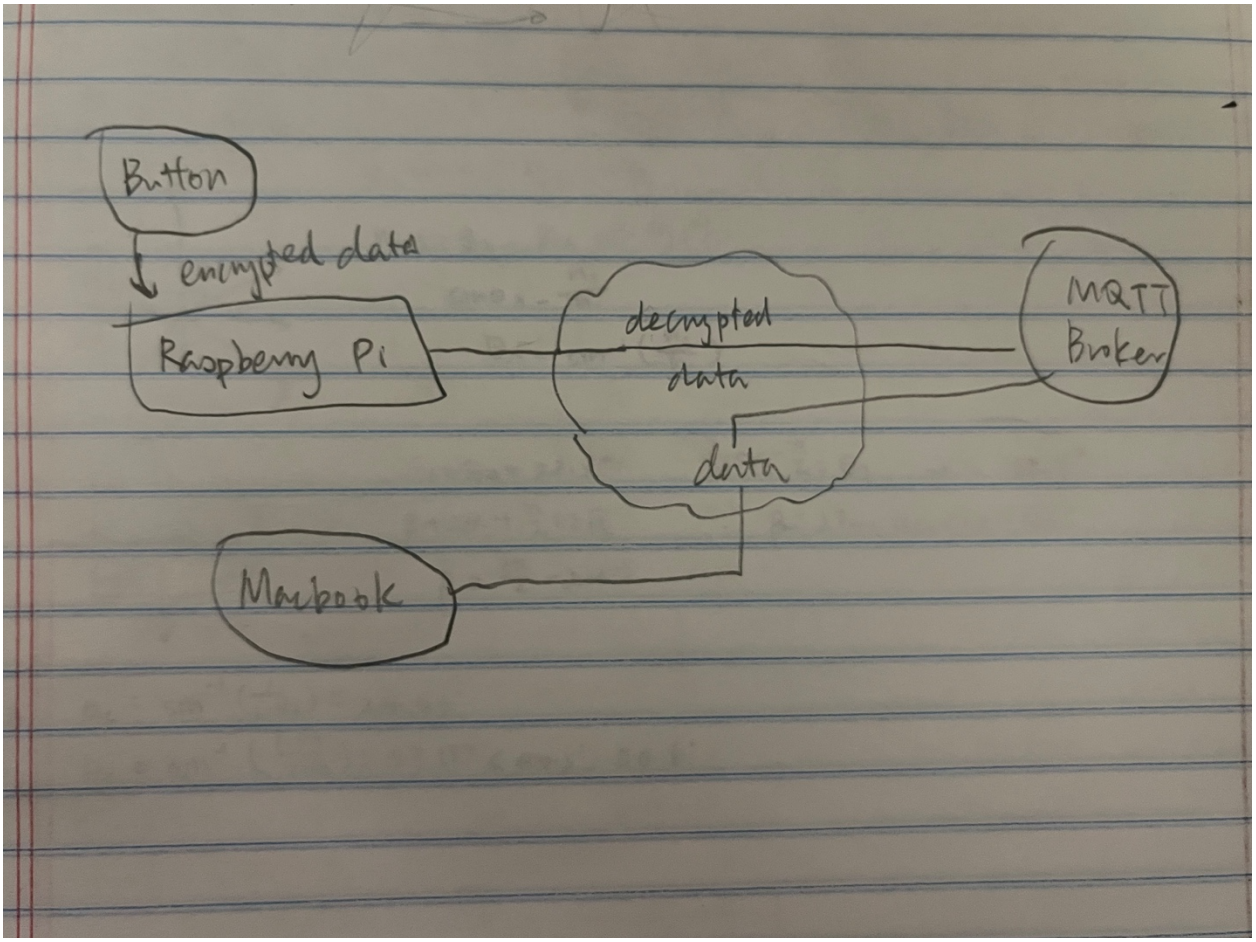
Kevin Luo
Prof. Bhaskar Krishnamachari
EE250L
5 May 2023

Final Project Writeup

Functionality:

For the final project, I coded a simple Morse Code decoder that uses a GrovePi button as input. A short press (0.2-0.4 seconds) represents a dot, a long press (0.6 seconds and above) represents a dash, and any interval longer than 0.8 second in between two signals represent a space that separates the letter before and after. The program allows the user 10 seconds to input any legible morse code. Once the 10 seconds elapse, the program will decode the input sequence and output the decoded message. The decoded message is then transmitted to my PC and displayed on a simple web front. This project is a prototype of what could be used remotely for encrypted communications between two nodes.

Block Diagram:



Design:

The GrovePi buttons run on an analog port that either produces a high or low input depending on if the button has been pressed or not. To process these inputs accordingly, and to

judge if the button press meets the time threshold that we use to differentiate between a dot and a dash, I choose to read the button input every 0.2 seconds and write a function that determines the duration of each press by counting up the number of consecutive 1s and 0s. Any consecutive occurrence of 1s that exceed three counts as a dash and a singular or double occurrence of 1s count as a dot. To separate two encoded letters, I also count up the number of consecutive 0s, where four or more 0s signify a space. After concatenating a string composed only of dots, dashes, and spaces, I use a Morse Code dictionary and split the string into individual letters. I then run a for loop through the string, comparing each letter with the Morse Code dictionary, and eventually obtain a decoded message. After utilizing the data processing method specified above, the decoded message is then published to a mosquito broker on port 1883 using MQTT protocol under the topic "button/morse_code", which a separate program running on my PC is subscribed to. The MQTT program on my PC not only receives the decoded message via the connection, but also runs a Flask program that updates an index.html file I coded every time a new message is delivered.

Reflection:

Originally, I planned to use audio input for morse code, where I would then either filter or transform the audio to categorize each audio spike into dots or dashes. The idea didn't work for several reasons. First, it was hard to find a clear indicator that can consistently differentiate between the two different audio input (dots or dashes). Although dashes signify a longer pause between two spikes, the duration, frequency, and amplitude of the two prolonged spikes are still in large indistinguishable compared to a shorter spike for dots. A possible method to make the input distinguishable is by making the dots quieter and the dashes louder, so that an amplitude filter could be used to categorize different signals. The problem with this approach is that not only is it hard to find an accurate threshold value, it is also hard to correspond each audio spike to only one output since the duration of each spike varies, making it impossible to find a sampling window that works for all. I therefore wasn't able to find a viable method to implement an audio morse code decoder and have to resort to an input method that's easier to manipulate and decrypt. However, the button input method is a lot less powerful compared to audio due to the slow speed it's able to intake any signal. A series of three coded letters take approximately 10 seconds to be registered, which isn't an ideal speed if the user wishes to encode an entire sentence. If a viable solution could be implemented for the sound input method, using sound would be a much better method for morse code input.