


GSoC Proposal for BeagleBoard.org

BeagleBoard.org
Feb 11, 2026

Table of contents

Chapter 1

Introduction

Extending `librobotcontrol` and `rcpy` support to BeagleBone AI, BeagleBone AI-64 and BeagleV-Fire with enhanced GPIO implementation. Maintaining real-time control via hardware-acceleration (PRU/FPGA).

1.1 Summary links

- **Contributor:** Nunnapas Temridiwong
- **Mentors:** Jason Kridner, Deepak Khatri
- **Code:** TBD
- **Documentation:** TBD
- **GSoC:** TBD

1.2 Status

This project is currently just a proposal.

1.3 Proposal

1.4 About

- **Forum:** u/ntem (Nunnapas Temridiwong)
- **OpenBeagle:** ntemridiwong (Nunnapas Temridiwong)
- **Github:** NunnapasTem (Nunnapas Temridiwong)
- **Discord:**
- **School:** University of Florida
- **Country:** United State
- **Primary language:** English
- **Typical work hours:** 9AM-6PM US Eastern
- **Previous GSoC participation:** First Time Applicant

Chapter 2

Project

Project name: Extending `librobotcontrol` and `rcpy` support to BeagleBone AI, Beagle Bone AI-64, and BeagleV-Fire - Nunnapas Temridiwong

2.1 Description

librobotcontrol is a C library designed to facilitate robotics software development on Linux embedded systems. It provides APIs for interacting with hardware elements generally used in robotics application, including motor controls, real-time operations, and communication protocols (e.g. I2C, UART, and SPI). Currently, *librobotcontrol* primarily supports AM335x-based boards such as BeagleBone Black/Blue. However, the library still faced several challenges:

- **Depreciated and SoC-specific I/O Interfaces** - The I/O interface still relies on Linux sysfs.
- **Lack of Support for New Hardware** - Although `librobotcontrol` v1.1 has been initiated to port the library to BeagleBone AI (BBAI), BeagleBone AI-64 (BBAI-64), and BeagleV-Fire (BB-Fire), this version is yet to be complete and fully tested.

rcpy is a Python 3 library with binding functionalities to `librobotcontrol`. Similar to its peer, the library currently does not provide support for BBAI, BBAI-64, and BeagleV-Fire.

Thus, the goal of this project is to extend support for `librobotcontrol` and `rcpy` to newer boards and restructure the library to be more portable for future development.

With the following goals in mind, these are the project's core objectives:

1. **Extend hardware support of `librobotcontrol` and `rcpy` to BBAI, BBAI-64, and BeagleV-Fire.**
    - Complete porting the library, initiated in v1.1, to BBAI/BBAI-64/BeagleV-Fire.
    - Clean up the existing modifications in v1.1.
    - Implement BeagleV-Fire real-time solution.
2. Migrate I/O codebase from *sysfs* to native implementation against character device driver.
3. Optimize math functions for DSP units available in newer boards.
4. Update documentations, examples and testings.

2.2 Expand Hardware Support

1. **librobotcontrol v1.1 Validation**

Initially, every hardware interface implemented in v1.1 will be cross-checked against the AM5729 (BBAI) and TDA4VM (BBAI-64) technical reference manuals to ensure register mappings, clock configurations, and peripheral control logic match the silicon specifications. This validation includes ensuring pinmux settings, power domains,

and clock gating are appropriately configured for each peripheral (e.g., PWM, I2C, SPI, UART). Special attention will be given to memory-mapped I/O regions and the correctness of device tree overlays. The PRU (Programmable Real-Time Unit) firmware for both AM5729 and TDA4VM will be reviewed and restructured to isolate board-specific behavior, enabling modular reuse across platforms. Once validated, PRU-accelerated functions such as encoder reading and PWM generation will be benchmarked against existing implementations to verify their performances.

2. **BeagleV-Fire Real-time solution**

Since BeagleV-Fire does not own built-in PRUs, I plan to leverage the Low-latency I/O RISC-V CPU core in FPGA fabric project implemented by Atharva Kashalkar for PRU-accelerated tasks. This dedicated processing unit runs independently from the main Linux system, enabling deterministic control loops required for real-time robotic applications. A lightweight RPC mechanism will be developed to communicate between the main processor and the real-time core via shared memory or character device interface.

2.3 I/O Native Implementation

To replace the deprecated `sysfs` interface and ensure long-term portability across architectures, low-level I/O access will be re-implemented using direct system calls via native assembly for each supported architecture, including armhf (for BBAI), aarch64 (for BBAI-64), and riscv64(for BeagleV-Fire). The implementation will focus on interacting with character device interfaces through raw ioctl system calls. This architecture-specific approach will not only modernize the I/O module of `librobotcontrol` but also increase its portability to other SoCs and emerging RISC-V platforms.

2.4 DSP Optimization

To fully utilize the onboard DSP cores available in BBAI (C66x) and BBAI-64 (C7x), the math module will be utilizing TI's highly tuned DSP libraries. This includes DSPLIB for C66x, and TIC7x DSPLIB for C7x. These libraries will be integrated with `ibrobotcontrol` in a modular way, allowing for conditional compilation depending on hardware capabilities.

2.5 Software

- C programming
- Assembly (ArmHF, Aarch64, riscv64)
- Visual Studio Code
- QEMU
- Device Tree
- TI DSP Library
- FPGA
- Python 3

2.6 Hardware

- BeagleBone AI, BeagleBone AI-64, BeagleBone Fire
- BeagleBone Robotics Cape
- Logic Analyzer
- JTAG Debugging

Chapter 3

Timeline

Provide a development timeline with 10 milestones, one for each week of development without an evaluation, and any pre-work. (A realistic, measurable timeline is critical to our selection process.)

Note: This timeline is based on the official GSoC timeline

3.1 Timeline summary

Date	Activity
Feb 27 - Mar 24	Connect with possible mentors and request review on first draft
Mar 25 - Apr 8	Complete prerequisites, verify value to community and request review on second draft
Apr 9 - May 7	Complete prerequisites, verify value to community and request review on second draft
May 8 - Jun 1	*Community Bonding Period (May 8 – June 1)*
Jun 2 - Jun 8	*Introductory video (June 2 - June 8)*
Jun 9 - Jun 15	*Milestone #1: Validate librobotcontrol v1.1 (June 9 - June 15)*
Jun 16 - Jun 22	*Milestone #2 – Validate librobotcontrol v1.1 and refactor PRU firmware (June 16 - June 22)*
Jun 23 - Jun 29	*Milestone #3 – BeagleV-Fire RT core communication and basic RPC (June 23 - June 29)*
Jun 30 - July 6	*Milestone #4 – Native I/O implementation in assembly (June 30 - July 6)*
July 7 - July 13	*Milestone #5 – Midterm wrap-up and documentation (July 7 - July 13)*
July 14 - July 18	*Submit Midterm Evaluations (July 14 – July 18)*
July 19 - July 25	*Milestone #6 – DSP integration and optimization (July 19- Aug 25)*
July 26 - Aug 1	*Milestone #7 – rcpy compatibility and bindings update (July 25 - Aug 1)*
Aug 2 - Aug 8	*Milestone #8 – Upstream preparation and testing (Aug 2 - Aug 8)*
Aug 9 - Aug 15	*Milestone #9 – Final testing and bug fixing (Aug 9 - Aug 15)*
Aug 16 - Aug 24	*Milestone #10 – Final submission (Aug 16 - Aug 24)*
August 25	*Final Submission (Aug 25)*

3.2 Timeline detailed

3.2.1 Community Bonding Period (May 8 – June 1)

- Engage with the BeagleBoard and librobotcontrol communities via Discord and the Beagle Forum.
- Study hardware documentation for BBAI (AM5729), BBAI-64 (TDA4VM), and BeagleV-Fire (MPFS025T).
- Analyze existing work in `librobotcontrol v1.1` and identify architectural dependencies.
- Set up cross-compilation toolchains and QEMU for BBAI, BBAI-64, and BeagleV-Fire.
- Finalize the project roadmap, break down the objectives with mentor feedback, and draft a development strategy.

3.2.2 Introductory video (June 2 - June 8)

- Record an introductory video outlining project goals and background.

3.2.3 Milestone #1: Validate librobotcontrol v1.1 (June 9 - June 15)

- Set up cross-compilation toolchains for armhf, aarch64, and riscv64.
- Verify existing librobotcontrol v1.1 builds and runs on BBAI and BBAI-64.
- Begin initial testing of GPIO, PWM, and I2C interfaces on all target boards.

3.2.4 Milestone #2 – Validate librobotcontrol v1.1 and refactor PRU firmware (June 16 - June 22)

- Begin refactoring PRU firmware for BBAI and BBAI-64, isolating board-specific logic.
- Ensure encoder and PWM examples run deterministically using PRU.
- Benchmark PRU-based functionality against AM335x-based implementation.

3.2.5 Milestone #3 – BeagleV-Fire RT core communication and basic RPC (June 23 - June 29)

- Study Atharva's GSoC project for FPGA-based RT core on BeagleV-Fire.
- Implement and test a RPC mechanism using shared memory or character devices.
- Validate backward compatibility for real-time I/O interactions.
- Create mock test applications to demonstrate communication between main CPU and RT core.

3.2.6 Milestone #4 – Native I/O implementation in assembly (June 30 - July 6)

- Implement character device access and ioctl calls in assembly for armhf (BBAI), aarch64 (BBAI-64), and riscv64 (BeagleV-Fire).
- Abstract architecture-specific system calls with clean C bindings.
- Begin integration testing across all three platforms.

3.2.7 Milestone #5 – Midterm wrap-up and documentation (July 7 - July 13)

- Finalize and test all hardware backends for PRU and RT-core support.
- Submit a detailed midterm report with architecture diagrams, benchmarking results, and test coverage.
- Update library documentation for new board support.

3.2.8 Submit Midterm Evaluations (July 14 – July 18)

- Submit midterm evaluation via GSoC portal.
- Meet with mentor for feedback session and realignment for second phase.

Important: **July 12 - 18:00 UTC:** Midterm evaluation deadline (standard coding period)

3.2.9 Milestone #6 – DSP integration and optimization (July 19- Aug 25)

- Integrate TI DSPLIB for C66x on BBAI and C7x DSPLIB on BBAI-64.
- Offload compute-heavy math routines to DSP (e.g., filters, PID controller).
- Enable conditional compilation for DSP acceleration.
- Benchmark CPU vs DSP performance for selected tasks.

3.2.10 Milestone #7 – `rcpy` compatibility and bindings update (July 25 - Aug 1)

- Update rcpy Python bindings to support newer board APIs.
- Ensure compatibility with both new native I/O backend and legacy API.
- Refactor documentation and examples to highlight Python usage on BBAI and BBAI-64.

3.2.11 Milestone #8 – Upstream preparation and testing (Aug 2 - Aug 8)

- Clean up and modularize changes for `librobotcontrol`.
- Submit patches and PRs to upstream repositories (librobotcontrol & rcpy).
- Perform integration tests with Robotics Cape on all target boards.
- Gather community feedback for API changes and improvements.

3.2.12 Milestone #9 – Final testing and bug fixing (Aug 9 - Aug 15)

- Conduct end-to-end testing for robotics scenarios.
- Address any bugs found during testing.
- Finalize board-specific documentation and setup instructions.

3.2.13 Milestone #10 – Final submission (Aug 16 - Aug 24)

- Record final project demo video showing upgraded `librobotcontrol` running on all supported boards.
- Submit final code, docs, and deliverables to GSoC portal.
- Write up key learnings and recommendations for future work.

3.2.14 Final Submission (Aug 25)

- Submit final project video and final work to GSoC site.
- Complete final mentor evaluation.

Important: **August 19 - 26 - 18:00 UTC:** Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

August 26 - September 2 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

3.2.15 Initial results (September 3)

Important: **September 3 - November 4:** GSoC contributors with extended timelines continue coding

November 4 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 11 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with extended deadline

Chapter 4

Experience and approach

Through my undergraduate studies in computer engineering, I've gained hands-on experiencesin embedded systems, low-level development, and c programming:

- Worked on a year-long animatronic project involving real-time motor control, sensor feedback, and hardware integration.
- Conducted research in side-channel attacks, gaining experience with hardware timing and low-level system behavior.
- Developed and worked with a custom RTOS, implementing priority scheduling, inter-process communication (IPC), and multi-threading to support real-time operations.
- Comfortable reading technical reference manuals, and datasheets for SoC-level hardware bring-up and validation.
- Experienced in using a logic analyzer for debugging protocols such as I2C, SPI, and UART.

4.1 Contingency

I will utilize the following resources in case I get stuck on my project and my mentor isn't available:

- BeagleBone AI, BeagleBone AI-64, BeagleV-Fire docs.
- BeagleBone Device Tree.
- am57xx, TDA4VM, MPFS025T TRMs.
- BeagleBoard community, and robotics communities.
- Linux kernel mailing list.

4.2 Benefit

This project will extend supports to more BeagleBoard devices as well as significantly improves the `librobotcontrol`'s and `rcpy`'s capabilities:

- Adding architecture-specific supports which increases the library's long-term viability.
- Replacing Depreciated `sys` with faster character device driver.
- Fully utilizing DSP hardware available in the newer boards.
- Maintaining real-time control via PRUs and FPGA-based RISC-V. Utilizing previous GSoC project.
- Improving documentations and testings.

4.3 Misc

- Submitted a Pull Request for Cross Compilation: #211