Gautier Dagan
James Willis
EECS395- Tang. Interaction
Morse Code interpreter

Introduction

For Our creative Stretch project, we created a device with the Arduino that will allow a user to type Morse code using a button, which then prints the Morse code message out to the serial port reader in real time. Below we detail how we went about creating this project.

Prior Research

Before this project we did not know too much about Morse code, or Arduino. Armed with some basic knowledge about C, We began by looking at some of the example embedded C code provided in the Arduino IDE. The first example we looked at was Blink, the sensor version of Hello World. After getting an idea about the structure of an Arduino program, We looked at BlinkWithoutDelay- we needed a way to measure the time the button had been held without blocking. Their method of use of the milli() function is an integral element of our code. Finally we looked at the SoftwareSerialExample. This showed us how to interface with and print to the Serial Port, and display the text in the Arduino serial port reader.

In addition to researching techniques for writing Arduino programs, we also looked into the syntax and conventions of Morse code. We used the chart here (Wikipedia's infographic of Morse Codes and conventions).

An Iterative Process

Our approach to this project was to iteratively build out a more and more complete Morse Code machine. The first thing we did was to build the physical circuit. We hooked an LED up to an output pin on the Arduino, and did the same with a button and an input pin (input vs output is actually handled in the code). Once we had the physical layer in place we began coding.

The first iteration of the code was simple. We created a program that would light up the LED when the button was pressed. We thought the LED feedback was important, not only for debugging of any potential hardware issues, but also to provide feedback to the user that their button presses were doing something, like the click a traditional Morse Code unit has on each press.

Once we successfully completed that, we revised the code such that it printed to the Serial Port. This was relatively simple- when the button was seen as pressed we printed a message and toggled a Boolean indicating such. When the button was release the Boolean was return to 0. This was used to prevent printing occurring on

each iteration of the loop function. The next step was to distinguish between a dot and a dash. To do this we stored a global variable containing the time the button was pressed. When the button is then released the difference in time is checked and one of three results occurs- The press was shorter than a dot and nothing is printed, the press length was between the time for a dot and dash and a dot is printed, or the press was longer than the time for a dash and a dash is printed. At this point we almost had a working Morse code generator- we just need to add in spaces.

To add in spaces we had a similar process to checking for a dot vs a dash. When the button is released a bool is toggled up. When this Boolean is high the loop checks for whether its been long enough for a space to be printed, at which point it will print a space and toggle a Boolean indicating that a space has been printed since the last button press. This prevents infinite spaces printing. The space Boolean is then toggled down the next time the button is pressed. A similar solution was implemented to tell when a character starts and ends. At this point in the process we had a working Morse code writer; all that remained was to handle the translation of a series of dots and dashes to letters.

To print characters we stored a series of dots and dashes as chars in an array using C's enumerations. When a dot or dash is entered its placed in the next spot in the array (mod 4). When the character-separating amount of time has passed between button presses we check if the pattern in the array matches any of the letters of the Morse code alphabet. If so we print that letter to the Serial Port and empty the array. Otherwise we just empty the array