Abstract
The goal of the project was to build an alarm clock with creative snooze options to help people get out of bed in the morning without hitting snooze excessively. The alarm clock accomplishes this by escalating its snooze tactics from the standard beeping, to a cognitive puzzle, all the way to the clock physically running away. The clock that was built was able to accomplish each of these stages. The only current limitation is that the clock body was not built sturdily enough to repeatedly fall off a bedside table in order to run away each morning. However the hardware could be redesigned to fulfill this requirement without changing the functioning software and electronics.
## Contents

Abstract .......................................................................................................................... 0
Motivation ......................................................................................................................... 2
Solution ............................................................................................................................ 2
  Block Diagram .............................................................................................................. 3
Parts .................................................................................................................................. 3
  Microcontroller ............................................................................................................ 3
    Hardware ..................................................................................................................... 3
    Software ..................................................................................................................... 3
  Pushbutton Switches .................................................................................................... 4
  Piezo Buzzer .................................................................................................................. 5
  LCD Display .................................................................................................................. 5
  Adafruit Trellis .............................................................................................................. 6
  Logic Level Shifter ....................................................................................................... 6
  Power Regulator ......................................................................................................... 7
  DC Motors .................................................................................................................... 7
  H-Bridge ....................................................................................................................... 8
Results .............................................................................................................................. 8
Appendix A ..................................................................................................................... 9
  Task Division ............................................................................................................... 9
Appendix B ..................................................................................................................... 10
  Parts List ...................................................................................................................... 10
  Schematic ..................................................................................................................... 11
Appendix C ..................................................................................................................... 11
  References .................................................................................................................. 11
Motivation
The problem that many people, young and old, face each morning is the task of getting out of bed. Whether it’s because the user is still tired, his bed is warmer than his room, or his mind just won’t leave that last REM cycle, he has a tendency to hit the snooze button repeatedly in an effort to sleep just a little bit longer. Surveys have found that more than half of American adults hit the snooze button every morning. The problem is that hitting the snooze button is unhealthy and leads to worse performance throughout the day. This is because the sleep you get between snooze cycles is light sleep and is less restful than getting five more minutes of REM sleep. For those who snooze regularly, the goal is to reduce the reliance on the snooze button by snoozing for 10 minutes or less each day. Most users snooze at 5 minute intervals, so after the initial alarm goes off, the clock would allow two snoozes at the maximum.

Solution
In order to truly wake up, the user’s brain needs to be cognizant, and the user has to physically get out of bed. Since the clock allows two snoozes, the first snooze needs to wake up the brain, and the second needs to get the body moving. To accomplish this, the first snooze alarm goes off with the set tone, just as it did for the initial alarm. But this time, a simple game similar to Simon will light up on the keypad. The user must match the lit up pattern in order for the alarm to turn off and for the next snooze countdown to start. When the second snooze alarm goes off the tone starts playing again, however this time the clock itself runs away. Since the user can no longer reach out of bed to hit the snooze button, he must physically get out of bed to turn off the alarm.
Block Diagram

Figure 1 – Top Level Block Diagram. All Timers listed on the left, and parts listed across the bottom. Main program stages are shown at the top along with each stage’s interactions with hardware and software. Clock generated signals are shown in red, I²C is shown in green, and all other signals are shown in black.

Parts

Microcontroller
MSP430F5529 Launchpad

Hardware
The clock uses the MSP-EXP430F5529LP Launchpad Evaluation Kit. It has two onboard programmable buttons and two onboard programmable LEDs. The Launchpad has a built in debugger that can be disconnected as necessary, 40 header pins, and both 3.3V and 5V outputs. It can run up to 25Mhz, and holds 128KB Flash memory as well as 8KB RAM.

Software
The MSP430 was programmed using TI’s Eclipse based Code Composer Studio. Some example programs from Code Composer, such as those for PWM, were referenced, but none were copied directly or used as-is. The main features of the software are the adapted state machine (seen in the figure 2 below), software debouncing for the buttons using Timer A2, a pseudo-random number generator (PRNG), the Simon game, and the Piezo buzzer tones.
The Simon game uses the PRNG to determine a sequence of three unique numbers from one to sixteen and to light up this sequence on three of the 16 buttons of the Trellis. The program then waits for the user to press three buttons on the Trellis, and if these three buttons are the same as the generated sequence then the stage is passed. If not, the game starts over and a new sequence is generated.

To play different tones on the Piezo buzzer, the frequency for each note of two octaves of the chromatic scale were defined. These notes were then combined into various arrays to play specific musical sequences, such as the melody of “In the Hall of the Mountain King”. A time array was also built for each sequence to determine how long each note should be played for. The musical sequence array runs on Timer A1 as a PWM for the buzzer. The time array runs on Timer A0 to signal when the Timer A1 should run the next tone from the musical sequence array.

**Program Flow**

![Flow chart of main program. Main program states are Set Time, Set Alarm, Set Snooze Interval, Set Tune, Update LCD, Play Game, and Runaway Mode](image)

**Pushbutton Switches**

Two mini pushbutton switches are used to control the program flow and to set the time, alarm time, snooze time, and alarm tone. These switches are single pole single throw switches (part number not available, purchased from the GMU ECE Lab from their supply of leftover parts from previous kits). They are debounced in software, and are interfaced using the 3.3V power source and the internal pull-up resistors on pins 1.2 and 2.2.
Piezo Buzzer
PS1240

Figure 3 - Buzzer breadboarded to ground and the microcontroller pin

This Piezo buzzer is built for 3-30V and can be quite loud if driven at full voltage, but the buzzer will break quicker if it’s always run at high voltages. In this case, the buzzer runs at about 3V depending on the PWM period and frequency as needed to produce each different tone. To create a tone, the buzzer is connected to a PWM signal generated by the microcontroller on Timer A1 at pin 2.0. The PWM runs different frequencies depending on the notes to be played for each tune. The buzzer can play frequencies from 2KHz to 10KHz quite well, but most frequencies are centered around 4KHz as that is where the buzzer is loudest.

LCD Display
TC1602A-01T LCD

The LCD screen used in this project is a 16-character-by-2-row display, featuring white text and a blue backlight. While this particular screen was sold by Adafruit, it is based upon a Hitachi HD44780 display driver chip, and pin-compatible replacements may be purchased from a variety of vendors. The microcontroller communicates with this display using a total of six GPIO pins: a register-select line that dictates whether instructions or data are being written to the display; an enable line that starts the data read/write operation, and a 4-bit data bus, over which two successive nibbles are sent to transmit one 8-bit character.

Figure 4 - LCD breadboarded and running the "Set Time" functionality. The LCD was connected through a logic level shifter, as shown on the left

Because this screen is so ubiquitous among hobby electronics vendors, it was easy to find code examples online. A basic library written for an MSP430G2553 microcontroller was found, and was modified for the MSP430F5529 by simply changing pin assignments and device-specific header file dependencies. This library allows a string of characters to be written to the display using only a single line of code.
**Adafruit Trellis**

HT16K33 Controller

An Adafruit Trellis is used as the user interface for the ‘play game’ function of the alarm clock. The Trellis is a 4x4 array of elastomer buttons, each illuminated by its own LED. The array of buttons and LEDs is read and written by a Holtek HT16K33 controller. This controller communicates with the microcontroller over an I2C bus, while also driving an additional interrupt line whenever a button is pressed.

![Adafruit Trellis](image)

**Figure 5 - Adafruit Trellis. [Left] All LED’s soldered onto the board and a logic level shifter in place to drive the Trellis, [Right] The back of the Trellis board along with the rubber buttons**

Adafruit supports the Trellis by providing an Arduino library written in C++ that handles all of the I2C communication and memory-mapping necessary to read and write the keys. In order to make this library compatible for the microcontroller, it had to be edited to use only C-language commands. It was also necessary to replace the Arduino I2C functions with corresponding commands to the microcontroller’s USCI module, which was used for the I2C communications.

**Logic Level Shifter**

EzSBC LS1 Bi-directional Level Shifter

![Logic Level Shifter](image)

**Figure 6 - Breadboarded Logic Level shifter**

As both the LCD screen and the Trellis were specified as 5V units, it was necessary to obtain a level-shifter circuit to translate from the microcontroller’s 3.3V logic to the screen and Trellis’ 5V logic. In addition, this level shifter had to be bidirectional in order to support the I2C communication between the MSP430 and the Trellis. The level shifter that was ultimately used was a small generic level-shifter circuit board purchased from eBay. For each of the eight channels on this level-shifter board, two 10K resistors and a MOSFET were used to realize the circuit described by Philips Semiconductors’ “AN97055 Bi-directional level shifter for I²C-bus and other systems” applications note, which can be found in Appendix D of this report.
Power Regulator
Turnigy 3A UBEC

The voltage regulator used to power all the 5-Volt components in the alarm clock is a Turnigy brand 3-Amp universal battery elimination circuit, originally intended for radio-controlled hobby applications. This regulator is a DC-DC switching converter which accepts an input ranging from 6 to 23 Volts, and which supplies a constant 5-Volt output with up to 3 Amps of current. As an added feature, the voltage regulator also has an integrated piezo buzzer that alerts the user when the input supply has dropped below 6 Volts. A 9-Volt battery is used as the input supply for this regulator.

DC Motors
HN-35GMB DC Geared Motor

The Motors on the clock are DC geared motors rated for up to 12V at 310rpm. For convenience, the motors were run on a 9V battery. This also limited the speed of the motors to a reasonable pace for the user to be able to catch it in the morning. Each motor has two terminals, and connecting voltage through either terminal runs the motor either clockwise or counterclockwise. In this case, the terminals were connected to the H-bridge and were only enabled in one direction, as the clock does not have a reason to roll backwards in its current design. The motors were connected to enable signals, PWM signals, and voltage through the H-bridge, as described below.
One key factor was the lag between the two motors. Each motor runs at a slightly different speed because of uneven wear and physical construction. To compensate for this lag, the naturally faster motor must be driven at a slightly lower speed. When its speed is reduced by the correct factor, the motors will turn at the same speed and the clock will roll forward without unintentionally turning to either side. Through testing it was determined that the faster motor must be run at 80% of the speed of the slower motor. This factor was accurate when running the motors from 40% to 90% of their 12V range, and when running from 50% to 100% of the supplied 9V range.

**H-Bridge**

DF Robot DRI0002 Dual Motor Controller

The Dual Motor Controller sounds like it takes care of more steps in controlling the motor, but it’s actually just a well-mounted H-bridge. The motor controller breaks out the motor pins, power, and ground into wire clips. And it breaks out the enable and PWM pins onto header pins to be easily connected to a microcontroller. Since the motor’s power source could be up to 46V, there is a power regulator with a cooling board attached to the back.

![H-bridge closeup](image)

*Figure 9 - [Left] H-bridge closeup, [Right] pinout of connections to the H-bridge (copied from the datasheet)*

To control the motors, the microcontroller must send both enable signals and PWM signals to the H-bridge. The PWM signals were generated on Timer B0 on pins 3.5 and 3.6, with separate voltage levels for each motor to make up for the lag that is present in the motors as explained above. The motor enable pins were controlled by general I/O pins 4.1 and 4.2. When the clock hits the run-away stage these pins were set high, and when the snooze button is pushed to turn off the alarm these pins were reset low. The h-bridge power supply was connected to the 3.3V output of the microcontroller, and the motor power supply was connected directly to a 9V battery.

**Results**

The clock runs as specified in the project goals. The clock is accurate and the time can be set. The alarm goes off at the set time, and it comes back on after the set snooze period. The tone plays each time the alarm or snooze goes off, and it can be set to play different melodies depending on the selection. After the first snooze period, the Trellis lights up a random sequence of three buttons which the user must match in order to snooze the alarm. And after the second snooze period the clock runs away until the user turns off the alarm.
All functionality, as described above, fulfills the project goals. One problem that was noticed, but was determined to be outside this project’s scope, was that the clock is not shock-proof. This would be necessary if the clock was to roll of the user’s bedside table each morning when it runs away.

One lesson learned along the way was that many parts, especially those sold by Adafruit, require 5V logic, but others require 3.3V logic. The microcontroller cannot supply both logic levels at once, so logic level shifters were required for some parts. Another problem that was encountered was that the piezo buzzer is not very loud. Realistically, this buzzer might not wake up the user in the morning. For this project, there wasn’t time to improve the buzzer. But options to solve this problem include replacing the part with a louder buzzer, increasing the supplied voltage to the buzzer, or run it in differential drive with both pins connected to the microcontroller at opposite polarities so the generated square wave is double in amplitude.

Appendix A

Task Division

Lakshmi Meyyappan
Project lead – in charge of scheduling, project division, and all deliverables and videos. Hardware lead – choosing and connecting DC Motors and H-bridge, building enclosure and attaching wheels and peripherals. Software responsibilities included PWM control.

William Diehl
Software lead – building and programming the main program pseudo-state machine. Programmed interrupts, software debouncing, and tone sequences for the buzzer. Led integration efforts, working with each team member as necessary to integrate each piece of code into the main program.

James Kaye
Electronics lead – connecting electronics for the LCD, Trellis, and Logic Level Shifters, and power regulator. Programmed the I²C connection for the Trellis, and worked with Will on programming the LCD. Vector boarded all components including power subsystem, and drew up final schematic.
**Cong Chen**
Secondary programmer – programming PWM for the Piezo buzzer, designing and programming all aspects of the Simon game.

**Appendix B**

**Parts List**
MSP-EXP430F5529LP Launchpad Evaluation Kit

2 Mini Pushbutton Switches (part number not available)

TC1602A-01T LCD

HT16K33 Controller

EzSBC LS1 Bi-directional Level Shifter

Turnigy 3A UBEC

HN-35GMB DC Geared Motor

DF Robot DRI0002 Dual Motor Controller
Appendix C

References
LCD Screen Library: [http://www.ece.utep.edu/courses/web3376/Lab_5_-_LCD.html](http://www.ece.utep.edu/courses/web3376/Lab_5_-_LCD.html)

AN97055 Bi-directional level shifter for I²C-bus and other systems, Application Notes: [http://www.adafruit.com/datasheets/an97055.pdf](http://www.adafruit.com/datasheets/an97055.pdf)

Adafruit Trellis Arduino Libraries: [https://github.com/adafruit/Adafruit_Trellis_Library](https://github.com/adafruit/Adafruit_Trellis_Library)
