2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #37377

WIP: Replication of a 1/5th-Scale Autonomous Vehicle to Facilitate Curriculum Improvement in Cyber Engineering

Dr. Wookwon Lee, Gannon University

Wookwon Lee, P.E. received the B.S. degree in electronic engineering from Inha University, Korea, in 1985, and the M.S. and D.Sc. degrees in electrical engineering from the George Washington University, Washington, DC, in 1992 and 1995, respectively. He is currently a full professor in the Department of Electrical and Cyber Engineering at Gannon University, Erie, PA. Prior to joining Gannon in 2007, he had been involved in various research and development projects in industry and academia for more than 15 years.

Joseph Mendez
Naveen Kumar Manimaran

WIP: Replication of a 1/5th-Scale Autonomous Vehicle to Facilitate Curriculum Improvement in Cyber Engineering

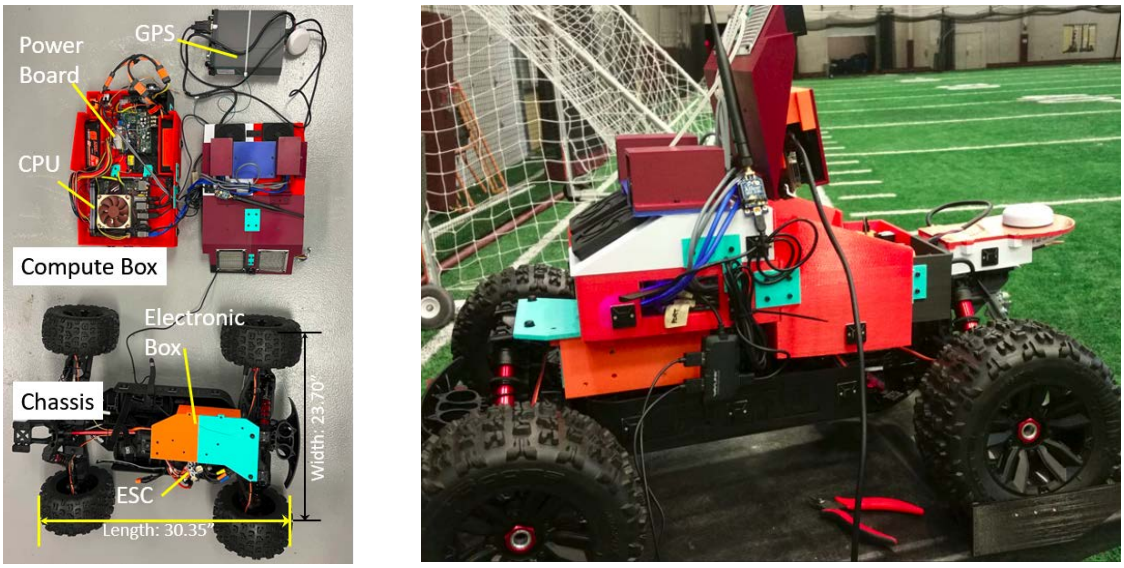
1. Background and Motivation

To respond to the industry trend and the recent nationwide initiative for producing engineering professionals in the cyber domain, our university launched an undergraduate degree program in cyber engineering three years ago. Cyber engineering combines the fundamentals of computer engineering, cryptography, and cybersecurity techniques to design, incorporate, and secure systems across the digital landscape. This includes, but is not limited to, embedded technology, autonomous technology, edge and end-point technologies. Compared to cybersecurity in general, however, cyber engineering still requires further refinement in its curriculum coverage. The current curriculum for the cyber engineering program at our university is centered on cyber physical systems (CPS) and their security including device-level security, boot security, and attack-resilient hardware/middleware. As an engineering curriculum, cyber engineering also requires a variety of hands-on laboratory-based learning as well. To better facilitate hands-on learning in a curricular setting, we have been developing a 1/5th-scale autonomous vehicle as a framework of cyber physical systems for a set of cyber engineering courses. For the development, we have adopted an existing 1/5th-scale autonomous vehicle known as AutoRally [1] which was developed as a high-performance testbed for self-driving vehicle research and funded by DARPA in mid-2010.

When we began our effort in late 2021, however, due to the nature of rapidly advancing products in embedded systems, sensors, and computing technologies, it quickly became clear that numerous parts used in AutoRally had already been discontinued and our effort to replicate it as a CPS platform encountered a series of challenges although much documentation and a detailed parts list were available. Having spent more than a year since then to find alternative comparable parts, construct the mechanical and electrical subsystems, and configure them as needed, our CPS platform is nearly ready for its initial field test on a football field. The overall development efforts have provided us with a great deal of insights on how we may apply our learning during the development phase to the curriculum improvement in cyber engineering. In this paper, we present details of our effort in developing a 1/5th-scale electric autonomous vehicle as a CPS platform from an AutoRally years after the DARPA project was completed. Also, using our cyber engineering curriculum as an example, we present a set of mappings of technical coverage between the CPS platform and core courses in cyber engineering.

2. Overview of 1/5th Autonomous Vehicle Platform

Figure 1 shows (a) the key components of our CPS platform and (b) the current shape of a completely assembled car ready for a field test. The key components are largely grouped into the chassis and the compute box. The chassis holds an electronic box and an electric speed controller (ESC) as well as sensors and batteries (not shown in the figure); the compute box contains a custom-built computer running on the Linux operating environment, a power board for DC-DC conversion from the batteries, and various sensors and electronic devices such as IMU, cameras, Wi-Fi modules, to name a few.



(a) Top view of key subsystems (b) Side view of the platform for a field test

Figure 1. A 1/5-scale autonomous vehicle under development as a Cyber Physical System (CPS) platform

For an autonomous vehicle that is battery-powered and electronically controlled for operation, the initial development of AutoRally began with converting the gasoline engine of a 1/5th-scale remote-control (RC) car into an electric engine. A step-by-step guide to construct the chassis of AutoRally is available in [2]. Since, as one of the challenges we encountered, however, the RC car with the gasoline engine was discontinued, we chose an RC car with an electric engine [3] for our further modification and development in reference to the step-by-step guide for the chassis of AutoRally. For the construction of a computer box, we followed a step-by-step guide in [4] as well as the software setup instructions [5], the operational procedure [6], and the configuration instructions in [7]. These documents served as very useful references for our development effort but also to some extent, due to a set of new off-the-shelf devices replacing obsolete ones, gathering additional information on our own was necessary. Below we present a summary of our activities and additional learning or clarifications needed to complete key tasks, while pointing to the referenced sources of the information as much as appropriate to avoid repeating the same instructions in this paper.

3. Key Activities for a Successful Development

3.1. Chassis

To operate the chassis with the battery, it was necessary to construct a capacitor pre-charge circuit according to the chassis instructions (pp. 6-9, [2]). For this, however, the battery leads were not altered but, instead, IC5 connectors with proper current ratings were used. This pre-charge circuit with an additional resistor in series with the capacitor is to prevent a spark from occurring when plugging in the batteries. The following shows a list of subsequent steps to complete the chassis:

- With the Castle Serial Link and Castle Link software downloaded from the Castle homepage (castlecreations.com), configure the ESC and Castle Serial Link device according to the instructions (pp. 36-37, [2])

- Configure the Futaba transmitter according to the instructions (pp. 35-36, [2]) and verify that the transmitter connects to the receiver.
- Connect the motor, ESC, receiver, transmitter, and batteries; calibrate the ESC according to the Castle ESC manual on line [8].
- Construct the Protoshield Assembly (pp. 37-45, [2]) and the Electronics Box circuit (p. 46, [2]). This step required much effort and time for accuracy and troubleshooting.
- With the code for the Arduino Due (i.e., autorally_chassis.ino) downloaded onto the device and perform minor tweaking for a simple remote-controlled operation in a lab setting, verify that the transmitter successfully switches between auto and manual using the buttons on the transmitter.

3D printing of the electronics box base (2 pieces), the electronics box lid (2 pieces), and the relevant support pieces was done with Fusion. Installing the electronics box components such as Protoshield assembly, multiplexer, relay, Castle serial link device, and receiver into the separate halves of the electronics box base was straightforward as well as mounting the Electronics Box on the chassis frame. 3D-printing the magnet holders (x4), hex adapters (x4), front hall mounts (x2), and rear hall mounts (x2), and testing the hall effect sensors was based on the instructions (pp. 24-28, [2]) although our parts were slightly different. As for the amount of time and effort on this development phase, all of this work was performed by a master-level graduate student for a period of about 2 months.

3.2. Compute Box

Compared to the construction of the compute box, the work on configuring the compute box was much more challenging for our development team of 2 master-level graduate students. For the compute box operation on the CPS platform, another Linux-based computer is required for the remote control and configuration purposes, which is referred to as the Operator Control Station (OCS) laptop. A summary of our activities to initially construct the compute box is provided below.

- Construction of a custom-built (mini-ITX form factor) computer running on the Linux operating environment.
- 3D printing of parts with Fusion for the compute box base and lid, giving attention to various mounting hole diameters for specific inserts; 3D printing of GPU cover/holder, microcontroller holder, and GPS box/lid.
- Construction of all cable assemblies (pp. 28-46, [4]), keeping cables longer than specified in the manual to allow for error during testing and ease of installation.
- Download Teensyduino [9]; construct the Run-stop Box (pp. 46-49, [4]); load the corresponding Arduino file onto the Teensy-LC; also load the other Teensy-LC board with the camera trigger Arduino file. Note that these Arduino files are available on a specific file path once the software tools [7] are downloaded and installed on the Linux-based computer in the compute box.
- Connect the power switch assembly, the hot swap board, the 2-pin power connector, and the power supply board. For lab testing, all other cables were disconnected from the power supply board and a lab power supply is used to apply ~22.6V to the 2-pin connector (Refer to Step ‘e’ (p. 85, [4]). Once the power supply board has been successfully powered with the lab power supply, follow steps ‘f’ through ‘l’ (pp. 85-87, [4]).

- Once successful, connect the rest of the compute box components per the “Installation and Routing” instructions (p. 50, [4]). It should be noted that the lab power supply must be able to supply at least 6 amps to allow the compute box to turn on and run at full load.

For the configuration of the CPS platform following the instructions in [5] and [7], the software setup is necessary on both the compute box and the OCS laptop and some other configuration steps are required only on one of the computers. The complete configuration steps can be summarized in the following 20 steps [7] - 1. Install Tools; 2. Configure IP addresses and `ssh` permissions; 3. Clock synchronization setup (`chrony` and `gpsd`); 4. Set `roscore` to auto start; 5. Set AutoRally `udev` rules; 6. Setup Compute Box Data Drive; 7. Change Power Button Behavior; 8. Disable Login and Lock Screen Password Prompts; 9. Setup on-board sensors; 10. Install M4api and Configure Cutoff Voltage; 11. Setup Cameras; 12. Configure XBees; 13. Configure GPS; 14. Configure Chassis Microcontroller; 15. Configure Compute Box Microcontroller; 16. Configure Run-stop Microcontroller; 17. Configure and Calibrate IMU; 18. Configure GPU; 19. Configure Platform-Agnostic Launch System; and 20. Verification.

Among these steps, some of the major challenges we encountered were for cameras, XBees, and GPS. For the cameras, the links provided in the instructions manual were obsolete and no longer active. The difficulty in configuring XBees was primarily due to the global shortage of the semiconductor devices and thus the availability of a specific model, XBee-PRO 900HP (S3B), and lack of instructions for other models of XBees for possible application to our CPS platform. The difficulty in configuring the GPS device was that the GPS device available and acquired in Jan. 2022 (Hemisphere GNSS P/N 940-4137-10 Phantom 34 Module) for our CPS platform was outputting the GNSS position data, e.g., with a prefix of $GNGNS, while the GPS device used in AutoRally and its instructions (Hemisphere Eclipse P307 GPS) was outputting the GPS positioning data, e.g., with a prefix of $GPGNS. Also, the AutoRally code was filtering out all $xxGNS messages other than $GPGNS. Errors were cleared once $GNGNS messages were allowed to be processed by the code and some delay adjustments were made in reading the incoming GNSS data through the COM port since the `udev` rules for the GPS port didn’t work in our application.

4. Relevance to Cyber Engineering Curriculum

In the emerging field of cyber engineering, embedded systems play a key role in technological advances and engineering education. The configuration and operation of the CPS platform require fundamental knowledges and technical skills in the Linux operating environment and interfacing with embedded systems that are placed on the CPS platform for the purposes of autonomous driving. With direct access to all configuration details and operational aspects, our CPS platform has a great potential to contribute to improving the cyber engineering education. Below using our cyber engineering curriculum as an example, we provide a set of content mappings between the technical knowledges that the CPS platform presents for student learning and possible integration of them into the course coverage.

Our BS program in cyber engineering requires 63 credit hours of major course work among 124 credit hours for the degree. Table 1 shows a set of major courses for the curriculum, excluding mathematics, science, and liberal studies courses [10].

Table 1. Major Courses in the Curriculum for BS in Cyber Engineering

FRESHMAN	
Fall	Spring
3 Intro to Networks/CIS 290	3 Digital Logic Design/ECE 140
	1 Digital Logic Design Lab/ECE 141
	3 Circuit 1/ECE 228
	1 Circuit 1 Lab/ECE 229
	3 Intro to C/C++/ECE 111
	1 Network Security Lab/CYSEC 101
SOPHOMORE	
Fall	Spring
3 u-controller Applications with IoT/ECE 245	3 u-controller Essentials for Cyber Appl/CYENG 225
3 Data Structure and Algorithm/ECE 217	**3 Embedded OS Appl. Programming/CYENG 220**
JUNIOR	
Fall	Spring
3 Trusted OS/CYENG 312	**3 Tech Selective**
3 Intro to Cyber-physical Syst/CYENG 237	1 Professional Seminar/ECE 380
1 Project Experience/ECE 381	**3 Secured Embedded System/CYENG 350**
3 Test, Measurement, and Control/ECE 243	
SENIOR	
Fall	Spring
3 Tech Elective 1	3 Technical Elective 2
3 Senior Design I/ECE 357	3 Senior Design II/ECE 358

In particular, the following CYENG core courses are relevant to the CPS platform and further development could be facilitated:

- Embedded OS Appl. Programming/CYENG 220
- Trusted OS/CYENG 312
- Intro to Cyber-physical Syst/CYENG 237
- Secured Embedded System/CYENG 350
- Technical Selective -- Embedded Kernel and RTOS/ ECE 311

Although two other sophomore-level classes, u-controller Applications with IoT/ECE 245 and u-controller Essentials for Cyber Appl/CYENG 225 could also use the CPS platform, it is not envisioned for course improvement as acquiring the knowledge about micro-controllers and applications can be achieved with individual embedded systems devices.

Embedded OS Application Programming/ CYENG 220 teaches the student how to architect an embedded Linux environment for a distributed co-operating multi-application environment. The course explores how to leverage the Linux programming, inter-process communication, and shell programming. Topics also include bootup, scheduling of applications, and load balancing across multiple cores. This course is a good fit to perform the configuration and operation of the OCS laptop as the CPS platform is configured via close communication between the compute box and

the OCS laptop both running on the Linux environment and interacting with various embedded systems.

Trusted OS/ CYENG 312 covers basic understanding and configuration for hardening and securing an embedded Linux operating system. Topics include boot-time configurations and forensics, user and directory hardening, application vulnerability minimization, and minimizing memory attacks. The course will focus on a common Linux distribution architecture, security modules, cryptography tools, and how the system works. The CPS platform can be the playground for applying the knowledges covered in this course in order to improve the security of the embedded systems on the platform as well as serve as the real-time system for improving student learning experiences in the course.

Introduction to Cyber-Physical Systems/ CYENG 237 covers cyber and physical systems developed via high-level modeling and virtual/real prototyping using MATLAB/Simulink as well as real prototyping of an autonomous driving robot for advanced implementation and verification. Although not mentioned earlier, the operation of AutoRally and thus, the CPS platform can be simulated and verified in an autonomous driving simulator. The virtual/real CPS devices created in this course can be integrated into the CPS platform to re-enforce student learning with a small autonomous robot being currently used in this course via our CPS platform that is much more complex.

Secure Embedded Systems/ CYENG 350 provides a hands-on approach of understanding cyber-attacks using only the processing power and memory of resource-constrained embedded devices, architecting and implementing a root of trust (RoT) embedded system from power-up, firmware launching, boot-loading, and applications following the various industry-trusted system paradigms. Although not directly related to configuration of the CPS platform, this course can use the CPS platform as a playground for students applying the knowledges covered in this course such as implementing and experiment with a root of trust.

Embedded Kernel and RTOS/ ECE 311 covers basic understanding of embedded kernel and real-time operating system paradigms. Topics include process management, process synchronization, and memory management. For this course, embedded kernel topics can be implemented on the CPS platform serving as an embedded-system platform and RTOS topics can be easily implemented on the real-time operating systems of the OCS laptop and compute box of the CPS platform.

The current instructions and hands-on learning in these courses are based on unit devices or relatively-simple, small-scale subsystems. Our CPS platform serves as a complete complex system for the primary functionality of autonomous driving with various subsystems and sensors integrated. As such, our CPS platform is considered to be an excellent framework for our project-based courses and also improved student learning experiences.

5. Concluding Remarks

We have presented a summary of development effort to create a $1/5^{th}$-scale autonomous driving vehicle as the CPS platform for curriculum enhancement in our cyber engineering program. The primary technical challenges in our development arose from the fact that some of the key

components of the vehicle became obsolete and/or discontinued. As such, part of the work done for the self-driving features of the original AutoRally required revision with much effort. In return, however, it provided us with an opportunity to acquire in-depth knowledges that can be applied to improving course coverage and thus curriculum in cyber engineering education. We hope that the information presented in this paper is useful to educators in cyber engineering in general as well as those in embedded and cyber physical systems intending to create a complex educational platform of hardware and software for their cyber/computer engineering curriculum.

References

[1] AutoRally: A high-performance testbed for self-driving vehicle research. [on line] https://autorally.github.io/

[2] Geogia Institute of Technology, AutoRally Chassis Instructions, ver. 1.4, June 2018. [on line] https://github.com/AutoRally/autorally_platform_instructions

[3] Horizon Hobby, KRATON 1/5 4WD EXtreme Bash Roller: Instruction Manual. [on line] https://www.arrma-rc.com/en/product/1-5-kraton-4wd-extreme-bash-roller-black/ARA5208.html.

[4] Geogia Institute of Technology, AutoRally Compute Instructions, ver. 1.4, March 2018. [on line] https://github.com/AutoRally/autorally_platform_instructions

[5] Software for the AutoRally platform. [on line] Last accessed on Feb. 12, 2023 at: https://github.com/AutoRally/autorally.

[6] Geogia Institute of Technology, AutoRally Platform Operating Procedures, ver. 1.3, February, 2017. [on line] https://github.com/AutoRally/autorally_platform_instructions

[7] Geogia Institute of Technology, Platform Configuration Instructions, ver. last edited, July 1, 2020. [on line] https://github.com/AutoRally/autorally/wiki/Platform%20Configuration%20Instructions.

[8] Tech Tip: Castle ESC Calibration – When, Why, Why and How. [online] Last accessed on Feb. 12, 2023 at: https://home.castlecreations.com/blog/2020/1/9/tech-tip-castle-esc-calibration

[9] Teensyduino: Arduino 2.0.x Software Development, ver. 1.57. [on line] Last accessed on Feb. 12, 2023 at: https://www.pjrc.com/teensy/td_download.html.

[10] Undergraduate Catalog, Gannon University, 2022-2023.