GSoC Proposal for BeagleBoard.org

BeagleBoard.org
Jul 02, 2025

Table of contents

Chapter 1

Introduction

Greybus is a lightweight, modular protocol initially developed for **Google's Project Ara**, a modular smartphone initiative, to facilitate efficient, plug-and-play communication between hardware components such as sensors, displays, and processors. It abstracts low-level transport layers like **SPI (Serial Peripheral Interface)**, **I2C (Inter-Integrated Circuit)**, **UART (Universal Asynchronous Receiver-Transmitter)**, and **GPIO (General Purpose Input/Output)**, offering a unified interface that simplifies device interaction while maintaining low latency and minimal resource overhead. Greybus operates over a variety of physical transports, including high-speed Unipro and wireless protocols, making it highly adaptable.

BeagleConnect, an **open-source wireless solution** from BeagleBoard.org, leverages Greybus over **IEEE 802.15.4 (6LoWPAN)** to enable seamless integration of sensors and actuators in IoT applications. This approach reduces software complexity by abstracting hardware-specific details, enhances interoperability across diverse devices, and optimizes power efficiency—critical for battery-powered embedded systems.

Currently, the Greybus module for Zephyr, a scalable real-time operating system (RTOS) designed for resource-constrained devices, exists as an **out-of-tree module**. This status complicates maintenance, hinders community testing, and limits contributions due to its separation from Zephyr's mainline repository. This project aims to **upstream the Greybus module into Zephyr's official codebase**, ensuring it becomes a fully supported, in-tree component. This will improve maintainability, enable broader community engagement, and seamlessly integrate Greybus within Zephyr's ecosystem.

By achieving this, the project will provide **native Greybus support** for platforms like BeagleConnect, eliminating reliance on external patches or custom kernel forks. This will streamline development workflows, enhance long-term support, and empower developers to utilize Greybus in Zephyr-based IoT and embedded projects with greater ease and reliability.

1.1 Summary links

- **Contributor:** Sahil Jaiswal
- **Mentors:** Ayush Singh, Jason Kridner
- **Code:** TBD
- **Documentation:** TBD
- **GSoC:** TBD

1.2 Status

This project is currently just a proposal.

1.3 Proposal

- Registered on relevant platforms:
  - **OpenBeagle:** Sahil Jaiswal
  - **Discord:** Sahil Jaiswal
  - **Beagle Forum:** Sahil Jaiswal
- Submitted a Pull Request for Cross Compilation: #202

1.4 About

- **Forum:** Sahil7741 on Forum
- **OpenBeagle:** Sahil7741 on OpenBeagle
- **GitHub:** Sahil7741 on GitHub
- **School:** Indian Institute of Information Technology Gwalior (IIIT Gwalior)
- **Country:** India
- **Primary languages:** English, Hindi
- **Typical work hours:** 9 AM - 5 PM Indian Standard Time (IST)
- **Previous GSoC participation:** First-time applicant

Chapter 2

Project

Project name: Upstream Greybus Module for Zephyr

2.1 Description

The Greybus module's current out-of-tree status in Zephyr stems from its origins as a standalone implementation, requiring developers to manually integrate and maintain it outside Zephyr's official repository. This isolation results in version mismatches, limited visibility to Zephyr's automated testing infrastructure, and a steep barrier to community contributions. This project seeks to **upstream the Greybus module as an official Zephyr component**, embedding it within the RTOS's core framework to enhance maintainability, accessibility, and integration.

Upstreaming will enable **native support for BeagleConnect and other Greybus-enabled hardware platforms** within Zephyr, removing dependencies on external patches or modified kernels. This will allow developers to leverage Greybus's modular communication capabilities directly in Zephyr applications, such as IoT sensor networks or industrial control systems, without additional setup overhead.

Greybus itself is a **lightweight, high-speed protocol** that abstracts hardware complexity, supporting operations like device enumeration, firmware updates, and data streaming over interfaces like SPI, I2C, UART, GPIO, and 6LoWPAN. BeagleConnect utilizes Greybus to deliver **wireless sensor connectivity**, enabling low-power, real-time interaction between Zephyr-based nodes and Linux hosts. By integrating Greybus into Zephyr, this project will bridge the gap between lightweight RTOS environments and robust host systems, fostering a cohesive ecosystem for embedded development.

2.2 Goals and Objectives

The **primary goal** is to transform the Greybus module into an official Zephyr component, ensuring seamless integration, community-driven maintenance, and robust testing support. The key objectives are:

- **Cleaning up and reviving the testing infrastructure**: Refactor the module's existing test suite to align with Zephyr's standards, incorporating automated CI testing for reliability.
- **Making MikroBUS manifest optional**: Modify the module to operate independently of MikroBUS manifests, enabling Greybus functionality without requiring a patched kernel on the host side.
- **Upstreaming the module into Zephyr**: Submit the refactored module to Zephyr's mainline repository, adhering to contribution guidelines and passing all quality checks.
- **Using Greybus abstractions in host firmware**: Integrate Greybus's modular APIs into Zephyr's host firmware layer, enabling efficient communication with peripherals and hosts.
- **Ensuring compatibility with BeagleConnect devices**: Validate the module's functionality with BeagleConnect Freedom and similar platforms, supporting wireless 6LoWPAN-based IoT use cases.

Implementation Plan

1. **Cleanup and Refactor Greybus Module**

- Analyze the existing out-of-tree Greybus codebase to identify deprecated functions, unused variables, or non-compliant code segments.
- Refactor the module to meet Zephyr's coding standards (e.g., K&R style, consistent naming conventions), improving readability and maintainability.
- If needed refactor Greybus APIs (e.g., control, data, and operation handlers) to be modular and reusable, ensuring they integrate smoothly into Zephyr's device driver model and host firmware.

2. **Revive and Enhance Testing Infrastructure**

- Refactor the existing test framework to support Zephyr's Twister tool, ensuring comprehensive coverage of Greybus functionality.
- Integrate tests into Zephyr's CI/CD pipeline via OpenBeagle, automating validation for every commit.
- Validate Greybus operations across SPI, I2C, UART, and GPIO using QEMU emulation and BeagleConnect Freedom hardware, confirming compatibility and performance.
- **CI Testing**:

  To ensure the Greybus module's reliability, Integrate it with Zephyr's **Twister** testing framework for continuous integration (CI). Twister will automate unit, integration, and stress tests across multiple targets:

  – **Targets**:

    ∗ **QEMU**: For emulating SPI, I2C, UART, and GPIO protocols, ensuring broad compatibility across simulated platforms.

    ∗ **BeagleConnect Freedom**: For real hardware validation of wireless 6LoWPAN communication and peripheral interactions (e.g., sensors, actuators).

  – **Testing Infrastructure**:

    ∗ **Unit Tests**: Validate individual protocol implementations, such as SPI data transfer accuracy or I2C device addressing correctness, in isolation.

    ∗ **Integration Tests**: Confirm Greybus communication between Zephyr firmware and host over TCP and 6LoWPAN, ensuring end-to-end functionality.

    ∗ **Stress Tests**: Simulate high-load conditions, like rapid GPIO state changes or continuous UART data streams, to verify stability under pressure.

  – **Implementation**: Develop Twister test scripts, using Zephyr's testcase.yaml format, and integrate them into the OpenBeagle CI pipeline. Tests will execute on each commit to detect regressions early.

3. **Decouple MikroBUS Manifest Support**

- Modify the Greybus stack to function independently of MikroBUS manifests.
- Ensure that standard Greybus interfaces work seamlessly without requiring kernel modifications or custom patches.

4. **Upstream the Greybus Module into Zephyr**

- Prepare patches following Zephyr's contribution guidelines (e.g., detailed commit messages, signed-off-by tags).
- Submit pull requests to Zephyr's GitHub repository, collaborating with maintainers to address reviews, resolve merge conflicts, and meet compliance standards (e.g., documentation requirements).
- Verify successful integration by building and testing the module within Zephyr's mainline branch.

5. **Documentation and Community Engagement**

- Create comprehensive documentation, including API references, integration guides for BeagleConnect, and troubleshooting steps for common issues (e.g., protocol mismatches).

- Develop sample applications (e.g., a 6LoWPAN sensor node) to demonstrate Greybus usage in Zephyr.
- Engage with Zephyr and BeagleBoard communities via forums and mailing lists, soliciting feedback and ensuring the module meets real-world needs.

2.3 Software

- **Zephyr RTOS**
- **C programming**
- **Device Tree**
- **Linux kernel & drivers**
- **QEMU for testing**
- **TCP/IP networking stack**
- **OpenBeagle CI**

2.4 Hardware

- **BeaglePlay or BeagleConnect Freedom** (for hardware testing)
- **Basic wiring for debugging**
- **Serial console and JTAG (for low-level debugging)**
- **Host system with Linux for development**

Chapter 3

Timeline

3.1 Timeline summary

Date	Activity
February 27 - March 24	Connect with possible mentors and request review on the first draft
March 24 - April 8	Complete prerequisites, verify value to the community, and submit the application
April 9 - May 7	Deep dive into **Greybus architecture**, Zephyr subsystems, and device driver model
May 8 - June 1	*Community Bonding Period (May 8 - June 1)*
June 2 - June 8	Start coding and create an introductory video
June 9 - June 15	*Milestone #1, Introductory video and Greybus setup (June 9)*
June 16 - June 22	*Milestone #2, Cleanup and revive testing infrastructure (June 16)*
June 23 - June 29	*Milestone #3, Decouple MikroBUS manifest (June 23)*
June 30 - July 6	*Milestone #4, Prepare for upstreaming (June 30)*
July 7 - July 13	*Milestone #5, Documentation and final testing (July 7)*
July 14 - July 18	*Submit midterm evaluations (July 14 - July 18)*
July 19 - July 25	*Milestone #6, Greybus peripheral integration testing (July 25)*
July 26 - August 1	*Milestone #7, Finalize upstreaming process (August 1)*
August 2 - August 8	*Milestone #8, Final documentation and cleanup (August 8)*
August 9 - August 15	*Milestone #9, Testing and bug fixes (August 15)*
August 16 - August 22	*Milestone #10, Final submission (August 22)*
August 25	*Submit final project video, submit final work to GSoC site and complete final mentor evaluation (August 25)*

3.2 Timeline detailed

3.2.1 Community Bonding Period (May 8 - June 1)

- Engage with Zephyr and BeagleBoard communities via Discord and forums to discuss Greybus integration challenges.
- Study Greybus protocol specifications (e.g., control protocol, operation types) and Zephyr's networking and driver subsystems.
- Set up Greybus module locally, configure BeagleConnect Freedom, and document initial findings on the BeagleBoard forum.
- Finalize detailed implementation roadmap with mentors, including test case priorities and upstreaming milestones.

3.2.2 Milestone #1, Introductory video and Greybus setup (June 9)

- Create an **introductory video** outlining project goals, Greybus benefits, and planned BeagleConnect use cases.

- Set up Greybus module within a local Zephyr build environment, configuring device tree bindings for SPI and I2C.
- Perform basic SPI and I2C communication tests using QEMU, verifying data transfer between emulated Zephyr nodes and a simulated host.

3.2.3 Milestone #2, Cleanup and revive testing infrastructure (June 16)

- Refactor the existing Greybus test framework to use Twister, converting legacy tests into testcase.yaml format.
- Develop initial Twister unit tests for UART (e.g., baud rate consistency, data integrity) and integrate into OpenBeagle CI pipeline.
- Validate Greybus SPI operations on BeagleConnect Freedom (or QEMU if hardware delayed), ensuring accurate data transfer to a connected sensor.

3.2.4 Milestone #3, Decouple MikroBUS manifest (June 23)

- Modify Greybus stack to remove MikroBUS manifest dependencies, replacing them with generic device tree parsing.
- Test GPIO functionality (e.g., pin toggling) without kernel modifications using QEMU, confirming independence from MikroBUS.
- Refactor Greybus APIs to support modular host firmware integration, documenting changes for upstream review.

3.2.5 Milestone #4, Prepare for upstreaming (June 30)

- Prepare initial patches for SPI and I2C support, including updated device tree bindings and driver code.
- Submit draft pull requests to Zephyr's repository, engaging maintainers for early feedback on code structure and compliance.

3.2.6 Milestone #5, Documentation and final testing (July 7)

- Write detailed documentation for Greybus module setup, including SPI/I2C configuration examples and API usage.
- Add Twister integration tests for TCP communication, validating Zephyr-to-host data exchange on QEMU and BeagleConnect Freedom.
- Resolve UART test failures (e.g., packet loss), ensuring stability across emulated and real hardware targets.

3.2.7 Submit midterm evaluations (July 14 - July 18)

- Document progress (e.g., refactored code, initial CI tests, draft patches) in a midterm report.
- Submit **midterm evaluation** to GSoC, incorporating mentor feedback.

Important: July 18 - 18:00 UTC: Midterm evaluation deadline.

3.2.8 Milestone #6, Greybus peripheral integration testing (July 25)

- Integrate GPIO-based peripherals (e.g., LEDs, buttons) over Greybus on BeagleConnect Freedom, verifying real-time control.
- Develop Twister stress tests for SPI (e.g., 1000 rapid data transfers), running on QEMU to confirm robustness under load.

3.2.9 Milestone #7, Finalize upstreaming process (August 1)

- Address Zephyr maintainers' feedback on SPI, I2C, and UART patches, revising code and test cases as needed.
- Ensure all protocol patches (SPI, I2C, UART, GPIO) are merged into Zephyr's mainline, passing CI and compliance checks.

3.2.10 Milestone #8, Final documentation and cleanup (August 8)

- Finalize documentation with 6LoWPAN examples (e.g., wireless sensor data streaming) for BeagleConnect integration.
- Perform final code cleanup (e.g., remove debug logs, optimize memory usage) and validate I2C stability with Twister on QEMU.

3.2.11 Milestone #9, Testing and bug fixes (August 15)

- Conduct real-world testing of Greybus UART and TCP on BeagleConnect Freedom, streaming data to a Linux host over 6LoWPAN.
- Debug and fix Twister stress test issues (e.g., GPIO toggle latency), ensuring reliability across all protocols.

3.2.12 Milestone #10, Final submission (August 22)

- Prepare a final project video demonstrating SPI, I2C, UART, GPIO, and 6LoWPAN functionality with BeagleConnect Freedom.
- Submit completed work (code, documentation, test results) to the GSoC site.

3.2.13 Submit final project video, submit final work to GSoC site and complete final mentor evaluation (August 25)

- Create a final **video demo** showcasing Greybus in action (e.g., sensor data over 6LoWPAN).
- Submit **final work** to the GSoC site, including links to merged Zephyr patches.

3.2.14 Final Submission (August 25 - September 1)

Important: **August 25 - September 1 - 18:00 UTC:** Submit final work and mentor evaluation.

September 1 - 8 - 18:00 UTC: Mentors submit final evaluations.

3.2.15 Initial Results (September 1)

Important: **September 1 - November 9:** Contributors with extended timelines continue coding.

November 10 - 18:00 UTC: Final submission for extended timeline contributors.

November 17 - 18:00 UTC: Final mentor evaluations for extended timeline projects

Chapter 4

Experience and Approach

I am confident in my ability to complete this project within the proposed timeline due to my **strong background in embedded systems, real-time operating systems (RTOS), and Linux kernel development**. I have successfully set up and run **Zephyr on QEMU**, explored **device driver development**, and worked on **low-level firmware development** for microcontrollers like the **STM32**. Additionally, I have hands-on experience with **communication protocols**, including **I2C, SPI, UART, and USB**, which are crucial for integrating Greybus support in Zephyr.

I have already set up the **Zephyr repository**, interacted with the community, and successfully tested **QEMU emulation**. My experience in **debugging hardware-software interactions**, analyzing **protocol-level communication**, and working with **device tree bindings** makes me well-equipped to implement and optimize Greybus modules in Zephyr. Furthermore, I have experience in **porting software to constrained hardware environments**, ensuring efficient resource utilization, which aligns well with the goals of this project.

To ensure steady progress, I will follow a structured approach, breaking the project into **incremental milestones** with well-defined objectives. I will work closely with my **mentor and the community**, actively engage in discussions, and document my findings in Zephyr's **forum and mailing lists**. By maintaining a **continuous integration workflow**, submitting **incremental merge requests**, and conducting **rigorous testing**, I will ensure that my contributions are aligned with Zephyr's upstream development.

Additionally, I am highly adaptable and proactive in troubleshooting, allowing me to quickly identify and resolve issues that may arise during development. My ability to **read and understand existing codebases, debug system-level interactions, and optimize performance** will be instrumental in successfully completing this project.

4.1 Contingency

If I get stuck and my mentor isn't available, I will:

- **Consult Zephyr and BeagleBoard community resources**, including the **Discord, mailing lists, and forums**.
- **Refer to the official Greybus and Zephyr documentation**, ensuring I have thoroughly explored all possible solutions.
- **Engage with other open-source contributors and developers** who have worked on similar projects.
- **Break down the problem into smaller, testable components**, debug using **QEMU, logs, and tracing tools**, and isolate issues systematically.
- **Work on parallel tasks**, ensuring overall progress isn't halted while waiting for mentor feedback.

My ability to **work independently**, coupled with my strong debugging skills and structured problem-solving approach, ensures that I will overcome obstacles effectively.

4.2 Benefit

This project will have a **significant impact** on the BeagleBoard.org community by:

- Enabling **Greybus support in Zephyr**, making it easier to integrate **modular peripheral communication** in **BeagleBoard-based systems**.
- Expanding the usability of **Greybus beyond Linux**, allowing **lightweight RTOS platforms** like Zephyr to **efficiently communicate with host devices**.
- Helping **developers and embedded system engineers** prototype **sensor-based and IoT applications** using **BeagleBone boards** and **other embedded hardware**.

Community members have highlighted the importance of integrating Greybus support into Zephyr:

> *"Currently, Greybus module for Zephyr is an out-of-tree module, making it hard to maintain and limiting community testing and contributions. Upstreaming this module would enhance the BeagleConnect Technology ecosystem, making it easier to test and maintain."* — BeagleBoard.org Community Discussion
>
> *"Greybus is an application layer protocol developed originally for Google's modular smartphone project Ara and is a part of the Linux kernel. Integrating Greybus with Zephyr RTOS can be thought of as a Remote Procedure Call (RPC) framework, offering tight integrations within Linux kernel subsystems."* — Zephyr RTOS Emerging Technologies

By successfully completing this project, I will contribute to making **BeagleBoard hardware more flexible and interoperable** in **real-time and low-power applications**.

4.3 Misc

I completed the general requirements listed in the General Requirements.

Here are some contributions I made to BeagleBoard, along with their merged request links:

- **Automated PDF Generation for Subdirectory Proposals** PR #65 - Ensured that *make latexpdf* generates PDFs for all proposal *.rst* files, including those inside subdirectories, eliminating the need for manual builds.
- **Fixed PDF Download Button & Standardized PDF Deployment** PR #69 - Fixed broken PDF download links by restructuring output paths and aligning local builds with GitLab Pages deployment, using *_static/* as the standard location.