Final Report: Light-based Morse code transmitter and receiver
Members: Kazem Ayatollahi and Christopher Palacios

Phase 0: Circuit and Schematics

The required circuit diagram is as follows:

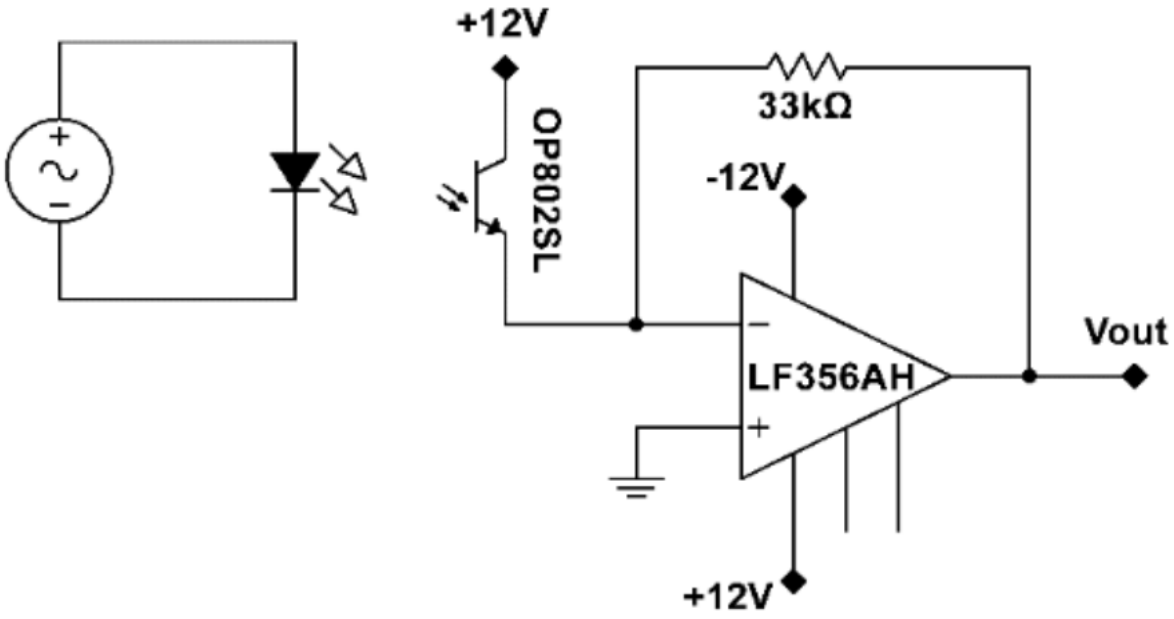


Figure 1: Left circuit sets up an LED. Circuit on the right sets up a phototransistor.

Note that we have used ADS (Analog Digital Discovery studio) as our scope and data acquisition tool.

The flow of the data and operations of the receiver and transmitter are explained using the following schematic:

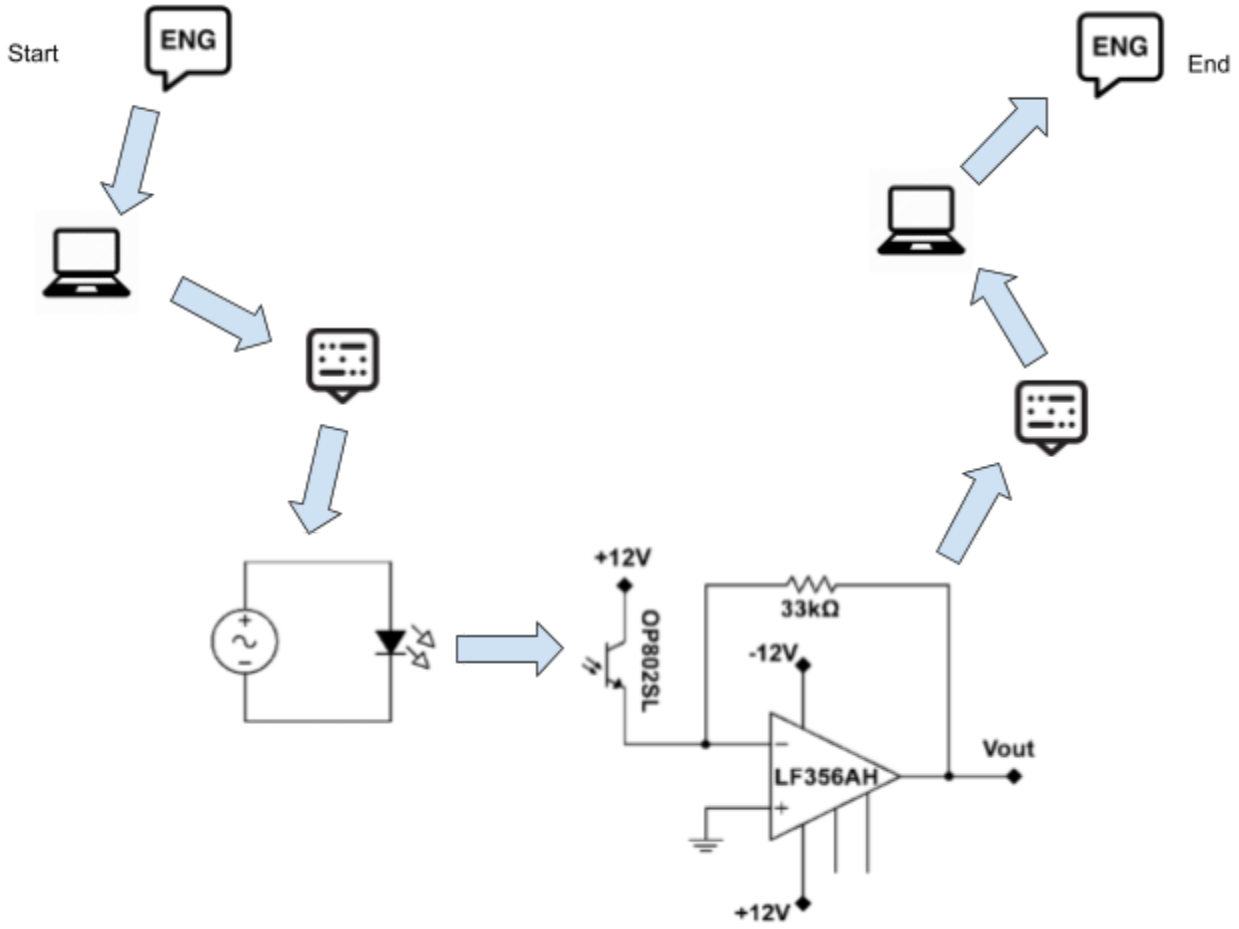


Figure 2: Diagram explaining the sequence steps taken by the user and Labview program

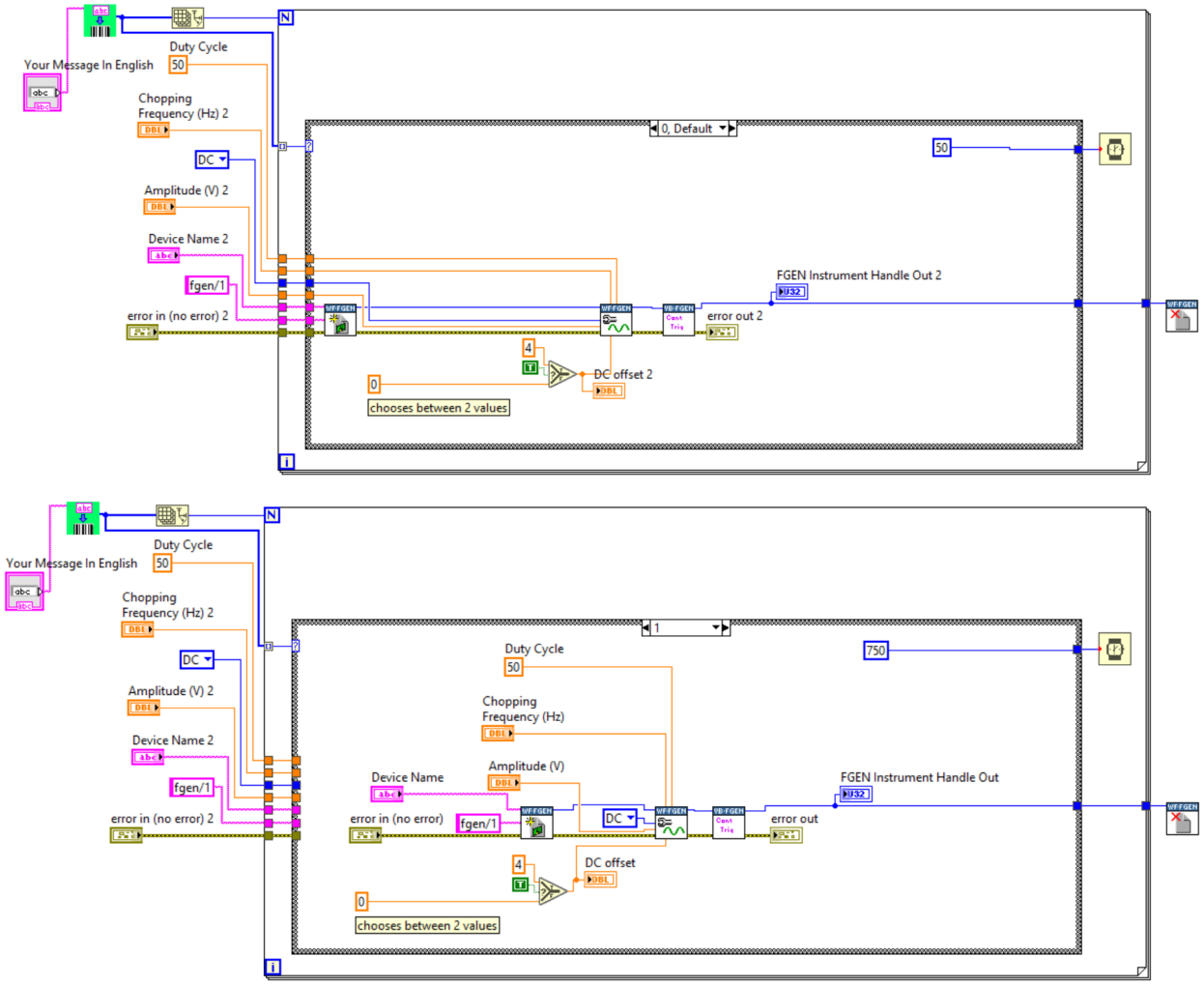
Overview:
User starts by inputting a customized message. This message is converted from a string of characters to an array of characters. This allows our Labview program to convert each character to its corresponding Morse code. This code is then converted into an array of time lengths that will set the LED on and off to be received by a phototransistor (OP802SL). The signal received is then converted back to a Morse code array based on those specified time lengths from the LED. Finally, the Morse code array is converted to its corresponding characters and the user running the receiver program will view an output text that the starting user transmitted.

Phase 1: Transmitter

Front Panel

Figure 3: User interface that collects the input message (string)

The final transmitter with 4 different cases:



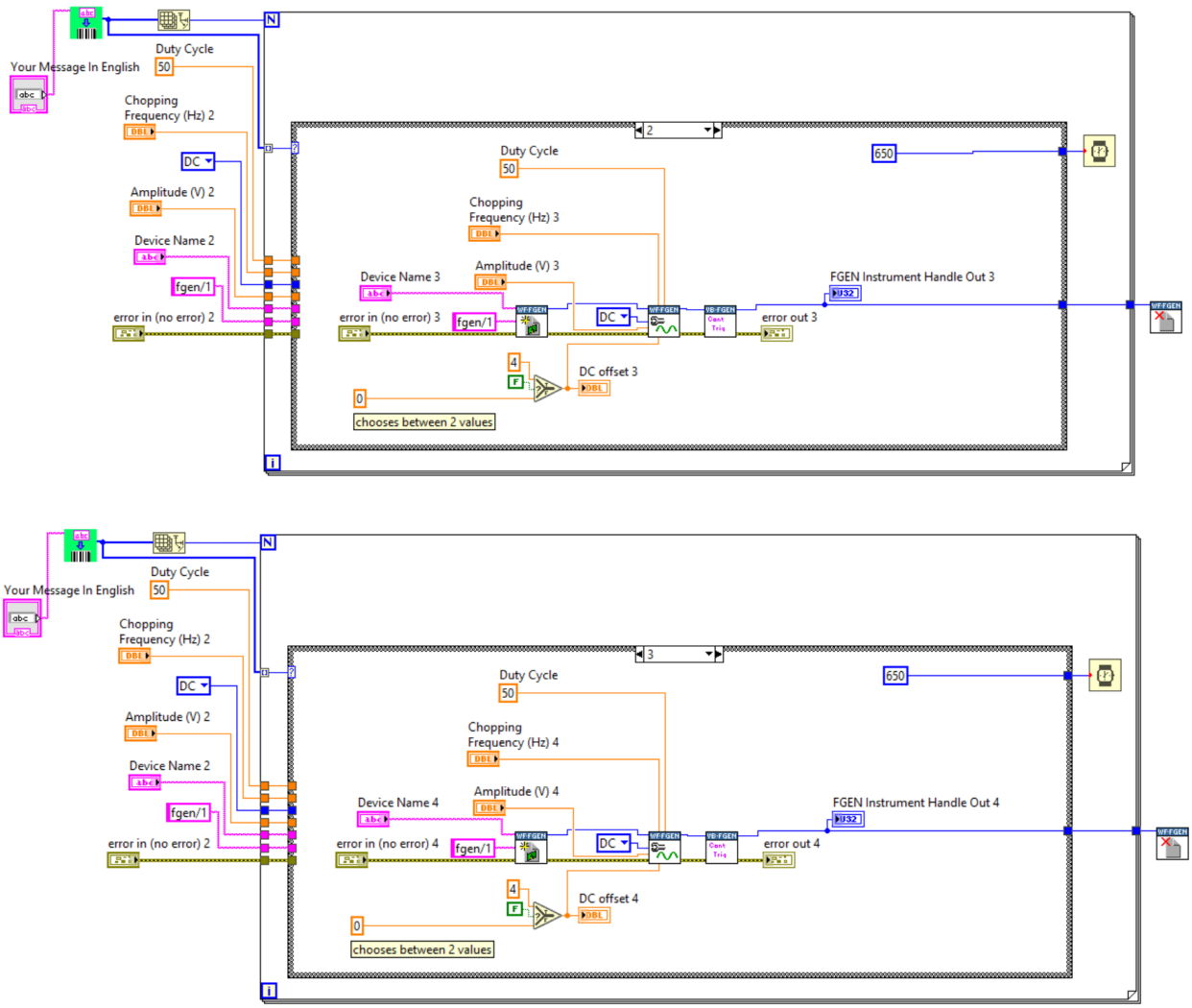


Figure 4: The first (top), second, third, and fourth (last) diagram show the case in which a zero, one, two, or three is passed from the English_to_Morse.vi (Figure 5), respectively.

Figure 4 displays the timing conventions given to each zero, one, two, or three digit. Dits (0's) are characterized by a 50ms LED flash and 750ms flash for dahs (1's). Spaces between characters are characterized by 650ms pauses for each 2 & 3 (see English_to_Morse.vi below for more detail). A 3 second wait time from when the phototransistor begins collecting data is required to begin reading the LED signal (more on that in the receiver section).

English_to_Morse.vi (top-left green sub vi from Figure 4):

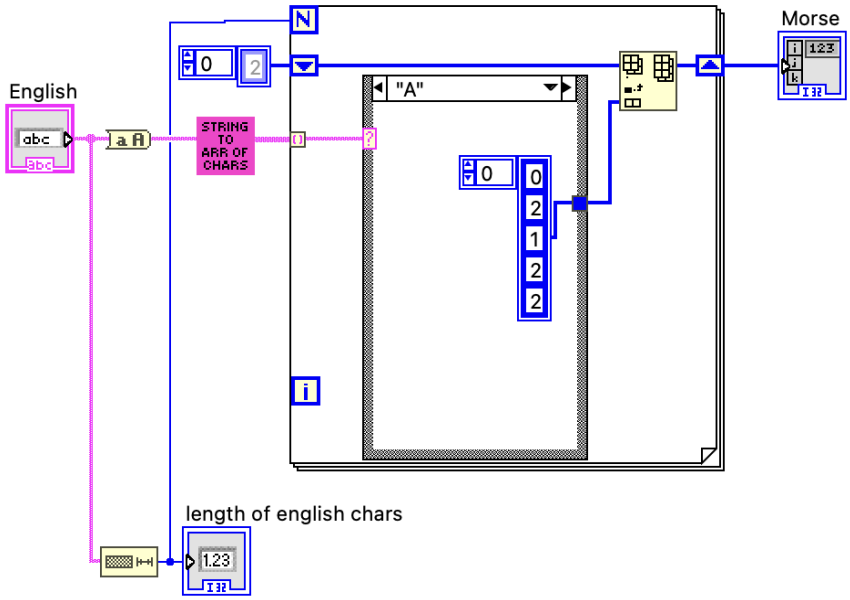


Figure 5: Converts string of text to an array of characters (see String_to_array.vi below) and finally into an array of integers based on each character's Morse code conversion

All the alphabet letters are included in the case structure. There are two times of light-off wait considered in our program.

a) Digit 2: turn LED off for 650 ms
b) Digit 3: turn LED off for 650 ms

Using these building blocks we have created all the necessary wait times.
There are 3 situations where we need to turn off the light:

1) Between dits (0's) and dahs (1's): implemented by a single digit 2
2) Between letters: implemented by two consecutive 2 digits (22)
3) Between words: Implemented by two consecutive codes 2 followed by a 3 digit (223). Note that the space character is considered a special character and is going to represent code 3 in the 223 sequence.

As an example, the letter B in Morse corresponds to (1000) which corresponds to 120202022 in our Labview program. If a space was entered at the end of the letter B, the code turns out to be 1202020223.

String_to_Array.vi (pink sub vi from Figure 5): 

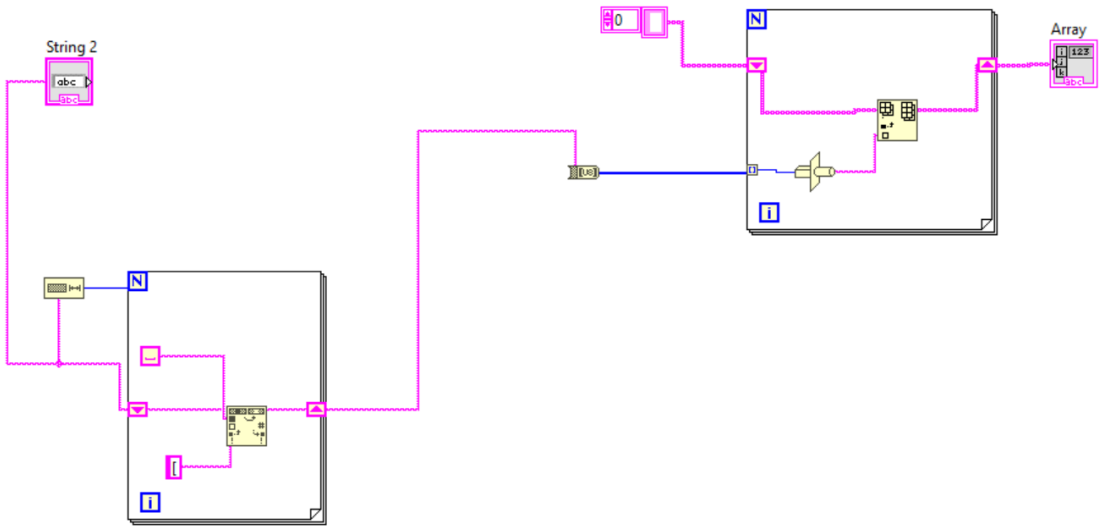


Figure 6: Converts an input string to an array of characters

Turns a string control to an array of characters. This will be necessary to convert each character to its corresponding Morse code.

Phase 2: Receiver

Front Panel:

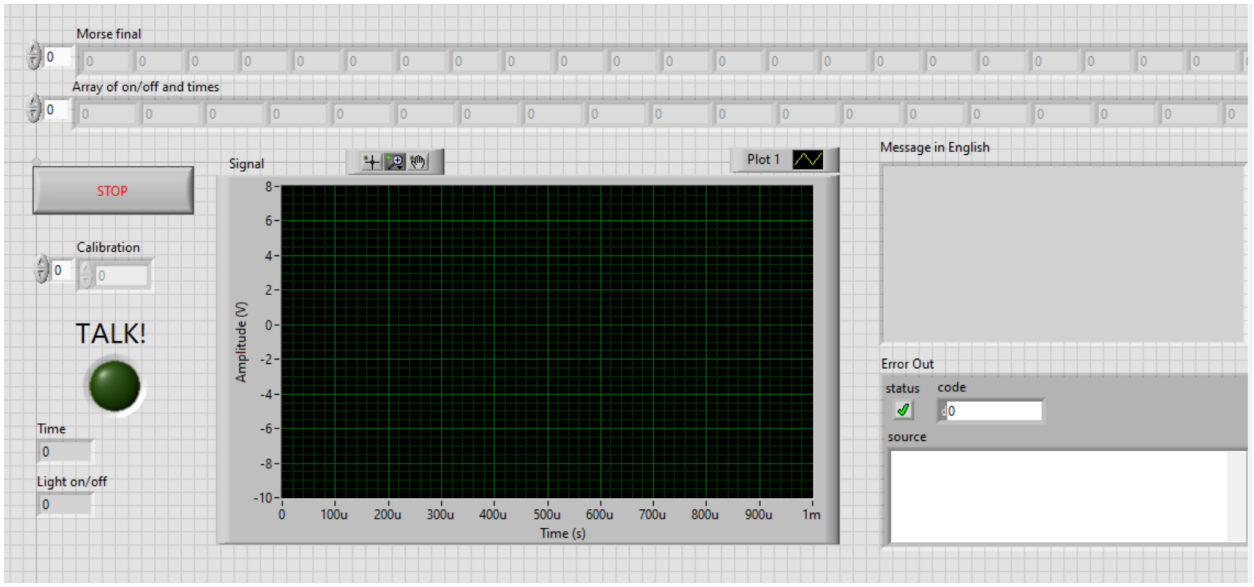


Figure 7: User interface for signal to string conversion output

The main receiver vi:

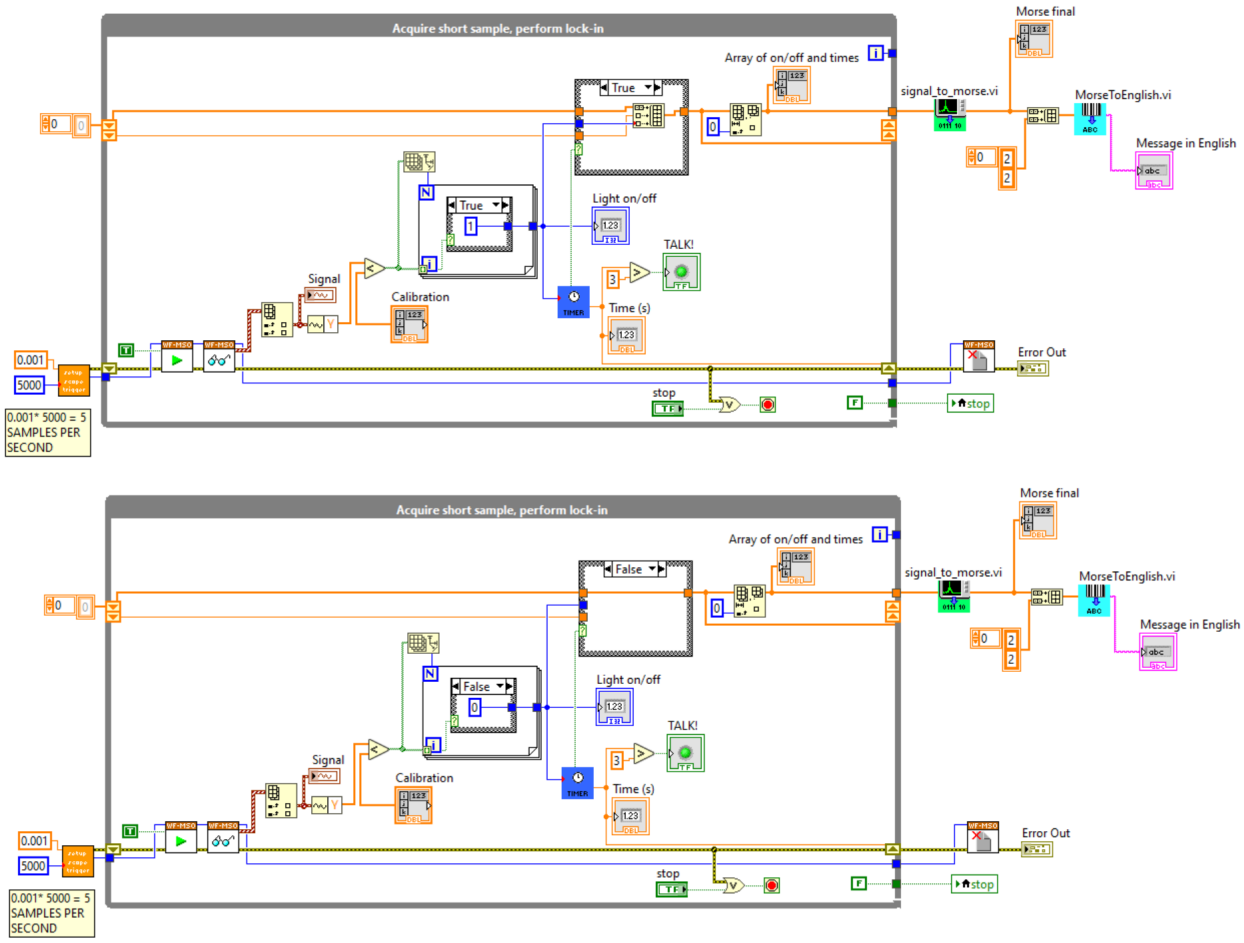


Figure 8: Initializes ADS scope to acquire data using the phototransistor

The receiver vi stores the output of the LED as 1 or 0 depending on if it's on or off, respectively. After the user starts acquiring data from the receiver, a 3 second delay is necessary before the LED begins transmitting. This ensures the initial off condition of the LED is not counted as a character. Each 1 and 0 is followed by the duration of which that condition was true. For example, consider the array 1, 0.5, 0, 0.5, 1, 2.6. This would mean the LED was on for 0.5 seconds and then off for 0.5 seconds, and then on again for 2.6 seconds.

The receiver vi (Figure 8) consists of 4 sub vi's:
1) Setup Scope:

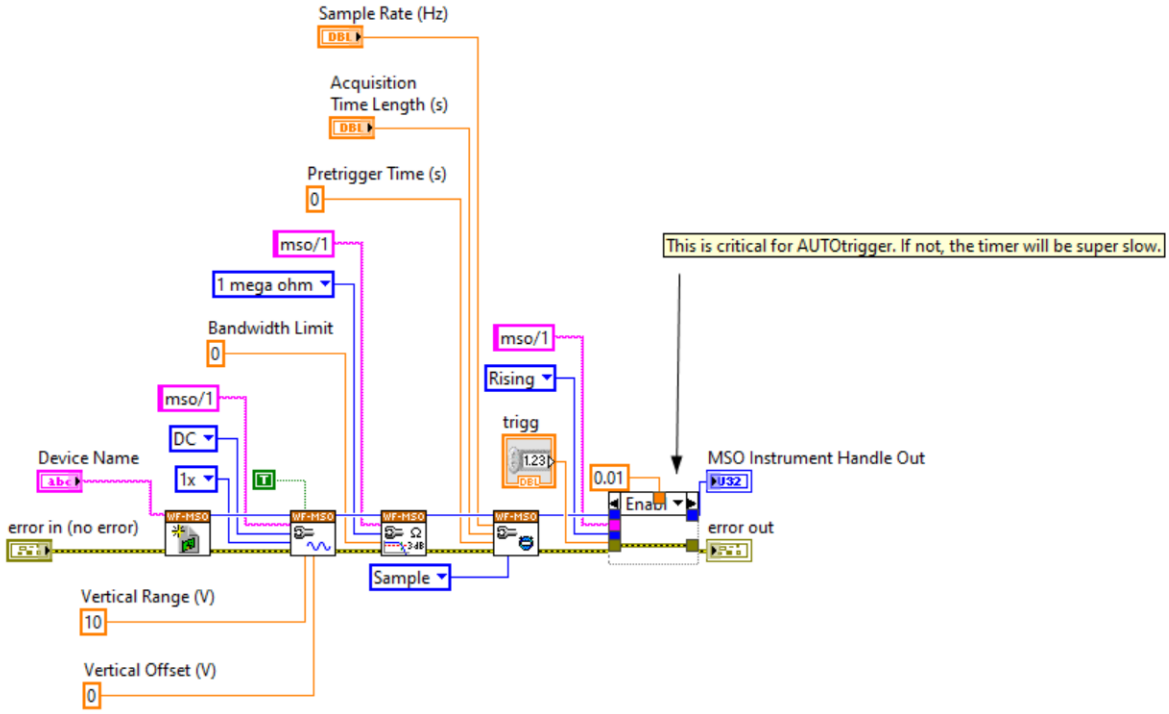


Figure 9: Initializing the ADS scope and trigger

The setup scope vi initializes the scope and trigger in order to start the data acquisition for the phototransistor. Note that the auto-trigger is enabled. This was accomplished by ensuring the trigger did not wait to receive irrelevant parameters. We obtained 5 samples/sec by setting an acquisition length of 0.001s and a sample rate of 5000Hz.

2) Timer:

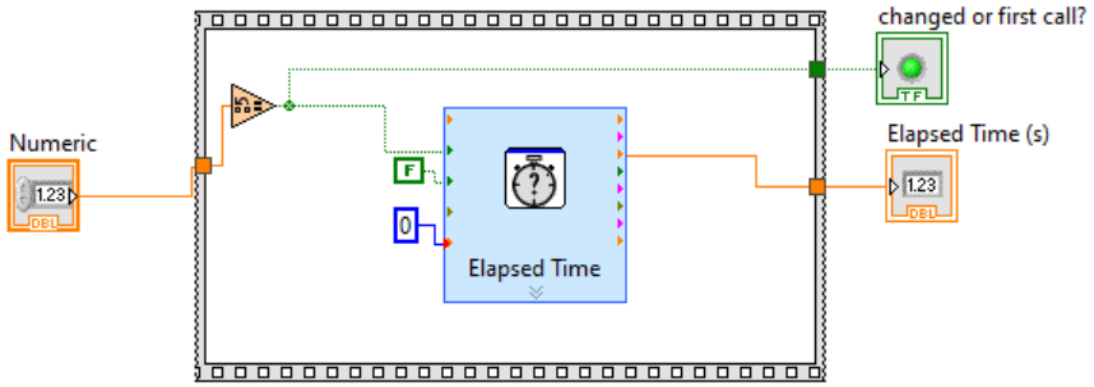
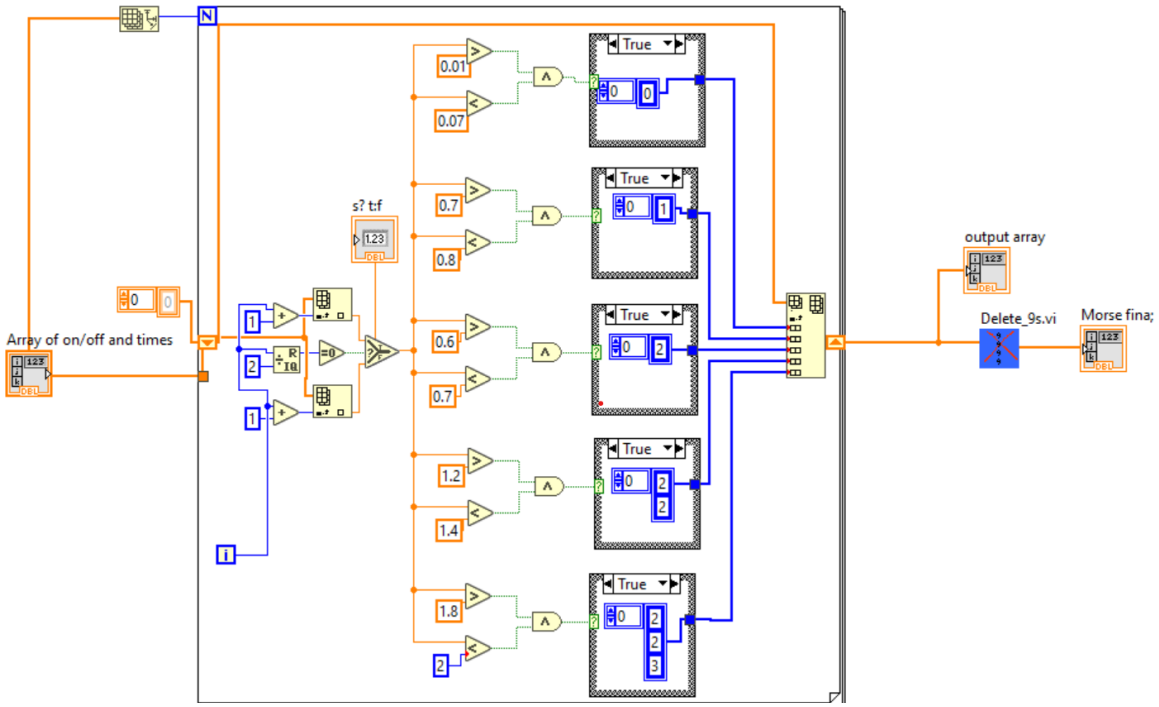


Figure 10: Measures the elapsed time from where you specify, and once the value of the input is changed, it rerecords the time measurement and returns the previous time.

3) Signal_To_Morse.vi



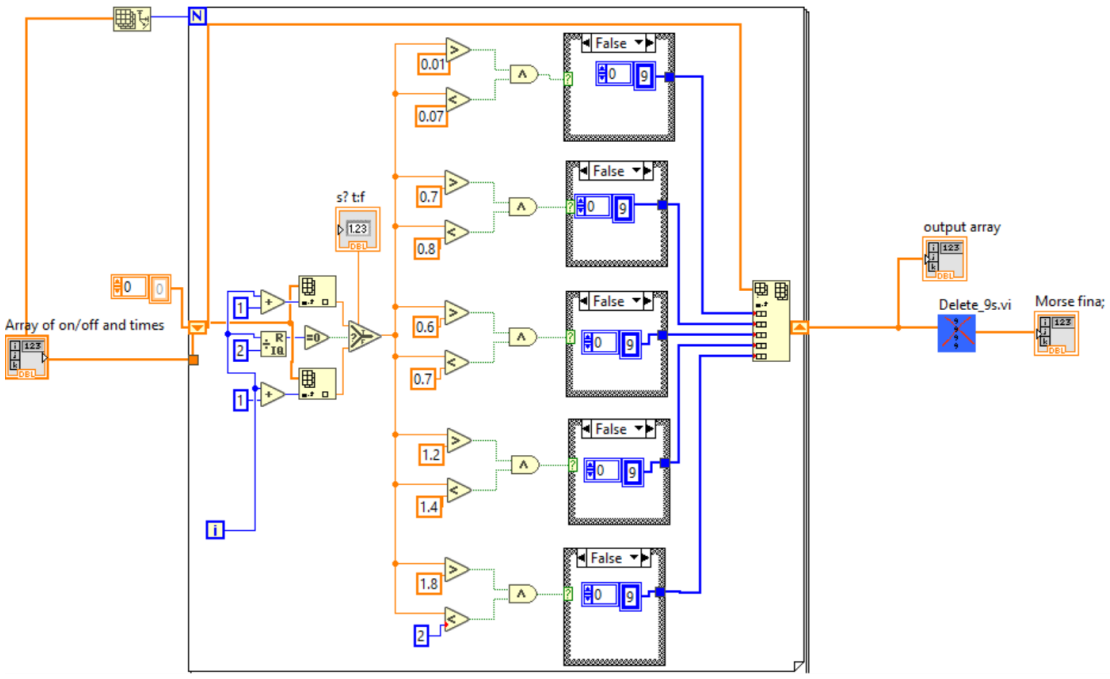


Figure 11: The input array of on and off times is converted to the original Morse code characterized by 0, 1, 2, and 3

The Signal_to_Morse.vi extracts the condition and time array, element by element, and decided which condition occurred based on the time specifications from Figure 4. If the case were not met (False), a 9 (arbitrarily chosen) is going to concatenate to the array which will eventually get removed. The Signal_to_Morse.vi includes the following sub vi:

Delete_9s.vi:

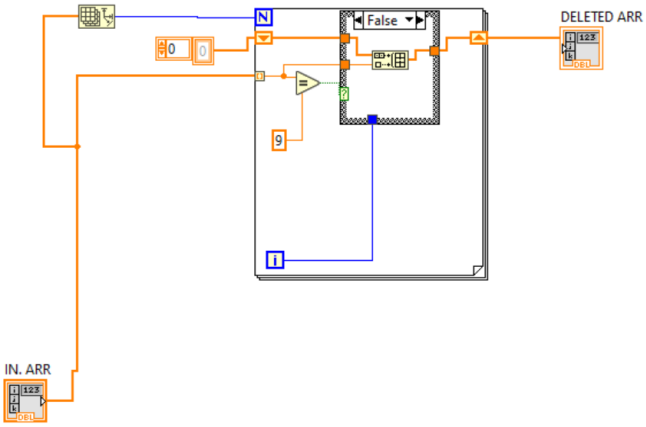


Figure 12: Deletes all the occurrences of the given element from the array of doubles. In our case, it deletes all occurrences of 9's from the array.

4) Morse_To_English.vi:

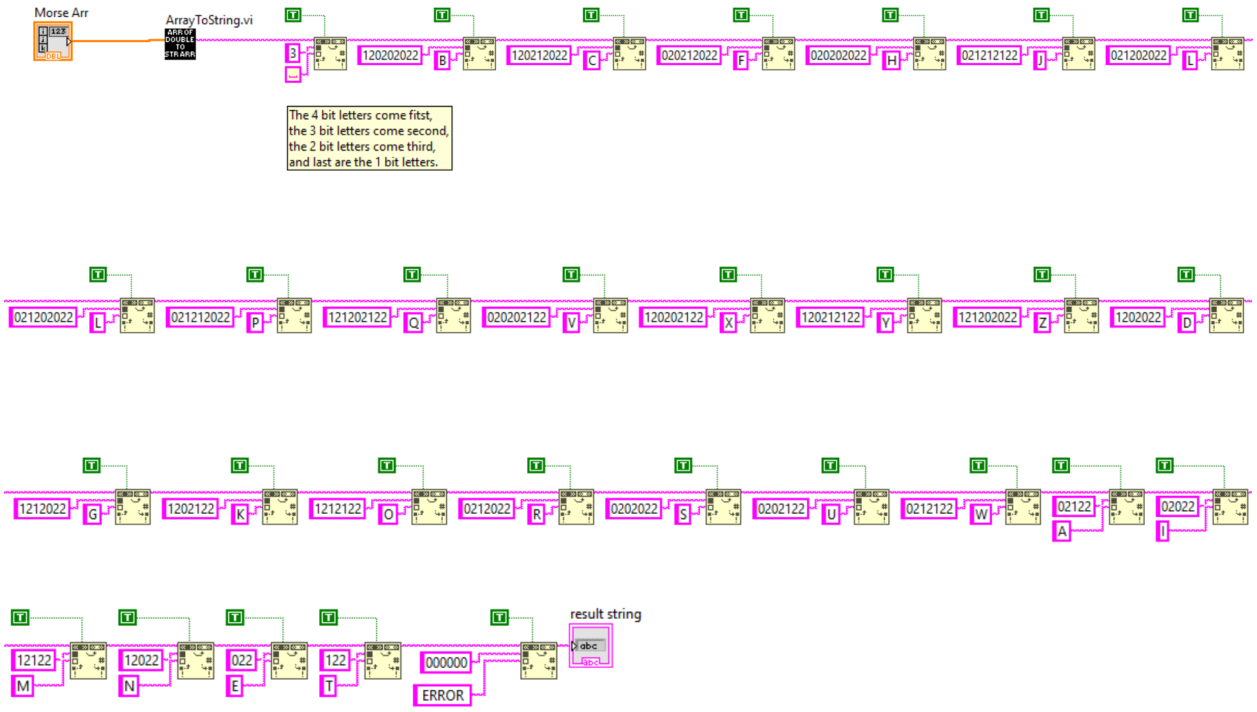


Figure 13: An array of Morse code corresponding to each letter is converted to an array of characters and saved as a string

The Morse_To_English.vi converts the numerical Morse code for an individual letter to its corresponding alphabetical character. First, the 4-bit characters are going to get replaced (such as B and C). Second, the 3-bit characters are going to be replaced (like O and U). Next, the 2-bit characters (e.g. A and N). Finally the 1-bit characters (T and E). This specific implementation is critical, so the lower-bit characters will not end up eating the higher-bit character's dots and dashes. (as an example a B corresponds to 1000 but a T is 1 and an E is 0. Hence a B if misinterpreted could be translated to a TEEE instead of a simple B). The Morse_To_English.vi contains the following sub vi.

Array_To_String:

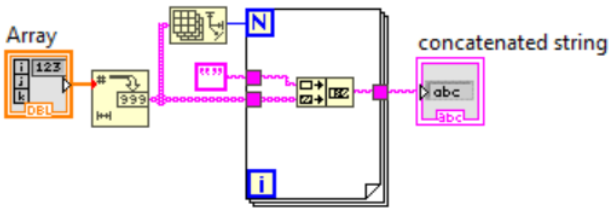


Figure 14: The input Morse code array is converted from an array of integers to a string of those same numbers

Since a convenient string-based built-in function was already available in the LabVIEW, we decided to transform our arrays of doubles into strings of numbers. This vi turns an array of doubles into a single string.

Final Remarks:
With the specified equipment and under normal room light conditions, we have tested to set apart the receiver and transmitter up to about 1.2 m. It seems that the distance can even increase significantly further as long as the phototransistor is able to collect a faint light signal. This could be further built upon by using transmitters and detectors outside of just visible light, allowing for a wide selection of communication methods.