Lab #4 Motion Detection Alarm (GPIO)
Summer 2021

Lab location: Online only

Lab Time: Thursday 7/13, 8:00am – 10:30am

Demo Due: The beginning of lab on Tuesday 7/20

Submission Due: Tuesday 7/20, 11:59pm

Introduction

The goal of this lab is for you to use GPIO (general-purpose I/O) and continue to exercise C and Assembly mixed programming.

The learning objectives are:

- **Develop a motion detection alarm** using a PIR (Passive Infrared) motion sensor and a buzzer
- **Exercise GPIO programming**
- **Exercise C and Assembly mixed programming (non-leaf functions)**

Prelab

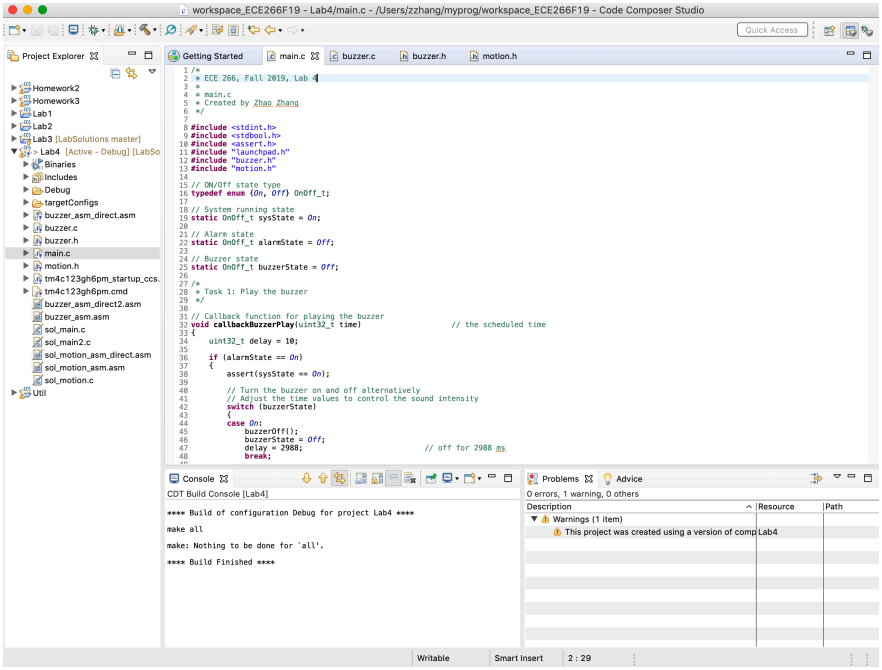
- Read the following:
  - Wiki page for the Grove buzzer: http://wiki.seeedstudio.com/Grove-Buzzer/
  - Wiki page for the Grove PIR motion sensor: http://wiki.seeedstudio.com/Grove-PIR_Motion_Sensor/
  - Grove - PIR Motion Sensor v1.2 Schematics
  - BISS0001 Datasheet (the IC used in the PIR motion sensor)
- **[Important] Read "TivaWare Peripheral Driver Library User Guide", Ch. 15 GPIO.** The document can be found under the "Resources" tab on Blackboard.

Part 1: Setup

One partner (with the help of partner): In your working directory, create a new folder named “Lab4”. Download “lab4_main.c”, “buzzer.c”, “buzzer_asm.asm”, and “buzzer.h” from Lab 4 attachments to this folder. Switch to CCS, create a new CCS project “Lab4” with an external reference to the “Lab4” folder. Remember to add the #include directory paths and the library file paths for the TivaWare and Util libraries. Switch to GitHub Desktop, commit the changes to your local repository, then push the changes to your group’s GitHub repository. Check on https://github.com to make sure the changes have been pushed into your group’s repository.

Other partner(s): Run GitHub Desktop. Pull the new changes from GitHub to your local repository. Check that Lab4/ and the new files appear in your working directory. Sw itch to CCS, create a new CCS project “Lab4” with an external reference to the “Lab4” folder. Remember to add the #include directory paths and the library file paths for TivaWare and Util libraries.

At this time, your CCS window may look like the following:



Attach the LaunchPad to the Grove base boosterpack, and then attach the Grove buzzer to jumper **J17** of the Gove base boosterpack. Double check that you have used the correct jumper.

Build and debug the project. The program will behave as follows:

- Initially, the LED is turned on in green.
- Push the SW1 button, the buzzer will start beeping periodically, and the LED is turned on in red.
- Push the SW2 button, the buzzer will stop beeping, and the LED is turned on in green.

Note: The buzzer sound can be disturbing in a quite environment. Be mindful of people around you. You may temporarily remove the buzzer if needed.

Read through the code in "main.c", "buzzer.h", "buzzer.c", "buzzer_asm.asm" to understand how the codes work. Pay attention to how buzzer-related functions are written in "buzzer.c" and "buzzer_asm.asm".

Part 2: Motion Detection

In this part, you will write a set of driver functions for the Grove PIR motion sensor. A Driver functions implement the basic operations of a peripheral device. You will write some function(s) in C and some in assembly. To see examples, review the code in "buzzer.c", "buzzer_asm.asm" and "buzzer.h".

Connect the **PIR motion sensor** to jumper **J16** of the Grove base. The motion sensor uses a single signal pin marked as D1. The other signal pin is not connected to the sensor and is marked as NC (Not Connected). When the motion sensor is connected to J16, the signal pin of the sensor is connected to the PC4 pin (Port C, pin 4) of Tiva C.

In CCS Project "Lab4", create a new C file named "motion.c", an assembly file named "motion_asm.asm" and a header file named "motion.h". Then,

- In "motion.h", add function prototypes for the functions that you plan to add to "motioin.c" and "motion_asm.acm".
- In "motion.c", write an initialization function for the PIR motion sensor. Note: See function buzzerInit() in "buzzer.c" for an example.
- In "motion_asm.asm", write an assembly function that reads the pin input from the motion sensor and returns true (in bool type) if the pin input is 1, false otherwise. Note: See functions buzzerOn() and buzzerOff() in "buzzer_asm.asm" for examples.
- Revise "main.c" to add a callback function that periodically checks the PIR motion sensor input and prints a message, e.g. "motion detected" or "motion not detected", in the CCS terminal.

Note: The buzzer is an output device and the PIR motion sensor is an input device. Therefore, your code should call different TivaWare GPIO functions.

Discuss with your partner(s) and assign tasks to each person. Carry out the tasks, integrate all the codes, then test and debug.

Part 3: Motion Detection Alarm

Further revise "main.c" such that:

1. Initially, the system is deactivated (sysState = Off).
2. If a user pushes SW1, the system should be activated (sysState = On). If a user pushes the SW2 button, the system will be de-activated.
3. When the system is deactivated, the LED should be turned off, and the buzzer should not make any sound.
4. When the system is activated and there is NO human motion around, the LED should be turned on and stay in the **green** color. The buzzer should not make any sound.
5. When the system is activated and there is any human motion around, the LED should be turned on in the **red** color and the buzzer should beep (as in Part 1).
6. ***Optional***: You may notice that the alarm is on and off, even if there is consistent human motion. That is the behavior of an IC in the buzzer module (the BISS0001 IC; see its datasheet on Blackboard). Revise your program so that the alarm stays on whenever there is any human motion detected in the past three seconds. Hint: Your code may memorize the last time that human motion is detected, and compare that time with the current time to decide if the alarm should be turned on or off.

Demo your system to your TA. Note: If you can demo part 3, you don't have to demo part 2.

Demo Requirement

At the time of demo, first show your source code to your TA and explain how it works. Each of you may have to answer questions from your TA, which test your understanding of the working details. Your answer may affect your lab grade.

Lab Report

Your group should write a lab report that includes the following contents.

1. Describe the responsibility of each person in your group and estimate the breakdown of contribution in percentage (e.g. 50% and 50% for a two-person group, or 33.3%, 33.3% and 33.3% for a three-person group).
   Note: If one contributes 40% or higher in a two-person group or 25% or higher in a three-person group, he or she may receive the same grade as the group gets.
2. Describe the two functions you write in "motion.c" and "motion_asm.asm", and explain how they work.
3. Describe the general idea how you revise "main.c" in Part 3 to implement the requirements.
4. How much flash memory does your program take? How much SRAM memory?
5. Estimate the number of hours in total you and your partners have worked on this lab.

Lab Submission

Make sure that you have pushed the latest version of your code into GitHub. **This step is required for you to get the grade.**

Then, submit the following items on Blackboard under Lab 4 assignment:

1. motion.c
2. motion_asm.asm
3. motion.h
4. The revised main.c
5. The lab report