ECEN 260 Lab Report

Lab #6: Digital Input & Output in 'C'

Name: Brodric Young

Lab Partner: Karl Richards & Brenden Clark

Date: 2/15/2025

Instructor: Brother Allred

Semester: Winter 2025

Lab Overview

This lab was about implementing a home security system using digital input and output in C on our Nucleo boards. By setting up I/O pins, MODER, PUPDR, and referencing pins in code and using bitwise operations, and wiring components, we created a system with three magnetic reed switch sensors simulating a front door, back door, and window. These sensors trigger LEDs to indicate alarm status. The objectives were to learn C programming for I/O operations, develop system specifications collaboratively, and practice pair programming. This lab applies what we've learned to a practical, real-world scenario, reinforcing concepts from class.

Test Plan and Results

#	Test Scenario	Expected Result	Observed Result
1	1. Press the "Arm" button. 2. Verify the "armed" LED lights up.	The "armed" LED should turn on, indicating the system is armed.	The "armed" LED turned on
2	1. Press the "Disarm" button. 2. Verify the "armed" LED turns off, and all alarm LEDs are off.	The "armed" LED and any active alarm LEDs should turn off.	The "armed" LED and any active alarm LEDs turned off.
3	1. Arm the system. 2. Simulate the front door opening by separating the reed switch. 3. Check the front door alarm LED.	The front door alarm LED should turn on and stay on until the system is disarmed.	The front door alarm LED turned on and stayed on until the system was disarmed.
4	1. Arm the system. 2. Simulate the back door opening by separating the reed switch. 3. Check the back door alarm LED.	The back door alarm LED should turn on and stay on until the system is disarmed.	The back door alarm LED turned on and stayed on until the system was disarmed.
5	1. Arm the system. 2. Simulate the window opening by separating the reed switch. 3. Check the window alarm LED.	The window alarm LED should turn on and stay on until the system is disarmed.	The window alarm LED turned on and stayed on until the system was disarmed.
6	1. Trigger one or more sensors to activate the alarm. 2. Press the "Disarm" button.	The "armed" LED and any active alarm LEDs should turn off, and the system should return to the disarmed state.	The "armed" LED and any active alarm LEDs turned off, and the system returned to the disarmed state.
7	1. Ensure the system is disarmed. 2. Simulate any sensor being triggered. 3. Check all LEDs.	No LEDs should light up, as the system is disarmed.	No LEDs lit up
8	1. Arm the system. 2. Simulate multiple sensors being triggered simultaneously. 3. Check all alarm LEDs.	All corresponding alarm LEDs for the triggered sensors should light up and stay on until the system is disarmed.	All corresponding alarm LEDs for the triggered sensors lit up and stay on until the system was disarmed.

Table 1: Test plan with expected and observed results.

Code

```
// armed state (initialize to disarm)

int armed = 0;


// infinite loop

while (1) {

      /* Part 5. Typed by Karl Richards. */

      // check if ArmButton is pressed

      if ((GPIOC->IDR & (1 << 13)) == 0) {

            // arm system

            armed = 1;  // set arm state

            GPIOA->ODR |= (1 << 5);  // turn on ArmStatus LED

      }


      // check if DisarmButton is pressed

      if ((GPIOA->IDR & (1 << DisarmButtonPin)) == 0) {

            // disarm system

            armed = 0;  // reset arm state

            GPIOA->ODR &= ~(1 << ArmStatusPin);

            GPIOB->ODR &= ~(1 << FrontAlarmPin);

            GPIOB->ODR &= ~(1 << BackAlarmPin);

            GPIOB->ODR &= ~(1 << WindowAlarmPin);

      }


      /* Part 6. Typed by Brenden Clark. */

      // if system is armed, check sensors

      if (armed) {

            // check the FrontSensor

            if (!(GPIOA->IDR & (1 << FrontSensorPin)) == 0) {

                  // system armed & front door is open:

                  GPIOB->ODR |= (1 << FrontAlarmPin); // turn on FrontAlarm LED

            }
            // check the BackSensor
            if (!(GPIOA->IDR & (1 << BackSensorPin)) == 0) {
                  // system armed & back door is open:
                  GPIOB->ODR |= (1 << BackAlarmPin);  // turn on BackAlarm LED
            }

            // check the WindowSensor
            if (!(GPIOA->IDR & (1 << WindowSensorPin)) == 0) {
                  // system armed & window is open:
                  GPIOB->ODR |= (1 << WindowAlarmPin);//turn on WindowAlarm LED
            }
      }
}
```

Conclusion

During this lab, a functional home alarm system was designed and implemented using digital I/O programming in C. By using reed switches, LEDs, and buttons, the system was able to simulate real-world scenarios, including arming/disarming and detecting triggered alarms. Testing covered many different combinations to ensure the system behaved as expected. Through this lab, we got a deeper understanding of GPIO programming, collaborative problem-solving, and debugging our code.

This lab built on previous material such as pull-up/down resistors, pin configurations, and GPIO operations but this time we did the programming in C. It took that material we learned and applied it to a real-world application. This lab also emphasized pair programming, which we'll be doing a lot of in the future as our projects and programs get more difficult and complicated. The skills we learned and used in this lab will be useful in designing systems for practical uses in the future.