Provide SPARC greth Network Drivers for lwip

Google Summer of Code Program 2025 Project Proposal

Prithvi Tambewagh
Discord : prithvi_t
prithvitambewagh789@gmail.com
Veermata Jijabai Technological Institute
Mumbai, India
Contact No.
Emergency Contact No.
RTEMS GitLab
GitHub
LinkedIn

Project Abstract:
This project aims to improvise and modernize the existing GRETH network driver in the Legacy Network Stack to lwIP (Lightweight IP), one of the latest networking stacks, improving performance and enabling long-term maintainability as the current implementation of this driver relies on Legacy Network APIs, while RTEMS is shifting to newer networking stacks like lwIP, it is difficult to use this driver in recent RTEMS applications. GRETH Driver is a driver of significant importance since it is made for LEON processors which are widely used in various space applications.

Project Scope: This is a Large Scope Project (350 hours)

Project Description:
The GRETH Ethernet Driver in RTEMS is based on the Legacy Network Stack, which is quite outdated. Nowadays, several RTEMS Applications work on newer networking stacks like lwIP. However, GRETH is, right now, incompatible with the lwIP networking stack and this project aims to solve this exact issue by porting the available GRETH Driver to lwIP. The porting involves restructuring the GRETH driver to be integrated with lwIP. This includes implementing essential components such as packet handling, interrupt mechanisms, and setup procedures for network initialization. The updated GRETH driver will lay the foundation for future enhancements like multicast support, checksum offloading, and advanced performance optimizations.

Benefits of porting GRETH Ethernet Driver to lwIP:
Having GRETH Driver ported to lwIP brings with it several benefits which include :

- ➢ Compatibility with more recent RTEMS applications

- ➢ Performance optimization
- ➢ Ability to run on memory-constrained devices (a notable advantage of using lwIP)
- ➢ Ease of future maintenance

Why I chose this project:
I have been working with embedded systems and I have recently been learning in depth about device driver development. This project seems to perfectly suit my interests and capabilities. I look forward to learning the intricacies of device drivers and networking in depth through this project while contributing to an RTOS like RTEMS!

Plan of Maintenance of GRETH lwIP Driver: This would involve :
- ➢ Routine checks for checking its compatibility with RTEMS and lwIP versions
- ➢ Debugging based on community feedback
- ➢ Ensuring documentation is up-to-date
- ➢ Periodic testing and code reviews

Project Deliverables
- ➔ GSoC Timeline 2025: *GSoC Timeline*.
- ➔ To ensure that coding begins by June 2nd, I have the following plan of action :
  - ❖ Set up RTEMS Development Environment.
  - ❖ Modify $HOME/quick-start/src/rtems/testsuites/samples/hello/init.c file and run this application on SPARC ERC32 BSP, hence completing the preliminary task for GSoC 2025.

```
*** BEGIN OF TEST HELLO WORLD ***
*** TEST VERSION: 7.0.0.9e7371709d6e7f5a6fc0a8c05cc7da30486b1af4-modified
*** TEST STATE: EXPECTED_PASS
*** TEST BUILD:
*** TEST TOOLS: 13.3.0 20240521 (RTEMS 7, RSB 04a84f76056858cd04e16c49cb844b7fe44b38df, Newlib 1b3dcfd)
Hello RTEMS from Prithvi!

*** END OF TEST HELLO WORLD ***
```

  - ❖ Since TSIM3 Evaluation version software was available to download freely for LEON3, I decided to test the GRETH Legacy Network Driver on it. However, I later learnt that it doesn't support GRETH. Started with simulation of GRETH Driver of Legacy Networking Stack on TSIM3 and as a first step simulated 'ticker' sample application on LEON3 Processor using TSIM3 simulator.

```
tsim> load /home/prithvi/quick-start/src/rtems/build/sparc/leon3/testsuites/samples/ticker.exe
  section: .text, addr: 0x40000000, size: 70816 bytes
  section: .rtemsroset, addr: 0x400114a0, size: 128 bytes
  section: .data, addr: 0x40013100, size: 1216 bytes
  read 1276 symbols
```

```
tsim> run
  Initializing and starting from 0x40000000


*** BEGIN OF TEST CLOCK TICK ***
*** TEST VERSION: 7.0.0.06e93a9e3b5ef37ac432aeaca1a46281a65f93d4
*** TEST STATE: EXPECTED_PASS
*** TEST BUILD:
*** TEST TOOLS: 13.3.0 20240521 (RTEMS 7, RSB 04a84f76056858cd04e16c49cb844b7fe44b38df, Newlib 1b3dcfd)
TA1  - rtems_clock_get_tod - 09:00:00   12/31/1988
TA2  - rtems_clock_get_tod - 09:00:00   12/31/1988
TA3  - rtems_clock_get_tod - 09:00:00   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:04   12/31/1988
TA2  - rtems_clock_get_tod - 09:00:09   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:09   12/31/1988
TA3  - rtems_clock_get_tod - 09:00:14   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:14   12/31/1988
TA2  - rtems_clock_get_tod - 09:00:19   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:19   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:24   12/31/1988
TA3  - rtems_clock_get_tod - 09:00:29   12/31/1988
TA2  - rtems_clock_get_tod - 09:00:29   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:29   12/31/1988
TA1  - rtems_clock_get_tod - 09:00:34   12/31/1988

*** END OF TEST CLOCK TICK ***
```

- Working actively on simulating GRETH Driver on a relevant Simulator, presently exploring [RTEMS-SIS (Simple Instruction Simulator)]

➔ June 2 onwards (coding begins) an overview of the whole GSoC Period -
- Create a fork of RTEMS lwIP support GitLab repository: [My fork].
- Give access to the fork to my mentors, RTEMS organization administrator, and other relevant community members.
- Create GRETH lwIP Driver capable of being initialized.
- Create GRETH lwIP Driver capable for packet transmission.
- Make GRETH lwIP Driver capable of packet reception.
- Implement interrupt handling in GRETH lwIP driver.

➔ July 14-18 (Midterm Evaluation) -
- Create a functional GRETH lwIP Driver which is capable of:
  - Driver Initialization.
  - Packet Transmission and Interrupt Handling related to it.
- Create tests for implemented features.

➔ July 18 - August 24 -
- Packet Reception and Interrupt Handling related to it.
- Complete remaining interrupt handling support.
- Testing of GRETH lwIP.
- Create tests for implemented features.
- Clean Up Code.
- Comprehensive Documentation creation.

➔ August 25 - September 1 (Final Evaluation) -
- Final Integration of GRETH lwIP into RTEMS Codebase.
- Documentation created for GRETH lwIP.

➔ September (Final Results Announced) -
- Continue with code clean up ensuring it is completed, if not previously.
- Continue with documentation completion, if not completed already.

- ➔ Post GSoC -
  - ❖ Contribute actively to RTEMS.
  - ❖ Seek technical advice from other RTEMS contributors and members.

Proposed Schedule

This project can be considered of the type where code can be ready for submission in increments at various stages, as elaborated below. Particularly, I aim to complete this project in 5 Phases, out of which

March 24 - April 8 18:00 UTC (Application Period)

- ➢ Interact with Project mentors as well as the RTEMS community to gain deeper insights into the project.
- ➢ Understand the expectations of the mentors from this project and finalize project goals.
- ➢ Aim to simulate existing GRETH Driver in Legacy Networking Stack.

April 8 18:00 UTC - May 8 18:00 UTC (Acceptance Waiting Period)

- ➢ Explore existing implementation of GRETH Driver in Legacy networking Stack.
- ➢ Complete simulation of existing GRETH Driver in Legacy Networking Stack, if not completed.
- ➢ Study RTEMS Networking Documentation related to lwIP.
- ➢ Update mentors with progress about my study for the project and seek feedback.

May 8 18:00 UTC - June 1 (Community Bonding Period)

- ➢ Set up RTEMS Development Environment.
- ➢ Understand existing lwIP implementation in RTEMS in detail.
- ➢ Update mentors with progress about my study for the project and seek feedback.

June 2 - July 13 (First Half)

1. **Phase 1 - Core setup and initial implementation**
   - ➔ Create a fork of RTEMS lwIP support GitLab repository: My fork.
   - ➔ Give access to the fork to my mentors, RTEMS organization administrator, and other relevant community members.
   - ➔ Driver Design and Preparation:
     - ❖ Analyze the existing GRETH driver to identify essential functionalities.
     - ❖ Determine the architecture and interfaces for the new GRETH-lwIP driver.
     - ❖ Create directory structure under rtemslwip/ directory in the fork :

```
rtemslwip/
├── greth/
│   ├── greth.h (Header file for GRETH driver)
│   ├── greth.c (Initialization, Data & Interrupt Handling)
```

```
    │ └── netstart.c (Set up networking)
```

- ➔ Initialization Module
  - ❖ Implement driver initialization functions:
    - `greth_init()`: Initialize GRETH Ethernet Hardware and make device ready for communication.
      - ★ Map Hardware Registers for GRETH Ethernet
      - ★ Memory allocation for driver state
      - ★ Allocate and initialize memory buffers for transmitting and receiving data.
      - ★ Setting configuration of GRETH Device
  - ❖ `greth_configure()`: Configure GRETH settings e.g. duplex mode, link speed, etc.
    - ★ Update registers to set specific configurations.
  - ❖ Initialize GRETH driver in lwIP.
- ➔ Since the packet transmission mechanism is yet to be implemented, the code until this phase **will not** be ready to be submitted for inclusion yet.
- ➔ By the end of Phase 1, the following outcomes are aimed to be achieved :
  - ❖ GRETH Driver capable of being initialized.

2. **Phase 2 - Packet Transmission**
   - ➔ Packet Transmission Implementation
     - ❖ Create functions for packet transmission:
       - `greth_send()`:
         - ★ Transfer packet data from lwIP 'pbuf' to GRETH transmission buffer.
         - ★ Notify GRETH hardware to begin transmitting packets.
         - ★ Handle interrupts and confirm if a packet is sent.
     - ❖ Integrate transmission logic with lwIP.
     - ❖ Validate transmission logic through basic test cases.
     - ❖ Implement Interrupt handling related to packet transmission.
   - ➔ Since the packet reception mechanism is yet to be implemented, the code until this phase **will not** be ready to be submitted for inclusion yet.
   - ➔ By the end of Phase 2, the following outcomes are aimed to be achieved :
     - ❖ Basic Packet transmission.
     - ❖ Tests for packet transmission functionality.
     - ❖ Interrupt handling related to packet transmission implemented.

July 14 - July 18 (Midterm Deliverable)

- ➔ Functional GRETH lwIP driver with:
  - ❖ Driver Initialization
  - ❖ Packet Transmission
  - ❖ Interrupt handling related to packet transmission
- ➔ Tests for implemented features.

July 18 - August 25 (Second Half)

1. **Phase 3 : Packet Reception**

- ➔ Packet Reception Logic:
  - ❖ Implement packet reception logic using the function :
    - `greth_receive()` :
      - ★ Check reception buffer for incoming packets
      - ★ Transfer incoming data to new pbuf structure for further processing by lwIP
  - ❖ Manage received packets
  - ❖ Write tests for packet reception
  - ❖ Implement Interrupt handling related to packet reception
- ➔ Since basic functionalities like packet transmission and reception are completed till this phase, code created until this phase **will be ready** to be submitted for inclusion.
- ➔ By the end of Phase 3, the following outcomes are aimed to be achieved:
  - ❖ GRETH Driver capable of receiving packets
  - ❖ Interrupt handling related to packet reception implemented

2. **Phase 4 : Complete Interrupt Handling Mechanism**
   - ➔ Interrupt Handling Implementation:
     - ❖ Incorporate interrupt handling logic in the driver
     - ❖ Test interrupt-driven packet communication.
     - ❖ Hence create functions like:
       - `greth_irq_handler()` : Respond to GRETH hardware interrupts
         - ★ Identify interrupt type (Tx, Rx, or error)
         - ★ Process incoming packets or handle transmission completions
   - ➔ Since the packet transmission and reception as well as interrupt handling mechanism is implemented, the code until this phase **will be ready** to be submitted for inclusion.
   - ➔ By the end of Phase 4, the following outcomes are aimed to be achieved:
     - ❖ GRETH lwIP interrupt handling is functional

3. **Phase 5 : Final Steps : Code Clean-Up, GRETH lwIP testing and Documentation**
   - ➔ Code Clean-Up
     - ❖ Clean Codebase
     - ❖ Ensure the code has proper formatting and well-commented
   - ➔ Test GRETH lwIP using methods like:
     - ❖ Virtual TAP interface
     - ❖ QEMU (GRETH Environment), etc.
   - ➔ Documentation
     - ❖ Finalize documentation covering:
       - Design
       - Implementation
       - Testing procedures
   - ➔ Integrate GRETH Driver with RTEMS lwIP
   - ➔ Since the GRETH lwIP driver is implemented, the code until this phase **will be ready** to be submitted for inclusion.
   - ➔ By the end of Phase 5 the following outcomes are aimed to be achieved :

- ❖ Cleaned up code
- ❖ Comprehensive documentation
- ❖ Final Integration of Code in RTEMS Codebase

Future Improvements

1. **VLAN Tagging :** This is a method of adding metadata (a VLAN ID) to network packets. This tag identifies which VLAN the packet belongs to, allowing devices to communicate within specific, isolated virtual networks. It has several benefits, like enhanced security, improved network performance, simplified network management, etc.

2. **Real-Time Network Monitoring:** It involves using a tool(s) for monitoring Network Statistics in Real Time which is useful as it helps in proactive issue detection, improved network performance, etc.

Continued Involvement

I aim to maintain an active role in RTEMS as an open-source contributor. I look forward to contributing to the development of robust, scalable features while enriching my technical knowledge. I would like to contribute to RTEMS actively even after completing GSoC.

Other Commitments

I have no other commitments during the GSoC period.

Eligibility

I confirm my eligibility for GSoC 2025 according to the rules: https://summerofcode.withgoogle.com/rules/

Major Challenges foreseen

- ➢ Since lwIP and Legacy Networking Stacks are significantly different, adapting existing GRETH Drivers to lwIP may require significant refactoring.
- ➢ Debugging at various stages of this project may be time-consuming due to the complexity of the porting involved.

References

- ➢ RTEMS GRETH Driver (Legacy Networking Stack) - GitLab
- ➢ RTEMS lwIP Networking Stack - GitLab
- ➢ RTEMS Legacy Networking Documentation
- ➢ GRLIB IP Core User Manual
- ➢ GRLIB IP Library User Manual

Relevant Background Experience

- ➢ I am proficient in C and C++. I have worked extensively with FreeRTOS and ESP32.
- ➢ I have been working with FreeRTOS and Embedded Systems for around 2 years now.
- ➢ I am fluent with version control systems like Git, GitLab as well as GitHub.

Personal

- I am Prithvi Tambewagh, Second Year Student pursuing B.Tech. Electronics & Telecommunications Engg. at Veermata Jijabai Technological Institute, Mumbai, India.
- I have a deep interest in communication and embedded systems.
- I have been working with ESP-IDF for more than a year now.
- I am part of the Society of Robotics and Automation at VJTI, where I have got the opportunity to take lectures of my juniors on various topics, including Pulse Width Modulation, Operators in C language, etc., and teach to an audience of 150+ students for each topic.
- I learned about RTEMS when studying Real Time Operating Systems. It caught my attention due to the fact that it is Open Source and its robust nature owing to which it is used on Space-based Embedded Systems!
- Resume: Link

Experience

- **Free Software Experience/Contributions (optional):**
  - As an introduction to Open Source Contribution, I participated in Hacktoberfest 2024 and made 4 PRs to 'Hacktoberfest' tagged Issues on GitHub related to Python, all of which are merged and hence received Level 4 badge.

- **Language Skill Set**
  - C
  - C++
  - Python

- **Related Research and Work Experience (if any)**:

  - **ESP32 Based Maze Solving Robot**: This robot solves a maze having white lines on a black background. I have used FreeRTOS for operating this robot. FreeRTOS is utilized in this project to create a task for maze solving, for real-time task scheduling, for managing timing, and for introducing appropriate delays. The lines are detected using an array of 5, TCRT1000 sensors. It uses Left Follow Rule (LFR) to navigate through the maze.