

MEMORIA DEL PROYECTO

Informática



aránzazu carnero tallón (53864) VICTORIA JIMÉNEZ BARBERO (54019)

Profesora Cecilia García Cena

1. **Objetivo**

El objetivo de este proyecto consiste en traducir un mensaje de texto introducido por teclado a código Morse, a través de Arduino, incluyendo luces y sonidos.

1. **Alcance**

El usuario indicará por teclado el motivo de su mensaje, es decir, si se trata de un mensaje de peligro o pidiendo ayuda, o por el contrario, si se trata de un mensaje ordinario. Dependiendo de la opción seleccionada, se encenderá un LED de un color u otro.

Debemos aclarar que está opción en la que el usuario indica el motivo de su mensaje sólo la hemos introducido en el código hecho en visual para obtener un código más complejo, dado que en un principio la idea era que simplemente se encendiera el LED.

A continuación, se le pedirá al usuario que escriba su correspondiente mensaje.

Por último, éste se traducirá a código Morse, es decir, cada letra a sus respectivos “puntos” y “rayas”. La secuencia creada saldrá en forma de pitidos por un zumbador y en forma de destellos luminosos por el LED.

1. **Lista de tareas**

El programa pedirá al usuario que elija el color del LED, dependiendo del motivo de su mensaje. Se indicará a qué color de LED corresponde cada tipo.

A pesar de que la idea principal era retransmitir la orden de socorro SOS, que es el código Morse más conocido, lo hemos adaptado para que sea el usuario quien introduzca el mensaje, creando un código más complejo con posibilidades infinitas. De esta forma conseguimos que el usuario interactúe de manera más activa en el proceso.

El programa esta diseño de tal manera, que en el caso de que algún caracter introducido sea incorrecto o no esté contenido en el código Morse, el programa notifique un mensaje de error, pidiendo al usuario que vuelva a introducir dicho mensaje.

La base del código consistirá en un *switch* que, a forma de abecedario, contendrá la traducción en forma de puntos y rayas de cada letra y número, incluyendo también la traducción de los signos más comunes (como pueden ser puntos, interrogaciones, exclamaciones, etc). Al programa también se le asignará la duración del pitido de la bocina y del destello del LED dependiendo de si es un punto o una raya, y el tiempo de apagado entre cada caracter.

Este proceso se repetirá de forma ininterrumpida con un bucle, hasta que el usuario decida salir del programa pulsando la tecla correspondiente, indicada en pantalla.

1. **Código C++:**

//Aránzazu Carnero Tallón (53864); Victoria Jiménez Barbero (54019)

//CÓDIGO MORSE. El usuario introduce el motivo del mensaje (alerta o normal), y el mensaje.

//El programa revisa que el mensaje este correctamente escrito, y lo traduce a código morse.

//El programa crea un fichero que contiene todas las frases que han sido introducidas durante el mismo, y el motivo de las mismas.

//El traductor se repite hasta que el usuario lo desee.

#include<stdio.h>

#include<windows.h>

#include<string.h>

#include<stdlib.h>

#define N 100

struct DATOS {

int opcion;

char cad[N], \*c;

};

int verificacion(struct DATOS);//verificacion del mensaje introducido por el usuario en la funcion main

void imprimir(struct DATOS);//traduccion del mensaje introducido por el usuario en la funcion main

int main()

{

struct DATOS datos;

int correcto;

FILE \*pf;

errno\_t err;

err = fopen\_s(&pf, "LIBRETA.txt", "a+");// fichero que recoge todos los mensajes introducidos por el usuario, y su motivo

do

{

printf("introduce el motivo del mensaje:\n1- mensaje de emergencia\n2-mensaje ordinario\n0-salir del programa\n");

scanf\_s("%d", &datos.opcion);

fprintf(pf, "%d\t", datos.opcion);

if (datos.opcion != 0 && datos.opcion != 1 && datos.opcion != 2)

{

printf("opcion intriducida no valida, vuelva a intentarlo\n");

}

switch (datos.opcion)

{

case 1:

printf("LED rojo ON\n");

do

{

printf("introduzca el mensaje a traducir. Puede escribir los simbolos mas comunes, espacios, y letras minusculas\n");

getchar();

gets\_s(datos.cad);//introduccion por el usuario del mensaje, para su proxima verificacion y traduccion.

fprintf(pf, "%s", datos.cad);

c = (char\*)calloc(strlen(cad) + 1, sizeof(char));

if (cad == NULL)

{

printf("NO HAY SUFICIENTE ESPACIO\n");

exit(-1);

}

if (verificacion(datos) == 1)

{

imprimir(datos);

}

} while (verificacion(datos) == 0);

break;

case 2:

printf("LED verde ON\n");

do

{

printf("introduzca el mensaje a traducir. Puede escribir los simbolos mas comunes, espacios, y letras minusculas\n");

getchar();

gets\_s(datos.cad);//introduccion por el usuario del mensaje, para su proxima verificacion y traduccion.

fprintf(pf, "%s", datos.cad);

c = (char\*)calloc(strlen(cad) + 1, sizeof(char));

if (cad == NULL)

{

printf("NO HAY SUFICIENTE ESPACIO\n");

exit(-1);

}

if (verificacion(datos) == 1)

{

imprimir(datos);

}

} while (verificacion(datos) == 0);

break;

}

} while (datos.opcion != 0);

fclose(pf);

system("pause");

}

int verificacion(struct DATOS datos) //funcion para comprobar que el mensaje introducido por el usuario es correcto. Sino, lo vuelve a pedir

{

int i = 0, correcto = 0;

while (datos.cad[i] != '\0')

{

if (datos.cad[i] >= 'a' && datos.cad[i] <= 'z' || datos.cad[i] >= '0' && datos.cad[i] <= '9' || datos.cad[i] == '.' || datos.cad[i] == ',' || datos.cad[i] == '?' || datos.cad[i] == '"' || datos.cad[i] == '!' || datos.cad[i] == ' ')

{

correcto++;

}

i++;

}

if (correcto == strlen(datos.c))//por cada caracter de la frase bien, sumamos 1 unidad al valor de correcto. si todos los caracteres de la frase estan bien, correcto= longitud frase.

{

correcto = 1;

}

else

{

correcto = 0;

printf("frase introducida incorrecta, vuelva a intentarlo\n");

}

return correcto;

}

void imprimir(struct DATOS datos)//funcion para imprimir por pantalla el codigo morse

{

int i = 0;

while (datos.cad[i] != '\0')

{

switch (datos.cad[i])

{

case 'a':

{

printf(".- ");

break;

}

case 'b':

{

printf("-... ");

break;

}

case 'c':

{

printf("-.-. ");

break;

}

case 'd':

{

printf("-.. ");

break;

}

case 'e':

{

printf(". ");

break;

}

case 'f':

{

printf("..-. ");

break;

}

case 'g':

{

printf("--. ");

break;

}

case 'h':

{

printf(".... ");

break;

}

case 'i':

{

printf(".. ");

break;

}

case 'j':

{

printf(".--- ");

break;

}

case 'k':

{

printf("-.- ");

break;

}

case 'l':

{

printf(".-.. ");

break;

}

case 'm':

{

printf("-- ");

break;

}

case 'n':

{

printf("-. ");

break;

}

case 'o':

{

printf("--- ");

break;

}

case 'p':

{

printf(".--. ");

break;

}

case 'q':

{

printf("--.- ");

break;

}

case 'r':

{

printf(".-. ");

break;

}

case 's':

{

printf("... ");

break;

}

case 't':

{

printf("- ");

break;

}

case 'u':

{

printf("..- ");

break;

}

case 'v':

{

printf("...- ");

break;

}

case 'w':

{

printf(".-- ");

break;

}

case 'x':

{

printf("-..- ");

break;

}

case 'y':

{

printf("-.-- ");

break;

}

case 'z':

{

printf("--.. ");

break;

}

case '0':

{

printf("----- ");

break;

}

case '1':

{

printf(".---- ");

break;

}

case '2':

{

printf("..--- ");

break;

}

case '3':

{

printf("...-- ");

break;

}

case '4':

{

printf("....- ");

break;

}

case '5':

{

printf("..... ");

break;

}

case '6':

{

printf("-.... ");

break;

}

case '7':

{

printf("--... ");

break;

}

case '8':

{

printf("---.. ");

break;

}

case '9':

{

printf("----.");

break;

}

case '.':

{

printf(".-.-.- ");

break;

}

case ',':

{

printf("--..-- ");

break;

}

case '?':

{

printf("..--.. ");

break;

}

case '"':

{

printf(".-..-. ");

break;

}

case '!':

{

printf(".-.-. ");

break;

}

case ' ':

{

printf(" ");

break;

}

}

i++;

}

printf("\n");

}