Embedded Assignment: Nim

Due: October 18, 2017 by 11:59pm

1 Overview

Nim is an ancient mathematical strategy game in which two players take turns removing objects from a set number of heaps. The goal of the game is to avoid being the player to remove the last object.

For this assignment, you will design, and build your own version of Nim using an Arduino platform. This assignment is designed to develop your skills in embedded programming and I/O concepts, which may be useful in your future projects.

You can read more about Nim at `https://en.wikipedia.org/wiki/Nim`.

2 Requirements

This assignment is divided into two parts: hardware design and software implementation. You will you build your own Nim game board to play a game between a human and a computer player. Your board should demonstrate the current game state and allow the player to select their moves. In addition, you will write the software to implement the logic for the game as well as a computer player.

2.1 Hardware Design

There are many interpretations for the game of Nim. As such, you can design the interface in any way you wish, so long as it conforms to the following requirements:

- Your game must have at least two heaps
- The heaps must be of size four or higher
- The starting player for the game (human or computer) must be selectable at the beginning of the game
- Your game must have a button to reset the game back to the beginning

In general, your game board will use LEDs to display the game state and accept user input from some number of buttons. It is up to you to determine what interface you want to provide, within the constraints of your Arduino hardware.

Here are some examples for how to display the game state using LEDs:

- 2 rows of 4 LEDs, each row indicating a different heap
- 1 row of 8 RGB LEDs, with different colors to denote each heap

You can use as many LEDs or buttons as you like to implement the game, within the power and I/O limitations of on your Arduino board.

Going beyond these requirements may be awarded with extra credit. Depending on your implementation, you may also require additional components (shift registers, etc.) to increase the number of inputs or outputs via multiplexing. If you have questions on designing your game hardware, or what constitutes extra credit, please feel free to consult the course staff for help!

For more information on components available to you, see Section 3.

2.2 Game Logic

All logic for your game should be implementing using your Arduino. Your game logic must conform to the following requirements:

- Your game should be played using *misere play*–that is, the player to take the last object loses the game.
- You can use any strategy you want to implement the CPU player, as long as it has the capability of winning a game when the option is available.
- The CPU player must respond in real-time. There should be no discernible lag between the player's move and the CPU player's response.
- When the game is over, you must have some way of displaying which player wins (perhaps by flashing a pattern on the LEDs). Once this occurs, the game state should be cleared and a new game should be started.

You can go beyond these requirements in any way you see fit (supporting both play styles, multiple strategies, game variants, etc.), which may qualify for extra credit. Any design decisions you make should be documented in the README file submitted with your work.

When designing your circuit and hardware, you should be thinking about the user interface: how can you make a game that is both fun and easy to play? Also, be sure to keep in mind potential usage issues. For example, what happens if multiple buttons are pressed at once?

3 Parts

You should have your own Arduino platform and breadboard to complete this assignment. In addition, each student is entitled to the parts listed below:

- Electrical wire
- Resistors
- Pushbutton switches

- RGB LEDs
- Diodes

Additional components (diodes, shift registers, etc.) may be available, but are subject to availability. To receive your components, please go to Prof. Reiss' office (CIT 403) to sign them out. You can keep the parts for the semester for future use (such as for your labs and final projects).

You can also supplement these parts with any other components you may have—please confirm your design with the course staff before building it, however.

Note: Neatness of your circuit is part of your grade! You may wish to spread out your design over more than one breadboard to keep your work neat. For details on grading, please see the Grading rubric in Section 9.

4 IoT Lab and Help Sessions

You will have access to the IoT lab as a space to work on your project, located on the 8th floor of the Sciences Library. The course staff will also schedule optional help sessions which you can attend for extra help—information on these sessions will be forthcoming.

If you wish, you you can solder your assignment to a protoboard after completing it. Instructions for soldering will be provided at one or more help sessions. Protoboards for soldering can be provided by the course, or you may use your own.

5 Collaboration

You can work on this assignment individually or in pairs. However, all work must be the creation of you and your partner and must not use an existing implementation for Nim. **If you are working in pairs, each team member must have a deep understanding of the entire project, not just their own part.**

6 Helpful Hints and Resources

- Hardware and software documentation on your Arduino can be found at `https://arduino.cc`. The library reference for the Arduino API can be found at `https://www.arduino.cc/en/Reference/HomePage`.
- Open Electronics Lab (`https://circuits.io`) is a free resources for prototyping circuits. This is a great way to try out your designs before you build them!
- By default, the Arduino IDE disables all compiler warnings. If you want to build with warnings enabled (often a good practice), you can turn them on in the IDE preferences.
- If you do not like using the Arduino IDE for editing code, you can edit files in your own editor and just use the IDE for compiling and loading programs onto the Arduino. If you want to do

this, select **Use External Editor** from the IDE preferences window, which will configure the IDE to auto-reload your files.

7 Submitting your work

You should write a README file documenting any design decisions you took when creating your work, any bugs you have in your code, and anything else you think the TAs should know about your project. If necessary, be sure to document any instructions to compile and run your program (this is not required if you are just using the Arduino IDE).

You can submit your work using the appropriate assignment on Canvas. Please submit a `zip` file containing your source code, documentation, and any other files necessary to build your project.

8 Evaluation

We will use "interactive grading" for this assignment: you will demonstrate your work to the course staff for a grade. Your grade will be based not just on the software functionality, but on the hardware design, efficiency, and documentation of your work.

The interactive grading will take place two to four days after the project is due. Instructions for signing up to meet with the course staff to demo your work will be provided as the deadline approaches—watch the mailing list for this information. You must demonstrate your work in order to receive a grade. Be prepared to answer question about both your hardware and software design. If you worked as a team of two students, **both** students should be prepared to answer questions about **all** components of the design.

9 Grading Rubric

Task	**Total Points**	**Points Earned**
User interface design: User inputs	20	
User interface design: Displaying game state	20	
Logic for reading/writing to I/O	20	
Game logic (obeying game rules)	10	
Computer player	10	
Hardware neatness	10	
Software design (using tasks, comments, easy to follow)	10	
Bonus: extra software/hardware features	(20)	
Bonus: Prototype soldered to protoboard	(10)	
Total	100	