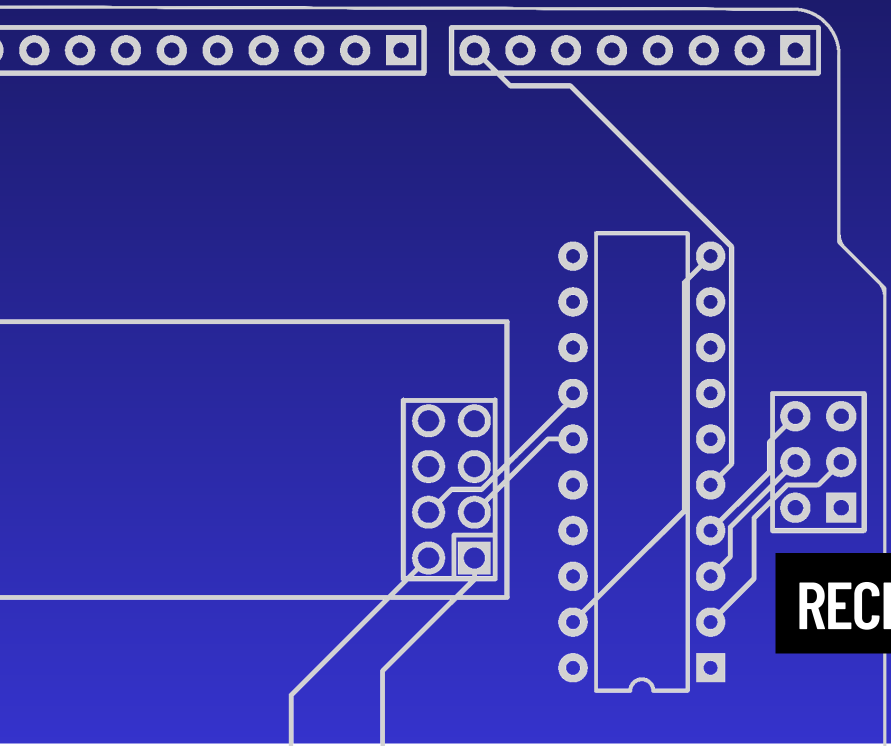
MLC

MORSE MESSENGER

..- -.- . ... ..-. / -- --- .-. ... . / -- . ...
. -. --. . .-.



RECIEVING

EASIER

RECIEVING

Before the world had smartphones, email, or even the humble landline, there was Morse code. Developed in the early 19th century, this revolutionary system of dots and dashes – often used by spies – was a way of encoding data such that it could be transmitted across oceans, via radio, and decoded by humans.

The table below shows how each character of the alphabet was encoded by a series of dots and dashes. Notice how the letter ‘E’ is a single dot, as this is the most frequent letter used in commination. On this page alone there are 64 occurrences of ‘e’.

A	● ▬	N	▬ ●
B	▬ ● ● ●	O	▬ ▬ ▬
C	▬ ● ▬ ●	P	● ▬ ▬ ●
D	▬ ● ●	Q	▬ ▬ ● ▬
E	●	R	● ▬ ●
F	● ● ▬ ●	S	● ● ●
G	▬ ▬ ●	T	▬
H	● ● ● ●	U	● ● ▬
I	● ●	V	● ● ● ▬
J	● ▬ ▬ ▬	W	● ▬ ▬
K	▬ ● ▬	X	▬ ● ● ▬
L	● ▬ ● ●	Y	▬ ● ▬ ▬
M	▬ ▬	Z	▬ ▬ ● ●

YOUR TASK

As a receiver you will be receiving a message from your partner that you will need to decode. Your partners message will be transmitted to your device in Morse code and your job is to translate this into comprehensive English.

STEP 1:

Firstly download the code stubs and dependent libraries from the following URL. If you already have an instance of a given library on your device, you won't need to install it again!

HTTPS://EXAMPLE-URL/RESOURCE.ORG

STEP 2:

Open the code in the Arduino IDE and find the subroutine below.

```
String DecodeMorse(String MorseCode)
{
   return "Edit this subroutine!";
}
```

MLC

This will be the function you are editing. You will have noticed the function takes in the encoded Morse message and return a String, ideally the translated English string.

That's where you come in.
Write a function to convert a string of Morse code into English. You may assume the message only consists of the following characters. (No Spaces)

●	▬	/
DOT (Full Stop)	DASH (Hyphen)	SLASH (Forwards Slash)

For example, The string:

```
..../././.-../.-../---/.--/---/.-./.-../-../
```

Would translate to:

HELLOWORLD

If you're unsure where to begin, consider looking up parallel arrays and substrings for Arduino, specifically the String() class. How this could benefit you here?

STEP 3:

Upload the code and test it by pressing the input button on your board, your should see a "CONGRATULATIONS" scroll across the screen. If so, wait for your partner to send you a message and have fun!

Make sure both you and your partner are using the same address (any 5 bit binary number) and the same channel (any integer between 0 and 255)

```
const byte address[6] = "00001";
const int Channel = 100;
```

These are the default values chosen for you, but change, at least, the channel to a value that your other classmates aren't using, this will ensure only you receive the intended message, not your classmates.

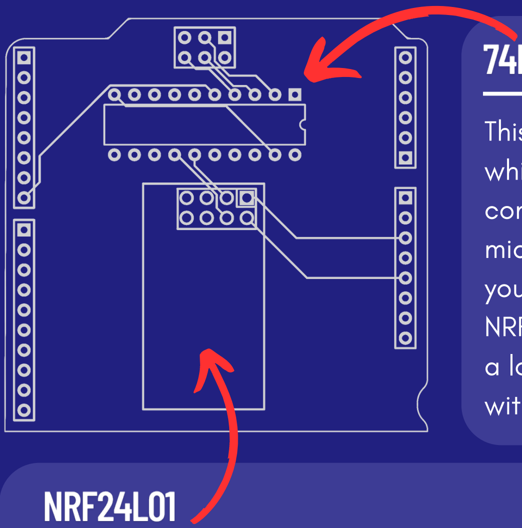
Or perhaps try to intercept one of their messages, like a real spy! To do this, you'll have to figure out which channel and address your target is using, without drawing suspicion to yourself. Good Luck!

EXTENSION:

What if your partner sends over a sequence of dots and dashes that have no English equivalent? How can you alter your code to handle these mishaps?

If you're interested in the rest of the code click on the UKESFMorse.h tab near the top of the IDE, here you can see where other libraries and subroutines have been implemented!

UNDERSTANDING THE CIRCUIT



74LVC245AN

This is a logic level converter which allows the radio module to communicate with the microprocessor beneath it. See if you can find out why the NRF24L01 radio module requires a logic level converter when used with the Seeeduino Lotus

NRF24L01

This is the NRF24L01 transceiver. It's a 2.4GHz radio module used for communication, some older phones would've used a similar circuit!

2.4GHz RADIO

Radio waves at 2.4GHz are part of the microwave spectrum, widely used in wireless communication. The "GHz" measures frequency—2.4GHz equals 2.4 billion cycles per second! Signals at this frequency have become essential in modern technologies.

One common application is Wi-Fi, which often operates in the 2.4GHz band. This frequency is favoured for its ability to travel farther and penetrate obstacles like walls, compared to higher frequencies. Bluetooth devices also utilize 2.4GHz signals to connect wirelessly, making it integral to smart gadgets and wearable tech.

However, because many devices use 2.4GHz, interference can sometimes occur, affecting signal performance. But by using different channels which use very slightly different frequencies or bands, we can filter out the noise by only listening to the frequency we want!

MLC