Real-Time Assignment: Cuckoo

Due: 11:59PM November 14, 2016

Contents

1 Overview
2 Parts, Consumables and Hints
3 IoT Lab
4 Constraints and Considerations
5 Useful Information
6 Optional Work
7 Collaboration Policy
8 Documentation
9 Submission
10 Evaluation
11 Testing

1 Overview

In this assignment, you will write a program that simulates a cuckoo clock running on the Arduino platform using the parts listed in Section 2. From Wikipedia: “A cuckoo clock is a clock, typically pendulum-driven, that strikes the hours using small bellows and pipes that imitate the call of the Common Cuckoo in addition to striking a wire gong. The mechanism to produce the cuckoo call was installed in almost every kind of cuckoo clock since the middle of the eighteenth century and has remained almost without variation until the present.” In general, the little cuckoo pops out from the clocks ornamental door and given that the clock is equipped with music sounds on the hour and half-hour; once at 1:00 and up to twelve times at noon or midnight. Your Arduino program should keep track of time and play the cuckoo sound k times according to the hour. The sound file can be found at
/course/cs160/asgn/cuckoo/sounds/cuckoo.au. For every half-hour, play the ding-dong sound file (/course/cs160/asgn/cuckoo/sounds/ding-dong.wav). This assignment focuses mostly on the software implementation. For the board design, the simplest approach is to connect one of the speaker legs to the digital/analog pin preceded by a resistor (minimum of 120Ω) and the other leg to the ground pin of the Arduino board. **Connecting the digital/analog pin directly to the speaker can possibly fry your Arduino board.** Be aware that the higher is the resistance, the lower will be the output volume. You may want to increment this initial design depending on how you decide to reproduce the sound file.

## 2 Parts, Consumables and Hints

Assuming you have your own Arduino platform and breadboard, each student is entitled to the parts listed below. You can pick them up at Prof. Reiss’s office (CIT Room 403).

1. Electrical wire
2. A 0.25W 8Ω sound speaker
3. Resistor(s)
4. Capacitors as needed (optional)
5. Transistors or op-amp chip (as needed)

**Note:** neatness of your circuit will be part of your grade.

In designing your circuit and software, keep in mind potential use issues. For example, what if multiple buttons are pressed at once?

## 3 IoT Lab

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. There will also be help sessions for you to optionally participate in. We have more information about these sessions once we have the times. If you wish, you can solder your assignment after breadboarding it. Soldering instructions will be provided during IoT lab help sessions. The solder-ready circuit boards will be provided by the course.

## 4 Constraints and Considerations

While implementing your Arduino solution, take in consideration the following constraints and considerations:
• Busy-waiting loops are forbidden (assume your program could be doing something else while the cuckoo sleeps or chirps).

• Audio data should be read from the Arduino platform, not from an external source (e.g., your computer, SD card, etc.)

• You can convert the provided audio files to other formats

• The Arduino language is a set of C/C++ functions that can be called from your code. All standard C and C++ constructs supported by avr-g++ (the compiler used for the AVR microcontroller on your board) should work in Arduino.

• You will need a way to synchronize or set the initial time of your clock. Consider using buttons to reset and jump through hours for testing and also for interactive grading. Alternatively, you can use the serial communication with your PC to set the time.

• Since there is no time for ordering external shields, you are limited to the hardware available for the course;

• The function millis() may not be accurate due to interrupt processing.

• If you need extra parts, talk to Prof. Reiss.

This exercise requires research on how to perform periodic, real-time tasks using interrupt routines, how waveforms work and how to reproduce them on the Arduino. It is your task to measure resource consumption and find trade-offs. Do not forget to document them. You should measure the accuracy of your clock over a 24-hour period and report it in your documentation. If time permits, you are encouraged to add other features to your clock (Section 5).

5 Useful Information

The following should help on your journey:

• You can compress the audio data for memory savings. You might want to take advantage of data redundancy, but remember that it can affect the quality of the sound output.

• wav2c\footnote{https://www.github.com/olleolleolelle/wav2c} is a program that converts audio wave files into AVR memory data. AVR is the microcontroller for the Arduino Uno platform. You can also use other tools to convert audio.

• If you run out of data space on your Arduino, you can store and read data from the program space\footnote{https://www.arduino.cc/en/Tutorial/Memory}.
• Because you are not allowed to play the sound data on a busy-waiting loop, consider using timer interrupt requests\(^3\).

• The web is your friend. Don’t fear to look for information when in doubt.

6 Optional Work

The following are worth extra points:

• Play a tic-tac sound to simulate the pendulum movement while cuckoo is sleeping (/course/cs160/asgn/cuckoo/sounds/pendulum.wav)

• Use a display to inform the current time

• Inform current time using synthesized voice (but be careful with the constraints)

• Consider other circuits for your board design (e.g., a DAC using multiple pins and a resistor cascade, amplifier using a transistor/op-amp chip, filtered output, etc.)

7 Collaboration Policy

Students can work individually or in pairs, however all the work must be the creation of you and your partner and not use an existing implementation for tic-tac-toe. If working in pairs, each member should have a deep understanding of the whole project, not just their part. Please use the mailing list or email the TAs if you have any questions.

8 Documentation

You should write a README file, documenting any bugs you have in your code, all the design decisions you took, and anything else you think the TAs should know about your project. Also be sure to document the instructions used to compile, install (if necessary) and run your program.

9 Submission

You should hand in your assignment by running /course/cs160/bin/cs160_handin cuckoo from your project directory. Please copy all the source code, documentation and the Makefile (if it exists) into your project directory, cd there, and run the handin script. Do NOT submit object or executable files and make sure your program compiles cleanly.

\(^3\)https://www.gammon.com.au/interrupts
10 Evaluation

We will have interactive grading for this assignment. Your grading will be based not just on functionality, but also on the design, efficiency, commenting, and documentation. The interactive grading will take place two to fours days after the project is due; location and hour will be announced later. If you do not show to your interactive grading and you do not have a valid excuse, you will be penalized. We will post information necessary to schedule an interactive grading session on the mailing list. Be prepared to answer questions as to your design (both HW and SW) and implementation.

11 Testing

To make sure you will not have unpleasant surprises during the interactive grading session, make sure you have extensively tested your board and software before submitting your code. Imagine you are releasing your cuckoo clock to the market (disregard aesthetics and usability factors for now). How people would interact with it? Will it ever cuckoo at the wrong time?