Advanced Embedded Software Development

ECEN5013-002 Spring 2019

Multi-controller System - Project 2 (240 Pts)

Due Dates:

- **Requirements Definition, Architecture, and Project Plan Document - No Later Than Wednesday, April 10th (Midnight) – earlier is encouraged**
- **Code Submission and Final Report - Sunday, April 28th (Midnight)**
- **Demo – To be scheduled by Doodle, but completed on or before Thursday, May 2nd.**

Revision 2019.04.03

1 Overview

In this project your team will apply many of the topics discussed in class, including concurrent and fault tolerance software concepts, to create a multi-controller "product" of your own choice using the BeagleBone Green, Tiva C-Series LaunchPad development board, sensors and output devices. The overall objective is to design and implement a monitoring/control system comprised of a controller/server node (Control Node) remotely connected to and controlling a remote sensor/output node (Remote Node). The actual top-level product application is your choice, but it will need to provide the capabilities and follow the guidelines and requirements detailed in the rest of this document. Operationally, the Remote Node provides sensing capabilities whose data is reported to the Control Node, which will make real-time decisions based on data and events, then provide control and feedback to the Remote Node. The Remote Node will drive output(s) for at least 2 devices. Notionally, the configuration is shown in Figure 1.

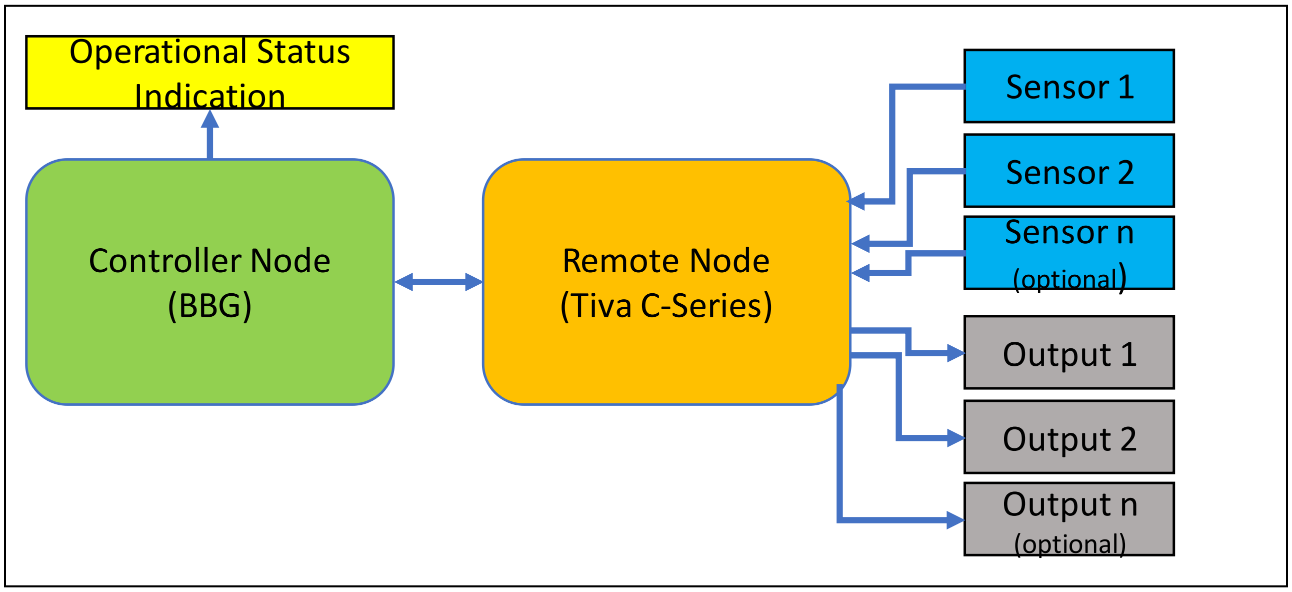

Figure 1 - Project 2 System Configuration

The Remote Node is comprised of the TI Tiva-C Series LaunchPad Dev Kit development board running your customized FreeRTOS image with multiple offboard sensors and output devices. It is connected to the Control Node, a Linux-based BeagleBone Green, via a communication link.

An example of the high-level product application might be gesture-based building entry system. The Remote Node sensors would detect the presence of the humans with gesture and temperature sensors, then relay the data or events to a Control Node that would chose to admit or deny access to a building based on some criteria, then appropriately command the Remote Node to a sequence the of inner/outer door locks. An example of incorporating system fault tolerance and/or fail-safe concepts in this system, if the connection is lost to the server (e.g. communication or power loss from a fire), the Remote Node unlocks the outer building door to allow safe exit by the entry occupants.

Another example would be a closed-loop motor controller system whose Remote Node uses the two sensors to measure the desired speed and actual motor speed, and whose outputs are the motor's driver control voltage (set via a DAC), and display. The goal, in this example, is to have the motor speed match the desired speed using a motor driver output. For the purposes of this project the Control Node would consume the sensor data, calculate the speed error and send control signals (feedback) to the Remote Node to adjust the outputs.

Again, these are only examples – for your project you will define the application and determine the control algorithm, appropriate sensors/output devices that includes at least 2 real sensors and 2 simulated or real output devices (display, relays, motors, actuators, etc.).

2 Learning Objectives

After completing this project, you will be able to:

- Build images and applications that apply concurrent programming techniques on Linux and FreeRTOS OSs
- Develop FreeRTOS hardware "driver" functionality for 2 different sensors
- Apply IPC and concurrency programming concepts to an application distributed between a 2 node system that executes a simple, control-loop or state machine feedback system using a Remote (sensor/output) Node and a Control Node
- Apply basic SW error reporting, fault detection and fault tolerance techniques to an application
- Apply basic SW engineering and project management techniques to a development project

3 System Capabilities

While you have the freedom to innovate your own product application utilizing a selection of sensors/outputs, the project shall embody the fundamental system capabilities described in this section as a part of the overall application as well as the project requirements.

1) The system shall start up and provide service automatically upon application of power without human intervention

2) The system shall employ a Control Node provide closed-loop or state machine control of a Remote Node which has sensors and outputs.
3) To support performance and failure analysis, the system shall support a logging function to record data, significant system “mode” switches and control events (e.g. start up, significant events, state machine changes, etc.), as well as error/faults/failures. These logs shall be readable/extractable from the Control Node for post-run or post-event analysis review.
4) Taking into consideration the product application, the system should support a degraded level of service in the absence/failure of any single sensor.
5) The system shall support appropriate error reporting, fault detection, fault tolerance, and fail-safe behavior.

4 Project Requirements

The project has technical, documentation and project management requirements which are detailed in the following sections and will be evaluated according to the project Rubric.

4.1 System Configuration

The system shall be comprised of two nodes, a Control Node and a Remote Node, that are connected via communications link.

4.1.1 Control Node Hardware/Software

- The Control Node shall be comprised of the BeagleBone Green running a customized Linux image.
- The Control Node shall have an alerting mechanism to indicate at least 3 levels of application operation – normal operation, degraded, failed/out-of-service. This mechanism is your choice and can be status LEDs.
- **Extra credit for operational status indicated by audible buzzer, or 7 segment display, etc.**
- The Control Node software shall provide the closed-loop or state machine control logic for the application.

4.1.2 Remote Node Hardware/Software

- The Remote Node shall be comprised of the Tiva C-Series LaunchPad board running your customized FreeRTOS image.
- The Remote Node shall utilize at least 2 sensors – one of which may be from Project 1 (e.g. temperature or Lux sensor) and one of which is not from Project 1. Both may be new (non-Project 1) sensors.
- **Extra credit for each non-I2C based sensor.**
- The Remote Node shall support at least 2 control “outputs” signals for the product application. They may be simulated outputs via a LEDs.
- **Extra credit for real output devices such as actuators, motors, displays, etc.**
- **Extra credit for additional output devices**

4.1.3 Control-Remote Node Connectivity

- The two nodes shall be connected or networked using any full-duplex method sufficient to support the application's need for communicating control/sensor information, transmit data, events, and logs and otherwise support the functionality expressed elsewhere in the requirements.
- The networking configuration of the system can be pre-administered (a static config). E.g. IP Addresses and Port numbers can be statically defined for the Control and Remote Nodes.

4.2 System Functionality

While the project's end application is your own choice, the system shall provide the common services and features described in this section.

4.2.1 Automatic startup

- The system nodes shall automatically (hands-free) start up upon the application of power.
- The product application shall support powering up
    - Control Node first
    - Remote Node first
    - Both nodes "simultaneously"
- Networking between the two nodes can be preconfigured (e.g. known ethernet IP addresses/port numbers).

4.2.2 Remote Node Sensing

- The Remote Node must periodically "measure" sensor data and detect appropriate events
- In the presence of the Control Node, the Remote Node shall report data/events to the Control Node control application
- In the absence of the Control Node, the Remote Node shall still autonomously sense data and appropriate events

4.2.3 Nominal Application using Closed-Loop or State Machine Control

- Using data and events from the Remote Node, the application code in the Control Node shall provide real-time decisions and provide feedback by sending control "signals" to the Remote Node output devices/interfaces.
- The application needs to support control functionality for your product's functionally, thus will need some requirements. You shall define at least three requirements for your product. These are to be included as part of Phase 1 Architecture, Design, Requirements document and will be used as part of the Project 2 Rubric.

  At least one of the three requirements shall define the control algorithm (e.g. control loop or state machine) behavior for the system.

  For example, given the gesture/temperature sensor door-entry system described in the Overview, a requirement might be - The system shall unlock the inner door lock in the presence of a human hand with the "peace" sign and the temperature sensor reads greater than 98.6°F.

4.2.4 Fault Detection/Tolerance Behavior

Applying techniques from Project 1 (such as BISTs) as well as other methods discussed in class, the system should detect errors and failures during startup and well as normal operation, then react accordingly.

- The Control Node shall detect the arrival and disappearance of the Remote Node
- The system shall detect faults/failures of any sensor at start up and during normal operation.
- In the event of a failure (node, comms, sensor, etc.), the system and nodes shall operate in an (predetermined) fault tolerant/fail-safe manner appropriate for the product application
- With both nodes operational and in communications, the system shall be able to provide a level of service given any single sensor failure (degraded operational state)
- In the absence of control-loop feedback from the Control Node, e.g. the Remote Node loses communication with the Control Node, the system shall go to an appropriate state (e.g. fail-safe, degraded capability, shutdown, etc.)

4.2.5 Logging

To support performance and failure analysis the system shall support logging and log management functions to record and retrieve data and events. The number of log files and how they are managed is shall be your own choice (e.g. on the Control node - one log file for Remote Node, another for the Control Node, or all records in one file).

The logging functionality requirements are:

- The Control node shall collect and record system events such as system "mode" switches and control events (e.g. start up, state machine changes, etc), errors, alarms, faults and actions. Examples of events that should be recorded include failure of a sensor, loss of communication with the Remote Node, and degraded operational state.
- Log files shall survive a reboot and power cycling of the system.
- The Control Node shall time-stamp and capture all log entries to a log file or files – Both Control Node and Remote Node events shall be logged in the Control Node.
- In the event of loss of communications between the nodes, each shall node continue to capture and log events.
- Logs shall be readable/retrievable from the Control Node for post-run or post-event analysis review.
- **[Extra Credit] Upon restoration of communication between the nodes, the Remote Node data/events shall be forwarded to the Control Node.**

4.3 Project Management Requirements

The project shall be developed in 3 phases.

1) Requirements Definition, Architecture, and Project Plan document and review
2) Code Development and Repo submissions
3) Demonstration of Capabilities and Final Report

A portion of the Rubric will be based on project management aspects.

4.3.1 Project Team

The project team shall consist of 1 or 2 members, exclusively of students from the class.

4.3.2 Requirements Definition, Architecture, and Project Plan document and review

Like any substantial commercial embedded software development project that is to meet project and product requirements and a deadline, you are to create a system and software architecture diagram along with some documentation of your design. This document shall be turned into Canvas before writing any code. It can be submitted earlier than the due date, if you wish to begin coding sooner.

The purpose of this activity is to:

1) Assist in assimilating and understanding the requirements of the project.
2) Reinforce good engineering practices
3) Use project management techniques to ensure an on-time, high quality deliverable e.g.
    - Guide/inform a development plan – e.g. Identify high risk areas to address first
    - Identify and address where complexity may exist in your design

The required Phase 1 document submission elements shall include:

- **Team Members** – Names of the 1 or 2 team members
- **Product Description** – Project name and a brief description of the overall capabilities of your product and application, including what sensors and control outputs you plan to use.
- **Requirements** – Provide the 3 or more requirements for the specific functionality you are implementing in the product. One of these requirements shall be for closed-loop or state machine control – reading sensors, a control algorithm of some sort, leading to feedback of control signals to outputs.

  Consider and write these as the application use-cases and test cases. These are specifically what you plan to implement and are above and beyond those described in Section 4.1 System Configuration and Section 4.2 System Functionality. See Section 4.2.3 Nominal Application using Closed-Loop or State Machine Control for an example.

- **Architecture Description** –Diagrams of the software design and the interactions between nodes, hardware and major software components including
    - Interaction diagram (e.g. message sequence diagram) showing the inputs, control loop/state machine algorithm interactions, and output sequence(s) between Nodes

    For each node a diagram and description include:
    - Tasks, modules, kernel, software communication interfaces, etc.
    - Existing libraries and design elements planning be used in your design and explicitly included in your compilation.
        - For example, in user space indicate you be using pthread for your task design
    - Indicate the hardware interfaces for sensors, outputs and external communications
    - Describe each task names, software structures, and responsibilities.
    - A first-cut definition of the needed API/functions for tasks and major functions
        - Indicate the type of interface and functional support needed. E.g. IPC types, get_temp() and get_light() API into the tasks to retrieve data.
    - Other important concepts will be what is user vs. kernel space software. Indicate if something is a kernel module or a system call. etc.

Note: You shall note on your figures and documentation where you intend use existing library code, drivers, etc. and what will be your own code.

- **Project Plan** - Using a planning tool, such as Project Libre or an Excel Gantt chart, show work tasks and effort in a "work breakdown structure" (WBS) of your project development, including the development of this Requirements, Architecture, and Project Document, as well as a breakdown of the coding development work (coding tasks/IPC/etc., integration, testing, milestones). Each item in the WBS should estimate the effort to perform the work. Identify any high-risk areas and development dependencies.

4.3.3 Code Development and Repo submissions

During the development and at its conclusion the project is comprised of multiple work-products – code, makefiles, project plan, etc. To support good SW engineering practices and hygiene, the following are required aspects of the work products and repository.

- Code shall be developed using the ESE Coding Standards including required source file comments. E.g. Comments in each file…
- Code shall be regularly committed to the team's project repository constructed in a reasonable directory structure
- Both partners shall comment and check-in code
- Repo/Code requirements
    - Directory name Project 2
    - These items must be check-in to project repo:
        - Project Plan
        - Makefiles (Linux) and CCS Project files (such that the Tiva project can be built)
        - Source code
        - Readme.txt – description of files/functions in directory
        - Executables
        - Final Report
        - Code Dump
- At final submission (project-2-rel tag) the repo shall contain the project's Linux makefiles and CCS workspace/project files with a buildable source tree for each of target board, and corresponding executables

If you created a private repository, you need to add the instruction team as a collaborator to the project.

4.3.4 Demonstration of Capabilities and Final Report

At the completion of the code development phase, there is a demonstration to allow the opportunity to show off and test the project's capabilities, the fruits of your hard work and share key learnings. Demonstrations will during class time and lab hours. A set of use-cases (tests) will be performed to exercise your product's capabilities as well as standard, common capabilities. E.g. Removal of a sensor device.

Final Report

The project's final report will be comprised of a test plan, test results from executing the project's test plan against the project-2-rel tagged code, reasons for failed test/missed requirements, and a brief description of your key learnings.

Test Plan – This is a set of tests and brief descriptions of how each test validates specific operational requirements of the system, including your own self-defined requirements.

Test Results – This is a list/table of tests and the corresponding results of running each with the results rated as Pass, Fail, Pass with exception (including the anomalous behavior and the potential reason)

Key Learnings – This describes your *individual* key learnings in the project. These can be high-level (e.g. architecture) or low level (e.g. Sockets between FreeRTOS and Linux are hard/easy because…)

Code Dump

This shall be a single file in a .pdf format that is submitted to the Canvas Project 2 Assignment page. It shall be generated from the git repository code tagged **project-2-rel.** This allows a check for plagiarism. No other formats will be accepted. As with all course work, any online sources must be cited in the code files and any preexisting library code must retain licensing/credits in the code. You shall comment your own code indicating the author. Failure to give credit is a violation of the Honor Code

- To generate the code dump report, "cd" to the project top-level directory, and execute this one liner command to copy all files and folder contents into a single file to submit.
  ```
  $ find . \( -name '*.c' -o -name '*.h' \) -exec cat {} \; > allcodefiles.txt
  ```

5 Resources

These resources may be helpful in the development:

Tiva-C Series Launchpad TM4C1294 Eval Kit

- http://www.ti.com/tool/EK-TM4C1294XL

TI Code Composer Studio

- http://www.ti.com/tool/CCSTUDIO

FreeRTOS

- https://freertos.org/index.html
- https://freertos.org/Documentation/RTOS_book.html
    - Hands-on Tutorial/Guide
    - V10 Reference Manual
    - Book Companion Source Code

BeagleBone Green

- http://beagleboard.org/green
- http://beagleboard.org/buildroot
- https://github.com/beagleboard/buildroot
- https://bootlin.com/doc/training/buildroot/buildroot-labs.pdf
- Lots more at bootlin.com…

Texas Instruments Temperature Sensor

- http://www.ti.com/lit/ds/symlink/tmp102.pdf

Broadcom Light Sensor:

- https://www.broadcom.com/products/optical-sensors/ambient-light-photo-sensors/apds-9301

Mastering Embedded Linux Programming, 2nd Ed, Simmonds.

6 Project Deliverables, Submissions, and Rubric

6.1 Tagging, Deadlines, Late submissions

- Your first repo tag "**project-2-rel**" will be considered the final submission for all Phase 3 materials. Submissions after that tag will not be considered, evaluated, or graded.
- Failure to apply the **project-2-rel** tag will result in 0 pts for the Phase 3 submittal.
- Late submissions will be assessed at 10% per day penalty.
- In any case, no submissions will be accepted after May 2nd.

6.2 Rubric

Item	Value (points)	Due Date
Phase 1- Requirements Definition, Architecture, and Project Plan document		April 10th@ midnight or sooner
Description/Requirements	10	
Arch. Description/Design	10	
Project Plan	10	
Phase 2 – Code Development & Repo Submissions.		April 29th@ midnight
Consistent use of Version Control	5	Regularly
Adherence to ESE Coding Style Guidelines	5	
Repository structure and content - inclusive of all required elements	10	
Phase 3 – Repo Tag for Demonstration of Capabilities and Final Report		April 29th @ midnight
Buildable controller project using a checkout from repo – 10 points for per board. (e.g. CCS Workspace/project files/source and Linux tree with Makefiles)	20	
Executables in repo	5	
Code Dump text file in repo	5	
Final Report	20	
Automatic System Startup	10	

Control Node HW/SW Core Operations (Tasking, IPC, Connectivity, etc.)	10	
Control Node Operation Status Reporting/Alerting	3	
Remote Node HW/SW Core Operations (Tasking, IPC, Connectivity, etc.)	10	
Remote Node sensor 1 functionality	15	
Remote Node sensor 2 functionality	15	
Remote Node output 1 functionality	5	
Remote Node output 2 functionality	5	
Demo of Self Requirement 1 that includes Closed Loop Control or State Machine behavior	15	To be scheduled after Final Report
Demo of Self Requirement 2	15	
Demo of Self Requirement 3	15	
Fault Management Handling	10	
Logging Functionality	12	
Extra Credit		
Control Node operational status as audible buzzers, 7 segment display, etc.	5	
Control Node collecting Remote Node logs after restoration of comm between nodes	5	
Tiva-C non-I2C sensor interface	10	
Tiva-C additional (beyond 2) different sensors	2-10	
Tiva-C non-LED output devices	2-10	
Tiva-C additional (beyond 2) output devices	2-10	
Application Complexity, Performance, Design	10	