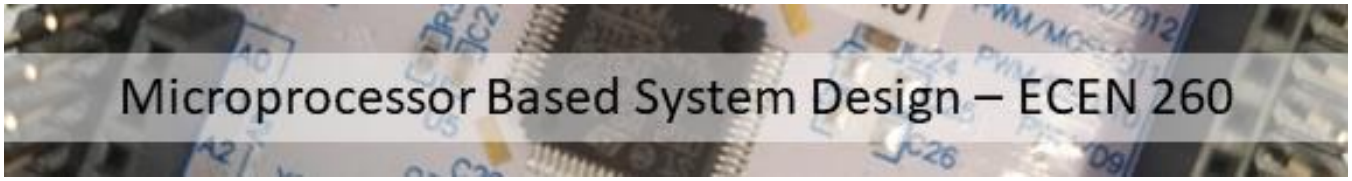

ECEN 260 Lab Report

Lab #07: Device Drivers

Name: Brodric Young

Lab Partner: Ben Jensen

Date: 2/19/2025

Instructor: Brother Allred

Semester: Winter 2025

Lab Overview

The objective of this lab is to develop device drivers for the STM32 microcontroller, test them with the alarm system circuit we made last lab, and learn how to use built-in Hardware Abstraction Layer (HAL) drivers. In part 1, we wrote our own GPIO drivers, peer reviewed code from lab partners, and then tested our drivers with the alarm circuit. In part 2, we used the I/O Configuration tool to generate code using the built-in HAL drivers. Then I added more code using the HAL driver to implement the same alarm system and tested to make sure it functioned the same way.

Schematic

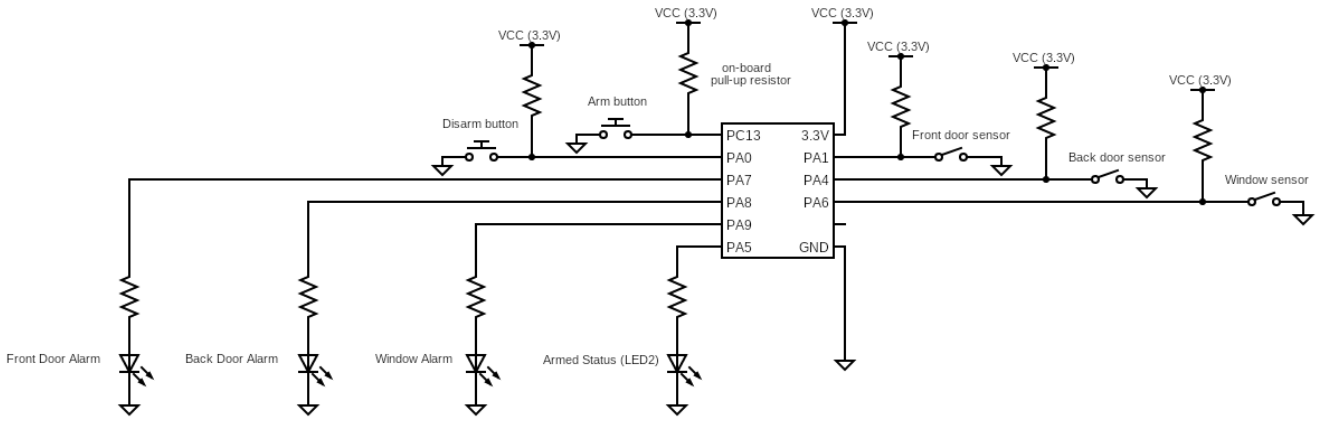

Figure 1: A schematic of the final system.

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

Hal Drivers main.c code snippet:

```
// if system is armed, check sensors
if (armed){
  // check the FrontSensor
  if (HAL_GPIO_ReadPin(FRONTSENSOR_GPIO_Port, FRONTSENSOR_Pin) != 0){ // if
reed is pulled away
        // system armed & front door is open:
        HAL_GPIO_WritePin(FRONTALARM_GPIO_Port, FRONTALARM_Pin, 1); // turn
on FrontAlarm LED
  }

  // check the BackSensor
  if (HAL_GPIO_ReadPin(BACKSENSOR_GPIO_Port, BACKSENSOR_Pin) != 0){ // if
reed is pulled away
        // system armed & back door is open:
        HAL_GPIO_WritePin(BACKALARM_GPIO_Port, BACKALARM_Pin, 1); // turn off
BackAlarm LED
  }

  // check the WindowSensor
  if (HAL_GPIO_ReadPin(WINDOWSENSOR_GPIO_Port, WINDOWSENSOR_Pin) != 0){ // if
reed is pulled away
        // system armed & window is open:
        HAL_GPIO_WritePin(WINDOWALARM_GPIO_Port, WINDOWALARM_Pin, 1); // turn
on WindowAlarm LED
  }
}
```

My Drivers main.c code snippet:

```
// if system is armed, check sensors
if (armed){
      // check the FrontSensor
      if (GPIO_ReadPin(FRONT_SENSOR_PORT, FRONT_SENSOR_PIN) != 0){
            // system armed & front door is open:
```

```
            GPIO_WritePin(FRONT_ALARM_PORT, FRONT_ALARM_PIN, 1); // turn on
FrontAlarm LED
      }

      // check the BackSensor
      if (GPIO_ReadPin(BACK_SENSOR_PORT, BACK_SENSOR_PIN) != 0){
            // system armed & back door is open:
            GPIO_WritePin(BACK_ALARM_PORT, BACK_ALARM_PIN, 1); // turn off
BackAlarm LED
      }

      // check the WindowSensor
      if (GPIO_ReadPin(WINDOW_SENSOR_PORT, WINDOW_SENSOR_PIN) != 0){
            // system armed & window is open:
            GPIO_WritePin(WINDOW_ALARM_PORT, WINDOW_ALARM_PIN, 1); // turn on
WindowAlarm LED
      }
}
```

HAL MODER configuration:

```
temp = GPIOx->MODER;

temp &= ~(GPIO_MODER_MODE0 << (position * 2u));

temp |= ((GPIO_Init->Mode & GPIO_MODE) << (position * 2u));

GPIOx->MODER = temp;
```

My MODER configuration:

```
switch (mode) {

      case GPIO_MODE_INPUT: // configure to 00

            GPIOx->MODER &= ~BITPAIR(pin); // reset both bits

            break;


      case GPIO_MODE_OUTPUT: // configure to 01

            uint32_t temp = GPIOx->MODER;    // read the MODER

            temp &= ~UBIT(pin);              // reset upper bit

            temp |= LBIT(pin);               // set lower bit

            GPIOx->MODER = temp;        // clock in entire configuration

            break;                      //in 1 clock cycle to avoid glitches

}
```

Conclusion

This lab focused on writing custom device drivers, peer reviewing code, and testing those drivers with an alarm system using reed switches and LEDs. By the end of the lab, we gained experience in writing and testing drivers to ensure they functioned right and learned the importance of code review. We also learned how to generate and use HAL drivers to do the same thing we wrote for our drivers. This can be seen in the code section by the main.c files having the same structure, just different driver function names and by the configuration operations using the MODER configuration as an example.

This lab connects directly to previous classes and labs by writing the code for the alarm system in a simplified way using drivers, both writing our own or the HAL generated drivers. It also highlights peer review to improve code which we'll be using in future classes and in future careers. Learning how to write, generate, and use drivers is going to be important as our circuits and code become more complicated and long.