Final Team Project
Morse Communicator

Due by Tuesday 3/21 @ 11:59PM

Submission: Upload all VIs and subVIs to Gradescope. Also submit a demo video to link: https://drive.google.com/drive/folders/1FEeXwAxbDNj3i25r4srD3LUdTE9XcqWo?usp=share_link

Morse Code Communication:

Morse code is a telecommunications method developed in the early 1800s. Using a series of dots (.) and dashes (-), it can encode the entire English alphabet, and can subsequently be used to communicate using the telegraph machine.
The rules used to distinguish between dots and dashes are:

1. The length of a dot is 1 time unit.
2. A dash is 3 time units.
3. The space between symbols (dots and dashes) of the same letter is 1 time unit.
4. The space between letters is 3 time units.
5. The space between words is 7 time units.

In this project, you will use an Arduino board, an LCD panel, an LDR resistor, and an LED to replicate a morse code communication transmitter and receiver protocol.

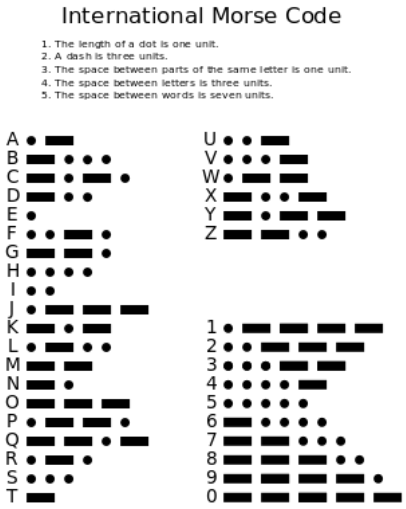


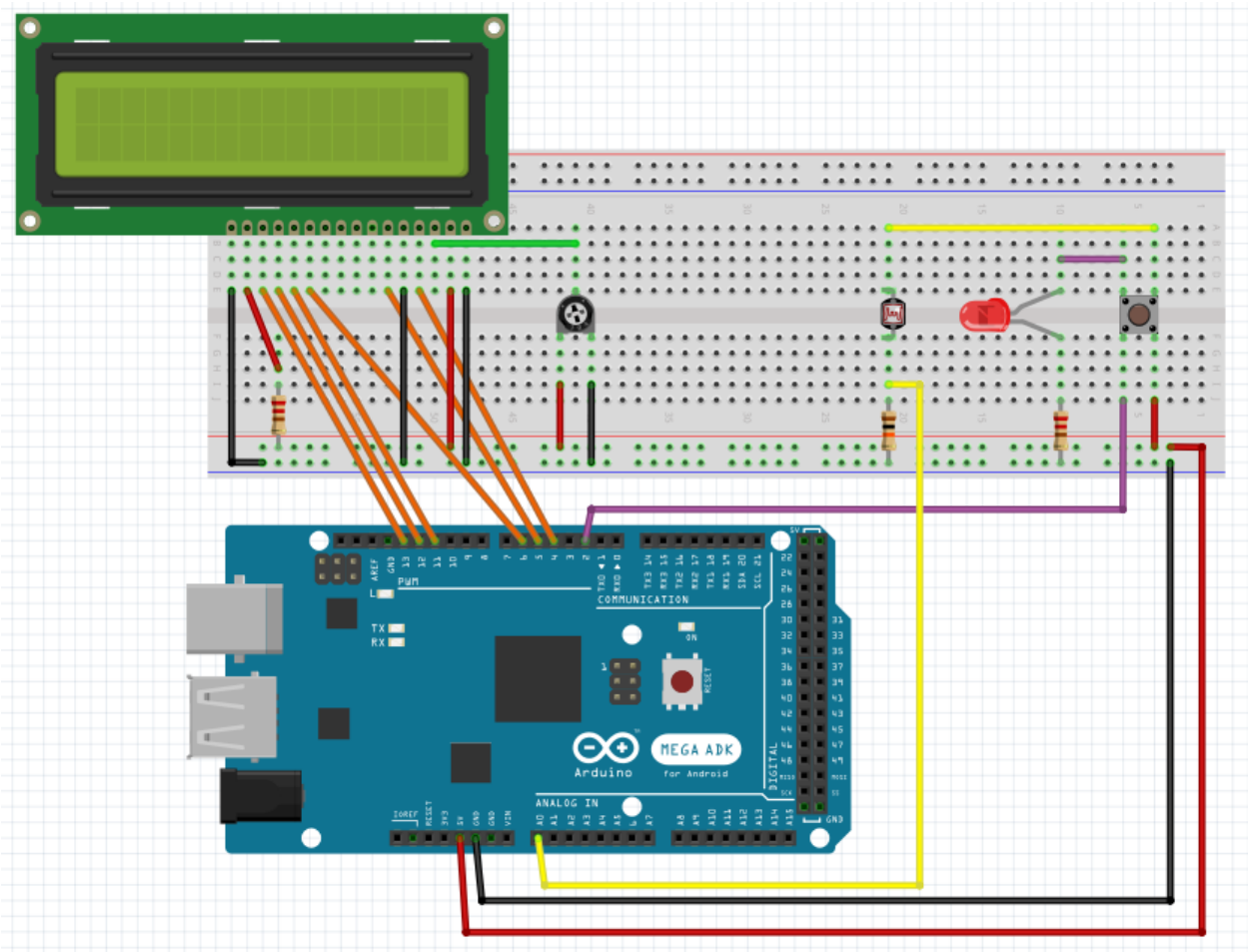
Figure 1: The international morse code

Challenge 0: Hardware Setup

The main hardware that will be used in this lab is an LED, LCD, and photoresistor. The LED will transmit morse code by flashing on and off. The Arduino will read the flashes transmitted by the LED using the photoresistor. Finally, the LCD will be used to display the received messages.

Components:

1. Arduino Uno
2. Photoresistor
3. Push Button
4. LED
5. $10\ k\Omega$ resistor (Photoresistor)
6. 2x $220\ \Omega$ resistor (LCD and LED)
7. LCD
8. Potentiometer



Challenge 1: Encoding and Decoding

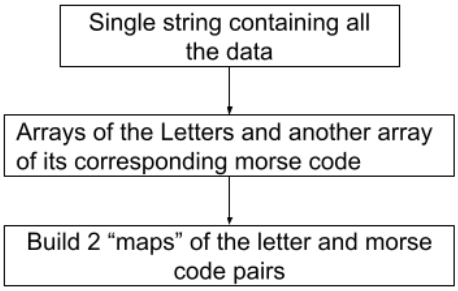
In this challenge, you will be learning how to read from the morse code lookup table txt file (morse.txt). You will be building two VIs - **morse code encoder** and **morse code decoder**. The two VIs will be responsible for converting text string to morse code strings and vice versa.

- **1.1: File Parser**
- **1.2: Morse Code Encoder**
- **1.3: Morse Code Decoder**

Comment your code as you go so others can understand!

1.1: File Parser

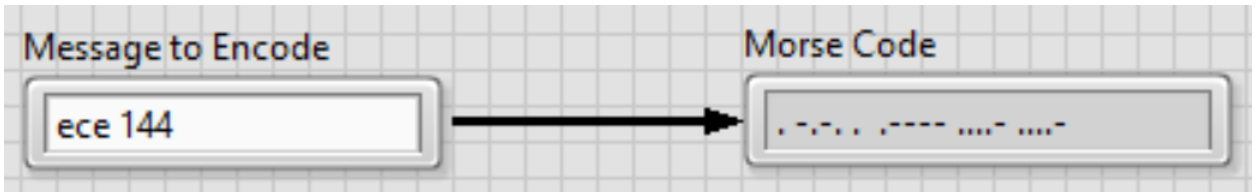
Using the morse.txt, you will be reading the entire file and extracting the information. Each line in the file contains a letter or number followed by the morse code representation. Your task is to read each line and store the letter or number and the morse code. Make sure that you store the information in such a manner that if you know the character, you can get the morse code, and vice-versa. Arrays are powerful but they may get complicated really quickly. Do you really want to iterate through an array while being inside subvi? Is there a faster way of doing this? Make a subvi for this and save it for later. **Hint:** What container is known for key-value pairs?

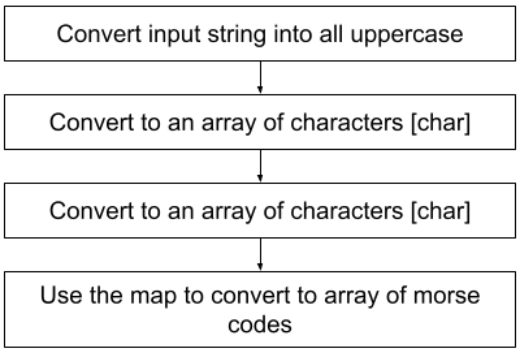


1.2: Morse Code Encoder

Now, we will want to encode any string message that is not case-sensitive. Please make sure the string is **upper-cased** before encoding. This will make things easier. Next, we will create our own “converter”. Given a string input, you will want to extract each character and store it in an array of strings. For example, “hello there” will become [“h”, “e”, “l”, “l”, “o”, “ “, “t”, “h”, “e”, “r”, “e”]. Each character can now be encoded to their respective morse code.

Using the container (map) from 1.1, find the corresponding morse code. This will also be an array of strings, but the only difference is that it will be the morse representation. Once you have this, the last thing you will do is to convert that array into a string to be shown as below. **Hint**: Use the String Palette, it has a lot of useful functions that will save you time.

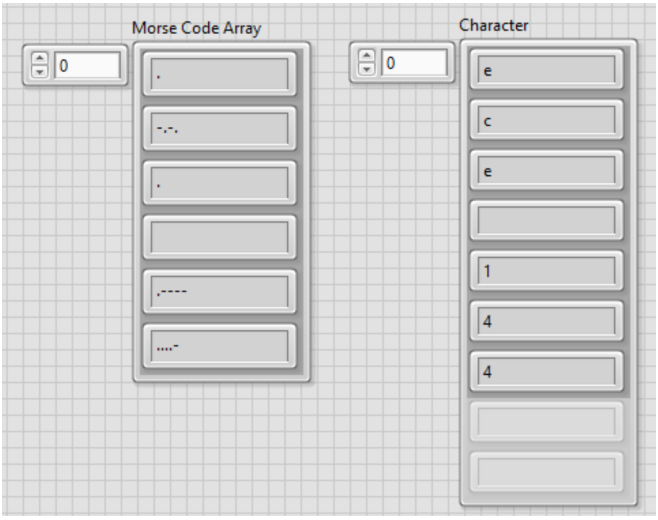




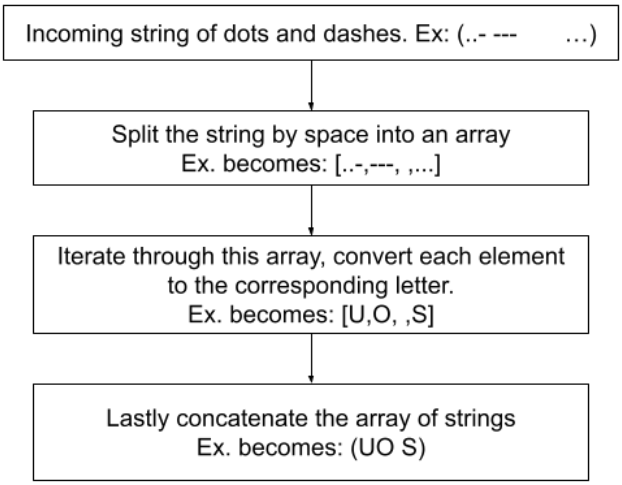
1.3: Morse Code Decoder

Lastly, we will want to decode the Morse code that we just created in 1.2. Assume that you only have the morse.txt information and the morse code string. How can we go backwards? Well the first step is to convert this string into an array of strings that contains each symbol and also of the spaces. Sound familiar? What is extremely important is to keep track of the spaces. If you have a ".", then a space, then "-", then ".", then "-", ".", then space. You need to make sure that your array is able to separate them accordingly. See image below.

For a separation of a word, there are 3 spaces and for a character it is 1 character. Now that we have the array, we need to iterate through it and grab the corresponding character from our container. Make sure you are keeping track of the spaces with a simple incrementer. If done, properly you should have the following:



Lastly, we will convert this string array into a string. You should have done this 1.2.



Challenge 2: Transmitter

In this challenge, you will be translating a string from user input into LED pulses.

2.1: String to Morse Code Encode (SubVI)

2.2: Morse Code to Boolean Array Encode

2.3: Write the Bool Array to LED Chan. Delay by Transmission Speed(ms)

Comment your code as you go!

2.1: String to Morse Code Encode

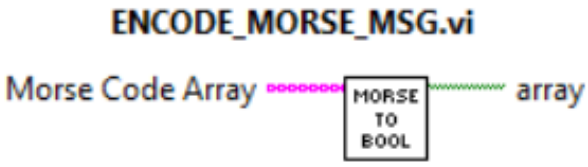
First step is to encode the user input string using the encoding function from Challenge 1. This will convert a text string of letters to a text string of morse codes. You can simply convert what you have from Challenge 1 to a SubVI. Here is an example:

2.2: Morse Code to Boolean Array Encode

Second step is to convert the morse code array to an array of booleans. For example, [“.”, “.”, “-”] morse code array will be converted to 

. Each dot will be one “T” value and each dash will be three “T” values in the array. Separate each dot/dash with a “F” value. At the end of each character, there will be two extra “F”. If you wish to send a white space character, that will be an array of seven “F” values.

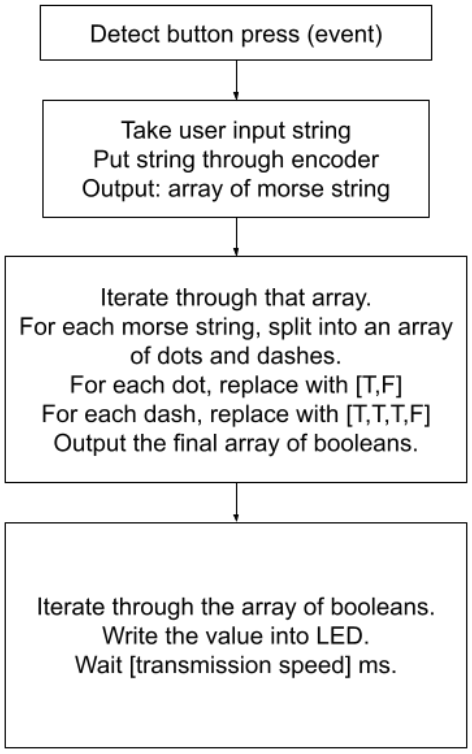
Hint: Try getting started by using a for loop to iterate through this array to append to an empty boolean array. For each dot, append [“T”, “F”]. For each dash, append [“T”, “T”, “T”, “F”]. For each space, append seven “F”s. Also make a separate default case to handle exceptions. You can make this a SubVI if you wish.



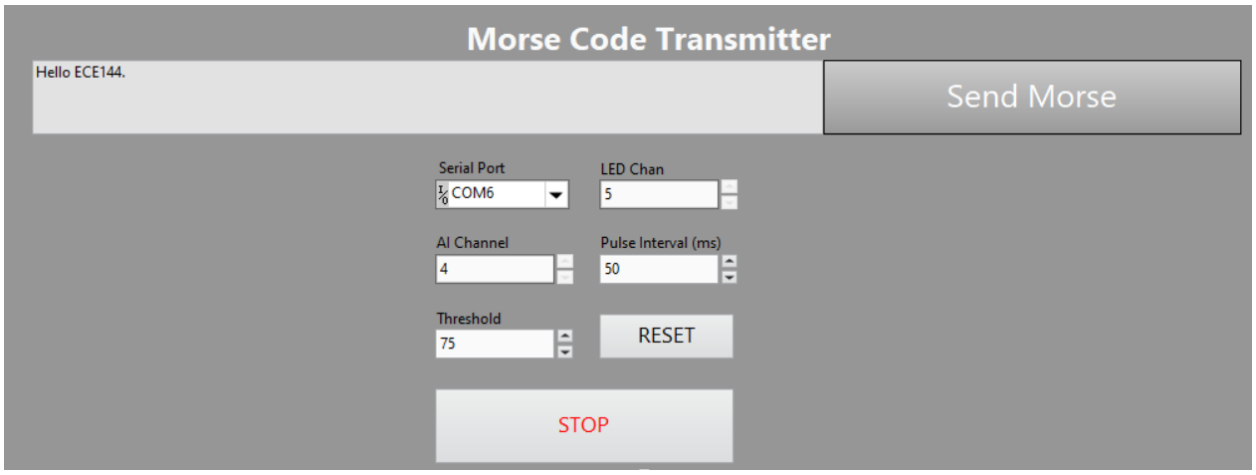
2.3: Write the Bool Array to LED Chan. Delay by Transmission Speed(ms)

Now that we are ready to send the morse code through the led. Import the SubVI from 2.1 and 2.2. Make a user input text field. Then connect the SubVIs to convert the user input to a boolean array.

Next, use a for loop to iterate through the boolean array. In each iteration, wait for transmission speed(ms).



Lastly, you should have the led transmitter code complete. To test it, put your LED transmitter in a while loop. Configure your code such that it will only send morse code when the user clicks on the “send” button. Tidy up the front panel. Here is a reference design:



Be creative and customize it!

Challenge 3: Receiver

There should be no data connection between your transmitter loop and receiver loops in LabVIEW. The only data path should be from the LED to the photodiode.

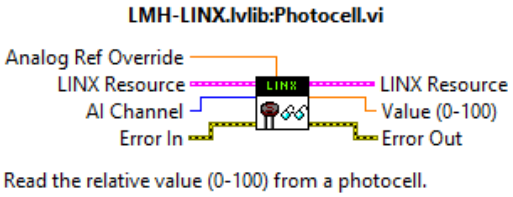
In this challenge, you will receive and decode the morse code sent by the transmitter loop that you just wrote above. Listed below will be the overall steps to create your receiver:

- **3.1: Signal Acquisition and Edge Detection**
- **3.2: Signal Translation to Morse Code**
- **3.3: Morse Decoding and Text Output**

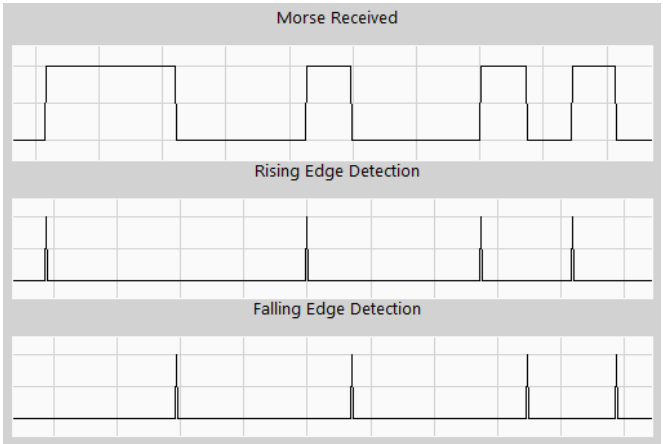
Just like the transmitter code you wrote above, your **receiver code should be encased in its own while loop for parallel operation** with the transmitter.

3.1: Signal Acquisition and Edge Detection (SubVI)

At this point, your LINX serial connection should be working well. To read the value of the photoresistor, simply use the **Photocell.vi**



Now that you have the reading from the photoresistor, you need to digitize the reading by choosing a threshold level. Anything above that threshold should be considered TRUE, and anything below should be considered FALSE. With the new digitized signal, **make a SubVI that can detect rising edges(FALSE to TRUE) and falling edges(TRUE to FALSE)**. The expected behavior of the SubVI is shown below.

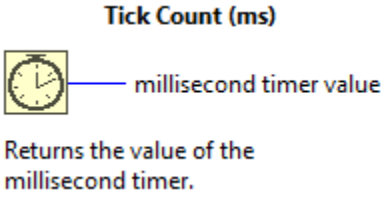


Hint: you can use a feedback node to save previous values to compare against new values in the SubVI.

3.2: Signal Translation to Morse Code

Morse code is made up of pulses of different lengths and spacings. Now that you can detect rising and falling edges in the signal from the photoresistor, you can extract the timing information of each pulse to differentiate between ".", "-", completed letters, and completed sentences.

Hint: Tick Count (ms) can be used to obtain current time. It can be stored in shift registers for you to determine elapsed time in a later loop.



To translate the signal into morse code, the edge detection output of your SubVI from 3.1 will be used.

- The amount of time between rising edge and falling edge will determine if a pulse is "." or "-".
- The amount of time between falling edge and rising edge will determine if a letter or word is complete.

There will be three cases from edge detection:

Case 1: Rising Edge

- Store current time

Case 2: Falling Edge

- Find difference between current time and time stored in case 1
- The time difference is used to determine if a pulse is "." or "-"

Case 3: No Edge

- If pulse is still being received, don't do anything
- If no pulse is received, check elapsed time to determine if a letter or word is complete

3.3: Morse Decoding and Text Output

In this section, we will be converting the morse code that you deciphered in 3.2 into alphabetic characters. In case 3 of 3.2, you checked for the end of a letter or word transmission. Whenever this is true, feed the morse code into the decoder from challenge 1 to turn it into an alphabetic character. Then append your character into a string indicator. If the end of a word is detected, append a space.

Challenge 4 [5pts]:

Be creative in your implementation of the Morse Code system. Extra credit points will be given out based on the transmission speed of the system that you built. Points will be based upon the time your system takes to send a message similar but not the same to the one below:

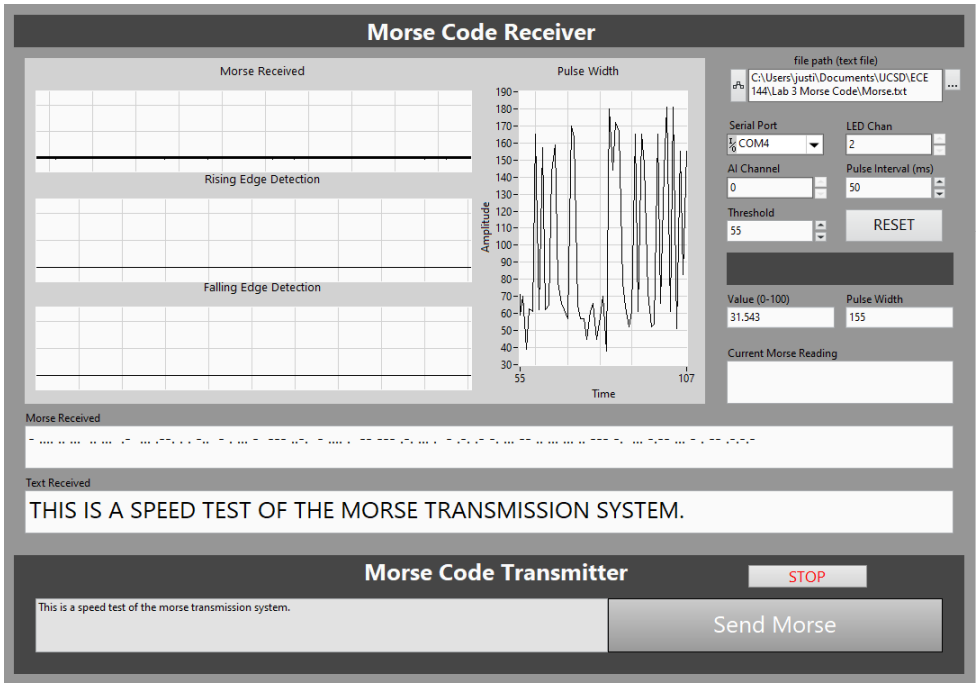
<u>This is a speed test of the morse transmission system. The faster your system transmits, the higher the credit.</u>

Transmission Speed:

- <1 Min: 3 points
- <45 Sec: 2 additional points

You can try different options to improve the transmission speed of your system. Things you can try include, but are not limited to: using two parallel led and photoresistor, using a different encoding instead of morse code, etc.

Front Panel Example



Congratulations! You just created a basic communications protocol in LabVIEW. You can now extend this project in several ways to simulate and understand any communications channel used today.


Piazza
TA