Ryan Comer – rnc525
Connor White – bcw824

Lab 3 Report

Alarm Clock

Requirements Document

Overview

Objectives

The objectives of this project are to design, build and test an alarm clock using the TM4C123 microcontroller and various circuit components. Some of the main goals include building and debugging modular software, processing user input, and to remove critical sections within our software. We will also learn how to generate sound using a square wave on a speaker component.

Process

The alarm clock project will be developed using the TM4C123 board, and the on-board switches will be used to handle user input. The system and all of its components will be built on a solderless breadboard and run on USB power. The on-board LED will be used as a heartbeat for the system. The speaker will be external and connected to the breadboard. There will be five hardware/software modules: switch/keypad input, time management, LCD graphics, heartbeat, and sound output. We will design and test each module independently from the other modules. After each module is tested, the system will be built and tested as a whole.

Roles and Responsibilities

The TA is our client for this system, and we (the students) are the engineers that were tasked to build the alarm clock. The purpose of this project is to build the system but also to learn how it works, and so it is the responsibility of the students to understand all parts of the system, even if they didn't work on that part. Because the system should be modular, we will split up the work between the 2 partners on the team, but we will both understand every component of the system for the demo.

Interactions with Existing Systems

Existing systems that our project will use include a TM4C123 microcontroller board, a solderless breadboard, and a ST7735 color LCD screen. The components will be connected together using the breadboard, and the microcontroller will be powered with a USB cable which then powers the entire system.

Terminology

Power budget	Maximum power that the power supply can drive. This limits the number of components that you can connect to a supply
Device driver	Interface between a device and a software module. An example would be the functions defined in ST7735.c
Critical section	Multiple threads access shared memory locations/variables. A higher priority thread has to interrupt another thread and write to a

	variable that the lower priority thread was reading.
Latency	The time between a service request and the process actually being serviced
Time jitter	The difference in the maximum latency and the minimum latency that occurs. Some systems have a strict minimum time jitter requirement.
Modular programming	Break up your software into modules with distinct tasks. Each of the modules should be kept separate so that you can easily debug or change specific features of your system.

Security
Software from Tivaware will not be used in this project. Instead, the software will interact with the board at a low level and directly write to the appropriate registers. Code written for this project can't be shared with any other EE445L students (past, present, and future). It is the responsibility of both partners to ensure that their code does not fall into the hands of other students.

Function Description
This section will detail the specific functionality of the system by listing the expected functions and describing how the components of the system are connected.

Functionality
The alarm clock system will be able to perform the following functions:

1. Display hours and minutes in both graphical and numeric forms on the LCD. The graphical output will include the 12 numbers around a circle, the hour hand, and the minute hand. The numerical output will be displayed in the corner on the same screen.
2. Set the current time by holding down SW1 (on-board) for 2 seconds.
3. Set the alarm time by holding down SW2 (on-board) for 2 seconds
4. Set the alarm type by holding down both SW1 and SW2 (on-board) for 2 seconds. SW1 and SW2 are then used to cycle through the possible alarm types. To select an alarm, the operator will press both SW1 and SW2.
5. Sound the alarm when the alarm time is hit. The alarm type that is played should be the one that the user chose
6. Stop the alarm by pressing any on-board button while the alarm is sounding.
7. LED heartbeat to show that the system is running

Scope
The lab is divided up into 3 parts: preparation, demo, and lab report. In the preparation, we are expected to have software written for each module as well as a main program to test the alarm. For the demo, we will show the TA our system with the alarm-selector as an extra feature. We will then answer any questions that the TA may have for our team. For the lab report, we will include this requirements document, a software diagram, hardware specifications, and answers to the questions shown in the lab manual.

Prototypes
A prototype system running on the TM4C123 board, ST7735 color LCD, and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

Performance
The system will be judged by three qualitative measures:

1. The software modules must be easy to understand, well-organized, and modular.
2. The clock display should be aesthetically pleasing and accurately display the time.
3. The operation of setting the time and alarm should simple and intuitive.

The system does not have any critical sections. Backward jumps in the ISR should be avoided if possible. User input is handled in the main program, but updating the LCD is done in an ISR. This isn't ideal because it increases the time for the ISR to finish, but it solved our problem of critical sections. Having the ISR that updates the time also update the display ensures that no 2 threads are trying to access this shared time variable. The average current on the 5V power will be measured with and without the alarm sounding. This will show how much power the alarm uses while it is sounding

Usability
User input is handled by using the 2 on-board switched on the TM4C123 microcontroller. The following list shows all of the input options for the user:

1. Set time – Hold down SW1 for 2 seconds. SW1 and SW2 are then used to increment the hours and minutes respectively
2. Set alarm – Hold down SW2 for 2 seconds. SW1 and SW2 are then used to increment the hours and minutes respectively
3. Change the alarm type – Hold down both SW1 and SW2 for 2 seconds. SW1 and SW2 are then used to cycle through the list of alarm types
4. Turn on/off alarm – press any button while the alarm is sounding

The user will be able to set the time (hours, minutes) and be able to set the alarm (hour, minute). There are no seconds displayed for the clock, and so the user can not set that value. When the time is set, the seconds are automatically set to 0. The digital time is displayed in military time with a rollover at 23:59.

The LCD display shows the time using a graphical display typical of a standard on the wall clock. The 12 numbers, the minute hand, and the hour hand are large and easy to see. The clock will also display the digital time in the top-left corner. Instructions on how to use the clock are also displayed in that corner.

Safety
The alarm sound will be VERY quiet in order to respect other people in the room during testing. A resistor connected to the base of the transistor is used to control the loudness of the speaker. Connecting or disconnecting wires on the protoboard while power is applied may damage the board.

Deliverables

Reports
A lab report is due on 09/23. The lab report will contain this requirements document, software and hardware designs, and answers to a few questions defined in the lab manual.

Audits

The preparation for the lab is due on 09/15 and will be graded by the TA.

Outcomes

There are three deliverables: preparation, demonstration, and report.

Hardware Design

This section will cover the hardware for the alarm clock system. There will be a PCBArtist circuit drawing of the system with descriptions of all the important components. The following figure shows the circuit diagram for the alarm clock system:

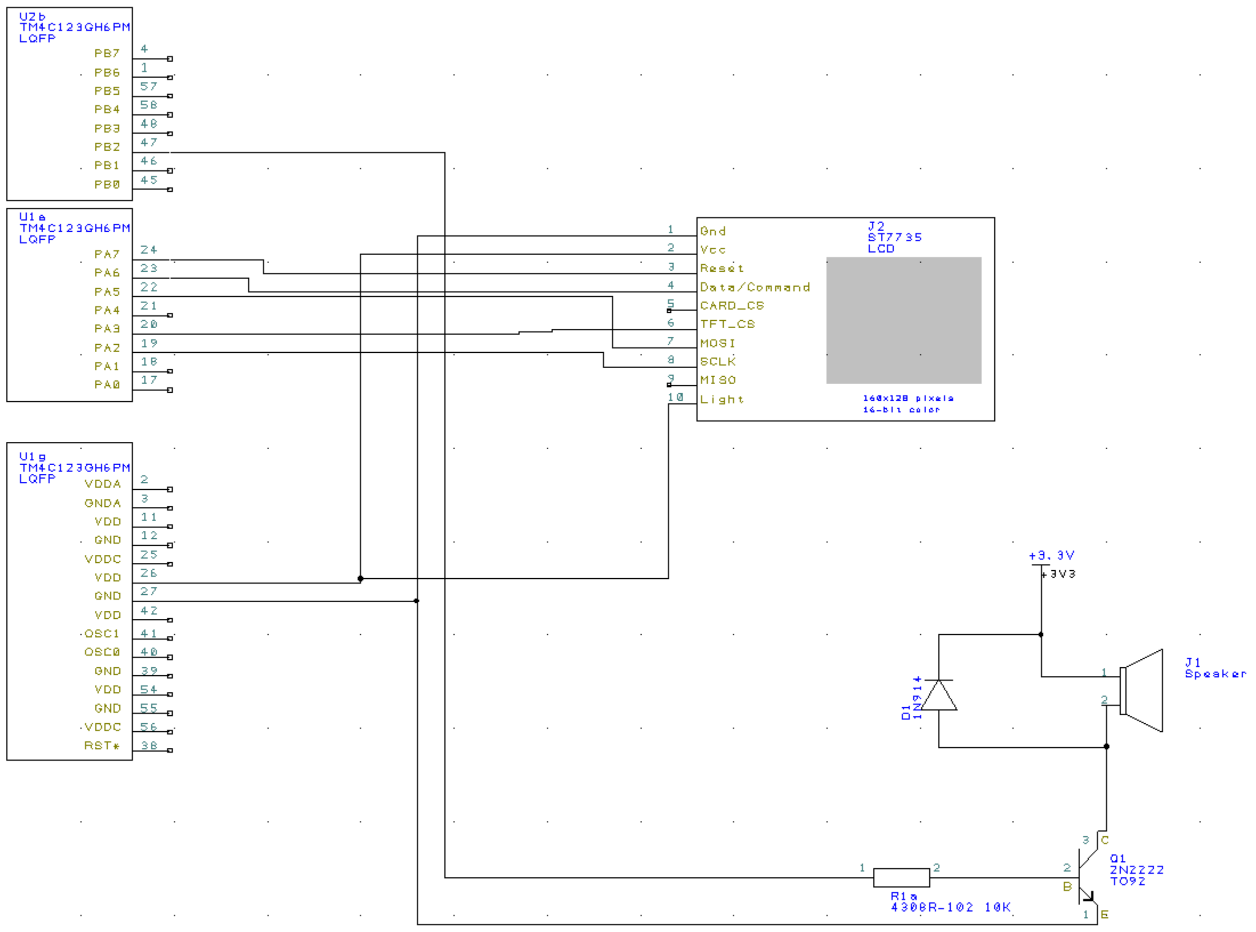

Figure 1: Circuit Diagram

The ST7735 display is used as a graphical output to the user for the system. The digital time and graphical clock will both be displayed using this screen while the system is running. The screen is updated in an ISR right after the time is updated. The connections from the screen to the board are the standard connections as described in ST7735.c and ST7735.h for the screen to work properly.

The speaker component is used to generate sound whenever the alarm goes off. The speaker is connected to VCC and the collector of a transistor. A diode is connected over the speaker to ensure that no backwards current goes through the speaker because this would create inductance and raise the voltage. The transistor is then connected to GPIO pin PB2. A square wave is generated by toggling PB2 at a certain frequency thus causing the speaker to generate a tone. The transistor acts like a switch, and sending current through the base of the transistor causes the switch to flip. Doing this at a certain frequency is what creates the square wave.

Software Design

This section will cover the software for the alarm clock system by showing the software diagram and call graphs for the different modules.

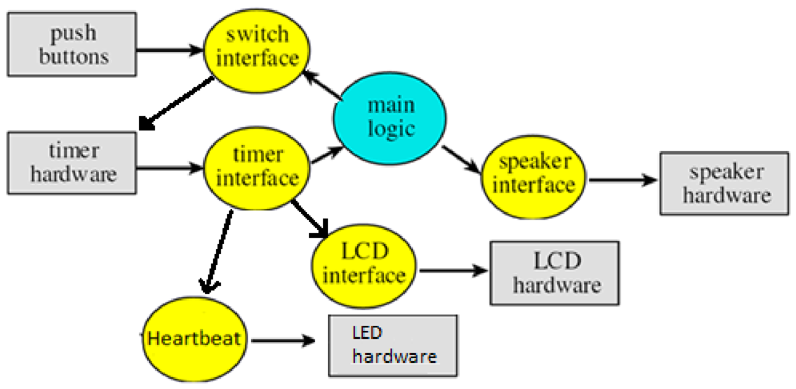

Figure 2: Software Diagram

Figure 2 shows a software diagram for the alarm clock system. The rectangles represent hardware components to the system while the ovals represent software modules. The on-board buttons are used for user input, and the main program will repeatedly check the user inputs by using the switch interface to check the button values. The switch interface uses the timer hardware to debounce the on-board switches. The value of the switch has to remain constant for a certain amount of time, and this time is measured using the timers.

The timer interface is the module that keeps track of the current time, and increments it every second. This is done using the on-board timer hardware. The timer interface also toggles the heartbeat for the system. After incrementing the time, the timer interface will then use the LCD interface to update the screen. This adds undesirable time to the ISR execution, but it solves our problem of a critical section. By having the same thread increment the time and update the LCD with the new time, there is never an opportunity for multiple threads to access the shared variable.

The speaker interface will toggle PB1 at a certain frequency to generate a sound on the speaker hardware. Different frequencies correspond to different sounds.

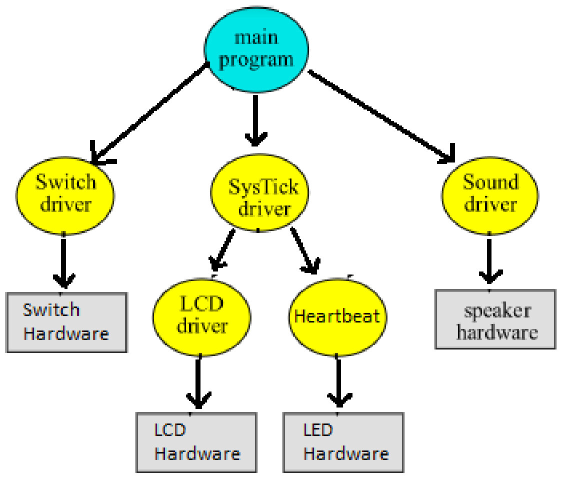

Figure 3: Call Graph

Figure 3 shows the call graph for the alarm clock system. This represents which software modules call functions on other modules. This diagram also shows dependencies in the system. For example, the heartbeat would not get toggled if there was a bug with the SysTick driver, or if the driver wasn't in the system at all.

Measurement Data

This section will cover the measurement data that was required for this lab.

RMS

The first measurement that we took was the RMS value of the voltage source for 5V and 3.3V. VBUS was used to get the 5V measurement, while the usual 3.3V pin was used for the 3.3V measurement. The RMS of the voltage shows us the average voltage that is being supplied, and we would expect these values to be close to the actual value of the voltage source. The RMS was measured using an oscilloscope with the 'Add Measurement' function.

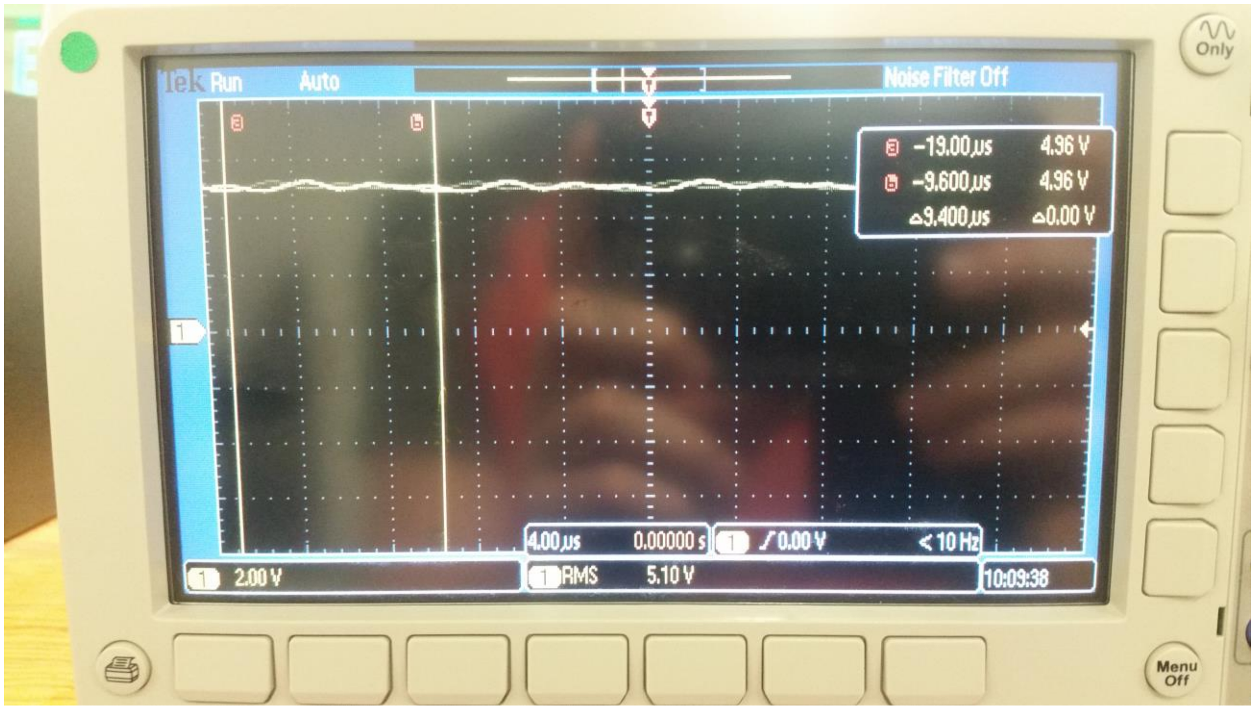

Figure 4: RMS Voltage for 5V

Figure 4 shows an RMS value of 5.10V for the 5V voltage source. This is farther from the actual value of 5V than expected.

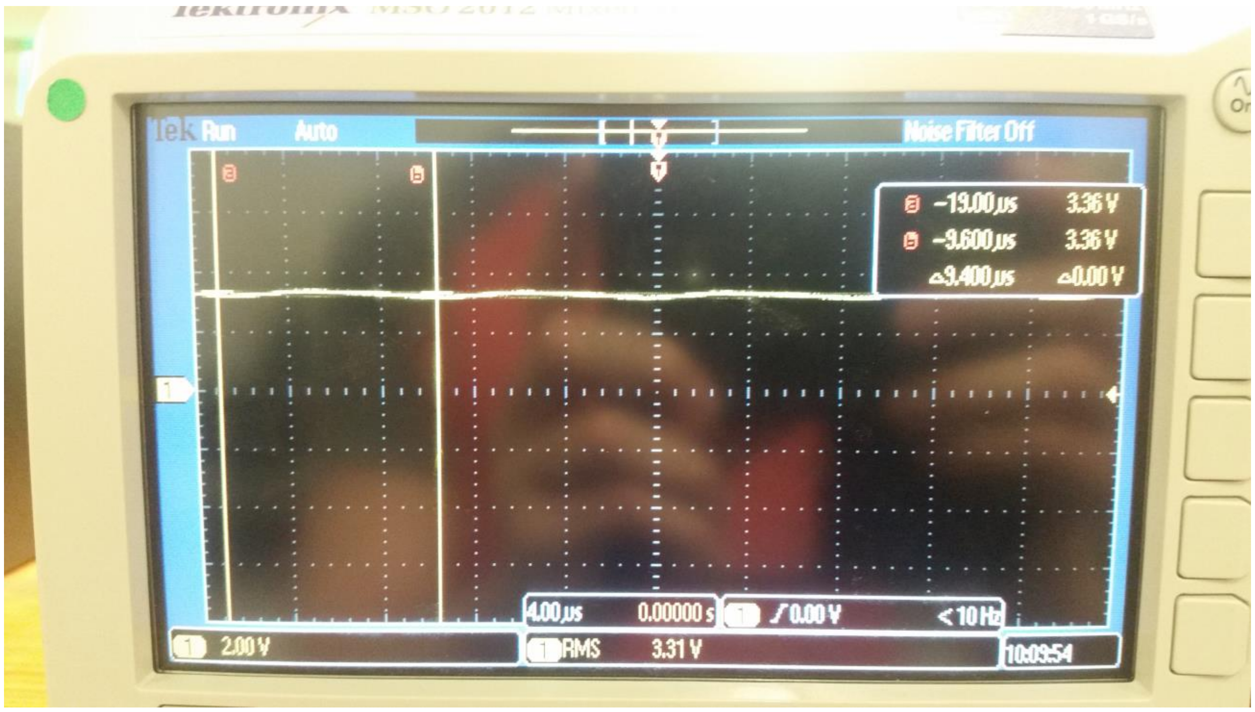

Figure 5: RMS Voltage for 3.3V

Figure 5 shows an RMS value of 3.31V for the 3.3V voltage source. This is closer to the actual value than the RMS of the 5V voltage source.

Voltage vs. Time (During alarm)

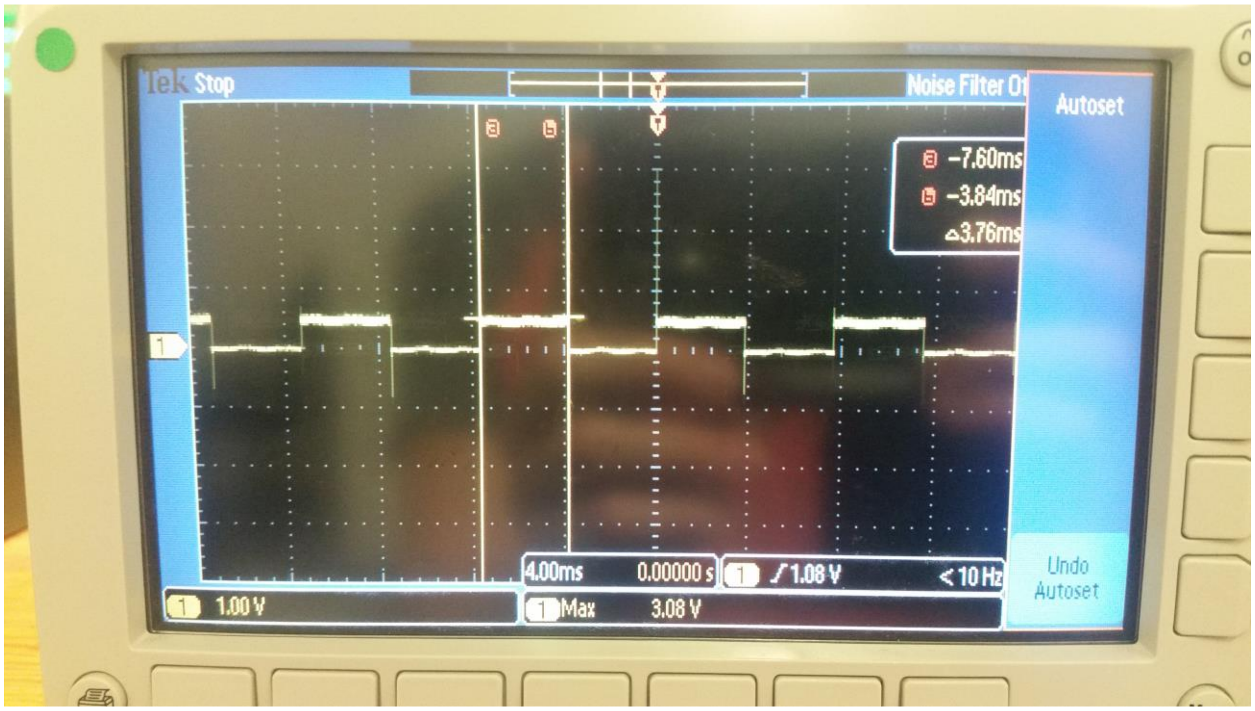

Figure 6: Voltage vs. Time

Figure 6 shows the voltage across the speaker over time during an alarm sound. As we expected, the voltage is a square wave, which is what causes the speaker to generate sound. The period of the wave is (3.76 * 2) = 7.52 ms, which corresponds to a frequency of 132.98 Hz (3rd octave C).

Current to Run Alarm

Another measurement that we had to take was the current required to run the alarm. This was done by using a power supply in the lab which shows how much current is being used. By measuring the current used by the circuit with and without the alarm, we can take the difference to find out how much current the alarm takes

Current Without Alarm	79 mA
Current With Alarm	91 mA

Table 1: Current with and without Alarm

Table 1 shows the amount of current that the circuit used with and without the alarm sounding. The speaker therefore takes (91 – 79) = 12 mA when the alarm is on.

Analysis and Discussion

1) Give two ways to remove a critical section.

1 - For each area of hardware that you access, make sure all accesses are made from the same thread.

2 - Disable interrupts

2) How long does it take to update the LCD with a new time?

0.0459 seconds

3) What would be the disadvantage of updating the LCD in the background ISR?

Updating the LCD in the background ISR will make the interrupt more intrusive as it will take more time.

4) Did you redraw the entire clock for each output? If so, how could you have redesigned the LCD update to run much faster, and create a lot less flicker?

We only changed the piece of the clock that changed. So, for example, when the time updated, we only changed the digital time and analog clock face. We did not change the title "Time: " or the alarm. Similarly, when changing the alarm, we only changed the digital alarm and not "Alarm: ", the digital time, or analog clock face.

We still could have improved this by shortening the watch hands so that they did not overlap the clock face background (numbers and circle). Then, when updating the analog clock face, we could have updated just the clock hands and not the background.

Still, as we only update the screen when the time changes (once per minute in normal mode or once each time the time or alarm are incremented in time or alarm set mode), the flicker was very minimal.

5) Assuming the system were battery powered, list three ways you could have saved power.

1 - We could have operated the speaker at a lower voltage so that it would have consumed less power.

2 - We could have optimized our clock to use fewer lit pixels (keep more pixels black) so that the total display brightness is lower.

3 - We could have turned off the display if the system hadn't been touched for a certain period of time.