```
/*********************************************************************
 Project     : Demo 3- Morse Tutor
 Version     : 1.0
 Libraries   : none
 Author      : Clive Maby
 Description : Generate Random Morse code A-Z, 0-9, A-Z & 0-9, Punctuation
               5 character groups by default

 Analogue Pins                    Digital Pins
 A0 - Random seed                 D2 - Audio Out
 A3 - Character Speed             D4 - Letters Switch
 A4 - Character Gap               D5 - Numbers Switch
                                  D6 - Letters + Numbers Switch
                                  D7 - Punctionation Switch
 *********************************************************************/

const int SpeakerPin = 2;                       // Set Audio out pin
const int GroupSize = 5;                        // Group Length
const int ToneFreq = 750;                       // Tone Frequency

int ditlength = 100;                            // Initial dot length
int dahlength = ditlength * 3;
int inter = ditlength;                          // inter character gap
int GroupGap = ditlength * 7;                   // inter group gap
int GroupCount = 0;                             // Group counter
int WordsPerMin = (analogRead(3) / 100);        // Character Speed
int charGap = analogRead(4);                    // Character Gap
int OptionPin=0;                                // Option selected
int RandomChar = 0;                             // Random Character selected
int Option =0;                                  // Current Option
int CW_Array_Start = 0;
int CW_Array_End = 0;

// Array for Letters, Numbers and Punctuation
const char CW_ARRAY[] = {
        'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R',
        'S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9',
         '/', '?', '(', ')', '"', '.', ',', ':', '-', '=',
 };

// Array for ditlength - Words Per Minute from 6 to 26 WordsPerMin
// Values must be /100 to determine ditlength (held as int as more efficient)
const int WordsPerMin_ARRAY[] = {175, 150, 125, 100, 90, 80, 70, 60, 50, 40, 30};


void setup()
{
  Serial.begin(115200);  // Set Comport Speed
  Serial.println("G4NAQ Morse Tutor");
  randomSeed(analogRead(0));

  // initialize the inputs (pins default to input so not set using pinMode) &
outputs
  pinMode(SpeakerPin, OUTPUT);
```

```
  for (int i = 4; i < 8; i++)
  {
   digitalWrite(i, LOW);
  }
  check_option();
}


void loop()   // Set WPM and inter group gap, work out CW option, generate random
character and sound it
{
  set_WordsPerMin();
  set_gap();
  check_option();
  RandomChar = random(CW_Array_Start,CW_Array_End);
  soundChar(CW_ARRAY[RandomChar]);
  check_GroupLength();
};


void check_option()  // Check CW Option based on switch position
{
  for (int i = 4; i < 8; i++) {
    OptionPin=digitalRead(i);
    if (OptionPin == HIGH) {  // Switch position is set if high
        if (Option != i) {    // Has position changed since last checked
            Option = i;       // Save new option
            GroupCount=0;     // Reset counter so display is correct
            Serial.println("     ");
            switch (i) {      // Check which option selected & choose range in CW
Array for characters
              case 4:
                  Serial.println("Sending Letters");
                  CW_Array_Start = 0;
                  CW_Array_End = 25;
                  break;
              case 5:
                  Serial.println("Sending Numbers");
                  CW_Array_Start = 26;
                  CW_Array_End = 35;
                  break;
              case 6:
                  Serial.println("Sending Letters & Numbers");
                  CW_Array_Start = 0;
                  CW_Array_End = 35;
                  break;
              case 7:
                  Serial.println("Sending Punctuation");
                  CW_Array_Start = 36;
                  CW_Array_End = 45;
                  break;
            };
         break;
        };
```

```
      break;
     };
   };
}


void check_GroupLength()   // Group length check
{
   GroupCount++;
   if (GroupCount == GroupSize) {
      delay(GroupGap);
      GroupCount=0;
      Serial.println("");
      }
}


void dit() // play a dot
{
  tone(SpeakerPin, ToneFreq);
  delay(ditlength);
  noTone(SpeakerPin);
  delay(inter);
}


void dah()  // play a dash
{
  tone(SpeakerPin, ToneFreq);
  delay(dahlength);
  noTone(SpeakerPin);
  delay(inter);
}


void set_gap()  // Set inter character gap
{
  charGap = analogRead(4);
  if (charGap < 100) {
     charGap = 100;
   }
   delay(charGap);
}


void set_WordsPerMin()  // Set Words Per Minute
{
    WordsPerMin=analogRead(3) / 100;
    ditlength = (WordsPerMin_ARRAY[WordsPerMin]);
}


void soundChar(char lookupChar)  // Sound selected character
{
```

```
 Serial.print(lookupChar);
 Serial.print(" ");
 switch(lookupChar)
{
case 'E':
  dit();
  break;
case 'T':
  dah();
  break;
case 'A':
  dit();    dah();
  break;
case 'O':
  dah();   dah();    dah();
  break;
case 'I':
  dit();    dit();
  break;
case 'N':
  dah();   dit();
  break;
case 'S':
  dit();    dit();    dit();
  break;
case 'H':
  dit();    dit();    dit();    dit();
  break;
case 'R':
  dit();    dah();    dit();
  break;
case 'D':
  dah();   dit();    dit();
  break;
case 'L':
  dit();    dah();    dit();    dit();
  break;
case 'C':
  dah();   dit();    dah();    dit();
  break;
case 'U':
  dit();    dit();    dah();
  break;
case 'M':
  dah();   dah();
  break;
case 'W':
  dit();    dah();    dah();
  break;
case 'F':
  dit();    dit();    dah();    dit();
  break;
case 'G':
  dah();   dah();    dit();
```

```
  break;
case 'Y':   dah();    dit();    dah();    dah();
  break;
case 'P':
  dit();    dah();    dah();    dit();
  break;
case 'B':
  dah();   dit();    dit();    dit();
  break;
case 'V':
  dit();    dit();    dit();    dah();
  break;
case 'K':
  dah();   dit();    dah();
  break;
case 'J':
  dit();    dah();    dah();    dah();
  break;
case 'X':
  dah();   dit();    dit();    dah();
  break;
case 'Q':
  dah();   dah();    dit();    dah();
  break;
case 'Z':
  dah();   dah();    dit();    dit();
  break;
case '0':
  dah();   dah();    dah();    dah();    dah();
  break;
case '1':
  dit();    dah();    dah();    dah();    dah();
  break;
case '2':
  dit();    dit();    dah();    dah();    dah();
  break;
case '3':
  dit();    dit();    dit();    dah();    dah();
  break;
case '4':
  dit();    dit();    dit();    dit();    dah();
  break;
case '5':
  dit();    dit();    dit();    dit();    dit();
  break;
case '6':
  dah();   dit();    dit();    dit();    dit();
  break;
case '7':
  dah();   dah();    dit();    dit();    dit();
  break;
case '8':
  dah();   dah();    dah();    dit();    dit();
  break;
```

```
  case '9':
    dah();   dah();    dah();    dah();    dit();
    break;
  case '/':
    dah();   dit();    dit();    dah();    dit();
    break;
  case '?':
    dit();    dit();    dah();    dah();    dit();    dit();
    break;
  case '(':
    dah();   dit();    dah();    dah();    dit();
    break;
  case ')':
    dah();   dit();    dah();    dah();    dit();    dah();
    break;
  case '"':
    dit();    dah();    dit();    dit();    dah();    dit();
    break;
  case '.':
    dit();    dah();    dit();    dah();    dit();    dah();
    break;
  case ',':
    dah();   dah();    dit();    dit();    dah();    dah();
    break;
  case ':':
    dah();   dah();    dah();    dit();    dit();    dit();
    break;
  case '-':
    dah();   dit();    dit();    dit();    dit();    dah();
    break;
  case '=':
    dah();   dit();    dit();    dit();    dah();
    break;
  }
}
```