Real-Time Assignment: Cuckoo

Due: November 24, 2019 by 11:59pm

1 Overview

In this assignment, you will write a program that simulates a cuckoo clock running on the Arduino platform using a few simple parts. 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 clock’s 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 task is to use your Arduino to keep track of the time and play the cuckoo sound \(k\) times according to the hour. This will involve learning how to perform periodic, real-time tasks on an embedded system.

2 Requirements

2.1 Hardware Design

The hardware design for this assignment is fairly straightforward, though you can extend it if you wish. The simplest approach to connect a speaker to your Arduino is to connect one of the speaker pins to a PWM pin using a series resistor of at least 100Ω, as demonstrated in the Sound lab. Make sure that you include the resistor: 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. We recommend starting with this approach first. After you have the rest of the assignment working, you are welcome to extend your hardware design to include additional components to improve the sound quality. See Section 4 for details.

Note also that you are not required to build a display to show the clock’s time on your breadboard. After you have the rest of the assignment working, you may do this for extra credit—see Section 4 for details.

2.2 Software Implementation

Your software implementation is responsible for two real-time tasks: keeping track of the clock time, and actually playing those sounds on the Arduino. You can design your software any way you wish, as long as it conforms to the following requirements:

- You must keep track of the time yourself using interrupts to provide your time reference. You are not permitted to use any existing C or Arduino Time libraries to do this for you.

- You must play three sound files: a “cuckoo” sound to be played at every hour, and a “ding-dong” sound to be played every half-hour and “tic tock” sound to simulate the pendulum movement every 5 second. Example sound files are provided in the course website. You can use other sound effects if you wish, but you need to keep in mind the space and sound quality constraints of your system.

- You can convert the provided sound files into any format you wish for playback on your Arduino. The recommended format for storing and playing sound files is to store 8-bit, unsigned samples with a sample rate of 8 kHz. This format is recommended to stay within the memory requirements of your Arduino. An example header file is provided for the given “cuckoo” sound to demonstrate how the data can be converted (more info in Section 5).

- 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, a different device, an external shield, etc.). Your implementation must support playing audio files (i.e., audio sample data), not just generate tones. You may not purchase any shields that are designed for playing Audio to supplement your work on this project.

You will almost certainly run out of RAM on your Arduino when storing the sound data. You can get around this (to some extent) by storing data in the Arduino’s program (flash) memory. See https://www.arduino.cc/en/Tutorial/Memory and https://www.arduino.cc/en/Reference/PROGMEM for details on how to do this.

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.

The function `millis()` may not be accurate enough for your application. In addition, note that all Arduino timing functions such as `millis()` and `delay()` utilize Timer0. If you reconfigure Timer0 to perform another function in your program, these functions will produce inaccurate results.

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. A good starting point for this is your implementation for the Sound lab, which demonstrated using two timers to play audio samples at a rate of 8kHz using a fast PWM approximation.

It is your task to measure resource consumption for storing and playing sound files and identifying tradeoffs—do not forget to document any design decisions you make in your readme! You should measure the accuracy of your clock over a 24-hour period and report it in your documentation (to do this, you can simply leave your clock running for 24 hours and report the time drift). If time permits, you are encouraged to add other features to your clock.
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:

1. Electrical wire
2. A 0.2W 8Ω sound speaker
3. Resistor(s)
4. Capacitors as needed (optional)
5. Transistors or Op-amps (optional)

Additional components 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.

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 Optional Enhancements

The following are worth extra points:

- Use a display to inform the current time (for example, make a binary clock, use a character or 7-segment display or 12 LEDs, etc.)
- Inform current time using synthesized voice (but be careful with the constraints)
- Consider other circuits to power your speaker (e.g., use a DAC, amplifier using a transistor/op-amp chip, filtered output, etc.)
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. 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

- 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.

- You can use external programs to convert audio files into samples to use in your program. wav2c\(^1\) is a program that can produce header files from audio data. Audacity\(^2\) can be used to manipulate audio files (including clipping, changing sample rate, etc.) and save them in various formats. Feel free to ask the course staff if you have questions on converting audio.

- 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.

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

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

\(^1\)https://www.github.com/olleolleolle/wav2c
\(^2\)https://www.audacityteam.org
to document any instructions to compile and run your program (this is not required if you are just using the Arduino IDE).

At minimum, your README should provide the following important design details:

- How you connected the speaker
- Results of testing the accuracy of your clock over a 24-hr period
- Memory usage of your sound files (how much space is required, and format in which you stored the data)
- Description of how you use interrupts to track the time and play sounds

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. You must demonstrate your work in order to receive a grade. Please make sure your project is testable during the interaction grading. (For example, shorten the period for each sound playing for the grading session). 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
<table>
<thead>
<tr>
<th>Task</th>
<th>Total Points</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timekeeping using interrupts</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Playing sounds at hour and half-hour intervals</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Storing sound data in Arduino memory (RAM, flash, etc.)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Software design (tasks, no delays, good interrupt usage, etc.)</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>README file (answers important design questions)</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td><strong>Bonus:</strong> Hardware to display time (LEDs, LCD display, etc.)</td>
<td>(20)</td>
<td></td>
</tr>
<tr>
<td><strong>Bonus:</strong> Improved sound output (DAC, op-amp, etc.)</td>
<td>(10)</td>
<td></td>
</tr>
<tr>
<td><strong>Bonus:</strong> Prototype soldered to proto-board</td>
<td>(10)</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>