High Assurance Controller of Self-balancing Robot

Capstone Project

Sponsored by: Matt Clark and Michal Podhradsky, Galois: galois.com

Background and Motivation

“The Department (of the Air Force)’s approach to software acquisition still trails current industry standards. Modern development tools have the potential to speed up both production and spiral upgrades while reducing cost, but leveraging them remains a challenge as top talent often lies outside the traditional Defense Industrial Base. Artificial intelligence (AI) will fundamentally change the character of warfare, so future Airmen must have systems that learn faster than their enemies. To harness this technology from commercial industry, we must design, acquire, and update software like them.”[1]

Autonomous cyber-physical systems provide the highest degree of challenge as tools and practices to design critical software with high assurance are in their infancy. Studies have shown that although some tools exist within this domain to make high assurance software, often graduating engineers do not understand how to implement these tools in a realistic design process.[2] This capstone project is exploratory, as Galois is looking for a demonstration of a workflow for a verification of inner control loops in cyber-physical systems.

The classic inverted pendulum problem is a long standing control problem that requires active balancing and non-linear control.[3] Verification of non-linear control systems are typically performed using several methods leveraging:

1. Stability, Observability, and Controllability using a set of linearized models[4]
2. Lyapunov stability of non-linear models[5]
3. Extensive simulation

---

[1] William Roper, “PRESENTATION TO THE HOUSE ARMED SERVICES COMMITTEE U.S. HOUSE OF REPRESENTATIVES”, 2018, https://docs.house.gov/meetings/AS/AS00/20180307/106892/HHRG-115-AS00-Wstate-RoperW-20180307.pdf

[2] Davis, Jennifer A., et al. "Study on the barriers to the industrial adoption of formal methods." *International Workshop on Formal Methods for Industrial Critical Systems*. Springer, Berlin, Heidelberg, 2013. http://loonwerks.com/publications/pdf/cofer2013fmics.pdf

[3] Pathak, Kaustubh, Jaume Franch, and Sunil Kumar Agrawal. "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization." *IEEE Transactions on robotics* 21.3 (2005): 505-513 http://www.academia.edu/download/4133309/p61.pdf

[4] Lavretsky, Eugene. "Adaptive control: Introduction, overview, and applications." *Lecture notes from IEEE Robust and Adaptive Control Workshop*. 2008. https://www.cds.caltech.edu/archive/help/uploads/wiki/files/140/IEEE_WorkShop_Slides_Lavretsky.pdf

[5] Ibanez, Carlos Aguilar, O. Gutierrez Frias, and M. Suarez Castanon. "Lyapunov-based controller for the inverted pendulum cart system." *Nonlinear Dynamics* 40.4 (2005): 367-374. http://www.wseas.us/e-library/conferences/venice2004/papers/472-109.pdf

These verification methods often rely on model based abstractions of the true dynamics. These model based abstractions do not segregate between uncertainty introduced by the design implementation, errors in software implementation, and environmental uncertainties. Often, sampling time, hardware variations are coupled with true environmental uncertainty, thereby putting more strain on the robustness requirements of the system.

The challenge of this project is to design and build a well understood, nonlinear active self-balancing (inverted pendulum) robot that, through software design and verification methods, is more robust and has increased performance over a baseline PID controller. The objective of this project is to increase the overall stability and robustness of the control design by isolating and verifying the underlying system design assumptions through modern software verification techniques, leveraging model checking techniques, and the choice of higher reliability processing infrastructures. You will describe how enforcing a strict sampling time, well formed algorithms, and adherence to specific invariants in the software make the system more efficient, the control algorithms tighter and more robust to environmental uncertainties.

This assignment will require you to leverage both the Rust programming language[6] and the HiFive1 Risc based Microprocessor.[7] The HiFive1 RISC-V based processor is Arduino compatible thereby allowing for the use of readily available hardware for sensor inputs.

To verify the correctness of your implementation, it is recommended that you create a model that is verifiable first. Several open source verification tools are capable of analyzing system models using model checking and analysis techniques. One verification tool you can use is the Kind2 model checker for verification. Your entire system should be modeled in a comparable modeling framework suitable for designing linear and nonlinear control systems. Some Matlab alternatives are GNU Octave, and Python with the Python Control Toolbox. [8] [9] [10]

Invariants must be generated during the continuous control design, such as boundaries on closed loop stability, and carried through to the discrete design implementation. It is also, highly valuable to separate your control design into a singular block or subsystem described as the “System Under Test” such that invariants can be modeled in Kind2. If you choose to simulate your system in Matlab Simulink, it is valuable to develop the control algorithms you plan to implement in real hardware as simulink Interpreted MATLAB functions (or matlab m-file functions). This enables cleaner architectural design approaches when translating simulation code to implementation code.

---

[6] Rust type safe language, https://www.rust-lang.org/en-US/

[7] Rust on the HiiFive Risc based processor, https://softsourceconsulting.github.io/rst/from-zero-to-rust-on-riscv.html

[8] https://www.gnu.org/software/octave/

[9] https://www.mathworks.com/products/simulink.html

[10] https://python-control.readthedocs.io/en/latest/

Objectives and Learning Outcomes

The team that works on this project will, with the support of Galois engineers, achieve the following objectives and learning outcomes. *Outcomes that are italicized are optional and will be supported/encouraged if there are team members with interest in the topic and relevant skills.*

- Gain experience blending formal verification with simulation using Octave and Kind2. Gain expertise in embedded Rust
- Gain experience with RISC-V microcontrollers
- Gain experience with advanced and nonlinear control
- Develop a simple self-balancing robot, design interface software components in Rust for interfacing sensors and motor actuation.
- Develop environment interface models and hardware abstractions for the Octave / Matlab simulation environment. Model the functional blocks in Kind2 / Lustre and compare closed loop simulation with invariant checks.
- Develop a nonlinear controller in Octave / Matlab m-files and Rust and formally verify as many properties as possible.
- *Develop a suite of verified nonlinear controllers and compare their performance with linear variants.*
- *Also, further simulation models could be realized using V-REP.*[11]
- Galois is hiring, and a successful capstone project could lead to an internship/permanent position.

Milestones

The team is expected to meet these milestones. *The stretch goals that are italicized are optional*. Each milestone has to be extensively documented so it can be reproduced **without** any input from the student team. The documentation is a hard requirement, and the milestone cannot be considered to be achieved without being properly documented. We hope the team will work on both parts A and B simultaneously. Galois will cover the cost of hardware, all required software is open source.

Part A: Hardware & Software

- Setup a build environment to compile and flash Rust code on HiFive1 RISC-V board[12]
- Construct a simple self-balancing robot similar to https://maker.pro/arduino/projects/build-arduino-self-balancing-robot using HiFive1 board instead of Arduino.

[11] http://www.coppeliarobotics.com/downloads.html
[12] Either bare-metal or using FreeRTOS, see https://github.com/riscv-rust/riscv-rust-toolchain/issues/29

- Identify the input / output relationship between the controller software that resides in the HiFive1 board and the external system interfaces (sensor accuracy, update rate, mathematical equivalent relationship of the sensors, slew rate of the motor / gears etc)
- Write a simple feedback controller (PID) that attempts to balance the robot. Document accuracy and performance characteristics.
- *Utilize Kind2's Rust code generation capability to synthesize code function definitions and overarching guarantees from the formal model.*

Part B: Control & Verification

- Develop and identify a model of a nominal self-balancing robot in Octave or Matlab. Reusing existing models is encouraged.
- Isolate the "Software under Test" as a severable block to analyze only the linear / non-linear controller for specific properties
- *Attempt to leverage Rust quickcheck[13] to verify the properties developed in Kind2 in the source files after the control design is implemented*
- Develop and identify a dynamics model of the robot in Octave or Matlab
- *Alternately, Rust code into a LEAN theorem definitions may be helpful.*[14]
- Develop and verify a nonlinear controller and compare its performance with the PID controller through both simulation and in the real system.

Student Skills

A team that tackles this ECE project will need team members that have expertise in, or are willing to build upon intermediate expertise in, each of the following:

- Embedded systems programming and debugging in Rust and C
- Control loop theory including advanced and nonlinear control
- Learn the concepts of applied formal methods and their realization via Kind2 and quickcheck.

[13] https://docs.rs/quickcheck/0.7.2/quickcheck/
[14] https://github.com/Kha/electrolysis