CIS 441 & 541: Embedded Software for Life-Critical Applications

Smart Pacemaker-Challenge Project

Milestone 2: Implementation

Due: 11:59 pm, Monday, 24 November 2024

(Will accept Wednesday without late penalty)

In this part, you are to implement (1) a pacemaker controller and (2) a heartbeat monitor application for storing and presenting the heartbeat information. (1) is to run on Arduino and (2) is to run on Codio. When implementing the pacemaker controller, you should follow a model-driven development approach, i.e., the implementation of your pacemaker should be based on the Uppaal models you constructed and verified in Milestone 1.

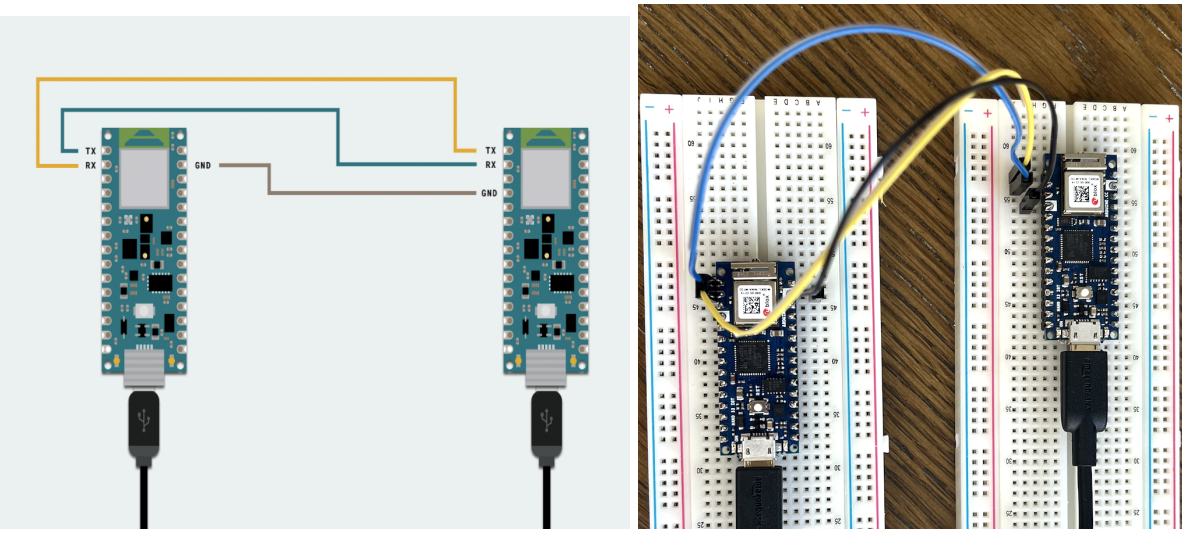
1. Pacemaker implementation and test generation

In this milestone, the heart signals are generated from a heart (simulator) running on a separate Arduino. The random heart will simulate all possible behaviors of a heart that ranges from a normally functioning heart that beats regularly to a problematic heart that beats randomly, within specified bounds. For example, suppose the parameters of lower and upper bounds for the next beat interval are 333 ms and 1500 ms, respectively. Here, whenever a heart beats itself or is paced (at time *t*), the next spontaneous beat will happen after randomly chosen time between (*t+333*) ms and (*t+1500*) ms.

When the heart simulator runs on one Arduino board, it outputs a signal if the heart beats itself. To detect this event from the pacemaker controller, either polling or interrupts can be used. With polling, the pacemaker controller periodically checks the communicating ports to see if the port values have changed. With interrupts, the pacemaker controller is interrupted to handle this signal and then resume its execution.

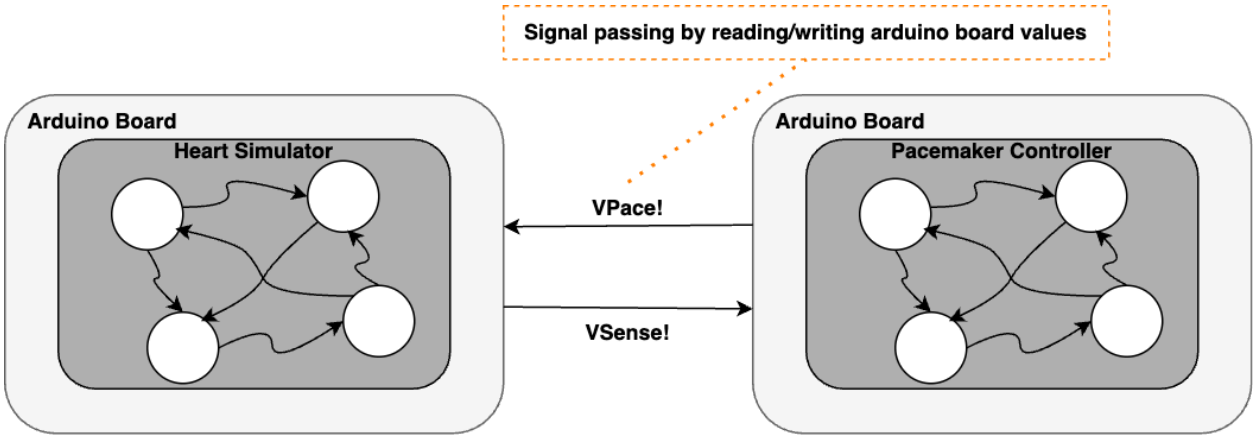
Upon detecting a heart signal, the pacemaker controller (on another Arduino) will decide whether (or when) to send a pacing signal to the heart, according to the pacemaker specification. The pacemaker controller will be able to communicate with the heart simulator using input and output pins.

To use UART communication between two Arduino boards, connect the TX, RX, and GND pins by crossing TX and RX, and connecting GND to GND between the boards as shown in the figure below:



In addition to the sensing/pacing communication between the two boards, a LED should blink whenever a sensing signal is generated by the heart, and another LED should blink whenever a pacing signal is generated by the pacemaker.

A typical system architecture is as follows:



2. Heartbeat Monitor

The heartbeat monitor has two features: 1) calculation of average heartbeats and 2) detect abnormal heartbeats. More specifically, the monitor calculates the average heart rate during the window of last W seconds at every P second interval (where W can be 20, 40 or 60 seconds and P can be 5, 10, 15 or 20 seconds).

For abnormal heartbeats, there are two scenarios possible. The first scenario is if during a time window, the heart is beating mainly by artificial beats produced by pacing signals of the pacemaker, then it is beating slowly or in other words not beating naturally. In this case, there should be an alarm alerting that in this window, the heart is beating slowly because the number

of pacing signals sent by the pacemaker in this window was above some threshold (you can come up with a value for this threshold). The second scenario is when the average heartbeat value is above the URL defined in Milestone 1, then it means the heartbeat rate is too fast. The monitor should detect either case and perform alarms (you may choose your own way to illustrate alarms on the monitor app on Codio). Moreover, two types of alarms should be distinguishable.

Submission. To submit your work, write a simple document describing the main logic of your code, as well as the necessary information on how to connect up the two Arduino boards provided to you. Also, record your monitor app on Codio. Zip this document and your project files (source code, hex files, recording, etc.) and submit to Canvas.

Last updated on 10/21/2025.