CSCE 462 Final Project

Row Pergfect 1.0

Brian McKeown

4/23/2024

Table of Contents

1. Executive Summary

The goal of this project was to design a smart rowing handle that was able to detect errors in one's stroke. These errors include pulling the handle to the left of your body, pulling the handle to the right of your body, rushing your arms back too fast, and pausing in the middle of your stroke. This was to be done by using two MPU 6050 gyroscopes and accelerometers. They would collect data from each side of the handle to get a more accurate reading of the handle's behavior then the data would be analyzed to output signals to LEDs corresponding to the behavior that was happening. Additionally, there would be a button to start and stop the collection of data.

2. Introduction

This smart handle is designed to replace the handle that is on the Concept 2 RowErg or any similar rowing machine. It is very important to maintain form when on these machines as losing posture and form can result in worse scores or even injuries. This project was designed to combat improper form in real-time. With the LEDs lighting up when you make a mistake you don't have to spend as much time thinking about your form. On the left is the machine it is meant to be used on and on the right is the handle it is meant to replace.

3. Challenge Summary

The largest challenge I faced was the short timeline. After leaving my previous team, I was left with only 1 week to develop my project. This presented a lot of challenges such as sourcing materials, lack of planning, and limited time to test. I solved this problem by working non-stop through the week to deliver the project. Another challenge I faced was that the IMUs were placed into the handle at an angle. This presented a ton of challenges with measuring accuracy and the effects of gravity were now effecting two different measurements. I solved this problem by using a transition matrix to rotate the values so that they were better in line with their true orientation. Another challenge was noisy data. The IMUs are not very expensive nor are they calibrated the best so the data polled from the sensors contained a lot of error. This was especially hard to deal with because the data had to be analyzed in real time which made things much more difficult. As data was being collected, we put it through a low pass filter and then took a moving average of the past three values to smooth out the data. This proved to work much better for analyzing the data.

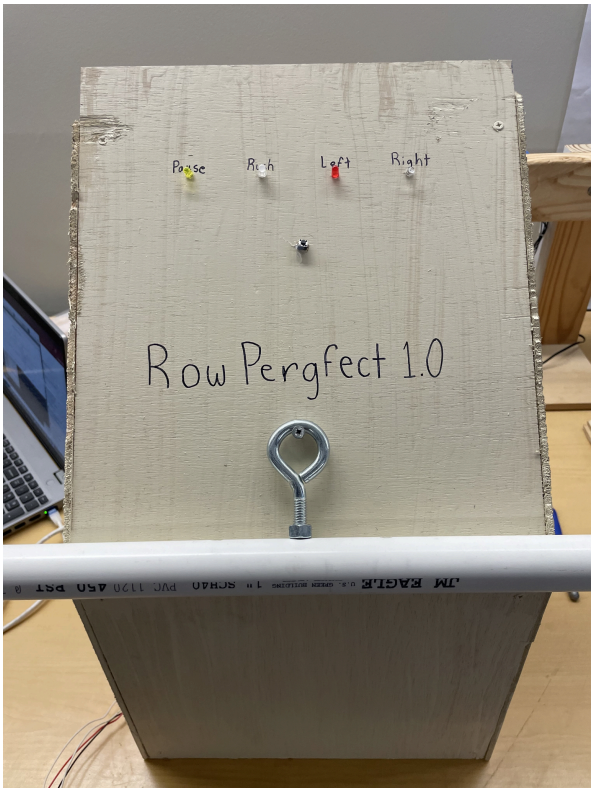
4. Problem Solving & Contribution Summary

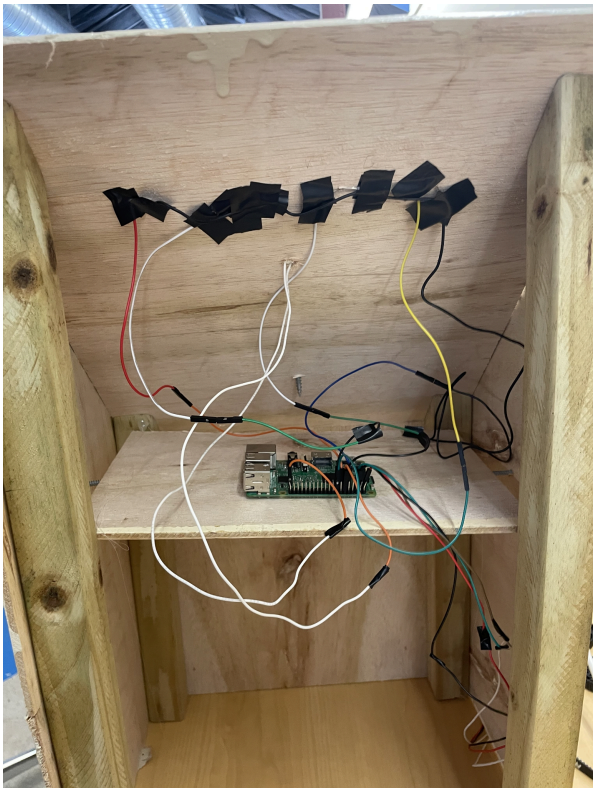
The only team member of this project was myself so all of the following contributions were my own. The project started with figuring out how to wire together two MPU6050s as I had never used I2C communication with two separate devices. All that was required was connecting the SCL and SDA lines to both of the devices and then addressing them by connecting one of the address lines to the power supply and one to the ground. Using the i2cdetect command in the terminal I could see both MPU6050s at the addresses 0x68 and 0x69. Next, I soldered the MPU6050s to long pieces of wire and measured them out so that one IMU could be placed on each end of the handle. After that, they were fixed in place inside the handle and it was on to testing.

As stated in the challenge summary, the IMUs were not fixed in place very well which led to a transition matrix being used to reorient their X, Y, and Z axis. After doing this, the handle was connected to the Concept2 RowErg and I started measuring data. I used Matplotlib to visualize the data. Additionally, I integrated the data to get velocity and position data. This allowed me to visualize the trends of the stroke. I used a switch variable and two different threshold values to measure the length of the stroke. I also took an average of the X acceleration values and if there was a value more than the 90 percentile plus 2 m/s^2 it would be marked as a rushed stroke. Then if there were any noticeable variations in the Y acceleration (which there shouldn't be as it is perpendicular to the range of motion) depending on the sign of the variation, the left or right LED would light up. Lastly, if the values for the acceleration stay close to the same value for more than four data points, it would activate the pause LED.

After, the wooden supplies for the housing were gathered, measured, cut, and painted. Then they were assembled into a box with a slanted roof and open back.

Lastly, all the remaining electronics that were still on the breadboard (the four LEDs and button) were soldered together and connected to the housing.





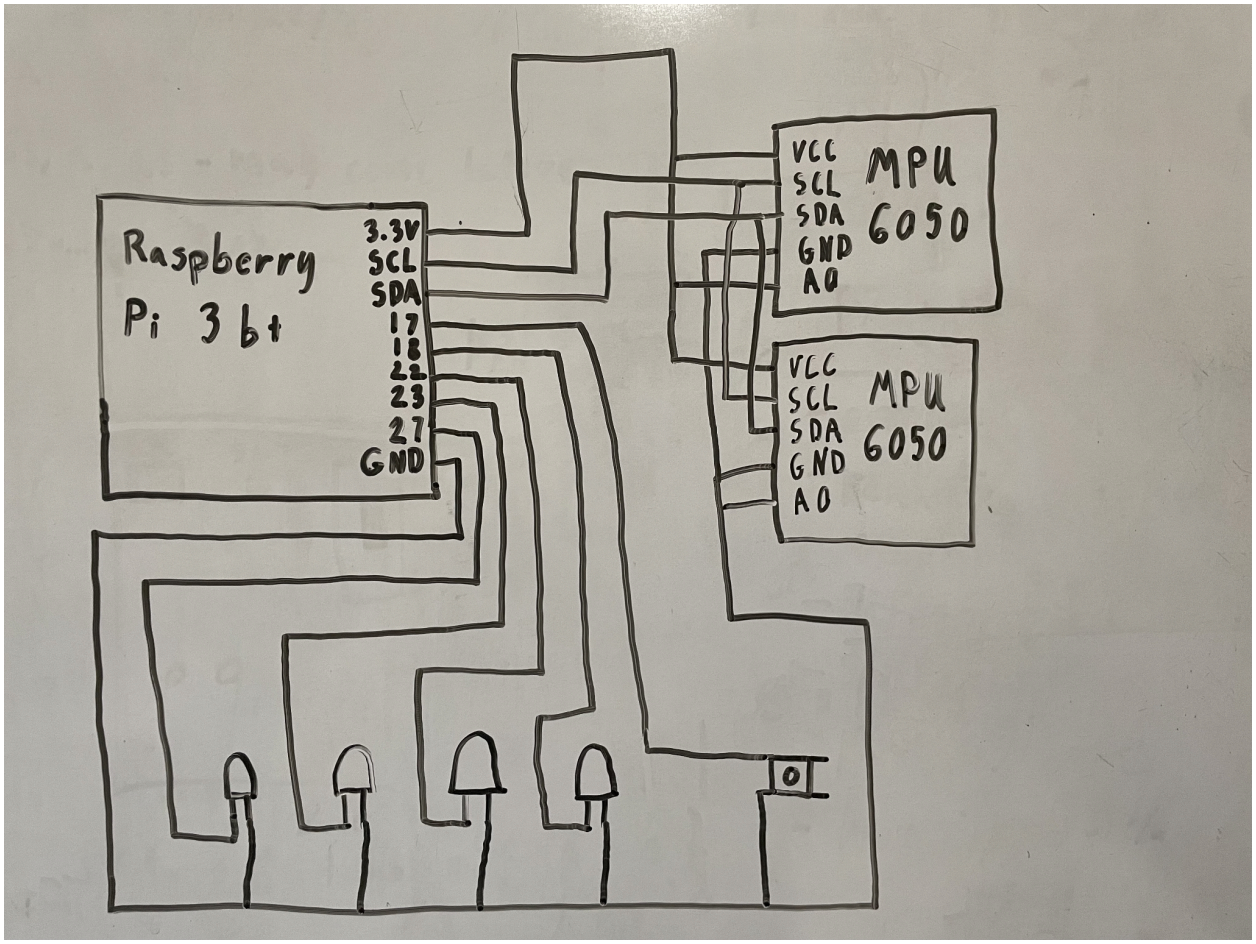
5. System Design

a. Physical:

Materials:

- 4 LEDs
- 2 MPU6050s
- 1 ft of 1 in. PVC pipe
- ¼ in. x 2-1/2 in. Stainless Steel Screw Eye
- 2 ft. x 4 ft. x ¼ in. plywood
- 2 in. x 2 in. x 8 ft. wood
- Wire (~30 ft.)

b. Electrical:



c. **Software:**

setup_gpio() - Sets up all necessary GPIO pins.

activate_led_#(duration=0.2) - replace # with the intended LED and it will send power to it for 0.2 seconds.

wait_for_start() - Waits for button press to continue any further.

button_pressed() - Checks if the button is pressed.

rotate_data(x, y, z, angle_deg=-45) - Rotates data through a transition matrix to get better aligned X, Y, and Z values.

get_smoothed_accel_data(sensor, alpha=0.5) - Smooths acceleration data.

low_pass_filter(data, alpha=0.5) - Sends data through a low pass filter.

moving_average(values, window_size=3) - Takes a moving average of acceleration values.

calc_stroke_rate(interval, rates_list) - calculates the number of strokes taken per minute.

calculate_average_y(accel_data_list) - calculates the average Y acceleration.

check_y_accel_thresholds(current_y, avg_y, flag_state) - checks if Y acceleration is past threshold to activate left or right LED.

check_stability_over_time(accel_data, time_step, duration, tolerance) - checks if acceleration data has remained relatively constant over a duration of time. Used to light up pause LED.

compute_90th_percentile(values) - computes value of 90th percentile in data set.

check_x_accel_threshold(current_x, x_values) - Checks if X acceleration is past the threshold value to light up the rush LED.

plot_data(time_step, accel_data_list) - plots data for visualization.

d. Link to code: https://github.com/BrianM53/Row-Pergfect-1.0

6. Outcomes:

For the most part, the prototype produced sufficient results that could be more refined with better software algorithms and more meticulous mechanical design. On the software side of things, the algorithms struggled to recognize each improper stroke technique at a good enough frequency. If I were to estimate the accuracy of each function, it would correctly output a pause almost 100% of the time, a rush 80% of the time, and left and right 80% of the time. Left and right also had difficulty distinguishing from one another and would sometimes light up the opposite LED. On the hardware side of things, the best improvement that could be made is taking out the IMUs and placing them back in a proper orientation. This would circumvent the need for the transition matrix to reorient the data as that was only a small fix to a larger problem. Additionally, the design

could have some connections soldered over again as some of the joints are not connected very well. This would give the project a better shelf life and make it mechanically more stable.

7. References:

https://chat.openai.com/ - some of the code was AI generated and just helped with other general questions.
https://www.circuito.io/ - helped with wiring diagrams to connect two I2C devices.
https://www.youtube.com/watch?v=UxABxSADZ6U&t=311s - Cool MPU6050 project that taught me techniques and gave me inspiration and direction on the project.