


GSoC Proposal for BeagleBoard.org

BeagleBoard.org
Feb 11, 2026

Table of contents

Chapter 1

Introduction

Librobotcontrol is a C library package that provides examples and testing utilities for robotic control projects using BeagleBone capes like the Robotics Cape from BeagleBoard.org. The BeagleBone Black, fully supports librobotcontrol through device tree overlays, enabling seamless hardware access.

BeagleBone AI supports librobotcontrol, but its implementation remains incomplete. To ensure proper functionality, drivers must be tested, and necessary modifications applied. Similarly, BeagleBone AI-64 can run librobotcontrol, but it lacks a refined implementation.

Librobotcontrol support of BeagleV-Fire is also to be verified and completed. Enhancing its cape gateware would improve flexibility and enable better compatibility with the Robotics Cape.

This project aims to update librobotcontrol for BeagleBone AI, BeagleBone AI-64, and BeagleV-Fire SBCs, refining device tree overlays and software support to bring Robotics Cape functionality closer to that of the BeagleBone Black.

Chapter 2

Summary links

- Contributor: Prithvi Tambewagh
- Mentors: Jason Kridner, Deepak Khatri
- Code: TBD
- Documentation: TBD
- GSoC: TBD

Chapter 3

Status

This project is currently a proposal.

Chapter 4

Proposal

- Created accounts across OpenBeagle, Discord and Beagle Forum
- The PR Request #204 for Cross-Compilation for ARM-Linux Platform
- Created the Project Proposal according to the prescribed template : Proposal Template

Chapter 5

About

- Forum: prithvitambewagh (Prithvi Tambewagh)
- OpenBeagle: prithvi_t (Prithvi Tambewagh)
- IRC: prithvit (Prithvi Tambewagh)
- Github: rkt-1597 (Prithvi Tambewagh)
- School: Veermata Jijabai Technological Institute
- Country: India
- Primary language: English, Hindi, Marathi
- Typical work hours: 9AM-5PM IST
- Previous GSoC participation: This would be my first time contributing in GSoC

5.1 Project

Project name: librobotcontrol support for newer boards

Chapter 6

Description

Objective: Expand *librobotcontrol* support to BeagleBone AI, BeagleBone AI-64, and BeagleV-Fire SBCs by adding its software support and implementing and validating core hardware functionalities.

Key Components:

- GPIO Control: Enable digital I/O operations for robotics applications.
- PWM Support: Implement motor and servo control via Pulse Width Modulation.
- I2C & SPI Communication: Provide robust support for various sensor and peripheral interfacing.
- UART Support: Ensure seamless serial communication.
- ADC Integration: Read analog values from sensors with high accuracy.
- Testing & Validation: Ensure correctness and efficiency with hardware testing.
- Documentation: Provide clear usage examples and technical documentation.

Understanding librobotcontrol: librobotcontrol is a C library designed for BeagleBone-based robotics applications. It provides low-level access to various hardware interfaces, making it a critical component for robotics development.

Chapter 7

Software

- Programming Language: C
- Compilation: GCC, Make
- Documentation Tools: Sphinx, Doxygen

Chapter 8

Hardware

- BeagleBone AI
- BeagleBone AI-64
- BeagleV-Fire
- BeagleBone Robotics Cape
- USB-to-serial adapters for UART testing

8.1 Timeline

Chapter 9

Timeline summary

Date	Activity
February 27	Connect with possible mentors and request review on first draft
March 9	Complete prerequisites, verify value to community and request review on second draft
March 15	Finalized timeline and request review on final draft
April 7	Submit application
May 8	*Community Bonding Period (May 8 - June 1)*
June 2	*Coding Begins (June 2)*
June 8	*Milestone #1, Motor Control Cleanup for Beaglebone AI & Introductory YouTube Video (June 8)*
June 17	*Milestone #2, Servo, Battery Monitoring, and GPIO Cleanup for BeagleBone AI (June 17)*
June 26	*Milestone #3, I2C and SPI Cleanup for BeagleBone AI (June 26)*
July 4	*Milestone #4, UART and ADC Cleanup for BeagleBone AI (July 4)*
July 9	*Milestone #5, Servo, Battery Monitoring, and GPIO Cleanup for BeagleBone AI-64 (July 9)*
July 14	*Midterm Evaluations (July 14 18:00 UTC - July 18 18:00 UTC)*
July 15	*Milestone #6, I2C and SPI Cleanup for BeagleBone AI-64 (July 15)*
July 28	*Milestone #7, UART and ADC Cleanup for BeagleBone AI-64 (July 28)*
August 7	*Milestone #8, Servo, Battery Monitoring, and GPIO Cleanup for BeagleV-Fire (August 7)*
August 17	*Milestone #9, I2C and SPI Cleanup for BeagleV-Fire (August 17)*
August 25	*Milestone #10, UART and ADC Cleanup for BeagleV-Fire (August 25)*
August 25	Submit Final Code and Documentation

9.1 Timeline detailed

Chapter 10

Community Bonding Period (May 8 - June 1)

- Study librobotcontrol repository.
- Study previous work on extending support of librobotcontrol on BeagleBone AI
- Identify hardware-specific differences across BeagleBone AI, BeagleBone AI-64, and BeagleV-Fire.
- Read BeagleBoard documentation and related specifications.
- Compare GPIO, I2C, SPI, PWM, UART, ADC implementations in the 3 boards.

Chapter 11

Coding Begins (June 2)

Initial implementation of project features begins.

Chapter 12

Milestone #1, Motor Control Cleanup for Beaglebone AI & Introductory YouTube Video (June 8)

1. Code Review:
    - Modularize motor initialization and control logic.
    - Validate .dto file for PWM pin configurations (P8.13, P8.19) which are enhanced high resolution PWM output pins of module 2.
    - Test motor functionality using Robotics Cape hardware.
    - Develop example scripts showcasing motor control features.
2. Introductory YouTube Video:
    - Create an introductory YouTube video for GSoC.
3. Deliverables after attaining this milestone:
    - Cleaned-up motor control code for BBAI.
    - Test scripts showcasing motor control.
    - Introductory YouTube Video.

Chapter 13

Milestone #2, Servo, Battery Monitoring, and GPIO Cleanup for BeagleBone AI (June 17)

1. Servo Control:
    - Optimize PWM signals for servo motors; update .dto configurations for servo pins (P9.14) which are enhanced high resolution PWM output pins of module 3 (3A here).
    - Validate servo motor responsiveness using Robotics Cape hardware.
    - Create example scripts demonstrating servo functionality.
2. Battery Monitoring:
    - Streamline voltage/current sensing logic and validate ADC channels for readings.
    - Test battery monitoring using Robotics Cape hardware.
    - Develop example scripts showcasing battery monitoring.
3. GPIO:
    - Simplify GPIO toggling and initialization; validate .dto files for GPIO pins (LEDs, buttons).
    - Test GPIO functionality using Robotics Cape hardware.
    - Create example scripts showcasing GPIO functionality.
4. Deliverables after attaining this milestone:
    - Cleaned-up servo control, battery monitoring, and GPIO code.
    - Example scripts showcasing these functionalities.

Chapter 14

Milestone #3, I2C and SPI Cleanup for BeagleBone AI (June 26)

1. I2C:
    - Refactor initialization, read/write functions; validate .dto files for I2C pins (P9_19 i.e. i2c4_scl in mode 7, P9_20 i.e. i2c4_sda in mode 7).
    - Test I2C communication with peripherals like IMUs (e.g., MPU-6050).
    - Develop example scripts showcasing I2C functionality.
2. SPI:
    - Optimize SPI data transfer logic; validate .dto configurations for SPI pins (CS, SCLK, MOSI, MISO).
    - Test SPI communication with devices like ADCs or displays.
    - Create example scripts for SPI communication.
3. Deliverables after attaining this milestone:
    - Cleaned-up I2C and SPI code for BBAI.
    - Example scripts demonstrating I2C and SPI functionality.

Chapter 15

Milestone #4, UART and ADC Cleanup for BeagleBone AI (July 4)

1. UART:
    - Streamline UART communication logic and validate .dto files for UART pins (P9.24 i.e. uart10_txd in mode 3, P9.26 i.e. uart10_rxd in mode 3).
    - Test UART communication with peripherals like GPS modules or other serial devices.
    - Create example scripts showcasing UART functionality.
2. ADC:
    - Optimize ADC input reading logic; validate ADC configurations, using pins like P9.39 (ADC Input Pin).
    - Test ADC functionality using Robotics Cape hardware and sensors like potentiometers or light sensors.
    - Develop example scripts showcasing analog input readings.
3. Deliverables after attaining this milestone:
    - Cleaned-up UART and ADC code for BBAI.
    - Example scripts demonstrating UART and ADC functionality.

Chapter 16

Milestone #5, Servo, Battery Monitoring, and GPIO Cleanup for BeagleBone AI-64 (July 9)

1. Servo Control:
    - Optimize PWM signals for servo motors; update .dto configurations for servo pins (P9.14) which are enhanced high resolution PWM output pins of module 2 (2A here).
    - Validate servo motor responsiveness using Robotics Cape hardware.
    - Create example scripts demonstrating servo functionality.
2. Battery Monitoring:
    - Streamline battery monitoring logic for stable readings; validate ADC configurations for voltage/current sensing.
    - Test battery monitoring on BBAI-64 hardware.
3. GPIO:
    - Simplify GPIO initialization and toggling; validate .dto files for GPIO configurations.
    - Test GPIO functionality using Robotics Cape hardware.
    - Create example scripts showcasing GPIO functionality.
4. Deliverables after attaining this milestone:
    - Cleaned-up servo control, battery monitoring, and GPIO code for BBAI-64.
    - Example scripts showcasing these features.

Chapter 17

Milestone #6, I2C and SPI Cleanup for BeagleBone AI-64 (July 15)

1. I2C:
    - Adapt initialization and data handling; validate .dto configurations for I2C pins (P9_19, P9_20).
    - Test I2C communication with peripherals like IMUs.
    - Create example scripts showcasing I2C functionality.
2. SPI:
    - Optimize SPI transfer logic; validate .dto configurations for SPI pins (CS, SCLK, MOSI, MISO).
    - Test SPI communication with devices like ADCs or displays.
    - Develop example scripts showcasing SPI functionality.
3. Deliverables after attaining this milestone:
    - Cleaned-up I2C and SPI code for BBAI-64.
    - Example scripts demonstrating I2C and SPI functionality.

Chapter 18

Midterm Evaluations (July 14 18:00 UTC - July 18 18:00 UTC)

- Submit fully cleaned-up and validated code for all BeagleBone AI and partial BeagleBone AI-64 features.
- Detailed examples and documentation.

Chapter 19

Milestone #7, UART and ADC Cleanup for BeagleBone AI-64 (July 28)

1. UART:
    - Streamline UART communication logic for 64-bit architecture; validate .dto configurations for UART pins.
    - Test UART functionality with Robotics Cape peripherals.
    - Create example scripts demonstrating UART functionality.
2. ADC:
    - Adapt ADC logic for accurate analog input readings; validate ADC configurations.
    - Test ADC functionality using Robotics Cape hardware and sensors.
    - Develop example scripts demonstrating analog input readings.
3. Deliverables:
    - Cleaned-up UART and ADC code for BBAI-64.
    - Example scripts showcasing UART and ADC functionality.

Chapter 20

Milestone #8, Servo, Battery Monitoring, and GPIO Cleanup for BeagleV-Fire (August 7)

1. Servo Control:
    - Optimize PWM signals for servo motors; update .dto configurations.
    - Validate servo motor responsiveness using Robotics Cape hardware.
    - Create example scripts demonstrating servo functionality.
2. Battery Monitoring:
    - Streamline battery monitoring logic for stable readings; validate ADC configurations for voltage/current sensing.
    - Test battery monitoring on BBAI-64 hardware.
3. GPIO:
    - Simplify GPIO initialization and toggling; validate .dto files for GPIO configurations.
    - Test GPIO functionality using Robotics Cape hardware.
    - Create example scripts showcasing GPIO functionality.
4. Deliverables after attaining this milestone:
    - Cleaned-up servo control, battery monitoring, and GPIO code for BBAI-64.
    - Example scripts showcasing these features.

Chapter 21

Milestone #9, I2C and SPI Cleanup for BeagleV-Fire (August 17)

1. I2C:
    - Adapt initialization and data handling logic for BV-Fire; validate .dto configurations for I2C pins.
    - Test I2C communication with peripherals like sensors and IMUs.
    - Create example scripts showcasing I2C functionality on BV-Fire.
2. SPI:
    - Optimize SPI transfer logic for BV-Fire; validate .dto configurations for SPI pins.
    - Test SPI communication with devices like ADCs or displays.
    - Develop example scripts for SPI communication on BV-Fire.
3. Deliverables after attaining this milestone:
    - Cleaned-up I2C and SPI code for BV-Fire.
    - Updated .dto files for I2C and SPI pins.
    - Verified example scripts demonstrating I2C and SPI functionality.

Chapter 22

Milestone #10, UART and ADC Cleanup for BeagleV-Fire (August 25)

1. UART:
    - Refactor UART communication logic for BV-Fire; validate .dto configurations for UART pins.
    - Test UART functionality with devices connected to BV-Fire.
    - Create example scripts demonstrating UART functionality.
2. ADC:
    - Optimize ADC input reading logic for BV-Fire; validate ADC channel configurations.
    - Test ADC functionality with analog sensors.
    - Develop example scripts for ADC functionality.
3. Final Video:
    - Submit final project video.
    - Complete mentor evaluations.
    - Prepare a final project report detailing the implementation, challenges, and results.
    - Submit final work to the GSoC site.
    - Submit all documentation, example scripts, and test results.
    - Ensure the project repository is clean, well-organized, and properly tagged for submission.
    - Record a final YouTube video showcasing the working project, key learnings, and outcomes.
    - Complete mentor evaluations and address any last minute feedback.
4. Deliverables:
    - Cleaned-up UART and ADC code for BV-Fire.
    - Comprehensive examples and final documentation.

22.1 Experience

I have experience in embedded systems and am actively pursuing Linux development, making me well-suited for this project. My work with *librobotcontrol* aligns closely with my interest in low-level hardware interactions, including GPIO, I2C, SPI, PWM, UART, and ADC. I have hands-on experience with OpenCV, TensorFlow/Keras, and real-time processing on embedded devices, demonstrating my ability to develop efficient, well-structured code.

I have interned under Dr. Sushama Wagh, HoD, Electrical Engg. Dept., VJTI in my first year of college. In this internship I learnt the fundamentals of Robotics, Communication Protocols (SPI, I2C, UART), different types of motors (DC, Servo), Microprocessors and Microcontrollers as well as designing and planning behind robotics and corresponding calculations. I have made a PID controlled robot which balances a cylindrical structure at the centre of a beam which is free to rotate about its central transverse axis, as my capstone project for the internship.

I have worked extensively with ESP32 using ESP-IDF. I have created a LFR based Maze Solving Robot based on ESP32, coding it on ESP-IDF platform, as a part of a 4 membered team. I have also taken lectures on topics like 'Operators in C language' and 'Pulse Width Modulation' as part of various workshops organized for our juniors by Society of Robotics and Automation in my college.

I have qualified for the 2nd Stage of E-Yantra Robotics Competition organized by Indian Institute of Technology, Bombay, as a part of a 4 membered team, for the theme 'Warehouse Drone' where I got acquainted with ROS2 and drone technology. The problem statement focused on scanning packages efficiently, navigating through a crowded airspace.

I have completed 2 relevant courses:

1. Linux Device Driver Development
2. Baremetal Assembly and C on RISC-V

I am confident that my structured approach and ability to adapt to new challenges will allow me to successfully complete this project within the given timeline. By breaking the project into well-defined milestones and leveraging my knowledge of Linux systems and embedded programming, I will ensure a smooth implementation, thorough testing, and well-documented deliverables.

My Resume : Resume Link

Chapter 23

Contingency

If I encounter challenges and my mentor isn't available, I will follow a structured approach to troubleshoot the issue:

1. Thorough Debugging:
    - Use tools like *gdb* and *strace* to analyze runtime behavior.
    - Check kernel logs (*dmesg*), I2C/SPI/UART bus activity, and GPIO state.
    - Compare behavior across BeagleBone AI, AI-64, and BeagleV-Fire to isolate board-specific issues.
2. Community Engagement:
    - Refer to BeagleBoard.org Forum and mailing lists.
    - Search and ask relevant questions on Stack Overflow, BeagleBoard GitLab issues, and Linux kernel forums.
    - Consult existing BeagleBoard documentation and other open-source references.
3. Incremental Development & Testing:
    - Break down the issue into smaller testable components.
    - Use simplified test scripts to isolate failing subsystems.
    - Implement alternative approaches based on available documentation.
4. Time Management & Parallel Tasks:
    - If blocked on one aspect, shift focus to another pending task (e.g., documentation or testing).
    - Maintain a detailed log of issues and attempted solutions for future debugging or mentor discussions.

These strategies ensure that I continue progressing even if a mentor is temporarily unavailable.

Chapter 24

Benefit

Successful completion of this project will significantly enhance the *librobotcontrol* library's capabilities for BeagleBone AI, AI-64, and BeagleV-Fire. The impact on the BeagleBoard.org community includes:

1. **Expanded Hardware Support:**
    - Developers will have a fully functional robotics control library across multiple BeagleBoard platforms.
    - Improved support for sensors, actuators, and peripherals will enable more robotics and automation projects.
2. **Better Documentation & Examples:**
    - New users will benefit from clear, well-documented API usage and example programs.
    - Consistent implementation across different boards will reduce confusion and development time.
3. **Performance & Stability Improvements:**
    - Optimized I2C, SPI, UART, and ADC implementations will improve real-time responsiveness.
    - Thorough testing ensures reliability for industrial and academic projects.

By contributing to this project, I will help strengthen the BeagleBoard ecosystem, making it easier for developers, researchers, and hobbyists to build robotics applications with BeagleBone AI-based platforms. .

Chapter 25

Misc

Completed all GSoC prerequisites:

- Created accounts across OpenBeagle, Discord and Beagle Forum.
- PR for cross compilation task : Pull request

Chapter 26

References

- BeagleBone AI Manual
- BeagleBone AI-64 Manual
- BeagleV-Fire Manual
- Robotics Cape Interface Specifications