Eduardo Ruiz Moreno

Arduino: decodificador y codificador de código Morse

El propósito de este proyecto es crear un programa de Arduino que sea capaz de traducir a morse una palabra o pequeña frase o viceversa. En teoría, se podría utilizar el mismo Arduino para ambos propósitos, pero por falta de espacio en la placa de pruebas se dividirá el proyecto en dos programas.

Ambos programas tendrán parte del código en común, pero se diferenciarán en cuanto a los dispositivos de entrada/salida utilizados.

Para empezar, habrá que declarar un par de vectores que contendrán la traducción al morse de cada número y letra del abecedario.

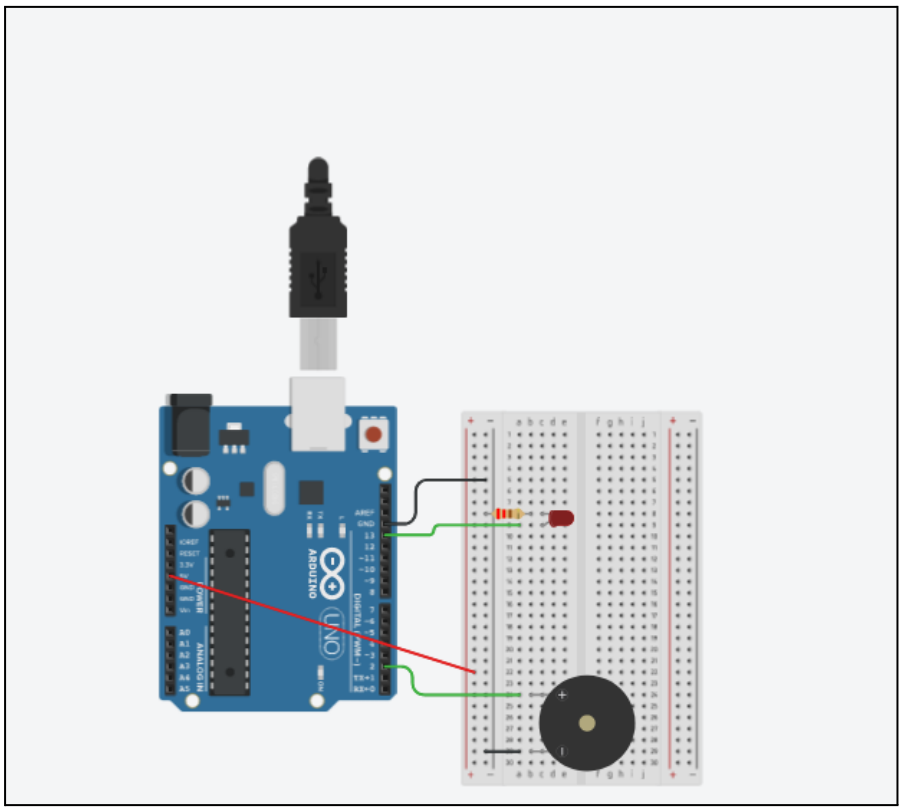
```
char* letters[] =
{
".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", // A-I
".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", // J-R
"...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.." // S-Z
};

char* numbers[] =
{
"-----", ".----", "..---", "...--", "....-", //0-4
".....", "-....", "--...", "---..", "----." //5-9
};
```

Además, se definirá una variable que indicará el tiempo de duración de un punto de código morse. El resto de medidas de tiempo derivan del punto. En este caso:

```
int dotLength = 150;

int dashLength = dotLength*3;  //Duración de una raya
int letterSpace = dotLength*3;  //Espacio entre letras
int wordSpace = dotLength*7;  //Espacio entre palabras
```

Por último, ambos proyectos usan una luz led para dar una indicación visual de que el programa funciona correctamente.

Codificador



Este proyecto es mucho más simple que el decodificador en cuanto a hardware. Solo se necesitará de una luz led y un *buzzer*. Ambos cumplen el mismo propósito: repetir, ya sea con sonido o luz, la palabra traducida a morse.

Setup

```
void setup()
{
  delay(500);
  pinMode(ledPin, OUTPUT);
  pinMode(tonePin, OUTPUT);
  Serial.begin(9600);
  Serial.println();
  Serial.println("-------------------------------");
  Serial.println("Morse Code encoder");

  Serial.println();
  Serial.println("-------------------------------");
  Serial.println("Type text and press Enter");
  Serial.println("-------------------------------");
}
```

Para el *setup()* simplemente se iniciarán los elementos hardware y se le indicará al programa que se va a utilizar el monitor en serie. Desde dicho monitor introduciremos la palabra o frase que queremos traducir y mostrará cada letra introducida junto a su traducción.

Loop

Una vez se haya completado el setup y el monitor en serie esté funcionando el programa espera a que se introduzca algo en el monitor. Una vez el usuario ha introducido algo desde el teclado, se realizan una serie de comprobaciones:

```
if (ch >= 'a' && ch <= 'z')
        { ch = ch-32; }
```

Primero, si el mensaje introducido es una letra minúscula, el programa la cambia a su equivalente en mayúscula restando el código ascii por 32.

```
if (ch >= 'A' && ch <= 'Z')
{
        Serial.print(ch);
        Serial.print(" ");
        Serial.println(letters[ch-'A']);
        flashSequence(letters[ch-'A']);
        delay(letterSpace);
}
```

Después, en el caso de que se haya introducido una letra, se procede a su traducción. En el monitor se imprimen tanto la letra introducida como su traducción. El vector declarado al principio del programa está ordenado en orden alfabético, por lo que para acceder a la traducción apropiada simplemente habrá que restar la letra introducida por A.

Por ejemplo, en el caso de que se haya introducido “A” (cuyo número Ascii es 65) se restará por 65 y por lo tanto se accederá a la posición 0 del vector, donde está almacenada la traducción del carácter “A”.

También se llama a la función flashSequence, explicada a profundidad más adelante en el documento. Servirá para activar los dispositivos hardware.

```
if (ch >= '0' && ch <= '9')
{
        Serial.print(ch);
        Serial.print(" ");
```

```
        Serial.println(numbers[ch-'0']);
        flashSequence(numbers[ch-'0']);
        delay(letterSpace);
}
```

Si se ha introducido un número se realiza el mismo proceso, excepto que se accede al otro vector y se resta por “0” (48 en ASCII).

Por último, si hay un espacio se imprime “_” y habrá un *delay* correspondiente a *wordSpace*.

Funciones Extra

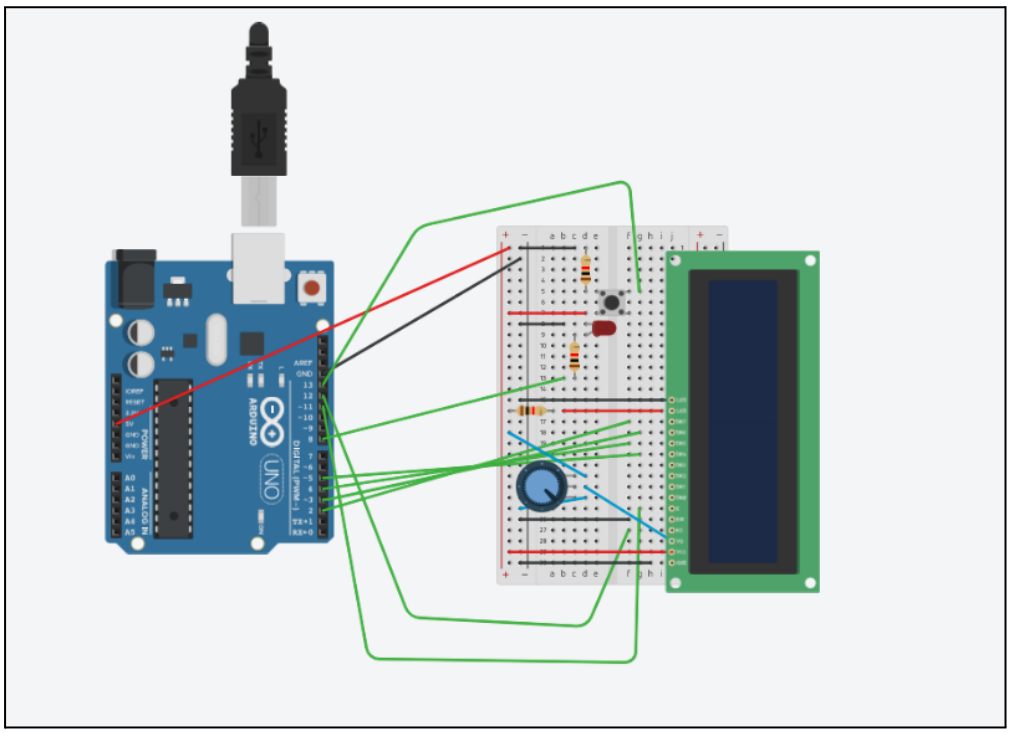
```
void flashSequence(char* sequence)
{
  int i = 0;
  while (sequence[i] == '.' || sequence[i] == '-')
  {
        flashDotOrDash(sequence[i]);
        i++;
  }
}

void flashDotOrDash(char dotOrDash)
{
  digitalWrite(ledPin, HIGH);
  tone(tonePin, toneFreq);
  if (dotOrDash == '.')
   { delay(dotLength); }
  else
   { delay(dashLength); }

  digitalWrite(ledPin, LOW);
  noTone(tonePin);
  delay(dotLength);
}
```

Estas funciones se ocupan de controlar el funcionamiento de la luz led y el *buzzer*. La primera llama a la segunda, pasándole como argumento cada componente de la letra o número en morse.

La segunda función inicia el funcionamiento de los componentes de salida y luego espera un determinado tiempo dependiendo de si el componente es un punto o una

raya. Una vez ha esperado ese tiempo, se paran los elementos hardware y vuelve a esperar un poco antes de seguir con el resto de la letra.

Decodificador



Para este proyecto se utilizará principalmente una pantalla lcd para mostrar la palabra traducida, complicando el desarrollo físico. Como método de entrada se utilizará un botón y, adicionalmente, habrá un led que se iluminará cuando se pulse el botón.

Setup

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup()
{
  delay(500);
  pinMode(buttonPin, INPUT_PULLUP);
  lcd.begin(16, 2);
  lcd.print("Morse decoder.");
```

```
  delay(500);
  lcd.clear();


  newLetter = false;
  newWord = false;
  keyboardText = false;
}
```

Se inicializan los componentes y se les da valor *falso* a una serie de variables que serán relevantes más adelante.

Cabe destacar que para trabajar con una pantalla lcd hay que utilizar la biblioteca apropiada.

Loop

```
if (digitalRead(buttonPin) == HIGH )
  {
        digitalWrite(ledPin, HIGH);
        newLetter = true;
        newWord = true;
        t1=millis(); //time at button press

        delay(debounceDelay);
        while (digitalRead(buttonPin) == HIGH )
        {delay(debounceDelay);}

        delay(debounceDelay);

        t2 = millis();
        onTime=t2-t1;

        digitalWrite(ledPin, LOW);
        delay(debounceDelay);

//check if dot or dash

        if (onTime <= dotLength*1.5)  //Se multiplica para permitir variación de tiempo
        {dashSeq = dashSeq + ".";}
        else
        {dashSeq = dashSeq + "-";}
  }
```

El programa espera hasta que se pulse el botón y calcula la duración de dicha pulsación. Dependiendo del tiempo calculado se guarda en un string un carácter “.” o “-”.

También se declara el inicio de una nueva letra y palabra y se enciende un led mientras se esté pulsando el botón.

```
gap=millis()-t2;
  if (newLetter == true && gap>=letterSpace)
  {
        for (i=0; i<=25; i++)
        {
              if (dashSeq == letters[i])
              {
                    keyLetter = i+65;
                    letterFound = true;
                    break ;
              }
        }
```

A continuación, si se ha empezado una nueva letra y no se ha cumplido el tiempo de separación entre letras se busca si la secuencia de puntos y rayas está en el vector de letras.

```
if(letterFound == false) //now check for numbers
{
        for (i=0; i<=10; i++)
        {
              if (dashSeq == numbers[i])
              {
                    keyLetter = i+48;
                    letterFound = true;
                    break ;         //don't keep checking if number found
              }
        }
}
```

Lo mismo pero para números.

Tras la búsqueda se imprime el número o letra en la pantalla (si no se encuentra la secuencia en ninguno de los vectores se imprime un error) y se declara que ha terminado la letra.

```
if (newWord == true && gap>=wordSpace*1.5)
        {
        newWord = false;
        lcd.print(" ");
        lineLength=lineLength+1;
        }
```

Si no se pulsa el botón antes de que se supere *wordSpace* se declara que la palabra ha terminado y se imprime un espacio. Si no se pulsa el botón durante aún más tiempo se borra la pantalla y se empieza de nuevo.