3-Node Morse Code Communication Network

Tianze Jiao, Hoang Mai Diem Pham, Ruixuan Shen, *Electrical Engineering Students*

***Abstract*—In current society, communication networks have become essential daily infrastructure. Our project is to use Morse Code to build a 3-node communication network. The main shortcomings are accuracy and speed, so we designed a system that can improve these issues. Our own communication protocol is the combination of Amplitude Shift Keying, Morse code with the addition of header, starting points, and ending points. We successfully built a communication system that can transmit and receive messages between three H7 boards with 100% accuracy and decent speed.**

***Index Terms*—Morse Code, DSP(DAC/ADC), Amplitude Modulation, sine wave, sampling, International Morse Code, Network Communication, Signal Generating Device, UART, CoolTerms.**

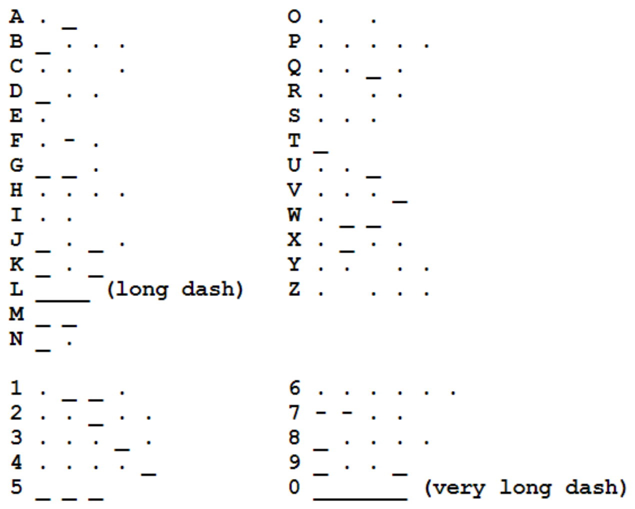
I. INTRODUCTION

A. History

A communication network, in its simplest form, is a set of equipment and facilities that provides a service: the transfer of information between users located at various geographical points. In current society, communication networks have become essential daily infrastructure. They provide flexible interconnectivity that allows the flow of mass information. By tracing back the history of telegraph communication networks, we could have a better understanding of how the networks function.

In 1837 Samuel B. Morse demonstrated a practical telegraph that provided the basis for telegram service, the transmission of text messages over long distances. In the Morse telegraph, the text was encoded into sequences of dots and dashes [1], corresponding to short and long pulses of electrical current over a copper wire. By relying on two signals, telegraphy made use of a digital transmission system. The Morse telegraph system becomes the precursor of the modern digital communication system in which all transmission takes place in terms of binary signals and all user information must first be converted to binary form [2].

In 1851 the first submarine cable was established between London and Paris. Networks of telegraph stations were established and messages were transmitted through cables over the entire continent. In these networks a message or telegram would arrive at a telegraph station, and an operator would make a routing decision based on the destination address. The operator would store the message until the desired communication line became available and then would forward the message to the next appropriate station. This method builds a foundation for later development of the communication networking systems like the Baudot system, a system that uses groups of five binary symbols to represent each letter in the alphabet [3]. It is also the reference for our project.



```
A . _                        O .   .
B _ . . .                    P . . . . .
C . .   .                    Q . . _ .
D _ . .                      R .   . .
E .                          S . . .
F . - .                      T _
G _ _ .                      U . . _
H . . . .                    V . . . _
I . .                        W . _ _
J _ . _ .                    X . _ . .
K _ . _                      Y . .   . .
L _____ (long dash)          Z .   . . .
M _ _
N _ .

1 . _ _ .                    6 . . . . . .
2 . . _ . .                  7 - - . .
3 . . . _ .                  8 _ . . . .
4 . . . . _                  9 _ . . _
5 _ _ _                      0 ________ (very long dash)
```

Fig. 1. The original Morse Code [4]

B. Constraints

- Global constraint:

Using a wired network will cost more than using a wireless network. The wired network mainly applies to small and medium network models. The design positions are already fixed. It faces many difficulties to extend the distance and deploy in rough terrain.

- Local constraint:

About the system design of this project, for the H7-board communication network, we have to use wire connection. This makes the hardware setup look a bit messy with lots of wires, and requires the presence of all three members (three laptops) to work on the 3-node communication.

With the limitation of the H7 memory size, we can not implement the code that requires large memory size in STM32. Our code needs to be simple or efficient enough to be run in a reasonable amount of time. Also, we spent lots of time to deeply understand how to use some built-in functions like ADC/DAC for transmitting signals.

II. MOTIVATION

The network communication built by students from previous years mainly has shortcomings in accuracy and speed. Accuracy is affected by noise distribution and signal receiving. We have designed an algorithm for dealing with noise distribution introduced and correctly receiving signals in the approach part. For speed, we decided to improve from the software side: coding styles/algorithms.

In the 113DA lab, signals were transmitted, processed, and output on a single board. So we choose the topic 3-node Communication Network to explore more features of the H7 board in connecting multiple boards together, whether or not the boards can correctly connect, transmit and receive the data provided by the user.

In previous labs, the board collected the signal by a buffer with a fixed size. We guarantee the H7 board can get a full signal by maintaining the voice or generating sine waves continuously before the program starts until the program stops running. However, for the communication between boards, we need a flexible buffer to store incoming signals with unknown size. We also need header or starting points to let the receiving board know when to start to store the main message. Our project requires us to dig into the operation of the DAC function on the transmitting board and the ADC function on the receiving board to solve this problem.

III. APPROACH

A. Team organization

The original plan was one person be responsible for the hardware setup and hardware debugging, one be responsible for the Morse code encoding and decoding, the other person be responsible for the code for communication. However, we later found these tasks cannot be conducted synchronously. The person who is responsible for the communication network could only start the work in the final stage of the project, after the other two people finish their jobs. We therefore modified the organization and decided the work for hardware setup, Morse code encoding and encoding as well as communication network would be evenly distributed among all three people and detailed work is determined weekly.

B. Plan

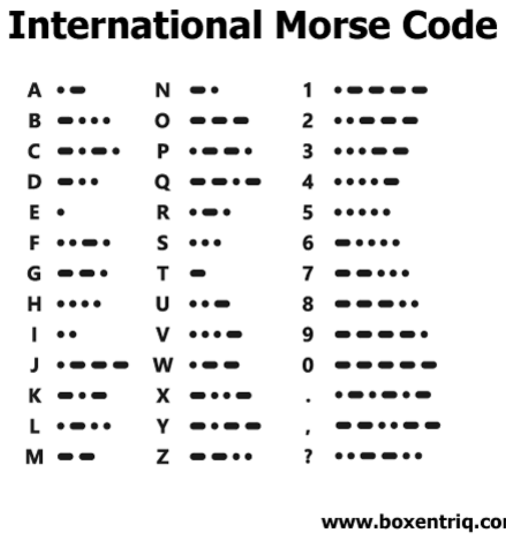
Plan	What happened
Week 1-3: Complete Hardware Setup and Encoder	We have finished the hardware setup. For the encoding part, we have only finished the coding part for conversion between Morse code and text messages based on the International Morse Code Translation table.
Week 4-5: Finish Encode Part and find a solution to input message	Encoding part was finished. With the help from TA, we learned how to use UART and CoolTerms to input the message.
Week 6: Finish Decode Part	Decoding part failed whenever the transmitted message has underscore. We have designed a new algorithm for the decoding part.
Week 7: Finish Decode Part & Start Working on Network Communication	We have finished decoding part, and implementing a temporary version of header for network communication
Week 8: Finish Most Part of Network Communication	We have designed the algorithm for the receiving part, and finished the coding part. However, we haven't tested our program.
Week 9: Finish Network Communication Part	Our program didn't work functionally as we expected. We have to discuss with TA and come up with a new algorithm for receiving parts. We have finished the coding part for the new design, and waited to test again in week 10.
Week 10: Finish Network Communication Part & Prepare for Final presentation	Our program for the receiving part failed again, the receiving board could not correctly catch the transmitting message. We have to use a simpler design for receiving parts due to time issues. We barely finished the network part. But the demo and presentation were done in the final week.

C. Standard:

We use international Morse code translation.

For Hardware Standard, the board we use for connection is NUCLEO-H743ZI2. The wires we used are 3.5mm male to male cable, and 3.5mm male to BNC. 3 laptop for transmitting/receiving signals.

For Software Standard, we use CoolTerm for UART and STM32CubeIDE 1.7.0 for coding.



International Morse Code

A	•▬	N	▬•	1	•▬▬▬▬
B	▬•••	O	▬▬▬	2	••▬▬▬
C	▬•▬•	P	•▬▬•	3	•••▬▬
D	▬••	Q	▬▬•▬	4	••••▬
E	•	R	•▬•	5	•••••
F	••▬•	S	•••	6	▬••••
G	▬▬•	T	▬	7	▬▬•••
H	••••	U	••▬	8	▬▬▬••
I	••	V	•••▬	9	▬▬▬▬•
J	•▬▬▬	W	•▬▬	0	▬▬▬▬▬
K	▬•▬	X	▬••▬	.	•▬•▬•▬
L	•▬••	Y	▬•▬▬	,	▬▬••▬▬
M	▬▬	Z	▬▬••	?	••▬▬••

www.boxentriq.com

Fig. 2. International Morse Code [5]

D. Theory

A communication protocol is a set of rules that governs how two or more communicating parties are to interact[3]. For this project, we designed our own protocol by combining Amplitude Shift Keying, Morse code and our own designed data frame.

Generally, when seeing the telegraph network from a service/architecture viewpoint, we can consider the following elements [3]:

1. A digital transmission system, which is the foundation for the network. It includes considering how the two digits to be sent, “dot” and “dash” in the case of Morse code, or “zero” and “one” in the case of Baudot and the transmission medium.
2. A framing method. It is required for indicating the beginning and end of messages and for taking the sequence of dots/dashes or zeros/ones and grouping them into characters, and in turn, meaningful messages.
3. A system for specifying the destination address of messages is needed. A routing procedure determines the path that a message follows across a network of telegraph stations interconnected by digital transmission lines.

For the design of the transmission system, we are using Amplitude Shift Keying (ASK) and Morse code. ASK is a simple technique to modify digital signals. For ASK modulation, the phase and the frequency of a carrier wave are unchanged. The carrier signal’s amplitude is varied according to the digitized modulating signal. [6]

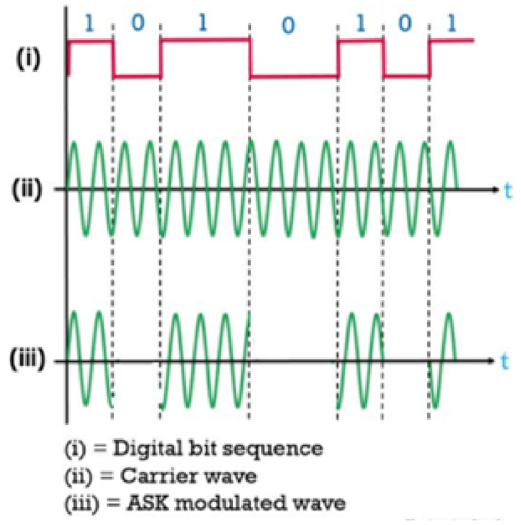


Fig. 3. Waveforms of amplitude shift keying. [6]

For our project, all characters in the input message are analog signals. However, after passing through the UART port, the signals will be processed on the H7 board in the form of digital signals. Also, based on the Morse code table in Fig.2, we only need two different amplitudes to represent the presence and absence of signal. The dot or dash is the presence of signal, and space is the absence of signal. Therefore, we take the idea about ASK modulation to apply to Morse code.

The duration of the signal’s presence or absence follows the rule of Morse code length [7]. Particularly,

1. The length of a dot is one unit.
2. The length of a dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

The disadvantage of ASK is that ASK is sensitive to noise, such as thermal noise, impulse noise, etc. The noise can be created from various phenomena such as heat or electromagnetic induction. The noise can combine with the signal to change the designed amplitude, which may cause the signal received at the receiving board to be wrong. To avoid this issue, we assigned the amplitude of the modulated sine wave around 2000 on H7 board (approximate 1.7V on the AD2), which is much larger than the noise magnitude.

For the transmission medium, we also chose wires to mitigate the effect caused by noise.

For the framing method and the system to specify the source address as well as the destination address, we are referencing a statistical multiplexing method. Statistical multiplexers provided a means for sharing a communications line among terminals. The Messages from a terminal are encapsulated inside a frame that consists of a header in addition to the user message. The header provides an address that identifies the terminal. The communication line transmits a sequence of binary digits, so a framing method is required to delineate the beginning and end of each frame [3].

E. Software/ Hardware

- *Software:*

<u>- Encode:</u>

1. Convert each character of the input into Morse code:

We represented a dot as ‘1’, a dash as ‘3’, a space as ‘0’. We recall the length of Morse code here:

1. The length of a dot is one unit.
2. The length of a dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units. For this item, instead of the space, in our project, we used the underscore ‘_’ between words, and the length of ‘_’ is seven units.

Character	Morse code	Encode Morse code
A	.-	103
B	-...	3010101
C	-.-.	3010301
D	-..	30101
E	.	1
F	..-.	1010301
G	--.	30301
H	….	1010101
I	..	101
J	.---	1030303
K	-.-	30103
L	.-..	1030101
M	--	303
N	-.	301
O	---	30303
P	.--.	1030301
Q	--.-	3030103
R	.-.	10301
S	…	10101
T	-	3
U	..-	10103
V	…-	1010103
W	.--	10303
X	-..-	3010103
Y	-.--	3010303
Z	--..	3030101
0	-----	303030303
1	.----	103030303
2	..---	101030303
3	…--	101010303
4	….-	101010103
5	…..	101010101

6	-….	301010101
7	--…	303010101
8	---..	303030101
9	----.	303030301
_		0000000

2. Generate sine waves corresponding to Morse code and output the signal through DAC function:

Each sine wave (1 cycle of sine wave) includes 32 points. Base on the length and the convention in previous parts, we generate sine waves as follow:

‘1’ = 1 sine wave (sine wave for 1 cycle)
‘3’ = 3 successive sine waves (sine wave for 3 cycles)
‘0’ = zero amplitude for 1 cycle

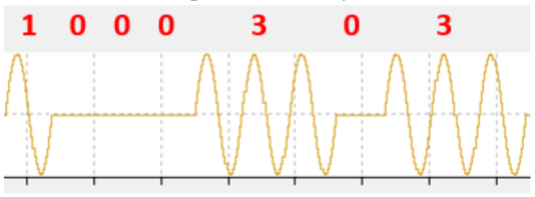


Fig. 4. An example of sine waves corresponding to Morse code

- <u>Decode:</u>

1. Convert received message array with amplitude index into a char array with index only ‘0’, ‘1’, and ‘3’ where ‘0’ refers to zero amplitude, ‘1’ refers to one sine wave, and ‘3’ refers to 3 successive sine waves.
2. Convert the char array with only ‘0’, ‘1’, and ‘3’ into a new char array with only dit, dash, slash, and space. For ‘1’/’3’ followed by one zero, converted into dit/dash; followed by three zeros, converted into dit/dash followed by space; followed by more than three zeros, converted into dit/dash plus space plus slash plus space.
3. Convert the char array with only dit, dash, slash, and space into a new char array containing the messages. The translation will be based on international Morse code translation.

- Communication:

We transmit signals in the order: starting point, header, message, ending point.

For the starting point, we use 8 successive square waves with starting amplitude 4095 to mark the starting point.

For header, we create an array of size 4, where the first two index represents source user, and the last two index represents destination user. User A, User B, User C, and all Users will be represented by index ‘00’, ‘01’, ‘10’, and ‘11’ respectively.

For the message part, there is no length limit. We will transmit the input message based on the encoding algorithm.

For the ending point, the algorithm is very similar to the starting point’s algorithm. Instead, we use 6 successive square waves with an ending amplitude 3000.

For the receiving part, we will first detect the starting point, and then detect the ending point. We will store the header part, message part, and the ending part all into a big global array. Then we will divide the big global array into header array, message array, and throw away the ending part.

- *Hardware:*

- <u>STM32H7 Nucleo-H743ZI2 (H7 board):</u>

H7 board is a low-cost device with a lot of features provided by the STM32H7 Series microcontroller, which allows us to build various prototypes [8]. Three nodes that we used in our project are three H7 boards. They are connected by 1m long wires.

PA3 is the input pin for receiving boards.

PA4 is the output pin for all boards.

To input the message to the transmitting board, UART is configured and CoolTerm is installed.

The sampling rate for both input and output in all boards is 8000 sps.

The ADC channel is in normal mode.

The DAC channel is in circular mode.

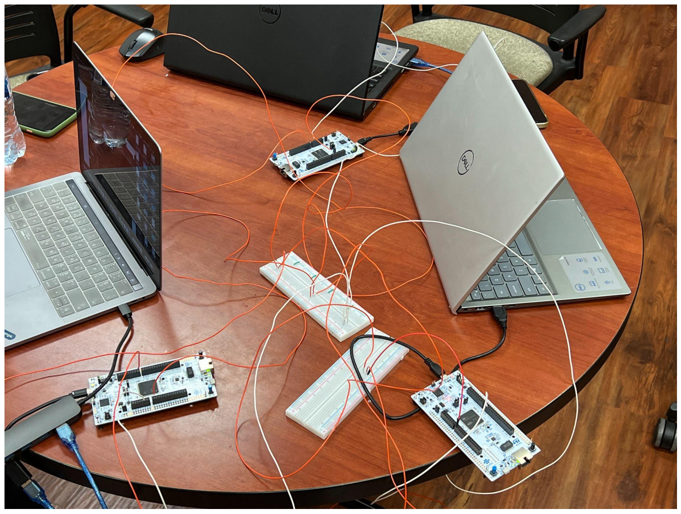


Fig. 5. The hardware setup of 3-board network

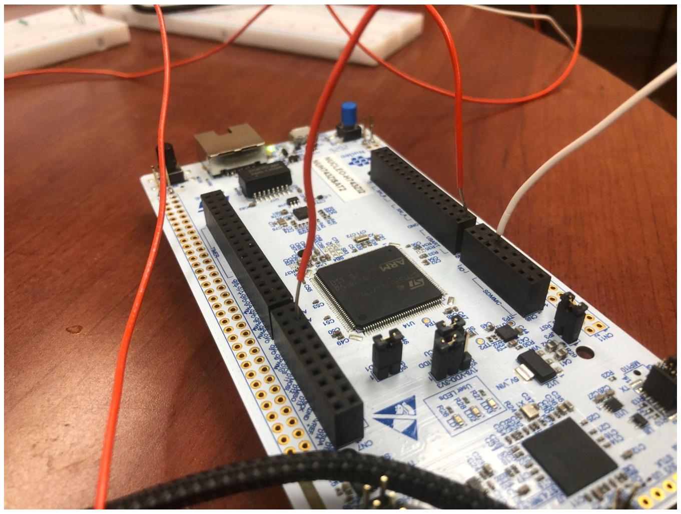


Fig. 6. The closed look at each board

- UART and CoolTerms:

As mentioned, we use the combination of UART and CoolTerm as a method for the user to input a message on the transmitting board.

UART (Universal Asynchronous Receiver/Transmitter) is a simple serial communication interface [9]. UART was already configured when we created new project in CubeIDE. In details, PD9 is set up for UART_Receive and PD8 is for UART_Transmit.

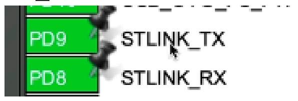
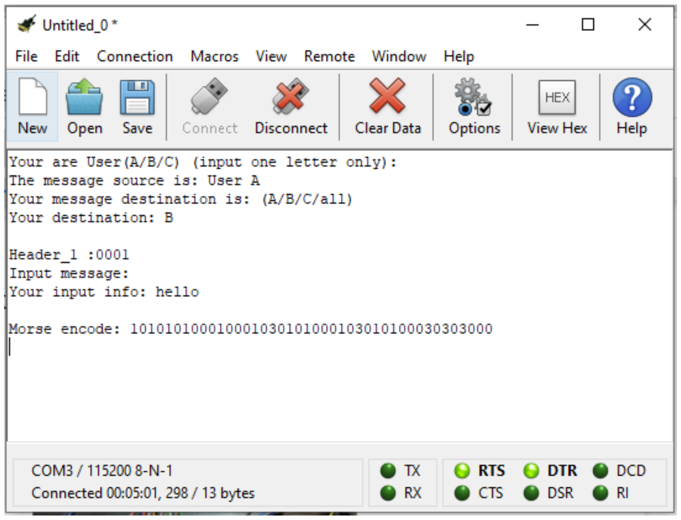


Fig. 7.. The UART is configurated in ioc. file

CoolTerm is a user-friendly terminal for serial communication with hardware that has been connected to the computer via serial ports [10]. In the Serial Port Options, we changed the Baudrate value to 115200.


Untitled_0 *
File Edit Connection Macros View Remote Window Help
New Open Save Connect Disconnect Clear Data Options View Hex Help

```
Your are User(A/B/C) (input one letter only):
The message source is: User A
Your message destination is: (A/B/C/all)
Your destination: B

Header_1 :0001
Input message:
Your input info: hello

Morse encode: 1010101000100010301010001030101000303030000
```

COM3 / 115200 8-N-1
Connected 00:05:01, 298 / 13 bytes
TX
RX
RTS
CTS
DTR
DSR
DCD
RI

Fig. 8. This is the interface of CoolTerms where the user enters the message on the transmitting board.

- AD2 (Digilent Analog Discovery 2):

AD2 is a USB oscilloscope, logic analyzer, and multi-function instrument that allows users to measure, visualize, generate, record, and control mixed-signal circuits of all kinds [11]. In this project, we used AD2 to check the output on the transmitting board. We needed to ensure that the transmitting board outputs correctly before connecting it to other boards.

For the setup:

+ The scope channel positive (solid orange or solid blue wire) on the AD2 was connected to the PA4 pin on the H7 board.

+ The scope channel negative (striped orange or striped blue wire) and the black ground wire on the AD2 both were connected to the ground pin on the H7 board.

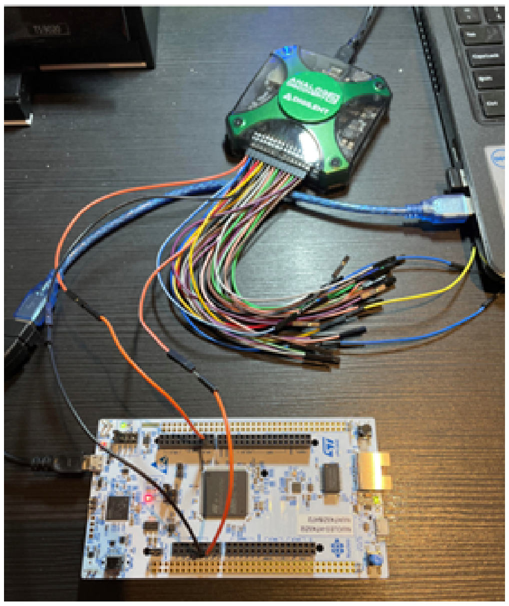


Fig. 9.. The setup of the connection between AD2 and H7 board

For example, with the input message "hi", the Morse code is 1010101000101000, we could see the output on the AD2:

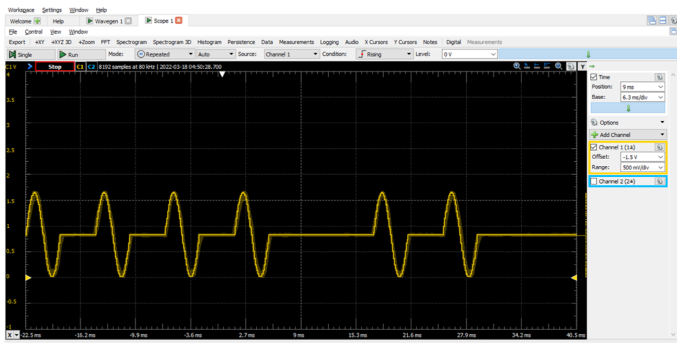


Fig. 10. An example of output signal on AD2

Then, after adding starting points, header, and end points to the main message, we would get the transmitting signal on the AD2:

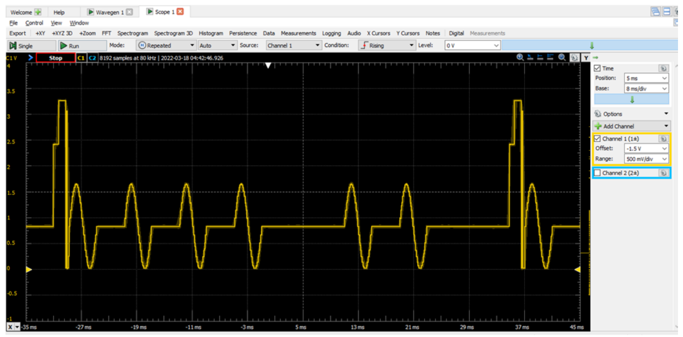


Fig. 11. An example of output signal on AD2 after adding starting points, header, and end points

F. Operation

1. Steps

❖ *Hardware setup:*

+ We need three H7 boards, three laptops, at least 12 long wires and 1 breadboard, then connect three boards as the picture below:

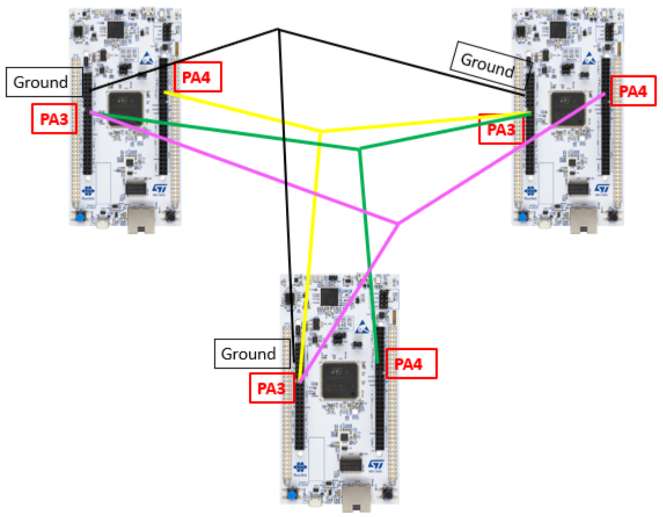
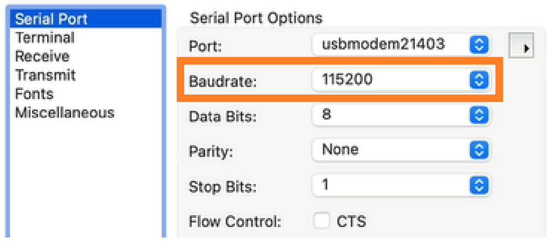


Fig. 12. The illustration of hardware setup of 3-board network

+ In .ioc file, choose the tab Clock Configuration, change the System Clock Mux to PLLCLK, make sure the timer clock is 75 KHz, and the Counter Period is 9374, which means the sampling rate is 8000 sps.
+ In the tab Pinout&Configuration, set PA3 as ADC1_INP15, and set PA4 as DAC1_OUT1
+ In the tab Analog, choose ADC1, choose DMA setting, choose the mode Normal. Also in the tab Analog, choose DAC1, choose DMA setting, choose the mode Circular.

+ UART is automatically configured so we do not need to do anything to activate it.
+ Install CoolTerms, then in the Serial Port Options, change the Baudrate value to 115200.



Serial Port Options

Port: usbmodem21403
Baudrate: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: CTS

Fig. 13. Change Baudrate value on the CoolTerms interface

❖ *Code:* in main.c

- Encode:
  + Code for converting each character of the input into Morse code (details were shown in the software part above).
  + Code for generating sine waves corresponding to Morse code and outputting the signal through DAC function (details were shown in the software part above).
  + Use the SWV Data Trace Timeline Graph in CubeIDE to verify the correctness of generated sine waves.

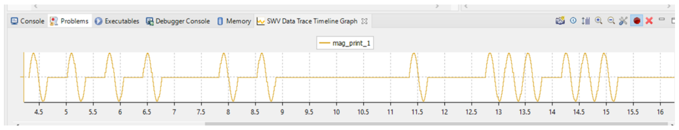


Fig. 14.. Using SWV Data Trace Timeline Graph to verify the sine waves

  + After that, pass the signal through the DAC function to output to other boards.
  + Use AD2 to check the output signal on the transmitting board (the hardware setup and examples were shown in the AD2 hardware part above).

- 3-board communication:
  + Add communication protocol to the main message, which includes starting points (8 digits with the amplitude of 4095), header (4 digits source + destination information), and ending points (6 digits with the amplitude of 3000). The full output on the transmitting board is shown below.

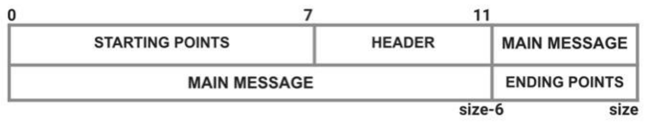


Fig. 15. The message frame is composed of four major parts: the starting points, the header, the main message and the ending points.

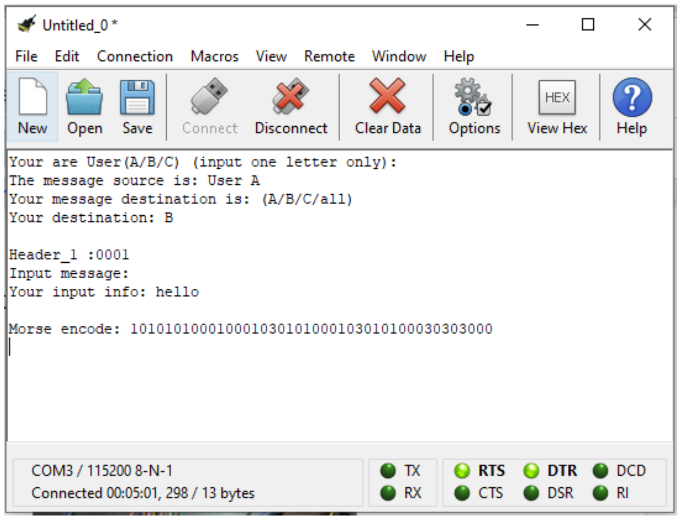
+ Use AD2 to check the output signal on the transmitting board after adding communication protocol.
+ Code to detect the starting points, detect the ending points, then extract and store the message into an array. After that, pass this array through the decoding part.

- Decode:
  + Code for converting the received message to the Morse code in the form of '0', '1', '3'.
  + Code for converting the array of '0', '1', '3' into the array of dit, dash, slash, and space.
  + Code for converting the char array with only dit, dash, slash, and space into a new char array containing the messages.
  + Use UART and CoolTerms to print out the message.

2. How to use the system:

Encoder:

When the user enters the encoder interface, it would ask the user's ID (user A, user B or user C) and the destination of the message (user A, B, C, or all users.) After choosing the source and destination, the program would ask the user to input the message that they want to send and automatically encode the text message in Morse code form as well as the header. User can verify the encoded message and the header through the CoolTerm interface. At this point, the message is sent out by the DAC and the program can loop back to the first stage, waiting for the user to send another message.


Untitled_0 *
File Edit Connection Macros View Remote Window Help
New Open Save Connect Disconnect Clear Data Options View Hex Help
TX RX RTS CTS DTR DSR DCD RI

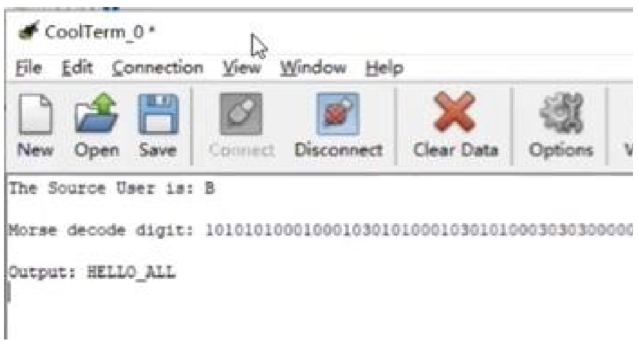
```
Your are User(A/B/C) (input one letter only):
The message source is: User A
Your message destination is: (A/B/C/all)
Your destination: B

Header_1 :0001
Input message:
Your input info: hello

Morse encode: 1010101000100010301010001030101000303030000
```

COM3 / 115200 8-N-1
Connected 00:05:01, 298 / 13 bytes

Fig. 16. This is the interface of CoolTerms where user A enters the message on the transmitting board.

Decoder:

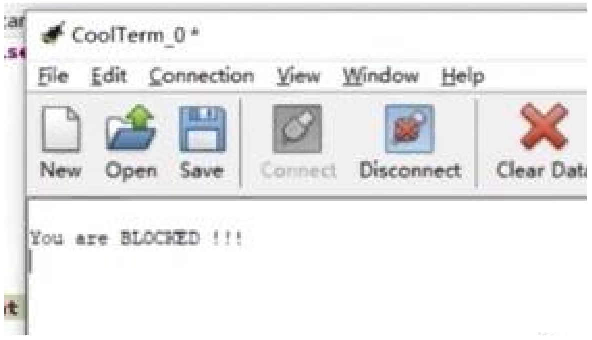
The program would constantly be in the idle state until a piece of message is received. Through the CoolTerm interface, the user can see the message received and its source if the user matches the destination of the message.


CoolTerm_0 *
File Edit Connection View Window Help
New Open Save Connect Disconnect Clear Data Options

```
The Source User is: B

Morse decode digit: 101010100010001030101000103010100030303000000

Output: HELLO_ALL
```

Fig. 17. This is the interface of CoolTerms where the user receives message from user B.

If the user's ID does not match with the destination included in the message's header, the CoolTerm interface would print "You are BLOCKED!!!"


CoolTerm_0 *
File Edit Connection View Window Help
New Open Save Connect Disconnect Clear Dat

```
You are BLOCKED !!!
```

Fig. 18. This is the interface of CoolTerms when the receiver board receives a message with unmatched destination ID. The user is blocked.

After receiving and decoding the message, the program would go back to idle state and wait for receiving a new round of messages.

IV. RESULTS

Demo Youtube Link:
https://www.youtube.com/watch?v=jjz9xdVJnYg

A. Describe:

- Accuracy

The transmit message is accurately and fully transferred to the receiving board. Success rate is 100%.

-Speed

Both transmitting signal and receiving signal can be done in a few seconds.

-Transmit Message Size

Transmitting message can't exceed the ADC buff size

-Header

Header works functionally in selecting source user, destination user as well as blocking user.

B. Discussion

- Accuracy

The challenge for accuracy is to deal with random noises and distribution. To overcome the challenge, we have set a range for detecting starting point and ending point instead of an exact amplitude.

For detecting the sine wave, we have set up an algorithm to use the half length index of each sine wave to determine if the sine wave exists or not.

In our case, our sine wave length is 32 indexes, so we will use 16 indexes to determine if a sine wave exists or not. In this way, even if some indexes of the sine wave are way off due to noises and distribution, we could still accurately determine if it is a sine wave or not.

-Speed

The main challenge for speeding up is to deal with generating sine waves, and ADC/DAC functions. It is hard to reach faster speed from software perspective, since these are all built-in functions, and we can't edit anything to optimize them. We can only improve the speed of the encoding and decoding part, which is already very fast and can be done in less than a second.

Another way is to improve speed up from a hardware perspective like using laptops with higher performance CPU, but it will be costly. Considering that we don't really need insane fast transmitting/receiving signal speed, we decided to not improve speed from a hardware perspective.

-Transmit Message Size

The transmitting message size is limited by the ADC buff size, since we only use one buff for receiving signals. One solution could be using two buffs for receiving signals. However, the challenge is that there is a gap between buff switches. Whenever buff is switched, some signals will be lost.

We have tried to receive the index one by one by setting the ADC buff size to one when passing into the function. However, we found out that there will also be a gap between each index to prevent us from reading successive indexes.
Then we tried to run two ADC functions one after one immediately so that the gap could be negligible. But the constraint is that we can't have processing code in between, which becomes meaningless because we can't use the values we extracted.

Finally, we tried to run two ADC functions in parallel in the way that one buff is full, we then fulfill the other buff and run the processing codes at the same time. But there is a software difficulty, since the ADC function is an embedded built-in function, and we don't know how to set it to work with processing code at the same time. We have tried several coding ways and algorithms, but all failed in accurately receiving signals.

Due to time issues, we have to give up using two buffs for receiving signals, and use only one buff instead. However, we can still set a very large buff for receiving signals with long messages. The trade off for creating a large buff is to take more memory space.

-Header

Current version of header has the functions for determining source, and destination. Both functions work appropriately.

We can add more functions to the header to build a more complex communication network. For example, we can add an index to indicate the current state of the user: 0 for idling state, 1 for receiving/transmitting state. Then when the user transmits a message to other users, the user can check other users' states to decide whether to transmit now or wait for them to have an idling state.

In this way, we can achieve non-stop discussion communication among three nodes in a one time run. However, it also requires modification for other parts, which require more time to complete such a complex system.

REFERENCES

[1] Bunnell J.B. The American, Continental, and International Morse Codes

[2] R. W. Burns, Communications: An international history of the formative years. Institution of Electrical Engineers, 2004.

[3] León García, A. and Widjaja. Communication networks. Madrid: MacGraw-Hill. 2004

[4] Telegraph Apparatus. [Online]. Available: http://jhbunnell.com/morsecode.shtml.

[5] J. Åhlén, "Morse code translator, decoder, alphabet," Boxentriq. [Online]. Available: https://www.boxentriq.com/code-breaking/morse-code.

[6] "What Is Amplitude Shift Keying (ASK)? Theory, Advantages, Disadvantages and Applications of Ask." *Electronics Coach*, 29 Sept. 2018, https://electronicscoach.com/amplitude-shift-keying.html.

[7] "Morse Code." *Wikipedia*, Wikimedia Foundation, 13 Mar. 2022, https://en.wikipedia.org/wiki/Morse_code.

[8] *Home - Stmicroelectronics.* https://www.st.com/resource/en/user_manual/dm00499160-stm32h7-nucleo144-boards-mb1364-stmicroelectronics.pdf.

[9] "Simple USB / Serial Communication with the CP2102N." *Vivonomicon's Blog,* https://vivonomicon.com/2019/10/25/simple-usb-serial-communication-with-the-cp2102n/.

[10] Jonathan Tan, et al. "Serial Terminal Basics with CoolTerm." *Latest Open Tech From Seeed,* 29 Sept. 2021, https://www.seeedstudio.com/blog/2021/01/20/serial-terminal-basics-with-coolterm/#:~:text=CoolTerm%20is%20a%20user%2Dfriendly,Windows%20%2F%20Mac%20%2F%20Linux%20devices.

[11] Oscilloscope, Logic Analyzer and Variable Power Supply." *Digilent*, https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/.