For our midterm project, we decided to create a small-scale casino on our microcontrollers. This casino would consist of multiple games, a player board to play the games, and a dealer board to display your opponent's move. Originally, we thought this would be simple through serial communication, but it ended up being much more complicated. This project gave us a deep dive into scheduling, communication, and data handling, all while using interrupts and external sensors. Our casino consists of a home screen to choose your game, a roulette game, and a blackjack game. Each game, and its functionality is controlled using external buttons, and the data is transferred through an EEPROM. Our home screen presents a welcome message along with your options for which games to play. Upon selection of a game, we enter the game's functionality. For roulette, the user picks a bet, (Red, Black, Even, Odd, or a number), and then submits it. The dealer spins a wheel to generate the number, and lets you know if your bet has won or lost. Again, this is all done through EEPROM and external buttons. Our blackjack game allows you to play a hand, see the dealers first card, and then decide whether or not you want to hit or stand. Once you either bust or stand, the dealer will play out the rest of the hand and let you know if you have won or lost. The base functionality of these games from a software point of view isn't overly complicated. If we wanted to code these games in python and just play in the terminal, this project would have been very simple. But the level of complexity skyrockets when each board needs to be waiting for another board to complete an action, and receive info once that action is done. Using interrupts for all the user interactions also increases the complexity.

Roulette needs to signal to the player board that the roulette button has been pushed. The dealer board is now waiting for a bet to be placed while the user toggles through bet options on the player board. Once a bet is selected, this value must be written to the EEPROM where the dealer reads it. Once the bet has been read, the dealer spins the wheel and displays the result and writes it to the EEPROM. On the player board, it is waiting for the result. Once it is read, it displays the result and then resets to play another round. This timing and data transfer requires very specific interrupt handling and scheduling. Blackjack needs to signal to the player board that the blackjack button has been pushed. The player board is now waiting for the start hand button to be pushed, and the dealer board is displaying the one card total and waiting for the player to finish their decisions. Once the player either busts or stands, it sends that message to the dealer board, and waits for the dealer to finish the hand. When the dealer board receives the player message, It finishes the hand and sends it's total to the player board where the totals are compared and the result is displayed. At first we thought we would be able to split this project into multiple parts so that we could each individually complete something. Once we started we realized this wouldn't be possible. The serial communication needs the game functionality to be able to be tested. The game functionality needs the serial communication to be able to be tested. And most importantly, each aspect of the project uses both boards with meant only one aspect could be tested at once. It is for these reasons that we all contributed to all aspects of the project.

Overall, this project gave us a great look into some content outside of the scope of the course. The board-to-board communication was both the most complex and difficult, but also most interesting part of this project. Integrating this new content with course covered content like interrupts and ISRs was truly a pleasure.