Dear Future Groups,

Congratulations on selecting or being put on one of the most challenging and rewarding projects at RIT, the Autonomous People Mover! This project is huge in scope, and we are sure that it may seem daunting to get brought up to speed on everything that has been done before you, but in an effort to get you started we are creating this document, as well as the Theory of Operations documents and subsystem videos. First, we will go over a quick history of the previous three phases of the project, and then go into a list of recommended (arguably required) knowledge for working on the cart, and finally list our suggestions for where you should start as far as future improvements

History

Up to the point of the creation of this document, there have been 3 phases of MSD projects to work on the APM, and a total of 4 teams. The original plan for the projects was to have the first phase convert a stock golf-cart into a remote control golf-cart. The second was to take a remote control golf-cart and make it basically autonomous, and the third to take a basically autonomous golf-cart and add more autonomous functionality.

However, as things go, phase one did complete the creation of a remote-control golf-cart, which worked for a very small period of time before having serious issues which lead to the start of phase 2 being an attempt to figure out why phase one's design failed. During phase 2, there was also a Computer Engineering team which attempted to get some of the sensors online and was crucial to influence the decision to use ROS. They however ran into some serious difficulty along with phase 2 as even at the end of their phase the issues with the phase 1 design was still not uncovered.

Having our MSD1 overlap with phase 2's MSD2 gave us insight into potential flaws and challenges we would face, so at the start of MSD2, we decided to re-design all of the electrical circuits and software for the cart. We completed a complete re-wiring of the cart, a new PCB design using newly designed circuits for controlling and filtering and level shifting, re-designed all the subsystems software, and in general re-designed the golf-cart as a computer controlled vehicle which could take input from a remote control or anything else. At the very end of our project autonomous functionality was achieved, but only in a very basic form, with the projects only real large flaw being localization.

Recommended Knowledge Before Start

The computer software for this project is written for ROS, if you are unfamiliar with the platform, it is crucial that you gain an understanding of how ROS works. The ROS.org website has a section for tutorials, and it is recommended that you complete as many as you can in preparation for this project. Pay careful attention to creating workspaces, packages, publishers and subscribers, tf, and launch files. Perhaps the most difficult to understand will be tf, but it is especially crucial to the function of the APM. An understanding of the C++ and python programming languages is an absolute necessity as well, so prepare yourself for that however you see fit. Also, if you need to adjust or add to the control subsystems like throttle, steering, brake, etc. you will need to be comfortable with Arduino - this is very easy however so if you aren't already leave yourself a little time for that. You should also be very well versed in usage of linux, and be very familiar with Git.

Future Work & Recommendations

While our projects ends with only basic autonomous functionality working, it is actually very close to some much more sophisticated functionality already with only a couple changes.

Firstly, the cart is only able to localize and get heading information from the cart odometry model, so that severely limits the ability of the cart to get a good desired heading. We had an IMU, however inside of the enclosure it produced mostly noise. Our first recommendation is that you invest in a high-quality IMU and mount it in a low-noise location. This will be able to fuse with the cart odometry (via the package robot_localization) and give a good heading which would immediately improve the performance of the autonomous mode significantly.

The next is to get a high-gain WIFI connection for the cart computer. This for the convenience of yourself as well as a potential source of internet for the RTK functionality for the RTK GPS. This leads to the next item, getting the RTK GPS working in Ubuntu. While not a necessity as the other GPS will do what you need it to, the RTK GPS offers unparalleled accuracy and works very well with RTKlib in Windows, getting that working on the cart would be a big plus. Make sure you design some way to see when it does and doesn't have RTK lock however, and adjust the covariances accordingly as when the RTK GPS does not have RTK lock it is worse than the regular GPS.

From a mechanical system standpoint, a new brake actuator might be wise, preferably smaller so the that mount does not contact the wheel at extreme steering angles and quicker with smaller stroke as much of the range of motion of the current one is not needed and operating quicker would improve emergency stop operation and general responsiveness.

Phase 1 had added a sensor which failed to detect the position of the brake pedal. You should strongly consider a design for detecting this as you can have an operator pressing the brake while the cart is trying to move forward and this can be an unsafe condition. The steering is working very well now in terms of accuracy and not making noise, but this is due to use RC low-pass filtering which leads to an accurate but not immediate steering system (~500ms delay). As a way to improve the platform, taking time to tune the R and C values could lead to a faster filter which still gets rid of the problematic noise leading to a faster reaction time, the PCB is designed to make the filter resistors and capacitors pretty easy to change. Also there is backlash in the steering gear so finding a way to reduce that will also help the accuracy and response time of the steering.

As a general note, please be aware that the LiDAR is the most expensive part of the project so far at $8000, so be very careful with it as a replacement would be very unlikely.

Contact Information

Question Topic	Who to Contact
General Questions, Software (Computer or Arduino), On-board Computer, ROSdue	Cody Smith (cds7494@rit.edu)
General Circuit Questions, PCB, GPS, Odometry	Alex Avery (aja9675@rit.edu)
Ultrasonics	Jess VanGiesen (jlv4851@rit.edu)
Project Management	Joe Hudden (jmh4206@rit.edu)
Mechanical Systems	Eric Schulken (ems1552@rit.edu)

We hope you find that we left you a well thought out platform for building upon, and as you develop new functionality please keep in mind the future of the project. Don't hesitate to reach out to experts. Work hard, and do us proud!

Regards,

Phase 3