# Examen Tecnico - Meli Morse

|  |  |
| --- | --- |
| Fecha | 22/12/2020 |
| Autor | Christian Di Costanzo |
| GitHUB | https://github.com/ChristianDC178/meli\_morse.git |

Contenido

[Examen Tecnico - Meli Morse 1](#_Toc59545030)

[Enunciado del problema 1](#_Toc59545031)

[Analisis del problema 1](#_Toc59545032)

[Desarrollo del algoritmo 3](#_Toc59545033)

[Explicacion del proyecto y codigo fuente 3](#_Toc59545034)

[Explicacion de MeliMorse.App 4](#_Toc59545035)

[Como probar el mensaje enviado en la consigna del examen 4](#_Toc59545036)

[Explicacion de decodeBits2Morse 4](#_Toc59545037)

[Como se puede compilar el proyecto para levantarlo 6](#_Toc59545038)

## Enunciado del problema

Un problema implícito para la interpretación de los mensajes en MORSE, es la velocidad de

trasmisión de los mismos. Naturalmente la velocidad de transmisión del código varía

ligeramente a lo largo del mensaje ya que, es enviada por un operador humano .

Veamos un ejemplo, el texto “HOLA MELI" en MORSE sería el siguiente:

**.... --- .-.. .- -- . .-.. ..**

Una transmisión en bits podría representarse de la siguiente forma (considerando el bit 1

como un pulso y el 0 como pausa):

**0000000011011011001110000011111100011111100111111000000011101111111101110**

**1110000000110001111110000011111100111111000000011000011011111111011101110**

**0000011011100000000000**

Como se puede ver, esta transmisión es generalmente precisa de acuerdo con el estándar,

pero algunos puntos, trazos y pausas son un poco más cortos o un poco más largos que los

demás en el mismo mensaje.

Tener en cuenta que no necesariamente la frecuencia de envío es constante (dado que es

un operador humano) pero sí se considera que la representación es consistente durante

todo el mensaje. Es decir, si un operador es lento o rápido, seguirá siendo lento o rápido a

lo largo del envío de ese mensaje. Ejemplificando, si los puntos para cierto operador son de

largo menor o igual a 5 bits, no pasará durante el envío que desee representar un trazo de

largo 4 o 5. El mismo concepto aplica a las pausas para la interpretación de la separación

entre letras, palabras y finalización del mensaje.

## Analisis del problema

Lo primero que se hizo para analizar el problema es identificar el mensaje representado en Bits que seria un mensaje en morse tomando el que indica el enunciado:

**0000000011011011001110000011111100011111100111111000000011101111111101110**

**1110000000110001111110000011111100111111000000011000011011111111011101110**

**0000011011100000000000**

**“Hola Meli”**

Ahora necesita desglozar el mensaje para representarlo en morse y luego en texto, se llega al siguiente analisis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bits | Cantidad | Morse | Letra Morse | Letra |
| 00000000 | 8 |  |  |  |
| 11 | 2 | . | .... | H |
| 0 | 1 |  |
| 11 | 2 | . |
| 0 | 1 |  |
| 11 | 2 | . |
| 00 | 2 |  |
| 111 | 3 | . |
| 00000 | 5 |  |  |  |
| 111111 | 6 | - | --- | O |
| 000 | 3 |  |
| 111111 | 6 | - |
| 00 | 2 |  |
| 111111 | 6 | - |
| 0000000 | 7 |  |  |  |
| 111 | 3 | . | .-.. | L |
| 0 | 1 |  |
| 11111111 | 8 | - |
| 0 | 1 |  |
| 111 | 3 | . |
| 0 | 1 |  |
| 111 | 3 | . |
| 0000000 | 7 |  |  |  |
| 11 | 2 | . | .- | A |
| 000 | 3 |  |
| 111111 | 6 | - |
| 00000 | 5 |  |  |  |
| 111111 | 6 | - | -- | M |
| 00 | 2 |  |
| 111111 | 6 | - |
| 0000000 | 7 |  |  |  |
| 11 | 2 | . | . | E |
| 0000 | 4 |  |  |  |
| 11 | 2 | . | .-.. | L |
| 0 | 1 |  |
| 11111111 | 8 | - |
| 0 | 1 |  |
| 111 | 3 | . |
| 0 | 1 |  |
| 111 | 3 | . |
| 000000 | 6 |  |  |  |
| 11 | 2 | . | .. | I |
| 0 | 1 |  |
| 111 | 3 | . |
| 00000000000 | 11 |  |  |  |

Se puede observar que la diferencia entre un punto y una linea en morse es de 2 a mas bits de diferencia , es decir si un punto es entre 2 y 3 , una linea podria ser representada desde 5 en adelante.

El analisis para la separacion de letras y palabras es un poco mas dificil de detectar ya que el patron no lleva una consistencia a lo largo del mensaje, hay separacion de letra con 1 , 2 y 6 bits, mientras lo que se observa en la separacion de palabras se puede ver con 5 bits, y tampoco existe otra muestra como para hacer otro analisis.

## Desarrollo del algoritmo

Para desarrollar el algoritmo de lectura del patron en bits se tomaron las siguientes pautas en base al enunciado:

* No todos los operadores tienen la misma velocidad en el envio del mensaje.
  + De este punto se podria suponer que para 3 operadores diferentes los puntos y rayas podrian ser de la siguiente forma:
    - Operador A:
      * Punto: 3 a 4 bits
      * Raya: 5 a n bits
    - Operador B:
      * Punto: 5 a 6 bits
      * Raya: 7 a n bits
    - Operador C:
      * Punto: 8 a 9 bits
      * Raya: 10 a n bits
* Por lo tanto para llevarlo a una formula:

Secuencia minima de bits = para rayas y puntos se toman los 1

* + Punto = (secuencia minima de bits + 1)
  + Raya = secuencia de bits – secuencia minima >= 2 bits de diferencia
* Para la separacion entre simbolos y letras se uso un analisis similar solo que:

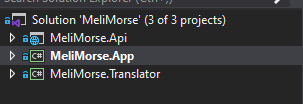
Secuencia minima de bits = para las separaciones se toman los 0

* + Separacion entre letras = secuencia de bits > ( secuencia minima + 2)

## Explicacion del proyecto y codigo fuente

Las caracteristicas del proyecto desarrollado son las siguientes:

|  |  |
| --- | --- |
| IDE | Visual Studio 2019 ( se puede usar VS Code) |
| Framework | NET Core 3.1 |
| Tipo de App | Consola para las pruebas y tiene la api rest |

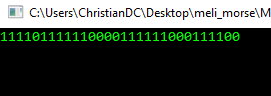


* **MeliMorse.Api** : contiene el proyecto para ejecutar la api que estaba requerida como plus
* **MeliMorse.App**: es una aplicacion de consola que sirve para hacer las pruebas
* **MeliMorse.Translator**: libreria que tiene una clase helper con el decoder , MorseDecoder donde contiene todos los metodos requeridos en el examen.

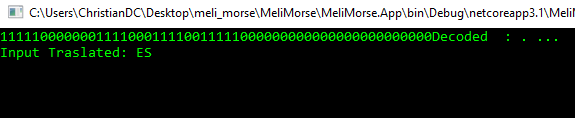
### Explicacion de MeliMorse.App

Se trata de una app de consola que cuando se ejecuta permite ingresar codigo morse, para simular los inputs fisicos se debe presionar la tecla 1 ya que no estan validadas las demas teclas por temas de tiempo. Por ejemplo para ingresar 111 , se debe presionar 1 x 3 y luego enter.

Una vez que se comienza a ingresar codigo , corre un contador para tomar en cuenta las pausas , dicho contador esta puesto en 800ms , luego se divide la pausa total en ms / 800 y se toma la parte entera del resultado para agregar 0.

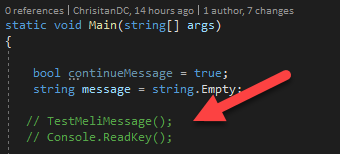


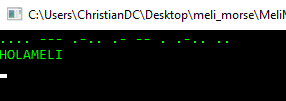
Cuando la pausa es igual o mayor a 10 bits se considera fin del mesaje y se muestra el morse por pantalla.



### Como probar el mensaje enviado en la consigna del examen

Descomentando las siguientes 2 lineas se puede usar la version de prueba que toma la secuencia de bits indicada en el enunciado del examen.



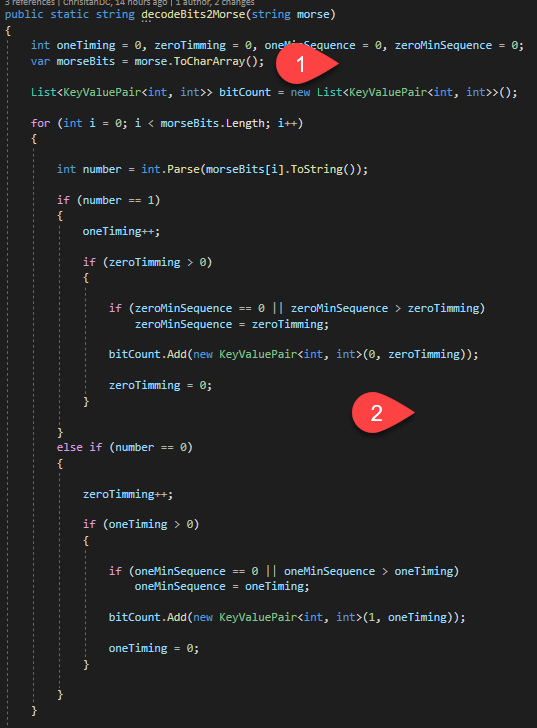


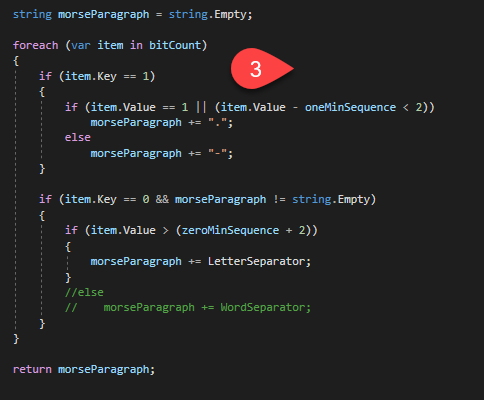
### Explicacion de decodeBits2Morse

El algoritmo hace lo siguiente:

1. Toma como entrada una cadena de bits emulada en un string, esto se hizo asi solamente como ejemplo y para hacerlo mas rapido , se podria tomar un array de bytes.
2. Itera por cada bit y va identificando lo siguiente:
   1. Timing: cantidad de veces seguidas del mismo bit , o sea 0 o 1
   2. Saca el valor minimo de ambas secuencias de bit para luego calcular que quizo introducir el usuario
   3. Agrupa en un listado con un Clave-Valor ( Key Value Pair ) el bit con la secuencia es decir por ejemplo:

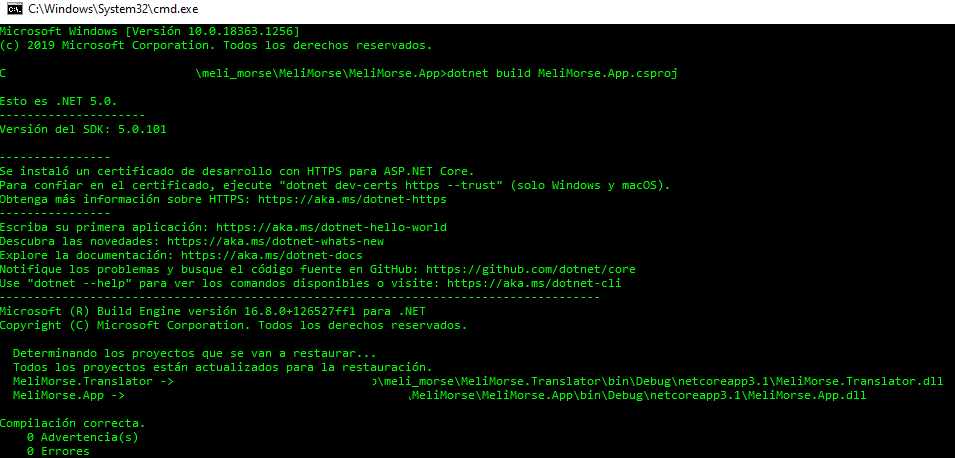
|  |  |
| --- | --- |
| Listado | |
| Key | Value |
| 0 | 6 |
| 1 | 3 |
| 0 | 2 |
| 1 | 3 |
| 0 | 6 |
| 1 | 4 |
| 0 | 2 |

3 – Luego que tiene todo identificado decodifica la secuencia usando el listado del punto 2. 



## Como se puede compilar el proyecto para levantarlo

Dentro del directorio MeliMorse.App ejecutar el siguiente comando: dotnet build MeliMorse.App.csproj



Correr el proyecto con el siguiente comando:

Dotnet run .\bin\debug\netcoreapp3.1\MeliMorse.app.exe

