Name: Calvin Heischman & Gabriel Moore

EID: CJH4858 & GAM3769

Semester: Spring 2024

Course: ECE445L

A) ***Objectives*:**

1. In a few sentences, describe the purpose of the lab and the features of your alarm clock.

This lab aims to build, test, and validate an embedded system. We created a gaming controller that is HID compliant thus allowing us to use it universally across several systems.

B) ***Hardware Design Deliverables:***

1. N/A

C) ***Software Design Deliverables:***

1. I have pushed my project to GitHub for grading (Yes/No):
2. Deliverable 1: System design diagram of the modules created.

A screenshot of a computer

Description automatically generated

D) ***General Deliverables:***

1. Deliverable 2: Software Tests

Button Publishing: When a button is pressed, the value is written both to the USB report and to a struct in the menu. If we want to verify that our button sampling is working properly, we can check either the usb report being sent (external) or print to the display to see what the tm4c is reading internally to make sure they match.

Device Descriptor: We define each entry in the string table, which combines to create the device descriptor. When the device is initialized, there is a verification function that checks if it was successfully received by the host before entering the main while loop. If we don’t properly describe the device, it will result in a specific fault at the main launch screen & we can check Wireshark to monitor the most recent data sent from the device to the host computer.

Menu functionality: We added multiple menu modes that change the button mappings in the HID report. This way, we’ve actually built our own debugging menu to configure the device based on specifications we need to test to compare both internal values with what gets sent to the external host.

1. Deliverable 3: Hardware Tests

Measured the current through several components to ensure they drew the right amount of power. Also checked for continuity throughout the systems to ensure that the current wasn’t being shorted anywhere in our circuit. Had components, like an LED that were used to quickly see if the board was being powered and if it had enough power to operate correctly.

1. Deliverable 4: Power consumption

25 + 160 + 3.6 + 25 + 7\*2 = 227.6 mA (max current draw)

1. Deliverable 5 (10pt EC): Characterization of the system

**CPU utilization:** We performed this measurement by sampling the time different before and after each module execution. This execution time was then taken as a ratio of the overall bus frequency (50 Mhz). All of the following measurements met our system requirements as they were nonintrusive. The primary source of error stems from frequency of execution, as the following threads assume that each module is executed on every check instead of the cases where they are not.

* + 1. Thread 1 (garf handler): The handler is only used to initialize everything. This means it is called whenever the device is plugged/unplugged into the computer. When in the handler, it’s using 100% of the CPU since it disables all interrupts. But it’s only called once, so after initialization, it’s negligible and functionally 0% of overall CPU usage.
    2. Thread 2 (buttons): 0.0125%
    3. Thread 3 (menu): 0.0125% when inactive, 0.0347% when active

**ADC jitter:** 1.17 us. We performed this measurement by taking multiple before/after timestamps when we sampled the button polling and comparing the time the polls took. This met our system requirements as the time it took to sample was nonintrusive and therefore didn’t interrupt our ability to frequently poll the usb host. We might have experienced error from the number of buttons pressed, as we poll all the buttons but interrupt based on which ones get toggled.

E) ***Analysis and Discussion Questions:***

1. N/A