Lab Three - Bus Communication: I2C and SPI

A bus is an interface that allows different components to communicate with each other, so that each component can be modularized and focus on a single task or duty within the larger scope of the system. For example, a modern personal computer generally contains a CPU for processing, memory (usually one of the variants of RAM) for storing programs and instructions that the CPU runs, a hard drive for storing files and user data, a GPU with an LCD display and keyboard/mouse peripherals for user operability, and a sound card. Computers may have less or they may have more components(e.g. laptops generally have WiFi cards for wireless connectivity to the Internet), but all components within a computer communicate with each other through a type of bus that makes it possible for all of the components to function together to allow your own computers to function.

In this lab, we will explore two bus protocols that are common and popular in embedded and IoT systems: SPI and I2C. We will use SPI to communicate with an accelerometer and I2C to communicate with an LCD screen and start building our smart watch. To begin building our smartwatch application we must complete the following on our ESP8266.

Part One: Displaying and Keeping Time

1. Obtain and modify the system time. When the ESP8266 boots up, its system time is reset to a default value; use the real-time clock (RTC) to set the system time.
2. Interface with the LCD screen through I2C and display the system time on the screen; the display should continuously update the time just like in a normal watch.
3. The OLED display shield comes with 3 buttons. Use these buttons to add functionalities to allow you to change the time and update the LCD display to reflect these changes.
4. Add in the functionality to view values from the light sensor. Additionally, adjust the display brightness or contrast depending on the brightness in the room (the display should get brighter if the room is dimly lit and more dim if it is bright out).

Part Two: Adding an Alarm

1. Add in an alarm functionality, that allows you to set an alarm. When the system time reaches this time, the vibration motor/piezzo should start vibrating to indicate that the alarm has gone off. Additionally, there should also be a visual indication.
2. Aim to combine the time and alarm functionalities into one system. In other words, combine parts one and two into one system.

Part Three: Interfacing with the Accelerometer

1. Interface with the accelerometer using SPI.
2. Use the accelerometer readings and implement text scrolling; if you tilt the accelerometer far enough in one of the four principal directions (up, down, left, right), the text on the screen should begin scrolling in the same direction. For example, if you tilt the accelerometer to the right, the text on the OLED screen scrolls to the right until it goes off screen before it reappears on the left side of the screen.

Group Members:

EECS 4764 Lab Three Checkpoints:

1. Display the time on the OLED display; the time should update like a regular watch. Be able to change time through the OLED display buttons.

2. Adjust screen brightness based on ambient light

3. Add alarm functionality; Be able to set alarm through OLED display and when alarm goes off, a visual and audio notification should go off.

4. Combine 1, 2, and 3.

5. Text scrolling based on accelerometer readings.