ECEN 260 Lab 05

Digital Input & Output in 'C'

Lab Objectives

- Learn how to program the digital I/O ports using the 'C' programming language.
- Learn how to reference specific pins in the code and identify them on the board.
- Implement a home security system that requires I/O to operate.
- Develop system specifications as a team.
- Practice *pair programming*.

Required Parts

- 1 Nucleo-L476RG development board
- 1 USB cable
- 1 breadboard (protoboard)
- 3 reed switches (each has two parts)
- 3-4 LEDs (you choose the colors)
- 3-4 resistors
- 1-2 push buttons
- jumper wires (the instructor has extras)

Overview

This laboratory has a real-world application. It simulates a home alarm system that includes 3 magnetic sensors: one **front door** sensor, one **back door** sensor, and one **window** sensor. Each sensor has two parts: a reed switch and a magnet. When they are separated, as when a door or window is opened, the reed switch opens and triggers an alarm. The alarms are simulated by LEDs in this lab.

Note: This lab is similar to the one described at the end of Chapter 5 in the *Programmable Microcontrollers* textbook from the reading.

Lab Requirements

You may use the structure of the included starting code ("scaffolding code") as a guide. Your system must have a way to become "armed" or "disarmed." When it is armed, you should have a way of showing when a sensor (front/back/window) is triggered and which sensor was triggered. That indication should stay active until the system is disarmed.

Program Specifications

The precise specifications of how you meet the requirements must be **determined by you and your lab partner(s).** Here are some things you can decide:

- What light will show which situation?
- Which button will arm? Which button will disarm?
- Will you wire the reed switches to be active high or active low switches?
- Which pins will we use for the inputs and outputs?

Connections

You will need 2 input buttons (“arm” and “disarm”), 3 input reed switches (“front” door, “back” door, and “window”), and 4 output LEDs (one to indicate the “armed” status, and each of the other three as an alarm for one of the door/window sensors. You may choose to use the green onboard LED (LED 2 - PA5) as one of the output LEDs – possibly as the “armed” LED. You may also choose whether to use the blue onboard button (Button 1 – PC13) as one of the input buttons.

Choosing Pins: When choosing pins for your inputs and outputs, only use pins on Ports A, B or C. To determine the location of pins, refer to Figure 1 below. Note that your code will reference pins using their Morpho names, but the board itself is labeled with the Arduino names. For example, to use PA8, you would reference Port A (GPIOA) pin 8, but the board itself labels that pin as D7. Remember that the blue Button 1 is *already* connected to PC13 and the green LED 2 is *already* connected to PA5.

Important! For this lab, do not use these pins: PA2/PA3 (used for communication), PA13/PA14/PA15 (used for debugging), PB4/PB5 (have a different default value than we are assuming), or PC14/PC15.

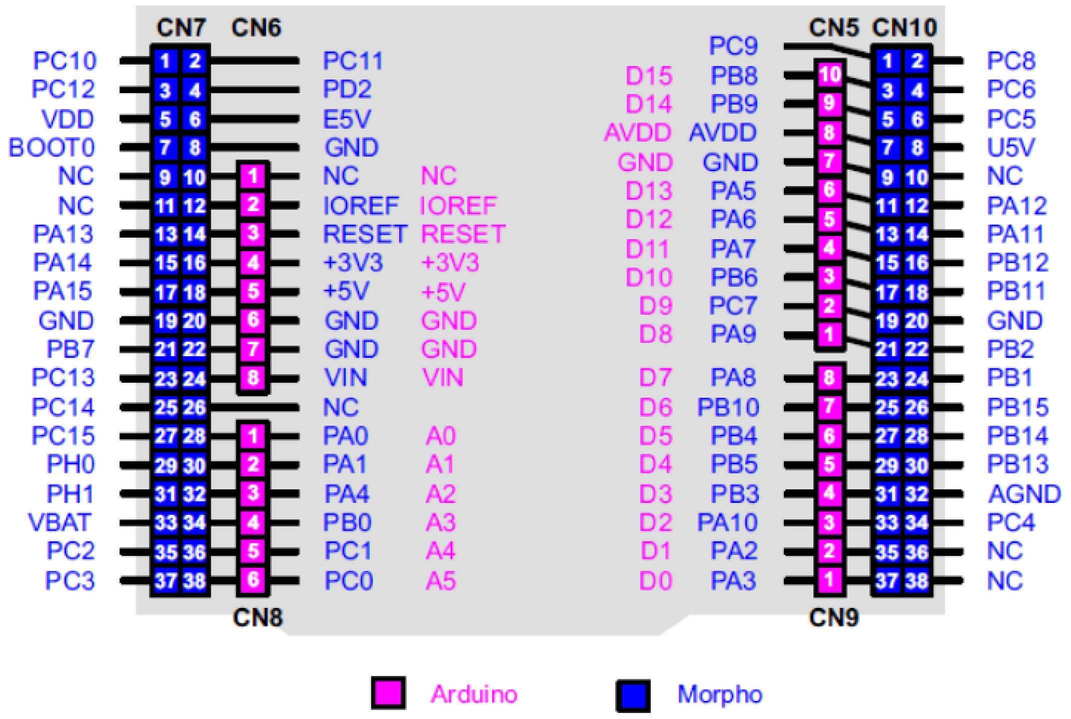

Figure 1: Nucleo-L476RG pinout diagram from user manual.

Current-limiting Resistors: Remember that each LED requires a current-limiting resistor so it doesn't burn out. The green onboard LED (LED 2, PA5) already has an onboard current-limiting resistor. Each additional LED you connect to your board as an output must have a current-limiting resistor.

Pull Resistors: Remember that two-way buttons and switches (like the push buttons and reed switches that we are using) require pull resistors to provide a "default" voltage level for when the button/switch is disconnected. Buttons/switches connected in an active-high configuration require pull-down resistors; buttons/switches in an active-low configuration require pull-up resistors. If you use the blue onboard button (Button 1, PC13), note that it is already wired as active-low with an onboard pull-up resistor. Each additional push button and reed switch will require a pull resistor. It is recommended that you use the internal pull resistors, enabling them using the Pull-Up/Pull-Down Register (PUPDR).

Lab Team Requirements / Pair Programming

Teams must be 2 or 3 people total. **You will practice a method called *pair programming.*** Pair programming is when two (or three) people code a project together, but under certain constraints.

Those constraints are that only one person can "drive" (write code) at a time; the other teammate(s) may assist verbally (navigate) but may not touch the computer when they are not driving. Sections of the code are labeled to specify who should drive for each section.

Starting Code (Scaffolding)

```
/* This program uses inputs from buttons and reed switch sensors
 * and outputs to status and alarm LEDs to create an home
 * security system.
 */

// INSTRUCTIONS FOR 2-PERSON TEAM:
// Person A needs to type parts 1, 3, and 5.
// Person B needs to type parts 2, 4, and 6.

// INSTRUCTIONS FOR 3-PERSON TEAM:
// Person A needs to type parts 1 and 4.
// Person B needs to type parts 2 and 5.
// Person C needs to type parts 3 and 6.


// GPIO Port struct
typedef struct{
	unsigned int MODER;   // offset: 0x00
	unsigned int OTYPER;  // offset: 0x04
	unsigned int OSPEEDR; // offset: 0x08
	unsigned int PUPDR;   // offset: 0x0C
	unsigned int IDR;     // offset: 0x10
	unsigned int ODR;     // offset: 0x14
} Port;
```

```c
// Address of the Advanced High-performance Bus 2 Enable Register
unsigned int* AHB2ENR = (unsigned int*) 0x4002104c;

// Base addresses of the GPIO Port control registers (SFRs)
Port* GPIOA = (Port*) 0x48000000; // base address of GPIOA
Port* GPIOB = (Port*) 0x48000400; // base address of GPIOB
Port* GPIOC = (Port*) 0x48000800; // base address of GPIOC

/* Part 1. Typed by __________. */

// input pins
#define ArmButtonPin    13 // PC13
#define DisarmButtonPin ?  // P??
#define FrontSensorPin  ?  // P??
#define BackSensorPin   ?  // P??
#define WindowSensorPin ?  // P??

// output pins
#define ArmStatusPin    5  // PA5
#define FrontAlarmPin   ?  // P??
#define BackAlarmPin    ?  // P??
#define WindowAlarmPin  ?  // P??


int main(void)
{
	// Turn on GPIO clocks (Ports A, B, and C)
    *AHB2ENR |= 0b111;

    /* Part 2. Typed by __________. */

    // Configure button pin as inputs
    GPIOC->MODER &= ~(0b11 << (ArmButtonPin*2));    // configure ArmButtonPin as input
    GPIO?->MODER &= ~(0b11 << (DisarmButtonPin*2)); // configure DisarmButtonPin as input

    // Configure reed switch pins as inputs
    ???; // configure FrontSensorPin as input
    ???; // configure BackSensorPin as input
    ???; // configure WindowSensorPin as input

    /* Part 3. Typed by __________. */

    // Enable pull resistors
    GPIO?->????? |= (0b?? << (DisarmButtonPin*2));
    GPIO?->????? |= (0b?? << (FrontSensorPin*2));
    GPIO?->????? |= (0b?? << (BackSensorPin*2));
    GPIO?->????? |= (0b?? << (WindowSensorPin*2));

    /* Part 4. Typed by __________. */

    // Configure LED pins as outputs
    ???; // configure ArmStatusPin as output
    ???; // configure FrontAlarmPin as output
    ???; // configure BackAlarmPin as output
    ???; // configure WindowAlarmPin as output
    // Initialize Alarm Status (turn off all LEDs)
    GPIOA->??? ?= ~(1 << ArmStatusPin);
    GPIO?->??? ?= ~(1 << FrontAlarmPin);
    GPIO?->??? ?= ~(1 << BackAlarmPin);
    GPIO?->??? ?= ~(1 << WindowAlarmPin);

    // armed state (initialize to disarm)
    int armed = 0;

    // infinite loop
      while(1){

            /* Part 5. Typed by __________. */
            // check if ArmButton is pressed
            if (???){
                  // arm system
                  armed = 1; // set arm state
                  GPIOA->??? ?= (1 << ArmStatusPin); // turn on ArmStatus LED
            }

            // check if DisarmButton is pressed
            if (???){
                  // disarm system
                  ???; // reset arm state
                  ???; // turn off ArmStatus LED
                  ???; // turn off FrontAlarm LED
                  ???; // turn off BackAlarm LED
                  ???; // turn off WindowAlarm LED
            }

            /* Part 6. Typed by __________. */
            // if system is armed, check sensors
            if (armed){

                  // check the FrontSensor
                  if ((GPIO?->IDR & ???) == ?){
                        // system armed & front door is open:
                        ???; // turn on FrontAlarm LED
                  }

                  // check the BackSensor
                  if (???){
                        // system armed & front door is open:
                        ???; // turn on BackAlarm LED
                  }

                  // check the WindowSensor
                  if (???){
                        // system armed & front door is open:
                        ???; // turn on WindowAlarm LED
                  }
            }
      }
}
```

Demonstration of Correct Operation

As soon as your system works correctly, record a video demonstration. Upload (1) that video and (2) a copy of your main.c file to the “Lab 05 Demo & Code” assignment.

Lab Report Instructions

Starting with this week’s lab report, your lab reports should now always include a “Test Plan and Results” section. The goal is for you to create a test plan that would be able to correctly identify whether or not your system met the requirements/specifications.

Your test plan should include several tests, each test with precise steps on how to complete that test. Your selection of tests should include common cases as well as edge cases / error cases as appropriate. (Test the extremes, or test what happens when the user does something unexpected.) Each test should also accurately indicate the specific expected results. Then, after creating your test plan, perform each test and record the *actual* results of each test.

Use the Lab Report Template provided in the “Lab 05 Report” I-Learn assignment. Submit your report there.