6.111 Final Project Design Proposal

Project team: Christopher Stephenson

Abstract:

The goal of this project is to produce a system that will decode Morse Code signals from a possibly noisy audio source, and display the decoded text on a screen. The system might also produce Morse Code signals via input from a keyboard.

The core of the project will be a Morse Code decoder that will take as its input a series of pulses, and a screen driver that will render the decoded text on a screen. Possible extensions to the project include: A Morse code detector which can 'guess' whether an arbitrary signal contains a Morse Code message, filters to isolate a Morse Code pulses from that signal and a Morse Code generator that produces a series of pulses from input via a keyboard.

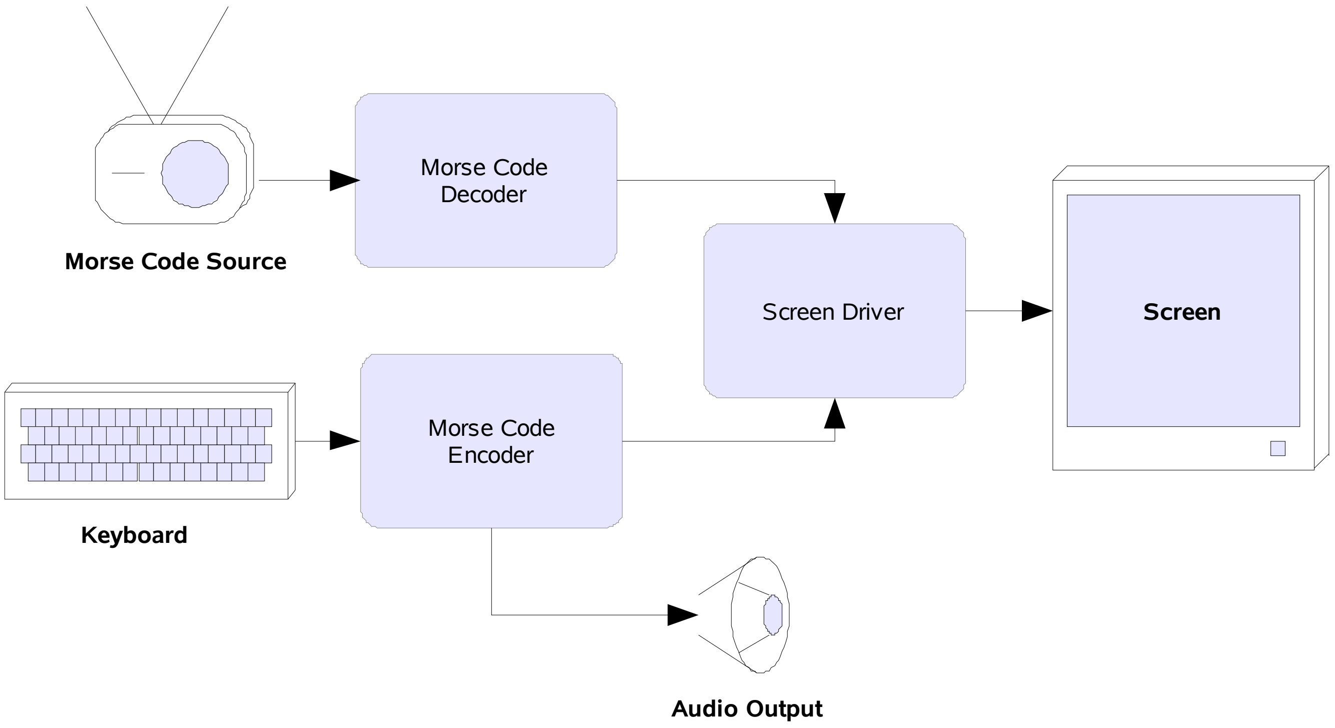
Proposal:

The aim of the project is to produce a digital system capable of decoding Morse Code, and displaying the output as text on a screen. As well as that, it should be possible to produce “perfect” Morse Code by outputting Morse Code encoded letters from a keyboard.

The project is neatly divided into three parts:

1. Morse Code Decoder
2. Morse Code Encoder
3. Screen Driver

These three parts will work together to produce a system that fulfills the above description as shown in the block diagram below:



A Block Diagram of the Basic System

Morse Code Decoder

The decoder development can itself be divided into three parts:

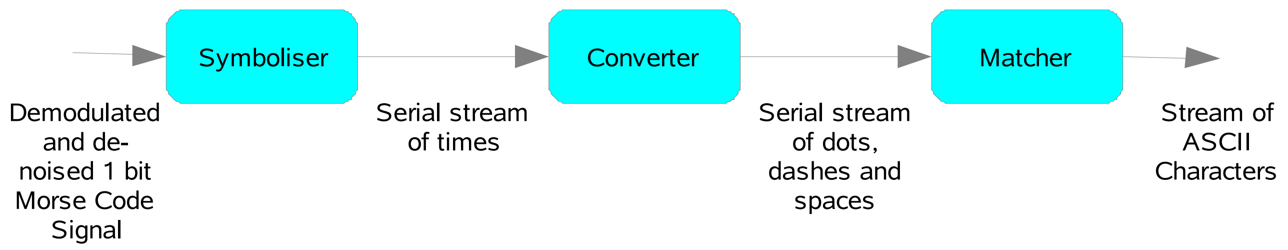
1. Simple Decoder
   Takes a "Perfect" Morse signal, and decodes it, producing a stream of ASCII characters

2. Morse Frequency Selector
   Given an arbitrary audio signal with Morse Code content (and possibly other noise) Identify the Morse Code signal, extract it and decode it.

3. Robust Decoder
   Improve the decoder to be able to handle Morse Code that is not perfect i.e. Human tapped Morse Code

The Frequency Scanner follows on logically from the decoder as Morse code is usually sent as a series of beeps over a radio channel. If the robust decoder is added, the system should be able decode Morse Code that comes from arbitrary sources such as a short wave radio where operators might still tap out Morse by hand.

The basic design for the decoder pipeline will be as follows:


Symboliser
Converter
Matcher
Demodulated and de-noised 1 bit Morse Code Signal
Serial stream of times
Serial stream of dots, dashes and spaces
Stream of ASCII Characters

Pipeline for the basic decoder

The Symboliser will convert an incoming 1 bit signal that switches on an off, into a serial stream of times – each time representing the length of time that the signal was either in the on or off state. These times are the interpreted as either dots, dashes, inter character spaces or inter word spaces by the converter module. This will probably be done by finding the "clock" of the Morse Code signal – i.e. The rate at which the code is being sent. The stream of symbols is then sent to the matcher which will output an ASCII character by matching the dots and dashes that arrive at its input.

The Frequency Selector would be perpended to the pipe line, and would take as its input an audio frequency signal. It would produce its output by examining the signal in the frequency domain, and then extract the signal of interest to be passed to the Symboliser. The exact details of the internals of this module are not determined.

Increasing the robustness of the decoder would probably mean extending the converter module t be able to handle Morse Code sent with variable speed and less strict timing.

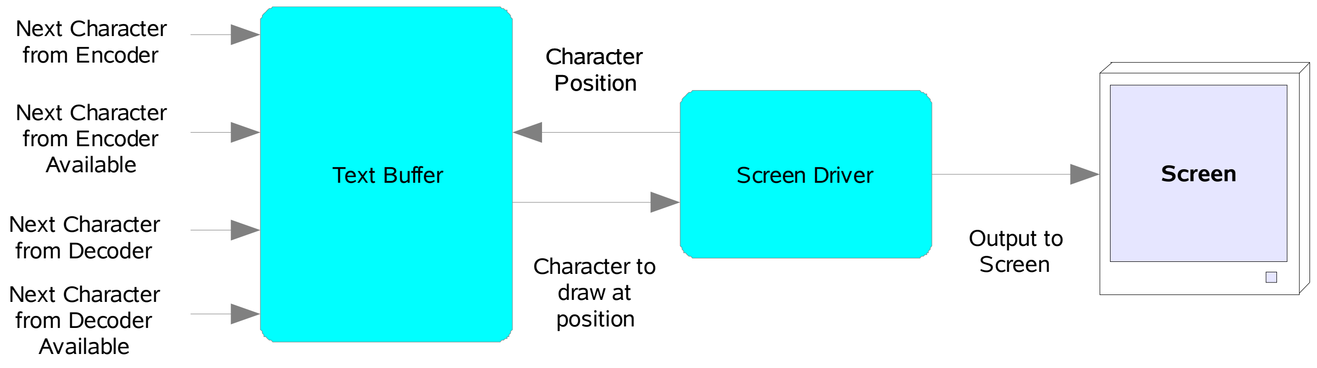
Morse Code Encoder

This module takes as its input a stream of ASCII characters. It will buffer them, and then produce a Morse Code representation of the characters. In addition to that, as the Morse for the character is being outputted, it will simultaneously, send that character for display on the screen.

Screen Driver

The screen driver is in actually fact not required to make this project work. It would be perfectly possible to send any output to the serial port on the lab kit and then view the results by connecting the lab kit to a computer and using software such as HyperTerminal to view the output. However, writing a custom screen driver for this project adds interest and makes the project more self contained – i.e. It could feasibly be adapted for use with any monitor including televisions, thus making it more portable.

The screen driver itself will be divided into two parts: A Text Buffer and the actual screen driver. This is summarized by the block diagram below:


Next Character from Encoder
Next Character from Encoder Available
Next Character from Decoder
Next Character from Decoder Available
Text Buffer
Character Position
Character to draw at position
Screen Driver
Output to Screen
Screen

Basic Block Diagram for Screen Driver

The Text Buffer is responsible for managing the text that will be displayed on the screen. This means that it will store the text and provide the character at a specified position to the screen driver. It will also be required to provide scrolling abilities – such that when the the last line of text is filled, the text is scrolled up the screen to make room for the next line of text.

The implementation of the Text Buffer will be based around a circular buffer. This makes scrolling easy by removing the need to recopy data into a regular rectangular buffer.

The Screen Driver Module simply outputs what it is told to the screen. It deals with the various timing issues for vertical retrace etc. As an extension, it could also support "Smooth scrolling"; that is scroll the contents of the screen up slowly when a new line is needed on the screen.

Extensions

If there is time, the following additional features could be implemented.

- True Full Duplex support

  The system should be able to "transmit and receive" on the same analogue channel at the same time. It must do this without trying to decode the Morse code that it has itself generated. As well as that, if it has "transmitted" over the top of an incoming signal, it should still be able to decode the incoming signal. This is not useful for standard short wave radios which are half duplex, but would be useful if the system was transmitting on a speaker and receiving on a microphone.

- Fist Identification

  During the second world war, radio operators were able to tell the identity of radio operators sending encrypted signals transmitted in Morse from the operators "Fist." This extra information proved vital in helping to crack the various codes being used.
  The system should be extended to try and guess the identity of an operator based on his fist. The guessed identity should be displayed alongside the text being received.