Lab 10

Implementing an Electronic Combination Lock

Due: Week of May 2, Before the start of your lab section[*]

This is a team-effort project. You may discuss concepts and syntax with other students, but you may discuss solutions only with your assigned partner(s), the professor, and the TAs. Sharing code with or copying code from a student who is not on your team, or from the internet, is prohibited.

In this assignment, you will collaboratively write code for your Arduino Nano-based class hardware kit to implement a simple embedded system. Specifically, you will immplement an electronic combination lock.

The instructions are written assuming you will edit the code in the Arduino IDE and run it on your Arduino Nano-based class hardware kit, constructed according to the pre-lab instructions. If you wish, you may edit the code in a different environment; however, our ability to provide support for problems with other IDEs is limited.

Please familiarize yourself with the entire assignment before beginning. Section 2 has the functional specification of the system system you will develop. Section 3 describes the few implementation constraints. Section 4 describes your options for storing data on and retrieving data from the Arduino Nano's EEPROM.

Learning Objectives

After successful completion of this assignment, students will be able to:

- Work collaboratively on a hardware/software project
- Design and implement a simple embedded system
- Expand their programming knowledge by consulting documentation

Continuing Forward

This penultimate lab assignment does not contribute to the final lab assignment. By integrating elements of what you learned in this course, and by demonstrating that you can review documentation to learn on your own, to design a small embedded system, you will show how much progress you have made this semester.

[*]See Piazza for the due dates of teams with students from different lab sections.

During Lab Time

During your lab period, coordinate with your group partner(s) to decide on your working arrangements. Unless you're only going to work on the assignment when you're together, you may want to set up a private Git repository that is shared with your partner(s). With your partner(s), think through your system's design and begin implementing it. The TAs will be available for questions.

No Spaghetti Code Allowed

In the interest of keeping your code readable, you may *not* use any **`goto`** statements, nor may you use any **`continue`** statements, nor may you use any **`break`** statements to exit from a loop, nor may you have any functions **`return`** from within a loop.

1 Scenario

"I have various teams working on different projects around here to improve security," Archie reminds you. He glances toward the Zoo's labs, where there's now a guy who looks like the actor who portrayed the fictional actor who portrayed the Norse god Odin, trying to avoid children while wistfully talking about raising rabbits in Montana. You briefly wonder why there are children someplace where there are also carnivorous megafauna, and then you remember that you work at a petting zoo. "What I need your team to do," Archie continues, "is make a combination lock so that only authorized people can get into our lab facilities."

2 Electronic Combination Lock Specification

1. A combination shall consist of three numbers in a particular order.
   (a) Each of the three numbers shall consist of exactly two hexadecimal digits.
   (b) It shall be possible for any two of the numbers to be duplicates of each other. It shall be possible for all three numbers to be duplicates of each other. For example, 98-98-98 is a valid combination.
   (c) Single-digit numbers must have leading zeroes. For example, 01-02-03 is a valid combination, but 1-2-3 is not.
2. When displayed on the **display module**, the combination's first number shall occupy the two leftmost digits; the second number shall occupy the middle two digits; and the third number shall occupy the two rightmost digits. Dashes shall be displayed between the numbers, for example: 12-34-56. If one or more of the positions has not had a number entered, then the digits for the unentered positions(s) shall be blank. For

example, if the first number has been entered but not the second and third, then the display will show 12- - .

3. The combination's numbers shall be entered using the **matrix keypad**. Buttons with decimal numerals (*0-9*) or alphabetic letters (*A-D*) shall be interpreted as having the corresponding hexadecimal numeral; the button with the octothorp (#) shall be interpreted as having the hexadecimal numeral *E*; and the button with the asterisk (*) shall be interpreted as having the hexadecimal numeral *F*.

4. When the system is powered-up, it shall be LOCKED, regardless of its state before losing power, and the display shall have all numbers blank: - - .

5. When the system is LOCKED, the system shall display the combination-entry display.

6. When the system displays the combination-entry display, it shall display the currently-entered cobmiantion (which might not be the correct combination), and the user shall be able to enter and change the currently-entered combination.

   (a) The system shall indicate to the user the position in the combination that they are entering a number for, by blinking the decimal points for that position's two digits on-and-off every half-second. For example, when all numbers are unentered and the user is being prompted to enter the first number, the display shall show:
   . .- - for a half-second, then
   - - for a half-second, then
   . .- - for a half-second, then
   - - for a half-second, and so on.

   (b) Hereafter, the blinking decimal points will be referred to as the *cursor*.

   (c) The cursor shall continue to blink while and after the user has entered numerals. For exmaple, after the user has entered "1" as the first digit of the first number, the display shall repeatedly show:
   1. .- - for a half-second, and
   1 - - for a half-second, then
   then after the user has entered "2" as the second digit of the first number, the display shall repeatedly show:
   1.2.- - for a half-second, and
   12- - for a half-second.

   (d) The user shall advance the cursor to the next number to be entered by pressing the **right pushbutton**.

      i. The cursor shall not advance until the user presses the **right pushbutton**.

      ii. If the cursor is positioned on the first number, then advancing the cursor places the cursor in the second nummber's position.

      iii. If the cursor is in the second number's position, then advancing repositions the cursor in the third number's position

iv. If the cursor is in the third number's position, then advancing the cursor places the cursor in the first number's position.

(e) The user shall be able to change the numbers they have entered by advancing the cursor from position to position.

(f) The user shall assert that they have entered the combination by pressing the **left pushbutton**.

(g) If the user has not entered numbers for all three positions before pressing the **left pushbutton**, then the system shall display Error for one second and then resume displaying the combination-entry display.

7. The **switches** shall be ignored while the system is LOCKED.

8. If the user had entered the correct combination when when they pressed the **left pushbutton**, then the system shall be UNLOCKED and shall display LAb oPEn.

9. If the user had entered an incorrect combination when they pressed the **left pushbutton**, then the system shall display bAd trY 1
bAd trY 2 or
bAd trY 3 for one second, depending on whether it was their first, second, or third attempt.

10. After the user's first and second "bad try" messages, the system shall resume displaying the combination-entry display.

11. After the user's third "bad try" message, the system shall be ALARMED.

    (a) When the system is ALARMED, it shall display ALErt ! and the **external LED** shall blink on-and-off every quarter-second.

    (b) When the system is ALARMED, it shall not accept further input until it has been powered-down and then powered-up.

12. When the system is UNLOCKED, the user shall be able to change the combination.

    (a) When the user places both **switches** in the right positions and then pressing the **left pushbutton**, the system shall be CHANGING.

    (b) When the system is CHANGING, it shall display EntEr for one second and and then shall display the combination-entry display, and the user shall be able to enter the proposed new combination, with the following change:

        i. Pressing the **left pushbutton** while the **left switch** is in the right position shall have no effect.

        ii. If the user places the **left switch** in the left position and then presses the **left pushbutton**, then the system shall be CONFIRMING.

(c) When the system is CONFIRMING, it shall display rE-EntEr for one second and hen shall display the combination-entry display, and the user shall be able to enter the proposed new combination, with the following change:

   i. Pressing the **left pushbutton** while the **right switch** is in the right position shall have no effect.

   ii. If the user places the **right switch** in the left position and then presses the **left pushbutton**, then the system shall display cHAnGEd if the confirmed combination matches the proposed combination, or nocHAnGE fif the confirmed combination differs from the proposed combination. The system shall then be UNLOCKED.

13. When the system is UNLOCKED, the user shall be able to lock the system by one of the two methods listed below. The system shall be LOCKED. It shall display cLoSEd for one seceond and then shall display the combination-entry display.

    (a) One option to lock the system is placing both **switches** in the left positions and then pressing both **pushbuttons** simultaneously.

    (b) The other option to lock the system is placing both **switches** in the left positions and then double-clicking the **right pushbutton**.

    - You only need to implement *one* of these locking mechanisms. With the Cow Pi's hardware, double-clicking is easier to detect with interrupts, and chording is easier to detect with polling.

14. Except as specified in Requirements 12 and 13, pressing the **pushbuttons** shall have no effect when the system is UNLOCKED.

15. The correct combination shall be persistent even while the system is unpowered. For example, if the user changes the combination, powers-down the system, and powers-up the system, then the correct combination shall still be the combination that they had changed it to before powering-down the system.

3 Constraints

You may re-use code from the previous lab assignments and from *cowpi.h*, or you can rewrite the necessary code using functions, macros, types, or constants that are part of the Arduino core;[1] functions, macros, types, or constants of avr-libc;[2] or one of Arduino's standard libraries.[3] You may *not* use a third-party library, not even one that the Arduino Libraries

[1] https://www.arduino.cc/reference/en/

[2] https://www.nongnu.org/avr-libc/user-manual/index.html

[3] https://www.arduino.cc/reference/en/libraries/ The standard libraries are those under the heading, "Official Arduino Libraries."

reference page links to (if it isn't among those available in the Arudino IDE's "Sketch" → "Include Library" menu under "Arduino libraries" then you may not use it).

You can use memory-mapped I/O registers, or you can use functions provided by the Arduino core read from and write to pins. You can use polling, or you can use interrupts. You can have as much or as little code in the **`loop()`** function as you deem fit.

Do *not* edit *cowpi.h*; all of your code must go in *ComboLock.ino*.

4 Accessing the EEPROM

To implement Requirement 15, you will want to save your lock's combination in your Arduino Nano's EEPROM. The Arduino Nano has 1KB of EEPROM, and you will use only a tiny fraction of this (depending on your implementation, you will probably use either 3 or 6 bytes of the EEPROM). While the EEPROM is not part of the ATmega328P's memory address space, you can access it, one byte at a time, using memory-mapped I/O. If you choose to use memory-mapped I/O to access the EEPROM, please read the ATmega328P datasheet,[4] §7.4 and §7.6. Alternatively, you can use the Arduino EEPROM library. If you choose to use the EEPROM library, please read the EEPROM library's documentation[5] and guide.[6] We recommend that you use the **`EEPROM.get()`** and **`EEPROM.put()`** library functions, as they work with arbitrary data types and write to the EEPROM in the most-efficient manner possible.

Clearing Part of EEPROM Memory

In the event that you forget your lock's combination (or that you need to set the combination before your first use), we have provided *ClearEEPROMPage.ino* that will set four bytes to 0. Using the Serial Monitor, you will be prompted to enter a number divisible by 4 (not to exceed 1023). The 4-byte EEPROM page starting at that address will then be set to 0. If you need to clear more than four bytes, press the RESET button on top of your Arduino Nano to run the program again. **NOTE:** after you type in the number, you will send that number to the Arduino Nano one of three ways:

Arduino 1.8 Use your mouse to click on the "Send" button.

Arduino 2.0 on Windows or Linux Press Control-Enter on your keyboard.

Arudino 2.0 on Macintosh Press Command-Enter on your keyboard.

[4]http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[5]https://docs.arduino.cc/learn/built-in-libraries/eeprom

[6]https://docs.arduino.cc/learn/programming/eeprom-guide

Non-Performing and Toxic Team Members

Non-performing team members If a student is habitually failing to be a contributing member of your team beyond your ability to resolve, you may ask a TA or the professor for help resolving the problem. In extreme circumstances, the professor may remove that student from the team. This can occur only after we've discussed your team's dilemma and concurred that no better option is available.

Toxic team members If a student's behavior is harming the other team members' ability to learn and perform, the professor may remove that student from the team. This can occur only after we've discussed your team's dilemma and concurred that no better option is available. If necessary, the case will be referred to Student Affairs for appropriate action.

We will find a way for an unfired team member to demonstrate mastery of the lab material and earn credit for the assignment without having to commplete the full assignment on their own.

A fired team member may complete the project on his/her own, or join a group with other fired team members (if any exist).

Turn-in and Grading

When you have completed this assignment, upload *ComboLock.ino* to Canvas.

This assignment is worth 80 points.

Rubric:

_____ **+2** Combinations are displayed as three 2-hex-digit numbers separated by dashes.

_____ **+4** Numbers are entered using the matrix keypad.

_____ **+2** The lock is locked when powered-up.

_____ **+4** When the lock is locked, it displays the combination-entry display, initially showing empty numbers (only dashes).

_____ **+4** The cursor indicating which number is being entered is represented as blinking decimal points in the relevant number position.

_____ **+4** The user can move the cursor between number positions by pressing the right button.

_____ **+4** After entering a combination, the user can submit the combination by pressing the left button.

_____ **+2** If the user tries to submit an incomplete combination, the lock displays "error" and then returns to the combination-entry display.

_____ **+4** When the user unlocks the lock, it displays "lab open".

_____ **+2** When the user mis-enters the combination, it displays "bad try" and the attempt number.

_____ **+4** After the first two bad tries, the user is given another opportunity to enter the combination.

_____ **+4** After the third bad try, the system displays "alert!", the external LED rapidly blinks, and the lock becomes unresponsive.

_____ **+4** The user can begin changing the combination by moving both switches to the right and pressing the left button.

_____ **+4** When the user begins changing the combination, the lock displays "enter" and then shows the combination-entry display.

_____ **+4** After entering a new combination, the user can begin confirming the combination by slidig the left switch to the left and pressing the left button.

_____ **+4** When the user begins confirming the combination, the lock displays "re-enter" and then shows the combination-entry display.

_____ **+4** After entering the confirming combination, the user can compare the new combination with the confirmed combination by sliding the right switch to the left and pressing the left button.

_____ **+4** If the user successfully confirmed the new combination, the lock displays "changed", and the new combination is the correct combination for future unlocking attempts.

_____ **+4** If the user failed to confirm the new combination, the lock displays "unchanged", and the previously-correct combination remains the combination for future unlocking attempts.

_____ **+4** The user can lock the lock by moving both switches to the left and *either* double-clicking the right button, *or* pressing both buttons at the same time. (Only one of these methods needs to be implemented).

_____ **+2** The inputs that are explicitly specified as having no effect in certain situations are ignored in those situations.

_____ **+4** The correct combination persists while Arduino is powered-down.

_____ **+2** The source code is well-organized and is readable.

_____ **Bonus +2** Get assignment checked-off by TA or professor during office hours before it is due. (You cannot get both bonuses.)

_____ **Bonus +1** Get assignment checked-off by TA at *start* of your scheduled lab immediately after it is due. (Your code must be uploaded to Canvas *before* it is due. You cannot get both bonuses.)

_____ **-1** for each **`goto`** statement, **`continue`** statement, **`break`** statement used to exit from a loop, or **`return`** statement that occurs within a loop.

Students' scores may be adjusted up or down as necessary if the team had an inequitable distribution of effort.

Epilogue

After fastening the new electronic combination lock to the lab door, Archie smiles and tells you that this was a job well done. With all of the excitement neatly wrapped-up and arriving at a satisfactory conclusion, you look forward to a boring career in which there's absolutely no screaming and running for your life.

The End...?

Appendix: Lab Checkoff

You are not required to have your assignment checked-off by a TA or the professor. If you do not do so, then we will perform a functional check ourselves. In the interest of making grading go faster, we are offering a small bonus to get your assignment checked-off at the start of your scheduled lab time immediately after it is due. Because checking off all students during lab would take up most of the lab time, we are offering a slightly larger bonus if you complete your assignment early and get it checked-off by a TA or the professor during office hours.

The check-off checklist will be added soon.