Smart Presentation Remote: Software Report

Margaret Garvan and Samuel Smith

Overview

The Smart Presentation Remote requires software to run on both the Atmel Xmega microcontroller as well as the host PC. Interfacing between the applications is done through Bluetooth. The microcontroller communicates through a Bluetooth UART device and the host PC connects to the device and treats it as a virtual COM port. Throughout the semester, we have written a functional remote control application that demonstrates the capabilities of our device, but the framework was not easily extensible. As our hardware is almost finished, we have allocated most of the remainder of the semester to software development.

Microcontroller Program

Our microcontroller program is relatively simple. It starts by initializing all the peripherals and drivers. It also draws the Gator Engineering logo to the screen on startup to demonstrate the features of the graphical LCD. The body of the program runs in a continuous loop in which it polls the gyroscope over I^2C. It also checks the keypad. Debouncing the keypad is performed by only transferring the keypad value during an interrupt routine which will be described later.

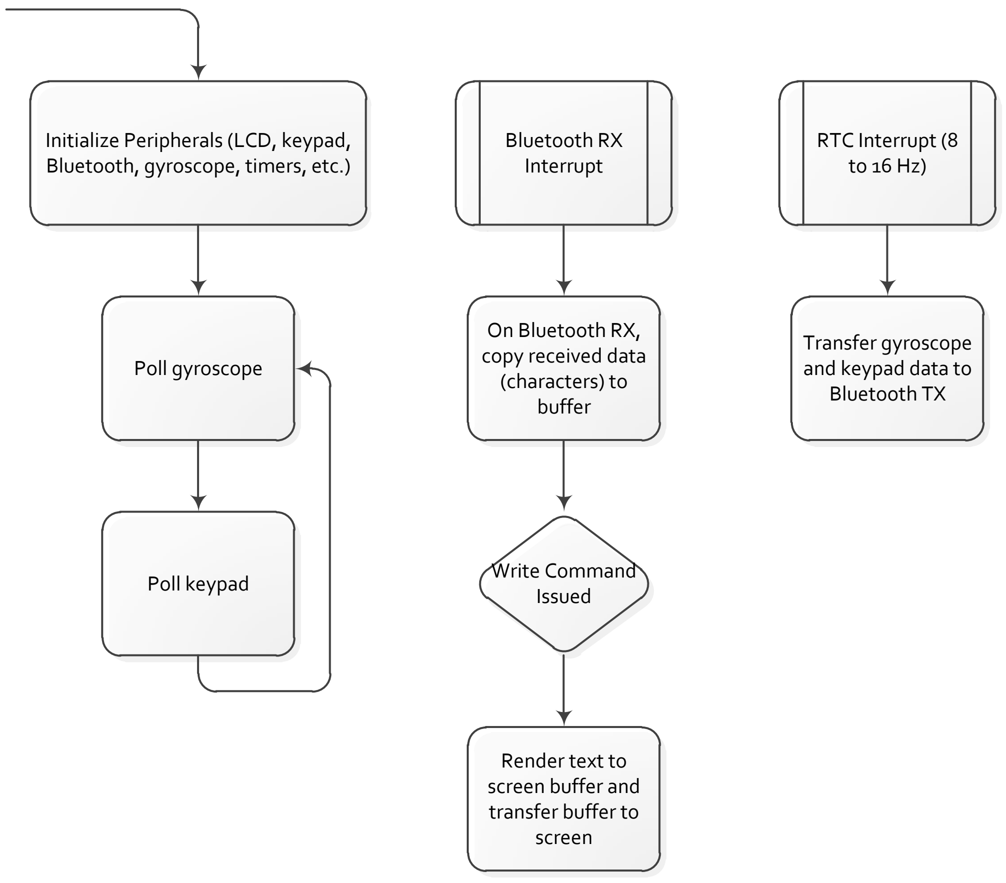
Interrupts are used to control the rest of the microcontroller program. The real-time clock is configured to generate interrupts at a rate of 8 Hz to transmit the keypad and gyroscope data to the host computer using the Xmega USART. We are hoping to increase this rate to 16 Hz with future optimization. The device implements a set of control words to manipulate device functionality from the host computer software. The final version of the software will implement control bytes for the following features:

1. Clear screen buffer
2. Move screen buffer pointer to next line
3. Render and draw text to screen
4. Laser pointer toggle

The current implementation is based on using the Bluetooth TX interrupt and transferring each read character (except control characters) to a small text buffer. When the render command is issued, the text is drawn to the screen. We are presently having some timing issues with copying all the Bluetooth data fast enough without errors and are considering various possibilities for optimization of the transfer such as lowering the baudrate to allow more time for each individual character to be processed.

Rendering the LCD contents is performed through the driver provided for the screen. The current implementation is somewhat inefficient because it requires the entire screen to be rendered and transferred at the same time. If time permits, we will modify the driver to allow for the transfer of single lines of the screen as opposed to drawing the entire screen all at once. This would be useful for features like the clock or timer, which are updated frequently and only take up a single line.

A flowchart for the primary functionality of the driver program is provided below:


Initialize Peripherals (LCD, keypad, Bluetooth, gyroscope, timers, etc.)
Poll gyroscope
Poll keypad
Bluetooth RX Interrupt
On Bluetooth RX, copy received data (characters) to buffer
Write Command Issued
Render text to screen buffer and transfer buffer to screen
RTC Interrupt (8 to 16 Hz)
Transfer gyroscope and keypad data to Bluetooth TX

Driver Application

The driver application for the PC is written in C# using Visual Studio 2010. A preliminary version of the driver just polled the virtual COM port to which the Bluetooth device was connected and moved the mouse based upon gyroscope readings. It could also emulate keyboard events and allow for simple control of a PowerPoint presentation. To demo writing to the screen, the program would place a clock on the screen as well as the current location of the mouse cursor. The old version of the program was very useful for testing purposes, but is now in the process of being completely rewritten from the ground up to allow for a more extensive software architecture.

The new version of the software features a graphical user interface for the driver that allows the user to easily connect to the remote and choose a premade control profile for the remote. Time permitting, a profile editor will be written, but this is not an immediate priority. Using a text editor a profile can be created that enables the following features to be accessed on the remote control:

1. Bind keys on the remote to keyboard events and mouse events.
2. Bind gyroscope readings to mouse movements with a specified gain (sensitivity) or other input events with a specified threshold.
3. Enable the timer feature. This can either be a simple clock or a stopwatch with start, stop, and reset buttons.
4. Enable the notes feature. Notes are selected from a menu in the GUI.
5. Enable the "buzzword" feature. A file containing buzzwords is selected in the GUI.

The current state of the driver application is that the old version is working very well with the current version of our microcontroller code. We have just started writing the new version, but we are anticipating a smooth development process as we have already figured out how to communicate with the microcontroller and perform most operations. We simply need to write a better program. Only a very small number of changes to the microcontroller code will be required to accommodate the new driver application. No flowchart is provided for the driver as the code is event-driven and object-oriented and thus not very transferable to a linear flow chart. A UML diagram showing all the classes and their interactions will be available in our final report.