Design Iterations

Software

The general methodology behind the code stayed relatively consistent throughout the design iterations, but several elements underwent changes. For example, between software versions 1.0 and 2.0, the team received the ST7565 LCD screen, which we used to replace the Arduino LCD shield in order to present a more appealing user interface to the potential users. So version 2.0 is essentially the same implementation of the code as version 1.0, but it includes the capability of interfacing with the more complex screen. To achieve this, an open-source user made Arduino library was utilized in order to create menu screens on the LCD screen. Furthermore, several methods were added to the code in order to generate menu screens.

However, this user-made library had several downsides. It was impossible to create more intuitive menus that better fit the desired functionality within the framework of this library. Furthermore, a glitch in the library caused the second menu item screen to appear first, which was undesirable. Additionally, cursor movements as a result of button presses was poorly coded in his library, causing button movements that were very “jumpy”, making it difficult to accurately navigate the menu screens. Most importantly, the void loop called an “update” function during every cycle that re-wrote the LCD screen output. Writing to this LCD screen is very time consuming, and therefore the speed at which the frequency pins could be modulated was limited to less than 16 Hz due to the loop cycling through at the slow rate of 16 Hz. In order to bypass this roadblock, a menu item was implemented that “escaped” from the void loop, and continually outputted the frequency and amplitude settings until told to stop. While this increased the maximum modulation speed of the frequency pins, the output could only occur on this screen, which was un-intuitive and not user friendly.

To overcome this, for version 3.0, the user-made library was abandoned. Instead, customized menu screens were written to better fit the functionality of the MARK 1 device. A short delay was added after the button presses so that the cursor movements in response to button presses were smooth, and so that one button press yielded one cursor movement. Furthermore, if statements were employed such that the LCD screen output would only be updated if a button was pressed. This drastically reduced the amount of time necessary to cycle through the void loop when no buttons were pressed, which allowed the frequency to be accurately modulated at frequencies greater than 500 Hz.

The final change made to the code was to the implementation of the pin changes to create frequency modulation of the H Bridge. Initially, an ‘if’ statement checked to see if the current time was within a certain tolerance of the time at which the current frequency dictated that a change should occur. If so, then the pin states were switched. This method was prone to errors though: The tolerance limited the speed at which the frequency pins could be modulated. The frequency could be modulated no faster than the tolerance level. Furthermore, the tolerance had to be precisely calibrated so that pins would switch at least once but no more than once each time that it was supposed to. The tolerance required to do this, however; is frequency dependent, and thus this method resulted in a frequency modulation highly prone to “missing” a change in frequency. Because accurate frequency modulation was vital to our device working properly, this error was unacceptable.

To rectify this problem, a function was written that defined a variable as sin(Frequency\*time)/abs(sin(Frequency\*time)). That is, this variable is a square wave, oscillating between -1 and 1 at the current frequency. This function then checks to see if the sign of this variable is the same as it was last cycle: if not, then this indicates that the frequency pins’ states must be changed. This method proved to be both much more robust as well as faster than the previous method.

Finally, it was determined that a non-trivial amount of time was needed to allow the transistors on the “open” half of the H Bridge to close before it was safe to open the other side. If the other side of the H Bridge was opened during this time, then “shoot through” currents would flow through the short circuit created by the opening transistors and the not-yet-closed transistors. This resulted in massive spikes of current through the transistors, causing them to heat up quickly. To avoid this, a delay was implemented in the frequency function such that the opening side of the H-Bridge opened a little after the closing side closed. This delay successfully limited the shoot through currents from occurring.