Project 2022

During the semester you used assembly code to manipulate the on board I/O devices of an AVR. As a result, you gained an understanding of how to interface and interpret the signals generated by these I/O devices. Some of the knowledge you obtained from the course are:

- Manipulate memory and memory mapped I/O elements to digitize analog voltages and display information.
- Serial communication to send and receive serial data using the USART.
- Use the processor's interrupt capability to respond to asynchronous events from one or more sources.
- Examine the signal operation of the above elements

In this project you will integrate all of the above elements and with additional code and hardware implement a Supervisory Control and Data Acquisition (***SCADA***) system. These types of systems are found in in a variety of real world applications. (e.g. manufacturing, power plants, etc). It will have following features and capabilities:

- The system should display a power up banner message with a version number ("***SCADA Mon v1.0***") on both the LCD and a terminal connected to the UART and the student's ID on the second line on both LCD and terminal for a duration of 4 seconds.

- The banner message on both the LCD and terminal will be replaced by another which will indicate whether the system is in ***Supervisor*** mode or ***Node*** mode configuration. The mode will be determined by a switch connected to PB0 pin on PORTB. The terminal connected to the UART will also reflect the power up mode state and display a related prompt (e.g. ***Svr#*** or ***Node>***). On the LCD, the 2nd line should simply indicate that it is waiting for a single character command input (e.g. ***Mode standby***). The mode selection can be switchable after power up, so it should be polled every so often. Switchover shouldn't lag by more than a second or two.

- In ***Node*** mode, the ADC will be used to monitor an analog input voltage (0 - 5V). When commanded from the terminal, the ADC will acquire 256 samples at a sample rate of one every 2mS and store them in a memory buffer. After the samples are acquired, the number of samples below (**L**) and above (**H**) a set trim level will be displayed. The trim should be a set at a default value of 2.5V at power up. On the second line of the LCD display, the two count values (in hex) will be displayed (e.g. **L:XX H:XX).** During the sampling interval, a 10 uS pulse should be generated on PE0 pin on PORTE for each sample.

- In ***Supervisor*** mode, the system will display a run time (***HH:MM*** format) from the time it powers up on the second line of the LCD. The time values should be kept in the data segment to make it easy to share them between code.

- Use a memory mapped two digit 7-segment LED to display seconds using available devices used in this course (e.g. '373, 4511).

- The above time keeping function in ***Supervisor*** mode will use an interrupt from an external clock generating circuit providing a reference frequency to maintain time (e.g. 555). The frequency used as a reference is 500Hz.

- The terminal commands sent to the AVR will depend on its mode and are shown below. Invalid terminal commands should show a corresponding error message for invalid commands or number entry related to mode. Also, confirmation of command completion

Supervisor Mode:	***Node Mode:***
• **R** - Reset run time display on LCD to zero**.**	• **A** - Acquire a new sample set and update LCD
• **P** - Pause run time operation (i.e. start/stop)	• **S** – Display the sum of the sample set **S:XXXX**
• **T** - Toggles LCD to ***MM:SS*** and ***HH*** on LED	• **T** – Prompts for and sets a new ADC trim level **XX**

- Program design should be modular by implementing the above features with subroutines and minimizing the main loop as much as possible, You can use the provided library code (as is) along with other code if duly referenced and grouped in an ***.inc*** file. For example, you will need a subroutine(s) which will converts the 2-digit ASCII characters into a hex or BCD values using arithmetic/logical instructions.

- In addition to the above, implement some unique additions or hardware to the above system. It will be worth 10% as part of the deliverable portion depending on complexity.

This project is an independent effort and each student is responsible for accomplishing its objectives. No assistance from the instructor will be provided except for clarification. Your embedded system comprising of hardware from the preceding labs should reliably demonstrate your code, otherwise a loss of marks will be incurred.

The implementation of the above objectives should have attained the following goals:

- Write code that meets stated objectives
- Make use of data sheet(s) to understand circuit operation
- Use measurement tools to build & troubleshoot hardware
- Technical report documenting project code and hardware
- Completion of project within alloted time

Due dates: refer to drop boxes for Deliverables & Report