2024 Fall Capstone Notebook (Sep 24- Nov 19)

Sep 24:

Progress

- Last week, finalized the flowchart design for the depth camera code
- With the flowchart finished, the C++ prototype for the depth camera can be modified. The current model is mostly for testing purposes and to confirm that the depth camera's data can be read
- The goal for the future is for communication directly between the camera and Pico microcontroller to be established with C++
- This will begin once the hardware components are finished, meanwhile we can brainstorm ideas for the upcoming LiDAR detection once it is confirmed to work with the hardware

Sep 26:

Progress

- Code to test the LiDAR will be written in Python
- Originally was going to create a diagram. However, the hardware implementation is not fully complete, and without it, fully visualizing the scope of the software necessary to implement GUIDE's goals will not be possible
    - Eg. Code that can communicate between hardware (Pico, LiDAR, depth camera, vibration motors)

Sep 29:

Progress

- Not much progress on software development today, the bulk of the work done Is still mostly on the hardware side of things
- Will continue to brainstorm how C++ can be used to help facilitate communication with the Pico and depth camera

Sep 31:

Progress

- Significant hardware progress has been made, and more information is available for development
    - Specifically, UART will be used to establish connections between all sensors and the Pico. Future research will now be focused on how UART communication can be facilitated in C++.

Oct 3:

<u>Progress</u>

- Sample files have been given to the software team from the hardware team to decode
- File output is according to the TF2-Luna documentation, which shows that the code is in packet format, future work will involve decoding the files and extracting the necessary information (distance)
- UART communication is not straightforward with C++

Oct 10:

<u>Progress</u>

- The output file has been partially decoded, with the following information present:
    - Each byte packet starts with the characters "YY", is header according to documentation
    - Distance data immediately follows this, and is encoded in little-endian
        - Specific distance data has not been decoded
    - Following distance is Signal Strength and Temperature, all 2 bytes length
    - The checksum is the final byte of the packet
- For C++ at least, the libraries available to decode the data (especially between hardware) are limited, and the complexity of the code rises significantly. As the main portion of the code should be data manipulation rather than data reading, Python may be recommended in the future

Oct 15:

<u>Progress</u>

- An example packet of data was fully decoded, the results being the following:
    - Packet: YY\x14\x00\x1f\x0e\x10\t\x0c
    - YY -> header
    - \x14\x00 -> Distance (result 19, manual shows that default unit is in cm)

    - \x1f\x0e -> Strength
    - \x10\t -> Most likely temperature
    - \x0c -> Checksum
- The output file was significantly more difficult to decode since through translation to a file, the characters would be encoded into a specific ASCII format. Since the hardware prototype was completed, direct communication between the LiDAR and Pico was established and the data could be directly read.
- The code to be able to do the above was made in Python, since the libraries available to establish UART connections are directly present and simplistic to use
    - The loss in optimization from C++ to Python seems to be negligible as the data without being rate limited is sent in less than 10ms consistently
    - Thus, all future code that directly handles data communication between hardware will be in Python

Oct 17:

<u>Progress</u>

- Now will be looking towards the depth camera
    - The data given by the depth camera is significantly more challenging to decode, as it includes information about the environment it is recording and sends it in ASCII format
    - Despite the above, the first few lines of information are similarly formatted to the LiDAR
- The LiDAR was tested to be incredibly precise and high range, and the data from this sensor is clearer than that of the depth camera
    - May explore a machine-learning approach where the depth camera's environmental data is used to determine whether an object detected by the LiDAR is an obstacle (example: a park bench vs. tree leaves)
    - The LiDAR distance information will have a higher weight than the depth camera's distance information in weight calculations

22 24 29 31

Oct 29:

- Original code that analyzed depth camera data has been converted from C++ to Python

- Currently decoding the data from the depth camera and analyzing which portions are relevant
    - The ASCII data is a heatmap of the area that the camera is scanning, this may be useful as a stretch goal to help interpret objects and could be a potential to integrate machine learning models
    - Before the ASCII data, there are data packets, like that of the LiDAR
    - Once decoded the algorithms that determines which portions of data from the LiDAR and camera are usable, what weight each has, etc. can be determined and implemented

Oct 31:

- Progression is underway for decoding the depth camera's data
    - Not much else to say on the software side of things

Nov 5:

- Depth camera data decoded
    - Bytes that contain the distance information have been found
    - Additional information includes average pixel distance and difference between closest and farthest pixels
    - Heatmap from pixel data generated, may be useful in adding optimizations for object detection
- Will continue working on debugging the distance output and optimizing the list of data generated by the camera

Nov 7:

- Considering working on a SMART algorithm that calibrates the walking stick for each user
    - The idea is that due to the angle adjuster pointing the sensors at an angle, the possibility of objects being detected at different distances may come to rise
    - While the above happening would result in only minor differences to range detection it is still useful as an optimization
    - To solve this, small subsets of data averages from the LiDAR and depth camera are taken into consideration every few seconds

Nov 12:

- Due to time constraints on project completion, the above algorithm will be difficult to test

- While the code for the above is complete, it will not be present in the final demo of the project
- It can still be considered as a stretch goal though

Nov 14:

- Created algorithm that determines when the vibration motors will go off at 2m, 1m, and 0.5m
    - Utilizes a combination of depth camera averages and LiDAR averages
    - Since several LiDAR readings happen during the duration of the camera reading, the camera reading must be non-blocking otherwise the loop running the code will stop midway through a camera reading
- Testing started today, some hardware issues caused the software implementation to be tested a little late but the code looks functional so far from the data gathered

Nov 19:

- Testing has been completed today
- Slowed down due to some last-minute hardware changes, but testing returns positive results
    - There are two vibration motors, one in the front, one in the back. The front motor activates at 2m, the back activates at 1m, and both activate at 0.5m to simulate vibration strength as an object gets closer to the user