CIS 4410 & 5410: Embedded Software for Life-Critical Applications

Smart Pacemaker-Challenge Project

27 October 2025

Objective

This project is to familiarize the students with the basic aspects of the model based design, verification, and implementation of life-critical CPS/IoT applications. In particular, the project is to learn how to develop life-critical embedded real-time systems and IoT applications.

Problem Description

You are to design a smart pacemaker. This IoT application contains three parts: a pacemaker, a monitor checking the status of the heart, and an application which collects and displays the heart-rate data from the monitor.

(a) Pacemaker (on Arduino)

A pacemaker is a medical device, which senses heartbeats regularly and sends electrical impulses to the heart in order to maintain the proper heart rhythm of a patient.

Timing constraints are one important aspect of correctness criteria. In this project, you will mainly focus on designing, verifying, and implementing a pacemaker controller with timing constraints. The modeling, implementation, and validation of a VVI mode pacemaker is the basic requirements for this project. To test the model and implementation of your pacemaker controller, you are also required to model and implement a simple random heart.

More information with the Pacemaker Challenge can be found at
· The specification document https://sqrl.cas.mcmaster.ca/_SQRLDocuments/PACEMAKER.pdf.
· The website https://sqrl.cas.mcmaster.ca/pacemaker.htm.

(b) Monitor (on Codio)

The monitor collects heart rate information from the pacemaker and calculates the average heart rate during the window of last W seconds at every P second interval (where W can be between 20 and 60 seconds and P can be between 5 and 20 seconds). Moreover, when the heart beats too quickly or too slow, the monitor will give an alarm. The heartbeat rate and the alarm information is displayed by an application (running on Codio). The MQTT will be used for communication.

Your project is divided into the following milestones that are to be carried out in sequence. **Each milestone has its own deadline**. **Please start early.**

Milestone 1: Modeling and verification of a pacemaker

You are asked to design a high-level model of the system. This model will serve as the blueprint for subsequent work. Based on the model, you will verify the model's properties to ensure the design is (conceptually) correct. You will also use this model to serve as a guide for implementation. You are to use the modeling language UPPAAL (http://www.uppaal.com) that was introduced in class. Using UPPAAL, you can create timed automata models and can verify properties specified in timed computational tree logics (TCTL).

You are asked to verify and assure that your model satisfies the required correctness properties. This phase will involve reading the pacemaker specification document, identifying correctness properties, specifying the properties in TCTL, and employing the UPPAAL tool to verify the correctness of your design.

Milestone 2: Implementation of a pacemaker

You are to implement a pacemaker using Arduino and a pacemaker/heart monitor on Codio. First, the implementation of the pacemaker controller must be consistent with the UPPAAL model. This consistency will be used as an argument for the correctness of the implementation in your assurance case (Milestone 3). A random heart simulator also needs to be implemented based on the UPPAAL model in Milestone 1 to test the pacemaker controller and the monitor. Second, you need to implement communication between the pacemaker and the monitor using MQTT. Third, you are to implement the monitor application on Codio that receives the heartbeat and alarm data from the device and displays it on your laptop.

Milestone 3: Validation and Assurance Cases

In this milestone, you are asked to develop an assurance case to argue the correctness of your pacemaker system. This involves providing arguments that the properties you have verified in Milestone 1 are also not violated in your implementation. This is done by profiling and/or testing of your implementation to gather enough evidence. Moreover, you are also required to validate the monitoring of the heart and communication between your Arduino and Codio using test cases. All the evidence is put together in the assurance case to justify your system-level claim.

Milestone 4: Demo & Final Report

Each group must submit a final report that includes the design models and correctness properties, implementation architecture, and assurance case. Also, each team is to present and demo their system during project presentation day.

Collaboration Policy

The project can be done in groups of two. It is **not** advised that each member take one milestone and work alone for the following reasons: First, the workloads of the milestones are different. Second, it would be beneficial for all members to know the process of the model-based software design of an embedded and cyber-physical system. Third, there will be questions in the 2nd exam on the different milestones of the project.

Assessment

The final grade for the project is based on the items below. The first six should be included in the project report.

1. The design model (15%)
2. Adequacy of correctness properties considered (15%)
3. Verification results, i.e., how well the model satisfies correctness properties (10%)
4. Implementation code (25%)
5. Provide an assurance case with convincing arguments for the correctness of your design/implementation (25%)
6. Member contributions to the project (5%)
7. Demo (5%)

Last update on 11/07/2025.