

2018/2019

Andrea Calleja Pedrote 53856. Stella Moral Salguero 54112. Marta Vinader Vega 54259.

Ingeniería ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA. etsidi.

TRADUCTOR DE CÓDIGO MORSE

# DOCUMENTACIÓN.

# OBJETIVOS.

El objetivo principal de nuestro proyecto es realizar un traductor de código Morse.

Este programa sería capaz de convertir un mensaje a puntos y rayas, es decir, código Morse.

La innovación de nuestro proyecto reside en que es capaz de detectar cuándo un mensaje es de peligro o cuando se trata un mensaje rutinario. Además, incorpora de forma visual luces de peligro (rojas) o luces que nos inspiran que el mensaje es un mensaje corriente (verde).

Además, para hacer más realista el proyecto, cada mensaje será emitido mediante sonidos a través de una bocina de forma que sea más tangible la traducción.

# MATERIAL UTILIZADO.

* Ordenadores.
* Compiladores de lenguaje C: DevC++, Visual Studio.
* Arduino.
* Placa de Arduino One (*protoboard*)
* Leds (verde y rojo)
* Bocina.
* Elementos de conexión entre leds, bocina y placa (cables)
* Cable de impresora para la conexión ordenador-*protoboard*.
* Resistencias de (220 ± 5%) Ω

# PROCEDIMIENTO SEGUIDO.

En primer lugar, nos reunimos todas con diferentes ideas, queríamos elaborar un proyecto diferente, creativo y distinto de un ascensor, semáforo, etc.

Empezamos a identificar que en nuestro proyecto queríamos tener luces y sonido, y la idea acabó convergiendo en un traductor. La forma de enlazar las luces y el sonido, en definitiva, una forma visual de hardware con un código que tuviera que ver con una traducción, fue un traductor de mensaje a código Morse.

Una vez tuvimos la idea comenzamos a elaborar el código, donde tuvimos diferentes problemas con la sintaxis, la algoritmia… Pero se fueron solucionando a medida que íbamos adquiriendo más conocimiento, entre las clases y las tutorías conseguimos solventarlo.

Después de realizar el código pasamos a montar el hardware, para ello tuvimos que buscar en diversos tutoriales/documentos como podíamos conectar los dos Leds que íbamos a utilizar con la bobina (si en serie o en paralelo) e ir asignando a cada elemento un pin dentro de la *protoboard*.

Una vez tuvimos el hardware listo, nos dedicamos a la conexión DevC- Arduino.

Hemos de decir que esta última parte del trabajo ha sido la más costosa, ya que nos encontramos con que no teníamos ninguna base y ante un problema que debíamos solucionar.

Para ello, investigamos acerca de la conexión y con diversas tutorías aprendimos a conectar Arduino con nuestro código.

En definitiva, el esquema básico que sigue nuestro programa sería el siguiente:

1. El usuario introduce el motivo de su mensaje: si se trata de un mensaje de alerta o peligro o, por el contrario, es un mensaje rutinario.
2. A continuación, el usuario introducirá el mensaje a traducir, teniendo en cuanta una serie de restricciones, dado que cierto símbolos no tienen traducción en Morse.

Hemos tenido en cuenta que el usuario quizás escriba letras mayúsculas. Para ello, el programa calculará su equivalente en minúsculas antes de traducirlas.

1. El programa procederá a traducir la frase del usuario a código Morse. Accederá a un fichero donde hemos escrito el alfabeto Morse en el que se debe basar, para que el código en C quede más limpio.
2. A continuación, se hará la conexión con el Arduino y se encenderá el led respectivo, junto con la bocina, dependiendo de si se trata de puntos o de rayas. Para los primeros la duración será menor que para los segundos.
3. Por último, se le mostrará la opción al usuario de seguir con el programa, a no ser que pulse una letra determinada.

# PROBLEMAS ENCONTRADOS.

El problema principal al cual nos enfrentamos fue a la hora de realizar la conexión entre el código de DevC y el de Arduino, ya que no teníamos ninguna base sobre qué sentencias escribir en un código u otro para ello, qué bibliotecas utilizar, etc. Cómo “encajar” un código en el otro, básicamente. Podríamos haber presentado cada código por su cuenta, pero nuestro objetivo era hacer un solo proyecto que integrara todos los códigos.

Otro problema relacionado con el anterior sería que, al tener que encender simultáneamente un led y la bocina, no podíamos hacer un *delay* entre un estado del led y otro ya que, de esta forma, el programa se quedaría parado y la bocina no emitiría sonido alguno. Para solucionarlo, usamos la medida del tiempo transcurrido entre dos eventos, sin necesidad, por tanto, de bloquear la ejecución con un *delay*.

Por otro lado, hemos visto que nuestros conocimientos en cuanto a todo lo relacionado con Arduino son bastante escasos. Hemos tenido dudas a la hora de escribir este código, las funciones asignadas, sus bibliotecas e, incluso al hacer las conexiones. Creemos que realmente es una herramienta muy útil que, bien utilizada, puede dar mucho juego y resolver numerosos problemas. Nos gustaría, por tanto, que se ahondara más en este tema en el futuro.

En general, estamos bastante satisfechas con los resultados, dado que los problemas comentados anteriormente los hemos ido solucionando con ayuda del profesor en tutorías, investigando por nuestra cuenta o incluso poniendo en común nuestras diferentes ideas.