Vehicle Level Software Design of the Florida Polytechnic Autonomous Golf-Cart

Heitor Tremura
Electrical and Computer Engineering
Florida Polytechnic University
Lakeland FL, 33805
Email: htremura3005@floridapoly.edu

Onur Toker
Electrical and Computer Engineering
Florida Polytechnic University
Lakeland FL, 33805
Email: otoker@floridapoly.edu

***Abstract*—In this paper, we summarize some of the design and implementation related aspects of the Florida Polytechnic Autonomous Golf-Cart, and present details of the low-level microcontroller based subsystem, two-way non-blocking communication details, and the lightweight software subsystem based on the co-operative multitasking approach. Autonomous vehicles research is becoming a larger field each day. As technology progresses, we want to develop more efficient and smarter vehicles which can transport us with little to no input from the motorist, and electric golf carts are valuable assets towards this goal. In the scientific literature, there are several published results and proposed architectures for autonomous golf carts, and we will review some of major ones, describing pros and cons of each design approach, and how they relate to our design and implementation. However, our main focus will be the vehicle level software design.**

I. Introduction

Golf carts have always been an attractive tool for autonomous vehicles research. Being relatively inexpensive and easy to acquire makes them ideal for research projects [1]. Research topics focused on golf cart-based matters often base themselves around a combination of many systems with varying complexity. Research is similarly performed on frameworks for autonomous systems in educational robotics, involving many solutions to comparable problems [2].

A great number of complex components must be involved in creation of an autonomous vehicle. This requires some architecture which can combine different systems. Because of the complexity of the problem, these parameters frequently require a sensor-fusion approach to make them work [3]. Inputs must be combined and weighed accordingly to make informed decisions when navigating to drive the relevant motors. Inputs can range from ultrasonic sensors, laser scanners, and thermal sensors to devices as simple as a compass. There must also be sensors to measure the angle of the steering wheel and speed of the wheels [3]. All these sensors send their information to some centralized computer which weighs the values and makes decisions based on its understanding of the current variables dictating the golf cart's position and speed. Oftentimes a CAN bus is used to coordinate the multiple sensors and devices which must communicate [3], [2].

In this paper, we briefly summarize the Florida Polytechnic Electric Golf-Cart and mainly focus on the low-level microcontroller based subsystem, two-way non-blocking communication between the microcontroller and the vehicle computer, and the lightweight software subsystem based on the co-operative multitasking approach.

This paper is organized as follows: In Section II we present a short summary of previously published autonomous golf-cart architectures, and in Section III we present the design and implementation related aspects of the Florida Polytechnic Autonomous Golf-Cart, details of the low-level microcontroller based subsystem, two-way non-blocking communication system, and the lightweight co-operative multitasking based software subsystem. In Section IV, we make some concluding remarks and discuss future research directions.

II. Review of Autonomous Vehicle Systems

Regarding the use of the word "Autonomous" there are several levels of autonomy as defined by SAE International [4]:

- Level 0: No Driving Automation.
- Level 1: Driver Assistance.
- Level 2: Partial Driving Automation.
- Level 3: Conditional Driving Automation.
- Level 4: High Driving Automation.
- Level 5: Full Driving Automation. Permits engagement of the Automated Driving System in all road conditions manageable by a human driver.

Organized on all the vehicles is an assortment of sensors. The following sensors are most commonly present on autonomous golf cart vehicles:

- Stereo RGB Cameras, [1], [2], [5], [6]. These are used in conjunction with sensor fusion to determine distance from vehicle.
- LiDAR, [1], [2], [6], [7]. LiDAR is used in much the same way as the RGB Camera setup. Light waves are emitted and return times are measured to determine distance from vehicle.
- GPS (Global Positioning System), [1], [3], [2], [5], [6], [7]. Likely one of the most crucial sensors for a great bulk of autonomous vehicles research. The GPS can determine the vehicle's position on the planet Earth.

Also present on many of the testbed golf carts is some way for a human operator to interact with the vehicle and provide immediate input. Often this is due to the autonomous golf cart vehicles being below or at Level 3 automation as defined by SAE [2], [4].

In the various golf cart architectures, there is a great difference in what algorithms and implementations are used for localization

SoutheastCon 2021 | 978-1-6654-0379-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/SOUTHEASTCON45413.2021.9401891

of the vehicle. They must also, using this localization, execute some decision making to determine the best course towards the goal while encountering the least obstacles. Examined in the papers are the following approaches:

- NDT-matching algorithm, [1]. Mapping and localization is handled using an open source Autoware "Normal-Distributions-Transform" laser scan matching [8]. Information from the Velodyne LiDAR is processed and aligned with a pre-computed 3D environment map.
- SLAM Algorithm, [2]. While in the designated autonomous mode the vehicle will drive by itself using SLAM (Simultaneous localization and mapping) in conjunction with Google Cartographer and a robot localization package in ROS (Robot Operating System) using the Velodyne LiDAR VLP-16, IMU VN-100, and Garmin GPS 18x.
- Wheel Odometry and Waypoint Navigation, [5]. By using a predefined starting position, it is possible to do localization with data obtained from wheel odometry. Using encoders on the wheels combined with circumference of the wheel, the final positioning is determined. This is used by navigation and path-planning algorithms to establish the current position and desired subsequent positions as waypoints.
- AMCL in ROS, [6]. An autonomous golf cart is simulated within Gazebo and researchers use this environment to implement AMCL, which is a localization system for a robot moving within a 2D space.
- Branch-and-bound algorithm, [7]. Using a node-based mapping system and a branch-and-bound algorithm, the computer can eliminate all of the paths which are not the shortest path.

After this review of published autonomous vehicle architectures, we are now ready to summarize some of the design details of the Florida Polytechnic Autonomous Golf-Cart System, which is presented in the next section.

III. Florida Polytechnic Autonomous Golf-Cart

Florida Polytechnic Autonomous Golf-Cart project is a joint research effort consisting of a large group of faculty and researchers. In this paper, we will focus on the low-level microcontroller system, and communication related issues.

Florida Polytechnic Autonomous Golf-Cart is a complex system with multiple subsystems, see Fig. 1. The final project objective is to have an autonomous vehicle testbed. The system developed in [9] has an on-board x86 PC as the master computer, and a stereo camera, a depth camera, a GPS/GNSS receiver, IMU sensors, and a Velodyne VLP-16 Lidar. This design is flexible to be used with various mechanical base systems. In [10], a low cost hardware-in-the-loop agent-based simulation testbed is proposed. In [11], an electric golf cart is first converted into a drive-by-wire system, and then using on-board computers various self-driving algorithms are tested. The autonomous vehicle system presented in [12] has Nvidia Drive PX2 as the master computer, and dSPACE Microautobox as the low-level control unit. Another interesting electric golf cart based design is discussed in [13].

The high-level block diagram of our proposed design is presented in Fig 2.

There will be an ARM Cortex-M4F based low-level controller directly interfaced to the linear actuators, and the DC motor

Fig. 1. Florida Polytechnic Autonomous Golf-Cart.

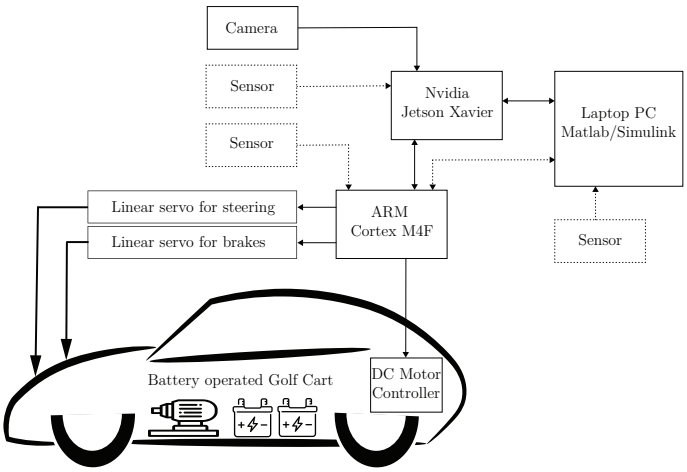

Fig. 2. High-level system block diagram for the proposed golf cart system.

controller of the golf cart. This will be a bare-metal system, and will support optional SPI and/or I2C sensors.

There will be also an on-board Nvidia Jetson Xavier, interfaced to a camera. The on-board Nvidia computer will have USB/Serial connection to the ARM Cortex-M4F board. Furthermore, the same Nvidia computer will support optional sensors, and can also be used a the master computer to run various AV algorithms without the help of the laptop PC. In this case, the AV algorithm has to be implemented in C/C++, Python, or in ROS framework.

Finally, there will be a laptop PC running Matlab/Simulink interfaced to both the Nvidia computer and the ARM Cortex-M4F board. The laptop PC will also support optional sensors, for example a lidar unit. Basically, the laptop PC can be used as the master computer to run AV algorithms, with or without the help of the Nvidia computer. The Nvidia computer can be completely turned off, or it can function as a slave computer. A typical application can be the Nvidia computer doing computer vision as a slave computer, and the master computer (laptop PC) using this Nvidia computer simply as another sensor block/subsystem.

The Nvidia Jetson Xavier is an 64-bit 8-core ARM computer

with 512-core Volta GPU. This will be used with an internal SSD and a external protective case. Compared to the Nvidia Drive PX2 used in [11], it is less powerful but much more economical.

A. Communication Abstraction Layer

We are currently developing a two-layer based solution for the microcontroller software. Our plan is to have a non-OS based lightweight solution using the co-operative multitasking approach. The first layer will be a communication abstraction layer, and will have the following functions.

- `is_RX_error()` checks for a general receive error, which includes but not limited to overflow, framing errors, and parity errors. This is a non-blocking function which returns almost immediately.
- `is_RX_ready()` checks whether a receive is completed, and a packet is ready to be read. This is a non-blocking function which returns almost immediately.
- `get_RX_data()` copies the complete received packet to a given location, and returns. This is a non-blocking function which returns almost immediately.
- `is_TX_error()` checks for a general transmit error, which includes but not limited to overflow, and other types of errors. This is a non-blocking function which returns almost immediately.
- `is_TX_ready()` checks whether the transmit system has a free buffer space to copy a new packet. This is a non-blocking function which returns almost immediately.
- `send_TX_data()` copies the complete packet to be transmitted from a given location to an internal buffer, and returns. This is a non-blocking function which returns almost immediately.

There will be certain data structures hidden from the user of this layer, and each of the above given functions may update these data structures if needed. Critical portions of the code will be enclosed between interrupt disable and interrupt enable commands, to make sure that critical sections run without any interrupts serviced during their execution.

B. Low-level Automation Layer

This layer sits on top of the Communication Abstraction Layer. Basically, there will be an infinite loop like code running on the microcontroller. There will be multiple tasks, but each task is supposed to be written as non-blocking. All tasks should return as quickly as possible. The following pseudocode explains the main design planned for this layer.

```
while (1) {
     if is_RX_error() rx_err_handler();
     if is_RX_ready()
        get_RX_data();
        // parse RX data and call relevant
        // tasks based on the received data

     // read sensors
     if is_TX_error() tx_err_handler();
     if is_TX_ready()
        send_TX_data();
     else
        tx_err_handler();
}
```

Once again all tasks are non-blocking, but the error handlers are allowed to be blocking and allowed to generate emergency stops. Typical tasks will be update of PWM duty cycle registers, reading or writing digital or analog ports, reading or writing timers registers, or limited amount of floating point operations without doing length vector/matrix type floating point calculations.

IV. Implementation of the Non-blocking Communication System

In this section, we will summarize our non-blocking communication system implementation. By using the hardware interrupts of the Atmel Atmega2560 microcontroller, the computer may send a message to the device without blocking its main program. Using these interrupts an implementation has been made which will accept short messages containing simple steering instructions such as ”Turn Right 500 steps” or ”Turn Left” or ”Stop”. The primary interrupts being used for these instructions are the Timer/Counter3 Overflow and USART0 Rx Complete vectors. Three pins are used to control the motor, two for direction of rotation (Clockwise/Anticlockwise) and a disable pin which will halt the steering motor. The final pin pulses to control the amount of rotation.

The USART (Universal Synchronous/Asynchronous Receiver-Transmitter) is used by the Atmega2560 as the communication protocol with the main computer over a serial connection via USB. The frame format is of 8bit data with 1 stopbit. These characters, when received by the Atmega2560, are stored in a buffer until a user-defined END_OF_MSG character is received. At that time Timer3 is used to implement a PWM signal that controls the rotation of a motor. This PWM signal is passed onto the pulsing pin and controls the amount of movement.

The API functions based off these requirements are as follows:

- `int numpulses(char *rxbuf)` takes a pointer to the buffer that holds the received message and parses the message to extract the number of pulses, returning it as an integer. If no number is given, then it will return 0.
- `void pulse(int p)` this function, when called with an integer input, will begin pulsing the pin connected to the steering motor the requested number of times by using Timer3 in PWM mode. If 0 is received then it will pulse continuously.
- `void rx_done_callback(char *rxbuf)` called when END_OF_MSG character is received and is passed a pointer to the message buffer. Currently parses the entirety of the received message for the desired action as well as whether we have a number of pulses. The developed low-level library will call this function whenever a complete message is received. However, it is the responsibility of the user to define the message handling logic, e.g. see Fig.3.

V. Conclusion

In this paper, we presented the low-level microcontroller based subsystem, two-way non-blocking communication details, and a lightweight software subsystem based on the co-operative multitasking approach for the Florida Polytechnic Autonomous Golf-Cart. This is a centralized design with a single microcontroller,

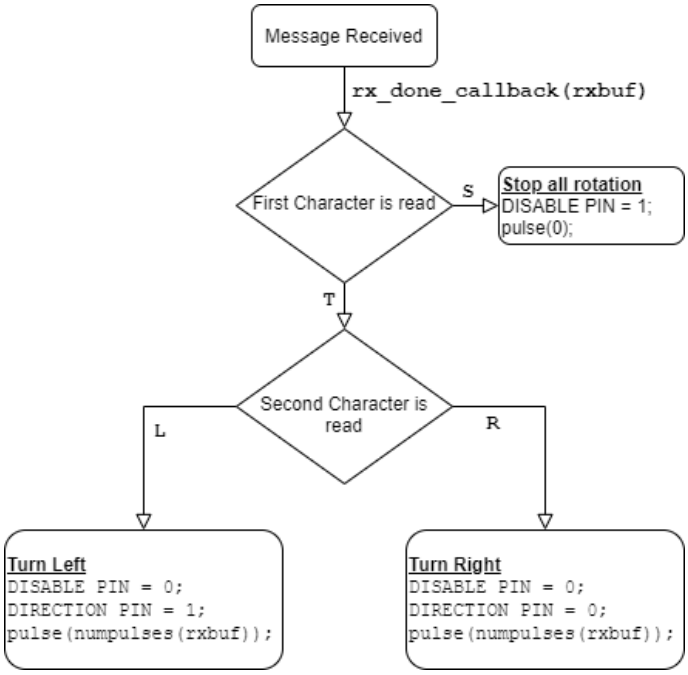

Fig. 3. Example message parsing and handling.

however as future research we are investigating a CAN bus like architecture with a distributed architecture.

VI. Acknowledgements

Authors would like to thank A. Sargolzaei, M. R. Khalghani, and Bruce Hicks for their collaboration, support, and help for the Florida Polytechnic Autonomous Golf-Cart. We would like to thank A. Sargolzaei for initiating the project, and coordinating the early design efforts, M. R. Khalghani and Bruce Hicks for the building the drive-by-wire subsystem, and M. R. Khalghani for designing the solar panel based renewable energy subsystem. This work has been supported in part by NSF grant 1919855, Advanced Mobility Institute grants GR-2000028, GR-2000029, and the Florida Polytechnic University grant GR-1900022.

References

[1] S. El-Tawab, N. Sprague, and A. Mufti, "Autonomous vehicles: Building a test-bed prototype at a controlled environment," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, https://doi.org/10.1109/WF-IoT48130.2020.9221222.

[2] H. Hafez, S. A. Maged, A. Osama, and M. Abdelaziz, "Platform modifications towards an autonomous multi-passenger golf cart," in *2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, https://doi.org/10.1109/NILES50944.2020.9257898.

[3] H. Somogyi, D. Pup, P. Koros, A. Mihaly, and A. Soumelidis, "Research of required vehicle system parameters and sensor systems for autonomous vehicle control," in *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2018, https://doi.org/10.1109/SACI.2018.8441008.

[4] "SURFACE VEHICLE RECOMMENDED PRACTICE," SAE International, 2018, https://www.sae.org/standardsdev/tsb/tsb004.pdf.

[5] A. Hussein, P. Marin-Plaza, D. Martin, A. de la Escalera, and J. Armingol, "Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, https://doi.org/10.1109/IVS.2016.7535372.

[6] I. Shimchik, A. Sagitov, I. Afanasyev, F. Matsuno, and E. Magid, "Golf cart prototype development and navigation simulation using ROS and Gazebo," in *MATEC Web of Conferences*, 2016, https://doi.org/10.1051/matecconf/20167509005.

[7] D. Gaynor, T. Latham, I. Anderson, and C. Johnson, "Autonomous Golf Cart," in *Proceedings of the 2013 ASEE North-Central Section Conference*, 2013, https://doi.org/10.1051/matecconf/20167509005.

[8] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, https://doi.org/10.1109/IROS.2003.1249285.

[9] Z. Gong, W. Xue, Z. Liu, Y. Zhao, R. Miao, R. Ying, and P. Liu, "Design of a Reconfigurable Multi-Sensor Testbed for Autonomous Vehicles and Ground Robots," in *Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, May 2019, https://doi.org/10.1109/ISCAS.2019.8702610.

[10] R. Barker, A. Hurst, R. Shrubsall, G. M. Hassan, and T. French, "A Low-Cost Hardware-in-the-Loop Agent-Based Simulation Testbed for Autonomous Vehicles," in *Proceedings of the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Auckland, New Zealand, Jul. 2018, https://doi.org/10.1109/AIM.2018.8452376.

[11] N. Nie, "The Self-Driving Golf Cart Project," https://neilnie.com/self-driving-golf-cart/.

[12] P. Pisu, "Autonomous Golf Cart Testbed," https://cecas.clemson.edu/pisugroup/autonomous-golf-cart.html.

[13] L. Hardesty, "Self-driving golf carts," http://news.mit.edu/2015/autonomous-self-driving-golf-carts-0901.