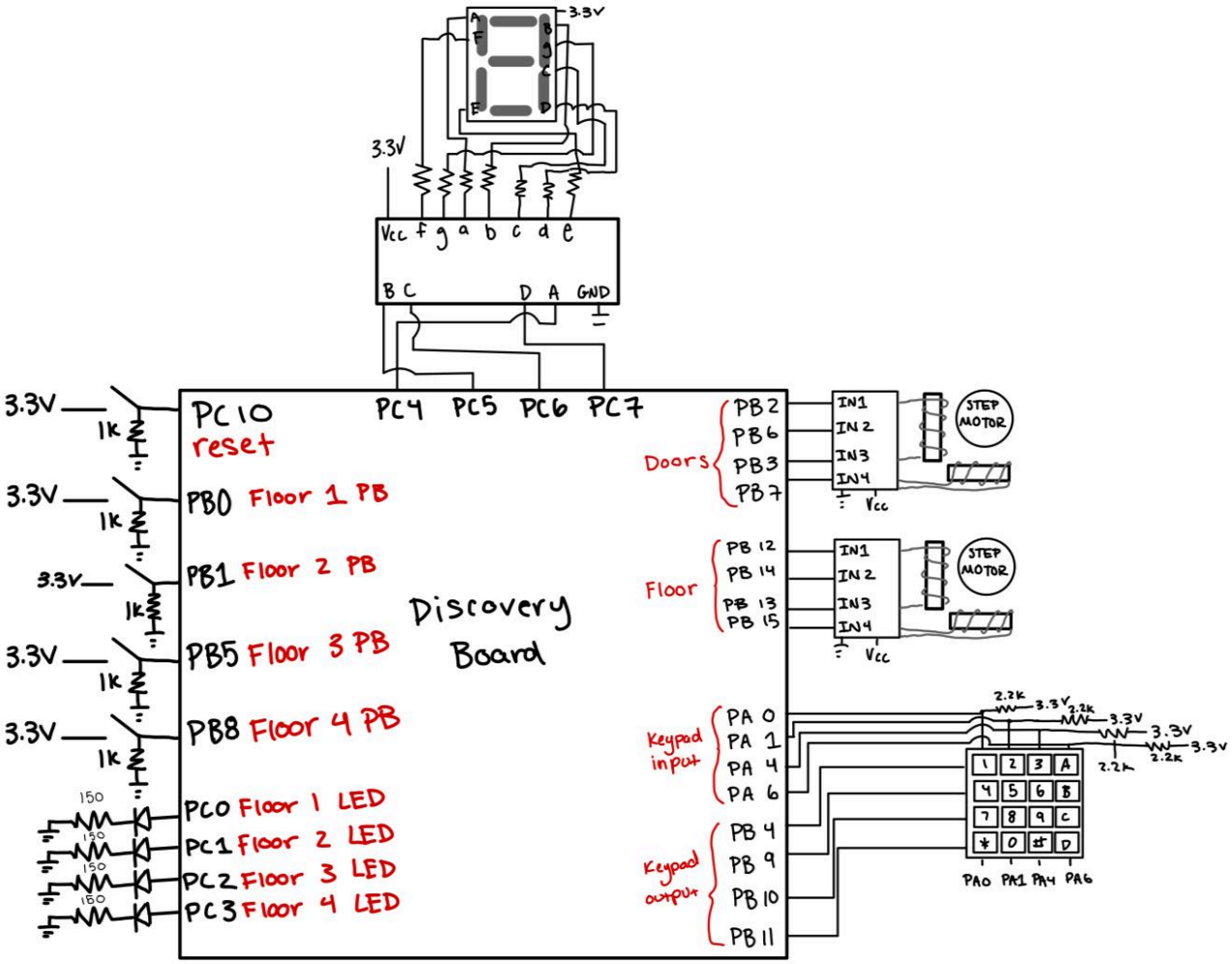
Term Project

Leslie McDonough, Cooper Sainiak, Jill Zitcovich, Allison Mitchell

Schematic:



Intro:

This report details the design and implementation of a microcontroller-based elevator controller for a four - floor system. The primary objective was to create a functional prototype that prioritizes simplicity while demonstrating core functionalities of an elevator system. By interfacing the STM32L476xx microcontroller with essential components such as a keypad, LED indicators of elevator calls, seven - segment display to display the current floor, push buttons and two motors, one to indicate the movement of the elevator and one for the movement of the door. All together, the project aims to showcase a viable solution for elevator control.

Operation of Implementation:

The STM32L476xx microcontroller is used as the core of our system. Through the use of LED's, they provide visual feedback indicating the floor that is selected outside the elevator by the user, by lighting up upon call. The elevator will then know to go to that floor picked by the user. Once inside the elevator, we implemented the required specifications by interfacing it with a four-by-four keypad and having user controlled input through push buttons, including a reset button to pull the elevator down to floor one and rest the series of calls. This is mapped to PC10.

We implemented GPIOC pins 0-3 as output for the LEDs, and GPIOB pins PB0, PB1, PB5, and PB8 as inputs with 150 Ohms pull-down resistors, in order of the respective floors. GPIOB pins PB2, PB3, PB6 and PB7 are implemented as outputs for the stepper motor for the doors. GPIOB pins PB12, PB13, PB14, and PB15 are implemented as outputs for the stepper motor for the floor movements. GPIOB pins PB4, PB9, PB10, and PB11 are implemented as outputs for the keypad (the rows), and GPIOA pins PA0, PA1, PA4, and PA6 are implemented as inputs for the keypad (the columns). Finally, GPIOC pins 4, 5, 6, and 7 are connected to the HEX board, as shown by the above schematic.

Within our main loop of the design we have implemented a program that utilizes software debouncing and checks for user input. This debouncing method creates a delay that ensures button presses are valid and avoid any noise - induced issues, which enhances the overall user experience. The program continuously monitors GPIO port data registers to detect button presses accurately and execute corresponding actions based on user input. The main loop then steps through the different initializations and sets of the project in order to achieve an operating elevator.

When checking for an external button press, we step into different segments checking if each button has been pressed. Testing each bit corresponding to the number button, if it is pressed the program then branches to this condition, illuminating the LED for the corresponding floor, indicating the arrival of the elevator. This then branches to a subroutine to indicate the called floor to be written to Tera Term, to provide a written description of the floor called to the user. For each implementation of the floor, the direction the elevator would be moving was taken into account. As calling the elevator to floor one, indicates it is only required to move down from upper floors. Though for the other floors, we had to take into consideration moving both up and down, this was distinguished by the rotation of the stepper motor.

For the two stepper motors representing the operation of doors and floor movements, the rotation of the motor requires 360^{o} per floor. To open or close the door, the motor rotates 180^{o}. This requires a value of 257 steps to rotate the motor the correct number of degrees for the door, or 514 steps to increase or decrease one floor level.

Finally, a physical tower of LEGOs, with 4 layers, was built with a small box to represent the elevator tower and car. The car is connected via string to the motor such that it moves upward

when the motor rotates clockwise and lowers downward when the motor rotates counterclockwise to show the elevator's progress through the floor.

Design Constraints:

Despite the successful implementation, the project faces certain design constraints that impact its scalability and robustness. Limited hardware knowledge, inherent to a single microcontroller setup, pose challenges in accommodating complex functionalities in a more efficient way such as attempting to use two microcontrollers. Developing and understanding a program that would utilize a second microcontroller would have divided the tasks between the microcontrollers and simplified the hardware design, though this would have needed double inputs in order for them to both know the current and called floor.

After implementing the majority of the program, we realized we had not written the program in a way that allotted for dynamic floor movement, so users are not able to call a floor while the elevator is in motion. Initially we believed this could be achieved by writing interrupts though realized by implementing SysTick Handler that would go off in order to scan the buttons in the keypad while the motors were rotating, though upon trying to implement this we realized we would need to restructure our code in order to account for this. After several attempts of implementing it within our current program, we were not able to achieve dynamic floor movement within the allotted time with our current understanding and level of assembly programming.

Neatness during board construction emerged as another constraint during the timeline of the project, requiring careful organization to optimize space utilization, while keeping in mind the relation between our set up and the board's functionality throughout the design process.

Due to the size of the available breadboards being a constraint to construction, requiring more thoughtful planning to fit the required hardware needed for the design, we decided to utilize two breadboards allowing for more space, and flow within the design.

Successes/Failures of Testing:

The testing phase involves iterative experimentation and refinement to ensure the system's reliability and functionality. While successful in detection button presses and illuminating the corresponding LED's, the implementation process revealed several challenges and areas for improvement.

Exploration of different implementation methods, including interrupt-based approaches, highlights the team's adaptability and problem-solving skills. Despite initial setbacks, the decision to employ a loop-based approach ensures a solid foundation for elevator functionality while addressing programming complexities.

Another aspect of the program that had to be worked around was the length of the program. With the program being longer we had to introduce the directive, LTORG. LTORG instructs the assembler to assemble to the current literal pool immediately. Default literal pools (LDR, etc.) can sometimes go out of range, causing the program to fail. By using LTORG after subroutine return instructions, the processor is able to execute constants as instructions. We also add in .w as it is 32 bit wide in coding. This was placed after subroutines prohibiting the program from running constants as instructions.