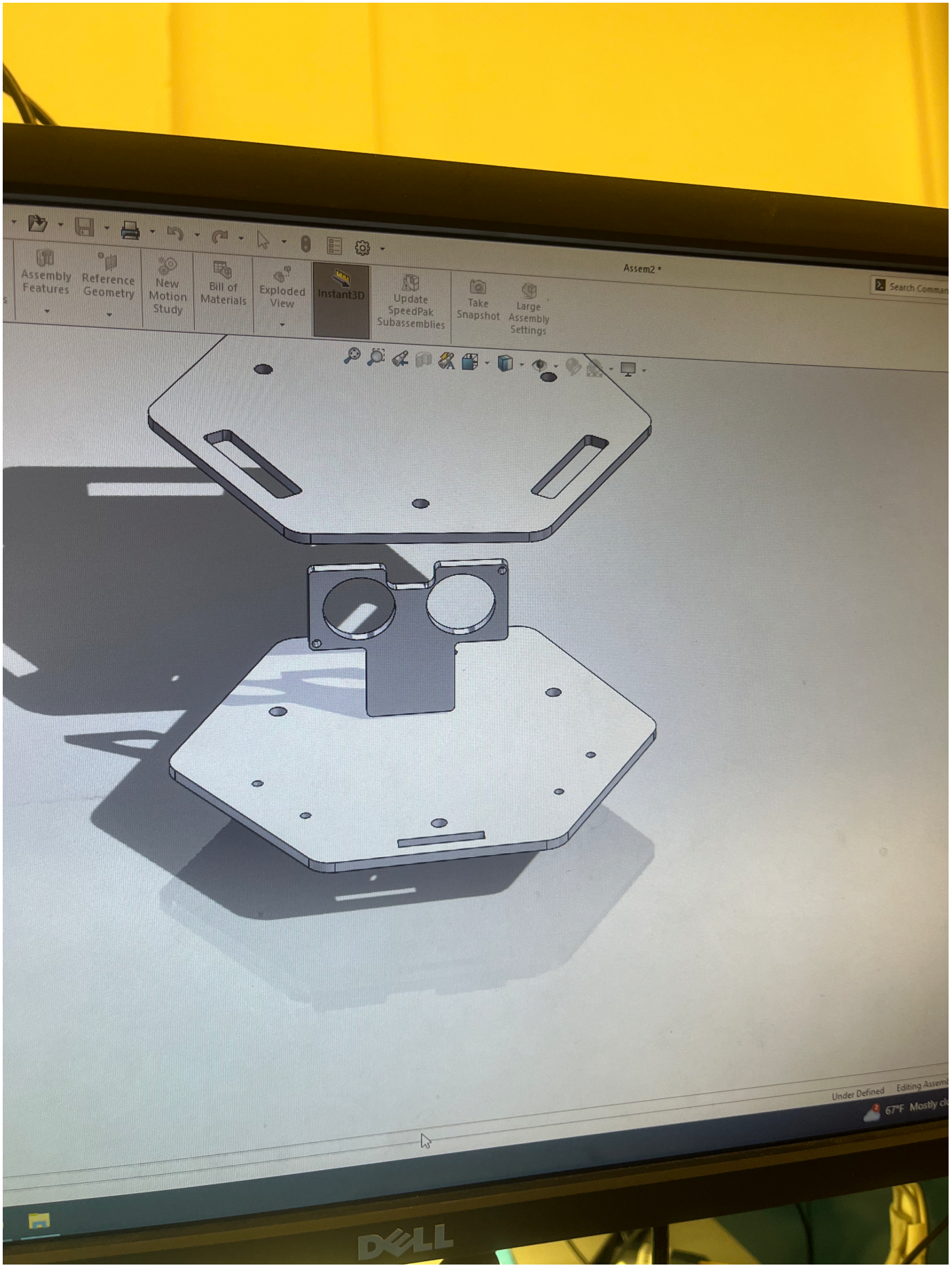
Milestone	Functionality Achieved	Distribution of Work
Sprint #1	1. Developed Wearable Controller Prototype using the FeatherS2. 2. Implemented motion sensing algorithms with the LSM6DS0 IMU. 3. Integrated the Flex Sensor for ADC-based speed control. 4. Developed basic motor control firmware for the ATmega using PWM signals for speed and direction.	1. Vidhu: FeatherS2 code for IMU and Flex Sensor integration. 2. Izzy: ATmega328PB motor control firmware development. 3. Helena: Hardware setup, wiring, and initial system testing.
Sprint #2	1. Established UART Communication between the two FeatherS2 boards. 2. Integrated gesture detection logic with motor speed control via PWM signals. 3. Assembled hardware components: Motor Drivers, Omni Wheels, and Power Systems. 4. Conducted movement tests to verify functionality.	1. Vidhu: UART communication code on both FeatherS2 boards. 2. Izzy: ATmega328PB firmware to decode UART data and control motors. 3. Helena: Hardware assembly, component mounting, and test setup.
MVP Demo	1. Developed and tested the Ultrasonic Sensor with the ATmega328PB. 2. Integrated obstacle detection logic to trigger the Buzzer via GPIO. 3. Refined gesture detection algorithms to improve accuracy. 4. Fine-tuned Flex Sensor ADC values for smoother speed control.	1. Vidhu: Ultrasonic sensor and buzzer control implementation. 2. Izzy: Refinement of gesture detection code for better responsiveness. 3. Helena: Conducted thorough testing of speed control accuracy.
Final Demo	1. Conducted full system integration for all subsystems. 2. Performed system calibration for accurate motion control, speed adjustment, and obstacle detection. 3. Conducted extensive performance testing for stability.	1. Vidhu: Final integration and calibration. 2. Izzy: Conducted performance testing and refined software logic. 3. Helena: In charge of documentation and demonstration preparation.

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

Sprint Review #1

Last week's progress

We have designed and modeled the housing for our robot. We then laser cut the parts out of acrylic.



We have also started implementing the gesture control by working with the IMU and feather S2 to determine the acceleration and rotation of the controller. The IMU is not outputting the measurements properly so it will require further debugging.

Current state of project

The project is coming along well. We are still waiting on the bulk of our parts to come in so we can start testing with the motors.

Next week's plan

Finish implementing the IMU detection. We will also start testing the flex sensor ADC. We will start assembing the physical robot using the laser cut parts and the other parts if they arrive.

Vidhu will finish debugging the IMU.

Izzy will assemble the robot.

Helena will test the flex sensor.

Sprint Review #2

Last week's progress

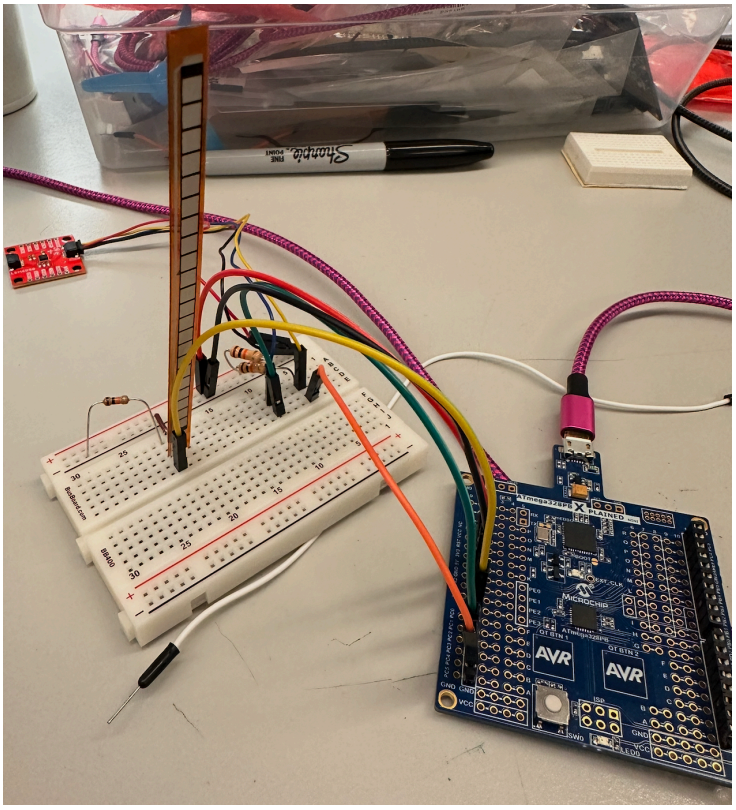
The picture velow shows varying outputs of the imu (need to calibrate in code the readings).

```
3591, 8903, 9185
-12128, -18410, 18749
777, 2005, 15403
-5943, -11782, 17919
-796, -2935, 16983
2116, -23, 16526
-471, 4154, 15605
-247, 2482, 19054
-237, 965, 16679
-179, -128, 16754
-203, -124, 16717
-139, -109, 16691
-214, -115, 16715
-16, -90, 16710
```

 1744395971360 some elementary calibration for the x acceleration showing the zero bias

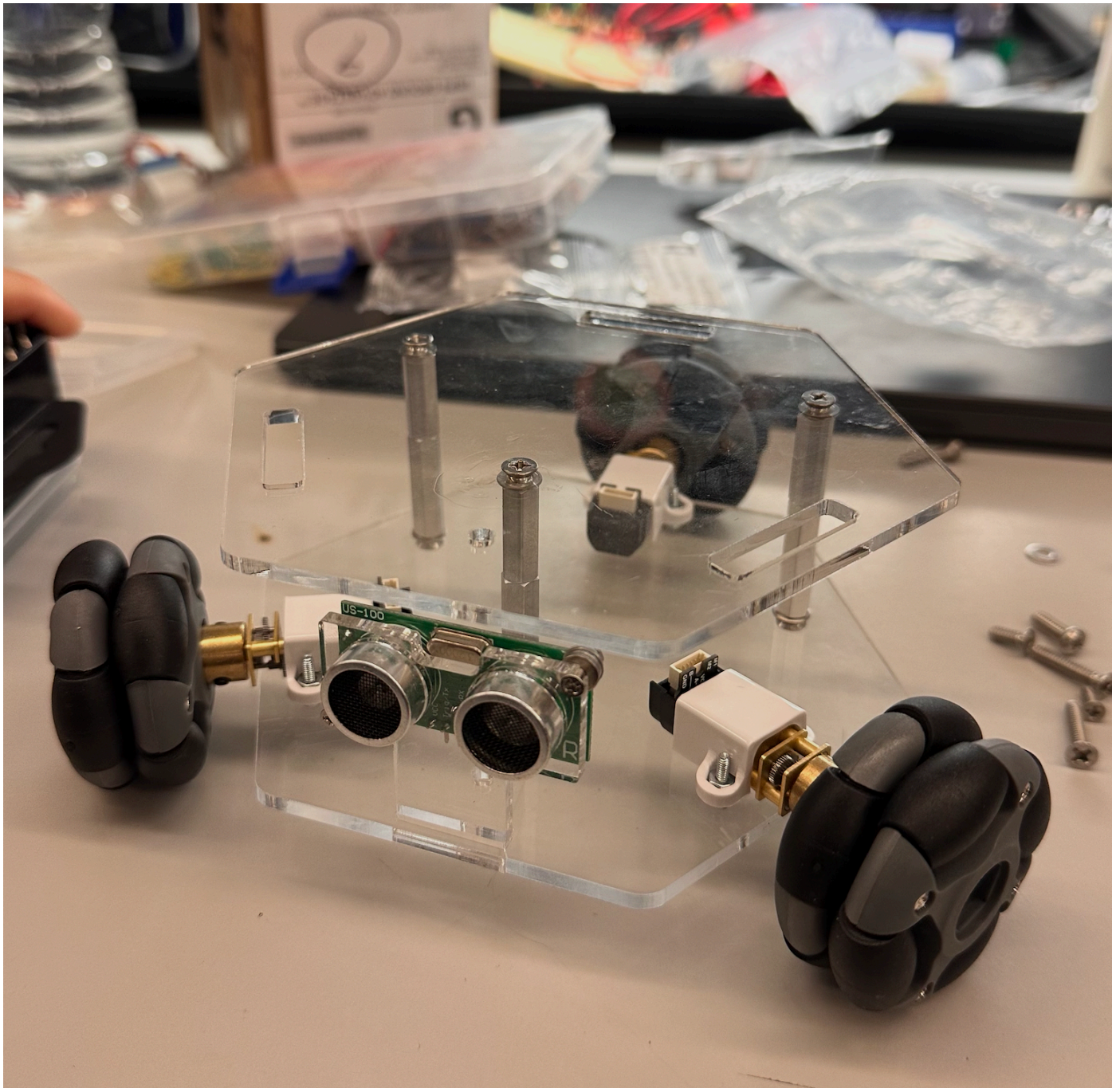
The flex sensor is working with its ADC pin as PC0. The ADC reads around 300 if the flex sensor is bent straight, and reads 0 if the flex sensor is completely bent.

```
Flex ADC:       0
Flex ADC:       291
Flex ADC:       291
Flex ADC:       291
Flex ADC:       197
Flex ADC:       199
Flex ADC:       187
Flex ADC:       181
Flex ADC:       171
Flex ADC:       158
Flex ADC:       149
Flex ADC:       141
Flex ADC:       130
Flex ADC:       119
Flex ADC:       115
Flex ADC:       106
Flex ADC:       0
```



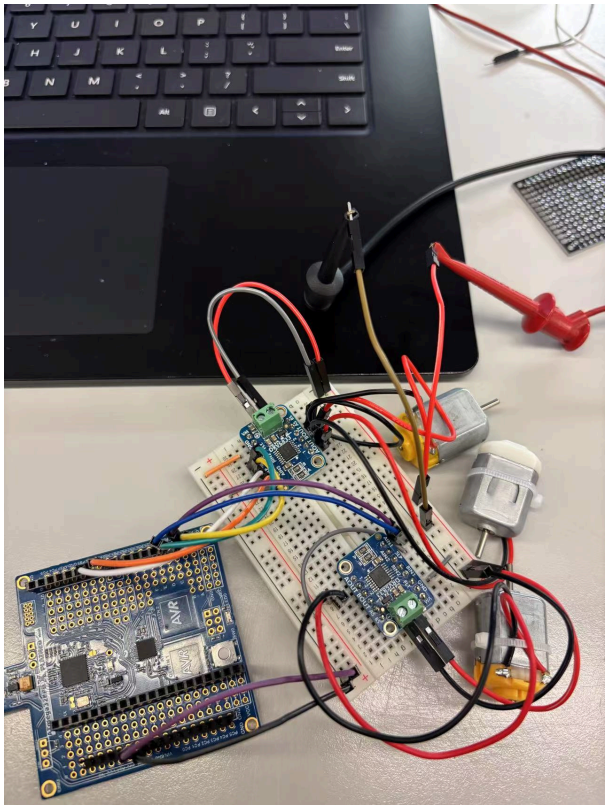
Current state of project

We've assembled the mechanical part of the rover robot, where the motor are attached to the bottom plate with the motor mounts, and the omni wheels are connected to the motor. The ultra sensor is also fixed in the bottom plate. The top plate will house the atmega and the other wiring. For the electrical part, the flex sensor (connected to ADC pin PC0) is working as expected, see code: https://github.com/upenn-embedded/final-project-s25-oranges/blob/main/flexsensor.c



Since the wires for our motor haven't arrived yet, we tested with the motors in the Detkin lab, and it functions properly as expected. The connections is as follows:

For motor 1, AIN1 goes to PB1, AIN2 goes to PB2, SLP goes to 5V on ATMega, AOUT goes to Motor M1 and M2.The green terminal is 5V and ground. See code (controllong a single Detkin motor): https://github.com/upenn-embedded/final-project-s25-oranges/blob/main/simple_working_motor_single.c. We followed similar wiring for the rest of the motor. Here is our motor prototype:



Here are detailed tables describing our wiring for the motors, motor driver, and ATMega328PB.

Motor Wiring:

Motor	Timer	Pin 1	Pin 2
A	1	PB1	PB2
B	3	PD0	PD2
C	4	PD1	PD3

Motor Driver Wiring:

Pin	ATMega328PB	Motor	External
VM	-	-	5V
GND	GND	-	-
BIN1	PD2	-	-
BIN2	PD0	-	-
SLP	5V	-	-
AIN2	PB3	-	-
AIN1	PB2	-	-
AS	-	-	-
BS	-	-	-
BOUT	-	M1/M2	-
AOUT	-	M1/M2	-

The ultrasensor along with the active buzzer is also working. Active buzzer is connected to PD6, the trig and echo pin of ultrasensor is connected to PC1 and PC2, and we used Timer2 to control the pulse. Buzzer sounds if distance is less than 15 cm. See code: https://github.com/upenn-embedded/final-project-s25-oranges/blob/main/ultrasensor_buzzer.c

```
Distance: 2 cm
Distance: 56 cm
Distance: 2 cm
Distance: 352 cm
Distance: 64 cm
Distance: 56 cm
Distance: 59 cm
Distance: 199 cm
Distance: 21 cm
Distance: 2 cm
Distance: 56 cm
Distance: 214 cm
Distance: 2 cm
Distance: 9 cm
Distance: 215 cm
Distance: 57 cm
Distance: 56 cm
Distance: 58 cm
Distance: 56 cm
```

Next week's plan

We will test the IMU with our calibrated code. We will also finish wiriing the motors with our motor drivers, and work on the encoding part. The electronics part should be integrating together, and we'll fix any potential conflicting issues. We will also work on EPS32 wiresless communcation

MVP Demo

1. Show a system block diagram & explain the hardware implementation.
2. Explain your firmware implementation, including application logic and critical drivers you've written.
3. Demo your device.
4. Have you achieved some or all of your Software Requirements Specification (SRS)?
    1. Show how you collected data and the outcomes.
5. Have you achieved some or all of your Hardware Requirements Specification (HRS)?
    1. Show how you collected data and the outcomes.
6. Show off the remaining elements that will make your project whole: mechanical casework, supporting graphical user interface (GUI), web portal, etc.
7. What is the riskiest part remaining of your project?
    1. How do you plan to de-risk this?
8. What questions or help do you need from the teaching team?

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.