Coding and amateur radio

Morse Game by M0KUK

Using microprocessors opens a new ability to easily create and modify the tool to provide bespoke solution. Morse code is very popular with amateur radio enthusiasts for its ability to provide sustainable means of communication even in times of weaker propagation. It is also a great way for those who like building their rigs as the construction of the transceiver can be much simpler than in case of voice modes.

About Two Hours

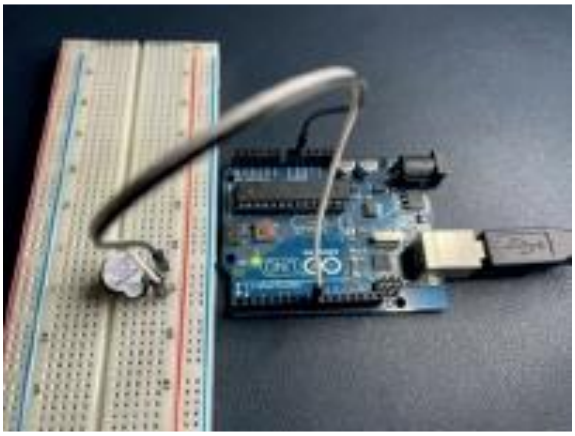
Kit List

- Arduino board (e.g. Arduino Uno or Nano)
- Buzzer or a small speaker
- Jumper/dupont wires
- Computer with Arduino IDE
- Breadboard (optional)

Instructions

If you don't have Arduino IDE please download and install it. Latest version is available on the Arduino webpage: https://www.arduino.cc/en/software or you can use Arduino Web Editor available from the same link.

For this example, we will be using Arduino Uno board, but any other board can be used. Please remember to adjust the pins to the board that you are using as this might be different.



You should be able to upload your sketch after a few steps to test it out and experiment with it.

Below listings aim on explaining critical parts of the code, however there will be some elements that were omitted in this document to make it easier to read. Feel free to refer to the Morse Game. ino file available here: https://github.com/M0KUK/MorseGenerator/blob/master/MorseGame.ino

1. Connect Buzzer (positive) to Arduino Uno to pin 8 and negative to ground (GND)
2. Open Arduino IDE and start new project
3. Start by defining connections (top of the file), tone and morse speed (length of dots and dashes or dits and dahs)

```
const int buzzerPin = 8; //defines pin that buzzer is connected to
const int morseTone = 600; //defines the tone for the morse sound
const int ditLength = 100; //dot length in milliseconds
const int dahLength = 3 * ditLength; //dash length in miliseconds (3x dot length)
```

4. Let's start with Morse version of 'Hello World!' calling CQ that is used to start communication. In setup section that starts with void setup() we need to play a tone, wait for the duration of dit or dah, make a short pause. Letter 'C' in morse is - . - . (dah, dit, dah, dit) and letter 'Q' is - - . - (dah, dah, dit, dah). So our code will look like this:

```
// Play CQ
tone(buzzerPin, morseTone,
dahLength); delay(dahLength);
delay(ditLength);
tone(buzzerPin, morseTone,
ditLength); delay(ditLength);
delay(ditLength);
tone(buzzerPin, morseTone,
dahLength); delay(dahLength);
delay(ditLength);
tone(buzzerPin, morseTone,
ditLength); delay(ditLength);
delay(dahLength); // end of character
tone(buzzerPin, morseTone,
dahLength); delay(dahLength);
delay(ditLength);
tone(buzzerPin, morseTone,
dahLength); delay(dahLength);
delay(ditLength);
tone(buzzerPin, morseTone,
ditLength); delay(ditLength);
delay(ditLength);
tone(buzzerPin, morseTone,
dahLength); delay(dahLength);
delay(dahLength);
```

Please note that within one character there is a slight delay of 1 dot between each dot or dah. And to mark end of each letter there is a longer pause equal to 1 dash (or 3 dots). Even though it's not used in this example after a full word we use longer pause of 3 dashes.

5. At this point you should be able to upload your sketch after connecting Arduino board to USB.

6. To make the code easier to read let's create a procedure to play dit and a procedure to play dah. To create a procedure we use command void with a name of the procedure. In our example we will create:

```
void ditPlay(){
  tone(buzzerPin, morseTone,
  ditLength); delay(ditLength);
}


void dahPlay(){
  tone(buzzerPin, morseTone,
  dahLength); delay(dahLength);
}
```

7. To use them we simply can type ditPlay() or dahPlay() what will simplify setup section of code to:

```
void setup() {
  // Play CQ
  dahPlay();
  delay(ditLength)
  ; ditPlay();
  delay(ditLength)
  ; dahPlay();
  delay(ditLength)
  ; ditPlay();
  delay(dahLength); //end of character
  dahPlay();
  delay(ditLength)
  ; dahPlay();
  delay(ditLength)
  ; ditPlay();
  delay(ditLength)
  ; dahPlay();
  delay(ditLength)
  ;
}
```

8. Further improvement can be achieved by mapping all letters arguments in our case accepts letter that we want to play: void playCharacter(char letterToPlay)

```
void playCharacter(char letterToPlay){
  String toPlay = morseCode[letterToPlay- ASCII_START]; //selecting right
  character for (byte i = 0; i < toPlay.length(); i++) //looping through all dits and
  dahs
  {

    play(toPlay[i]);
    delay(ditLength);
  }

}
```

We also used one more procedure to help us with the code:

```
void play(char
 whatToPlay){ if
 (whatToPlay == '-'){
 dahPlay();
 }

 else if (whatToPlay ==
   '.'){ ditPlay();
 }

}
```

and we used one more variable (array) to store all the letters in alphabetic order to be able to 'translate' letter to Morse code:

```
const String morseCode[] = {
  ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-
.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."

};
```

That allows us to have clear and readable lines in setup:

```
void setup() {
 // Play CQ
 playCharacter('C');
 delay(dahLength); //end of character
 playCharacter('Q');
}
```

9. Now we can use loop() section of the script for running a part of our program indefinitely. We can start with simple part generating a random letter, playing it and then waiting before playing a new character

```
void loop() {
  //Play random character
  char randomLetter = random(NUMBER_OF_LETTERS) + ASCII_START;
  playCharacter(randomLetter);
  delay(dahLength);

  delay(WAIT_TIME);
}
```

10. We could stop here and have a nice Morse tutor, but let's try to make it more interesting by adding the ability to provide the 'guess' of the heard letter. To achieve it we will use Serial monitor that will wait for keyboard input and allow us to communicate back to the user if the guess is correct or not. If it's former then the Congrats message will be shown and new letter will be played after a short pause. In the latter scenario the same letter will be played

```
void loop() {
  //Play random character
  char randomLetter = random(NUMBER_OF_LETTERS) + ASCII_START;
  playCharacter(randomLetter);
  delay(dahLength);

  while (guessLetter !=
    randomLetter){ if
    (Serial.available()){
      guessLetter = Serial.read();
      //Condition to make all characters uppercase
      if (guessLetter >= ASCII_LOWER_A && guessLetter <= ASCII_LOWER_Z){
        guessLetter = guessLetter - ASCII_LOWER_TO_UPPER;
      }

      //Presenting error message and replaying a
      character if (guessLetter != randomLetter){
        Serial.println("Not quite right. Try again!");
        playCharacter(randomLetter);
        delay(dahLength);
      }

    }

  }

  Serial.println("That's right! Congratulations");

  delay(WAIT_TIME);
}
```

Next Steps

- Try changing the tone and length of the sound
- Try adding numbers
- Try adding a switch to practice sending Morse
- Try adding more than one character, Q-Code or a random call sign.

More Information

- Full code can be found here: https://github.com/M0KUK/MorseGenerator/blob/master/MorseGame.ino
- Additional Arduino resources and examples are available here: https://www.arduino.cc/reference/en