Universidad Politécnica de Madrid

Grado en Ingeniería Electrónica y Automática

Asignatura

INFORMÁTICA

Curso 2018-2019

Universidad Politécnica de Madrid

Grado en Ingeniería Electrónica y Automática

Datos del Grupo

David Hergueta Soto (54666)
Guillermo Aparicio Voces (54483)

Introducción

La intención de nuestro proyecto ha sido la de realizar una alarma automática con ayuda de Arduino, y poder controlarla a través de un terminal PC, para ello hemos realizado un programa principal escrito en código C en el que hay diversas opciones con las que el usuario interactúa y con ayuda de comunicación serial, transmite las órdenes al controlador Arduino de la alarma.

La alarma necesita de una contraseña para funcionar, que se guardará en un archivo permanente de texto, de manera que cada vez que se inicie el programa la lea desde ahí y no sea necesario definirla. Si no hay ninguna creada, el usuario deberá crearla al iniciar el programa, además será necesario que Arduino permanezca conectado mientras se ejecute el programa para que funcione correctamente.

Desde el programa principal el usuario podrá: activar la alarma, desactivar la alarma, comprobar el registro de la alarma que el programa escribirá cada vez que la alarma es activada, desactivada o detecte alguna presencia; o cambiar la contraseña que haya ya definida.

En este documento queremos recoger una explicación del funcionamiento de nuestro programa así como incluir un flujograma de este para facilitar su comprensión, igualmente hemos subido un video a drive con una demo de funcionamiento del mismo:

https://drive.google.com/file/d/1vg8UWxDDfZtwGWAotqxK1oqjC9f1K7x0/view

Desarrollo

Programa principal

Código

Este es el código del programa principal que hemos implementado en nuestro proyecto de Visual Studio.

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<malloc.h>
#include<time.h>
//Biblioteca comunicación serial arduino
#include"SerialPort.h"

#define length 16 //Longitud máxima permitida en la clave de seguridad
de la alarma
#define MAX_DATA_LENGTH 255
#define ON 1
#define OFF 0
#define DETECCION 2

//Funciones para comunicacion serial
```

```
void autoConnect(SerialPort *arduino, char*, char*);
void Crear_Conexion(SerialPort *PuertoSerie, char *portName);
void CerrarConexion(SerialPort * PuertoSerie);
int readSerialPort(SerialPort * PuertoSerie, char *buffer, unsigned
int buf_size);
int writeSerialPort(SerialPort *PuertoSerie, char *buffer, unsigned
int buf_size);

// Funciones prototipo
char* Password(); //Función para introducir contraseña con asignación
dinámica
char* DefinePass(int flag); //Función para definir una contraseña para
la alarma. Si se quiere cambiar contraseña el flag debe ser 1, si no
se pone 0
void registro();//Funcion para ver cuando se activo la alarma
void fecha(int);//Funcion para escribir cuando se activa(ON),
desactiva(OFF) y detecta(DETECCION) la alarma en un txt
void Alarm(int flag, SerialPort *arduino, char* pass,char*
pass_aux,int* act); //Funcion para activar o desactivar la alarma

void main()
{
    char *pass; //Contraseña de la alarma

    printf("            SISTEMA DE ALARMAS KEEP'N YOU SAFE\n\n");
    pass = DefinePass(0); //indicamos la contraseña de la alarma

//CONEXION CON ARDUINO
    //Arduino SerialPort object
    SerialPort *arduino;
    // Puerto serie en el que está Arduino
    char* portName = "COM5";
    // Buffer para datos procedentes de Arduino
    char incomingData[MAX_DATA_LENGTH];

    // Crear estructura de datos del puerto serie
    arduino = (SerialPort *)malloc(sizeof(SerialPort));
    // Apertura del puerto serie
    Crear_Conexion(arduino, portName);
    autoConnect(arduino, incomingData, pass); //Dentro de esta funcion
escribiremos la aplicacion
}


void autoConnect(SerialPort *arduino, char *incomingData, char *pass)
{
    int act = 0; //Flag que se activa cuando la alarma esta activada

    // Espera la conexión con Arduino
    while (!isConnected(arduino))
    {
        Sleep(100);
        Crear_Conexion(arduino, arduino->portName);
    }
    //Comprueba si arduino está connectado
    if (isConnected(arduino))
    {
        system("CLS");
        printf("            SISTEMA DE ALARMAS KEEP'N YOU SAFE\n\n");
        printf("Conectado con Arduino en el puerto %s\n\n", arduino-
>portName);
```

```
        Sleep(7000);
    }

    // APLICACIÓN!! (Se ejecuta mientras el arduino esta conectado)
    while (isConnected(arduino))
    {
        char opc;
        char *pass_aux = NULL;

        readSerialPort(arduino, incomingData, MAX_DATA_LENGTH); //Lee
puerto serie
        if (incomingData[0]=='s') //Si el arduino ha detectado
presencia
        {
            Alarm(DETECCION, arduino, pass, pass_aux, &act);
            continue;
        }

        system("CLS");
        printf("            SISTEMA DE ALARMAS KEEP'N YOU SAFE\n\n");
        printf("Conectado con Arduino en el puerto %s\n\n", arduino-
>portName);

        printf("Que desea hacer?\n\n1.Activar alarma\n2.Desactivar
alarma\n3.Ver historial de detecciones\n4.Cambiar clave de
seguridad\n\n5.Salir\n\n");
        opc = _getch();

        //Si el arduino ha sido desconectado, se para el programa
        if (!isConnected(arduino))
        {
            break;
        }

        readSerialPort(arduino, incomingData, MAX_DATA_LENGTH); //Lee
puerto serie
        if (incomingData[0] == 's') //Si el arduino ha detectado
presencia
        {
            Alarm(DETECCION, arduino, pass, pass_aux, &act);
            continue;
        }

        switch (opc)
        {
            case '1':
            {
                if (act != 1)
                {
                    Alarm(ON, arduino, pass,pass_aux, &act);
                }

                break;
        }

            case '2':
            {
                if (act != 0)
                {
                    Alarm(OFF, arduino, pass,pass_aux, &act);
                }
```

```
            break;
            }

            case '3':
            {
                registro();
                break;
            }

            case '4':
            {
                printf("\nIntroduce la antigua clave de seguridad: ");
                pass_aux = Password();
                if (strcmp(pass, pass_aux) != 0)
                {
                    printf("Clave incorrecta!\n\n");
                    getch();
                    break;
                }
                else
                {
                    pass = DefinePass(1);
                    break;
                }
            }

            case '5':
            {
                printf("Esta seguro de que que quiere salir?
(s/n)\n\n");
                do
                {
                    opc = _getch();
                } while (opc != 's'&&opc != 'n');

                if (opc == 's')
                {
                    if (act != 0)
                    {
                        printf("Desactive primero la alarma\n");
                        Alarm(OFF, arduino, pass, pass_aux, &act);
                    }

                    if (act == 0)
                    {
                        printf("\nHasta luego!");
                        Sleep(700);
                        free(pass);
                        exit(0);
                    }
                    else //Si has introducido mal la clave para
desactivar la alarma vuelve al menu
                        break;
                }
                else if (opc == 'n')
                    break;
            }
            default:
                break;
        }
    }
```

```
    if (!isConnected(arduino))
    {
        printf("\nSe ha perdido la conexion con Arduino\n");
        free(pass);
        printf("Pulse una tecla para continuar");
        _getch();
    }
}

void Alarm(int flag, SerialPort *arduino, char *pass,char *pass_aux,
int *act)
{
    char sendData;

    if (flag != DETECCION)
    {
        printf("Introduzca la clave de seguridad: ");
        pass_aux = Password();
    }

        if (flag == ON)
        {
            if (strcmp(pass_aux, pass) == 0)
            {
                printf("Clave correcta\nAlarma activa\n\n");
                fecha(ON);

                sendData = 'a'; //Activa arduino
                writeSerialPort(arduino, &sendData, sizeof(char));
                *act = 1; //Flag para indicar que la alarma esta
activada
                printf("Pulse una tecla para continuar");
                _getch();
            }
            else
            {
                printf("Clave incorrecta\n\n");
                printf("Pulse una tecla para continuar");
                _getch();
            }
        }
    else if (flag == OFF)
    {
        if (strcmp(pass_aux, pass) == 0)
        {
            printf("Clave correcta\nAlarma desactivada\n\n");
            fecha(OFF);

            sendData = 'o'; //Desactiva arduino
            writeSerialPort(arduino, &sendData, sizeof(char));
            *act = 0; //Flag para indicar que la alarma se ha
desactivado
            printf("Pulse una tecla para continuar");
            _getch();
        }
        else
        {
            printf("Clave incorrecta\n\n");
            printf("Pulse una tecla para continuar");
            _getch();
```

```
        }
    }

    else if (flag == DETECCION)
        {
            fecha(DETECCION);
            do
            {
                system("CLS");
                printf("            SISTEMA DE ALARMAS KEEP'N YOU
SAFE\n\n");
                printf("PRESENCIA DETECTADA!\n\nIntroduzca clave para
desactivar: ");
                pass_aux = Password();
                if (strcmp(pass, pass_aux) != 0)
                {
                    printf("Clave incorrecta!!\n");
                    printf("Pulse una tecla para continuar");
                    _getch();
                }
                else
                {
                    printf("Clave correcta\nALARMA DESACTIVADA!\n\n");
                    fecha(OFF);
                    sendData = 'o'; //Desactiva el arduino
                    writeSerialPort(arduino, &sendData, sizeof(char));
                    *act = 0; //Flag para indicar que la alarma se ha
desactivado
                    printf("Pulse una tecla para continuar");
                    _getch();
                }
            } while (strcmp(pass, pass_aux) != 0);
        }
}

char* DefinePass(int flag)
{
    FILE *filepass;
    char *pass1, *pass2,*pass_aux;
    int i = 0;
    errno_t err_pass;

    //Asignacion de memoria para la contraseña
    pass1 = (char*)malloc(sizeof(char)*(length + 1));
    if (pass1 == NULL)
    {
        printf("No hay memoria disponible\n");
        printf("Pulse una tecla para continuar");
        _getch();
        exit(1);
    }
    *pass1 = NULL;

    pass_aux = (char*)malloc(sizeof(char)*(length + 1));
    if (pass_aux == NULL)
    {
        printf("No hay memoria disponible\n");
        printf("Pulse una tecla para continuar");
        _getch();
        exit(1);
    }
```

```
    *pass_aux = NULL;

    //Abrimos y leemos archivo de texto donde esta la contraseña
    err_pass = fopen_s(&filepass, "password.txt", "r");
    if (err_pass != NULL)
    {
        printf("El archivo password no se ha abierto corretamente\n");
        printf("Pulse una tecla para continuar");
        _getch();
        free(pass1);
        free(pass_aux);
        exit(1);
    }

    //Si el flag esta activado, se cambia la contraseña que esta ya
puesta
    if (flag == 1)
    {
        //Primero escribimos la clave que ya hay en un pass_auxiliar
        while (feof(filepass) == NULL)
        {
            fscanf_s(filepass, "%s", (pass_aux), _msize(pass_aux));
        }
        fclose(filepass);

        //Lo abrimos como solo escritura
        err_pass = fopen_s(&filepass, "password.txt", "w");
        if (err_pass != NULL)
        {
            printf("El archivo password no se ha abierto
corretamente\n");
            printf("Pulse una tecla para continuar");
            _getch();
            free(pass1);
            free(pass_aux);
            exit(1);
        }
        fprintf(filepass, "\0");
        fclose(filepass);

        //Lo volvemos a abrir como solo lectura
        err_pass = fopen_s(&filepass, "password.txt", "r");
        if (err_pass != NULL)
        {
            printf("El archivo password no se ha abierto
corretamente\n");
            printf("Pulse una tecla para continuar");
            _getch();
            free(pass1);
            free(pass_aux);
            exit(1);
        }
    }

    //Introducimos en pass1 la contraseña que haya definida en el
archivo
    while (feof(filepass) == NULL)
    {
        fscanf_s(filepass, "%s", (pass1), _msize(pass1));
    }
```

```
    //Reasignamos memoria para optimizar
    pass1 = (char*)realloc(pass1, strlen(pass1) + 1);
    fclose(filepass);

    //Si no habia ninguna contraseña previamente definida, primero la
creamos
    if (*pass1 == NULL)
    {
        do
        {
            printf("Defina una clave de hasta %d caracteres: ",
length);
            pass1 = Password();
            printf("Introducela de nuevo: ");
            pass2 = Password();

            if (strcmp(pass1, pass2) == 0)
                printf("Clave definida correctamente\n");
            else
            {
                printf("Las claves no coinciden\n");
                printf("Pulse una tecla para continuar");
                _getch();
                system("CLS");
                printf("            SISTEMA DE ALARMAS KEEP'N YOU
SAFE\n\n");
            }

            //Si queriamos cambiar la clave y no coincide, ponemos la
que habiamos grabado en el pass_auxiliar para no realizar cambios
            if (flag == 1 && strcmp(pass1, pass2) != 0)
            {
                strcpy(pass1, pass_aux);
                free(pass2);
                free(pass_aux);
                break;
            }
        } while (strcmp(pass1, pass2) != 0);

        //Abrimos de nuevo el archivo como solo escritura
        err_pass = fopen_s(&filepass, "password.txt", "w");
        if (err_pass != NULL)
        {
            printf("El archivo password no se ha abierto
corretamente\n");
            printf("Pulse una tecla para continuar");
            _getch();
            free(pass1);
            exit(1);
        }

        fprintf(filepass, "%s", pass1);
        fclose(filepass);
        if (strcmp(pass1, pass2) == 0)
        {
            free(pass_aux);
            printf("Pulse una tecla para continuar");
            _getch();
            free(pass2);
            system("CLS");
        }
```

```
        return pass1;
    }
    //Si ya existia una contraseña creada,  la funcion la devuelve
    else
        return pass1;
}

char* Password()
{
    char *pass, c = 0;
    int i;
    //Asignacion de memoria para la contraseña
    pass = (char*)malloc((length)+1);
    if (pass == NULL)
    {
        printf("No hay memoria disponible\n");
        printf("Pulse una tecla para continuar");
        _getch();
        exit(1);
    }

    //Compobacion de que como mucho tiene los caracteres que hemos
indicado. Cada vez que escribes un caracter, imprime *
    do
    {
        i = 0;

        while (c != 13) //Si no presionamos intro
        {
            if (i < (length + 1)) //Si la contraseña cabe en la
cadena, se va introduciendo
            {
                c = _getch();

                if (i == 0 && c == 8)
                    continue;

                if (c > 31 && c < 126)
                {
                    pass[i] = c;
                    printf("*");
                }

                else if (c == 8) //Si pulsa retroceso
                {
                    printf("\b \b"); //Mueve cursor y borra anterior
caracter
                    c = NULL;
                    i = i - 2;
                }
                else if (c != 13) //Eliminamos teclas especiales
                {
                    _getch();
                    continue;
                }
            }
            else //Si no entra siemplemente se imprimen los *, luego
te mandara repetir contraseña
            {
                c = _getch();
                if (c != 8 && c != 13)
```

```
                {
                    printf("*");
                }

                if (c == 8)
                {
                    printf("\b \b");
                    c = NULL;
                    i = i - 2;
                }
                else if (c != 13)
                {
                    _getch();
                    continue;
                }
            }
            if (c == 13) //Si pulsa intro
            {
                if (i < ((length)+1))
                    pass[i] = '\0';
                else
                {
                    pass[(length)+1] = '\0';
                }
                i--;
                printf("\n");
            }
            i++;
        }
        if (i > length)
            printf("Error de longitud, introducela de nuevo: ");
        if (i == 0)
            printf("La clave no puede estar en blanco, introducela de
nuevo: ");
        c = 0;
    } while (i > length || i < 1);

    //Reasignacion de memoria para optimizar
    pass = (char*)realloc(pass, i * sizeof(char) + 1);
    return pass; //Devuelve el puntero de la contraseña
}

void fecha(int flag) {
    time_t current_time;
    FILE *filetime;
    errno_t err1;
    err1 = fopen_s(&filetime, "history.txt", "a");
    if (err1 != NULL)
    {
        printf("El archivo history no se ha abierto corretamente\n");
        printf("Pulse una tecla para continuar");
        _getch();
        exit(1);
    }
    current_time = time(NULL);
    switch (flag)
    {
    case ON:
        fprintf(filetime, "La alarma se activo: %s",
ctime(&current_time));
        break;
```

```
    case OFF:
        fprintf(filetime, "La alarma se desactivo: %s",
ctime(&current_time));
        break;
    case DETECCION:
        fprintf(filetime, "Se ha detectado presencia: %s",
ctime(&current_time));
        break;
    }
    fclose(filetime);
}
void registro()
{
    FILE *filetime;
    errno_t err1;
    char *fecha;
    int num_pal=9, i,espacio=0;

    fecha = (char*)malloc(sizeof(char)*(15));
    *fecha = NULL;

    err1 = fopen_s(&filetime, "history.txt", "r");
    if (err1 != NULL)
    {
        printf("El archivo history no se ha abierto corretamente\n");
        printf("Pulse una tecla para continuar");
        _getch();
        exit(1);
    }
    while (feof(filetime) == NULL)
{
    for (i = 1; i < num_pal; i++)
    {
        fscanf_s(filetime, "%s ", fecha,_msize(fecha));
        printf("%s ", fecha);
        if (*fecha=='d' && *(fecha+2)=='s')
        {
            espacio = 1;
        }
    }
    fscanf_s(filetime, "%s ", fecha, _msize(fecha));
    printf("%s\n", fecha);
    if (espacio==1)
    {
        printf("\n");
        espacio = 0;
    }
}
    fclose(filetime);
    printf("\n\nPulse una tecla para continuar");
    _getch();
}


//Funciones para la comunicacion serial
void Crear_Conexion(SerialPort *PuertoSerie, char *portName)
{

    PuertoSerie->connected = 0;
    PuertoSerie->portName = portName;
    PuertoSerie->handler = CreateFileA((portName),
```

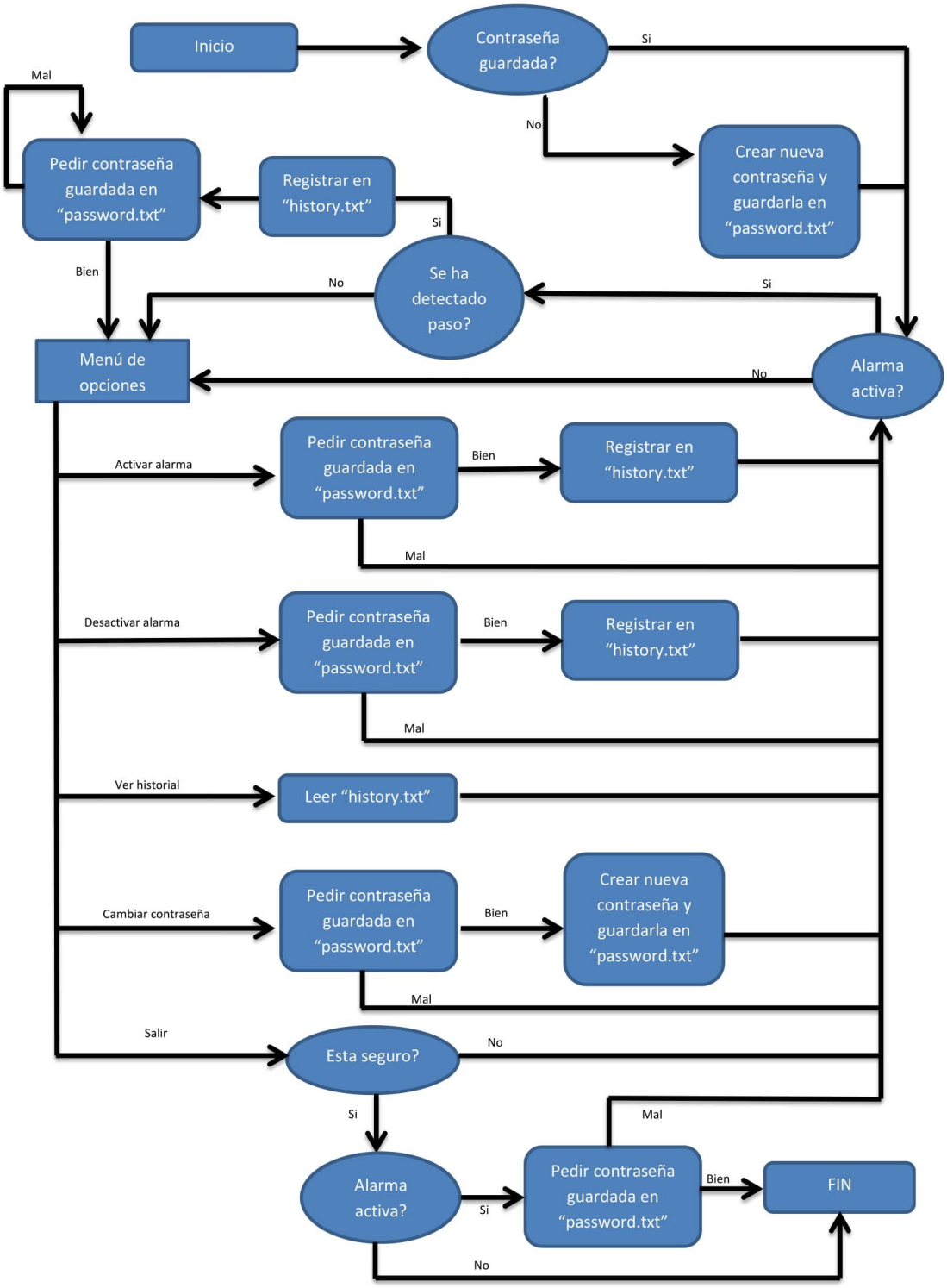
```
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL);
    if (PuertoSerie->handler == INVALID_HANDLE_VALUE)
    {
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
        {
            printf("ERROR: Handle was not attached. Reason: %s not
available\n", portName);
        }
        else
        {
            printf("ERROR!!!");
        }
    }
    else
    {
        DCB dcbSerialParameters = { 0 };

        if (!GetCommState(PuertoSerie->handler, &dcbSerialParameters))
        {
            printf("failed to get current serial parameters");
        }
        else
        {
            dcbSerialParameters.BaudRate = CBR_9600;
            dcbSerialParameters.ByteSize = 8;
            dcbSerialParameters.StopBits = ONESTOPBIT;
            dcbSerialParameters.Parity = NOPARITY;
            dcbSerialParameters.fDtrControl = DTR_CONTROL_ENABLE;

            if (!SetCommState(PuertoSerie->handler,
&dcbSerialParameters))
            {
                printf("ALERT: could not set Serial port
parameters\n");
            }
            else
            {
                PuertoSerie->connected = 1;
                PurgeComm(PuertoSerie->handler, PURGE_RXCLEAR |
PURGE_TXCLEAR);
                Sleep(ARDUINO_WAIT_TIME);
            }
        }
    }
    return;
}

void CerrarConexion(SerialPort * PuertoSerie)
{
    if (PuertoSerie->connected)
    {
        PuertoSerie->connected = 0;
        CloseHandle(PuertoSerie->handler);
    }
}
```

```
int isConnected(SerialPort *PuertoSerie)
{
    if (!ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status))
        PuertoSerie->connected = 0;
    return PuertoSerie->connected;
}

int readSerialPort(SerialPort * PuertoSerie, char *buffer, unsigned
int buf_size)
{
    DWORD bytesRead;
    unsigned int toRead = 0;

    ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status);

    if (PuertoSerie->status.cbInQue > 0)
    {
        if (PuertoSerie->status.cbInQue > buf_size)
        {
            toRead = buf_size;
        }
        else toRead = PuertoSerie->status.cbInQue;
    }

    memset(buffer, 0, buf_size);

    if (ReadFile(PuertoSerie->handler, buffer, toRead, &bytesRead,
NULL)) return bytesRead;

    return 0;
}

int writeSerialPort(SerialPort *PuertoSerie, char *buffer, unsigned
int buf_size)
{
    DWORD bytesSend;

    if (!WriteFile(PuertoSerie->handler, (void*)buffer, buf_size,
&bytesSend, 0))
    {
        ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status);
        return 0;
    }
    else return 1;
}
```

En el programa está incluido el archivo .c subido a Moodle necesario para llevar a cabo la comunicación serial con Arduino y la librería “SerialPort.h”, además que el propio código de la aplicación que nosotros hemos escrito.

También hemos usado la librería “time.h” para obtener la hora del ordenador mediante las funciones “time” y “ctime”.

Funcionamiento del programa



Al iniciar el programa, primero dentro de la función “main” introduce la contraseña de la alarma en el puntero “pass” que con la función DefinePass(0), le introduce la clave que hay definida en “Password.txt” (si no hay ninguna, te pide crearla).

Una vez copiada, inicia la comunicación serie con Arduino. Si se encuentra Arduino, la función “autoConnect” ejecuta la aplicación mientras la placa esté conectada.

Si está conectada, el programa despliega un menú donde están las opciones de la alarma (activar alarma, desactivar alarma, imprimir en consola el registro de la alarma, cambiar la contraseña o salir).

Cuando termine de ejecutar una una opción, se limpia la pantalla y vuelve a imprimir el menú con las opciones.

Para activar o desactivar la alarma, el usuario deberá introducir correctamente la clave de seguridad. Si la alarma se activa o se desactiva, el programa imprime en el archivo “registro.txt” la fecha y la hora a la que se ha realizado dicha acción.

Cada vez que el menú se actualiza, lee el puerto serie, si la alarma estaba activa y ha detectado paso, envía una señal al programa que cuando la detecta, limpia la pantalla, indica que se ha detectado paso, escribe en “registro.txt” la fecha y la hora a la que se ha activado la alarma, y hasta que el usuario no introduzca la contraseña correcta, no la desactiva.

Si el usuario elige “cambiar contraseña”, e introduce la clave correctamente, el programa llama a la función “DefinePass” con el flag=1 para que borre primero la contraseña definida en “password.txt” y pida introducir la nueva clave.

Si se elige la opción “salir”, el programa primero detecta si la alarma está activa (con ayuda de la variable “act” que cambia a 1 si está activa y a 0 si no lo está), y si lo está primero pedirá desactivarla. Una vez hecho, el programa finalizará.

Funciones empleadas

Explicaremos a continuación el funcionamiento de las funciones que hemos creado o modificado para la aplicación. El resto son las funciones que emplea la comunicación serial que simplemente hemos copiado del archivo de Moodle.

- char* Password()

  Esta función sirve para introducir contraseñas con ayuda de cadenas de caracteres (cuya longitud máxima podrá definir el usuario al inicio del programa variando “#define length”), generadas con asignación dinámica de memoria. Al introducir cada carácter de la contraseña, escribe en la ventana de comandos un “*”, hasta presionar intro.

  Si la contraseña está en blanco o tiene mayor longitud que la máxima definida por el usuario, no permitirá introducirla.

  Si la contraseña es válida, la función devuelve la dirección de memoria del puntero de la cadena de caracteres.

- char* DefinePass(int flag)

  Esta función sirve para definir la contraseña de la alarma, para ello, lee el archivo de texto en el que se aloja la contraseña, la guarda en una cadena de caracteres generada mediante asignación dinámica, y devuelve la dirección de memoria del puntero de la cadena.

  Si el flag es 1, la función primero borra la contraseña creada en el archivo de texto para cambiarla; si es distinta de 1, la función escribe en la cadena la contraseña puesta en el archivo, si está vacío, primero la crea.

- void Alarm(int flag, SerialPort *arduino, char* pass,char* pass_aux,int* act)

  Función que pide introducir la contraseña definida y activa o desactiva la alarma dependiendo del valor del flag (ON, OFF, DETECCION) enviándole una señal a la controladora de la alarma mediante el puerto serie (una “a” para activarla, o una “o” para desactivarla).

  Para ello es necesario introducirle el valor del flag, el puntero del puerto serie de Arduino (forma parte de las librerías de la comunicación serial), el puntero de la contraseña de la alarma, el puntero auxiliar en el que se guarda la contraseña que el usuario escribirá para compararla con la que hay definida, y la dirección de memoria de la variable “act” que cambiará cuando la alarma se active (1) o se desactive (0).

  Si la contraseña introducida por el usuario no coincide con la definida en la alarma, sale de la función sin enviar ninguna señal.

- void fecha(int)

  Esta función se encuentra cada vez que se active, desactive o detecte algo la alarma y lo guarda en un archivo txt. Esta funcion obtiene la fecha y hora del ordenador por medio de las funciones “time” y “ctime” contenidas en la librería “time.h”. A esta función se le introduce un flag que modifica el mensaje que se escribe en un archivo txt.

- void Registro()

  Esta función lee el archivo txt en el que ha escrito la función “fecha” e imprime su contenido por pantalla, añadiendo un espacio extra cada vez que se desactive la alarma. Se la llama en la opción del menú de ver registro. Usa asignación dinámica de memoria para recoger cada cadena de caracteres. Encontramos un problema al leer el archivo porque cada vez que encontraba un espacio paraba de leer, al final fue solucionado con un bucle for.

- void autoConnect(SerialPort *arduino, char*, char*)

  Esta función detecta si Arduino está conectado en el puerto serie indicado, y si es así, ejecuta la aplicación (que hemos escrito en el interior de la función). Si detecta que el Arduino ha sido desconectado, finaliza el programa.

  Es necesario indicarle: la dirección del puntero del puerto en el que está Arduino, el de la cadena de caracteres en la que se guardan los datos recibidos desde Arduino a través del puerto serie, (ambos forman parte del código para realizar la comunicación serial con Arduino), y el puntero de la cadena de caracteres en la que se guarda la contraseña de la alarma.

Programa de Arduino

Código

Este es el programa que se debe cargar en la placa para que funcione correctamente la alarma.

```
#include <LiquidCrystal_I2C.h> // Librerias para la lcd
#define distancialimite 10 //Introducir distancia a la que está el
tope, si es menor, la alarma se activa
LiquidCrystal_I2C lcd(0x27,16,2); //Pin SDA del lcd debe ir en A4 y
pin SCL debe ir en A5


//Prototipo funciones
void InicioLcd(); //Inicia la pantalla LCD
void SonarAlarma(int); //Reproduce el sonido de la alarma, debes
indicarle el pin del zumbador
int MedirDistancia(int); //Mide la distancia desde el sensor
ultrasonido, debes indicarle el pin de Echo

int Echo=13;
int Trig=12;
int Zumbador=3;
int Led[]={7,8}; //Led[0]=LedRojo Led[1]=LedVerde

void setup()
{
  int i;

  Serial.begin(9600);
  pinMode(Zumbador,OUTPUT);
  pinMode(Trig,OUTPUT);
  pinMode(Echo,INPUT);
   for(i=0;i<2;i++)
  {
    pinMode(Led[i],OUTPUT);
  }
  InicioLcd();

  //Juego de luces al iniciar
  for(i=0;i<2;i++)
  {
    digitalWrite(Led[i],HIGH);
    delay(50);
    digitalWrite(Led[i],LOW);
    delay(50);
  }
  for(i=1;i>=0;i--)
  {
    digitalWrite(Led[i],HIGH);
    delay(50);
    digitalWrite(Led[i],LOW);
    delay(50);
  }
}

void loop()
{
  long distancia;
  char active=0;
```

```
//ALARMA DESACTIVADA
  noTone(Zumbador);
  lcd.setCursor(0,0);
  lcd.print("     ALARMA     ");
  lcd.setCursor(0,1);
  lcd.print("  DESACTIVADA     ");
  digitalWrite(Led[1],HIGH);

  if (Serial.available()>0)
  {
   active=Serial.read();
  }

//ALARMA ACTIVA
  if(active=='a') //Si recibe una 'a' del programa, se activa la
alarma
  {
   //Primero borramos lcd y apagamos LedVerde
   lcd.clear();
   digitalWrite(Led[1],LOW);
   delay(500);

   while(active!='o')
   {
     //Cambiamos Led
     digitalWrite(Led[0],HIGH);

     //Cambiamos lcd
     lcd.setCursor(0,0);
     lcd.print("     ALARMA     ");
     lcd.setCursor(0,1);
     lcd.print("    ACTIVADA     ");

     //Activamos sensor ultrasonido
     digitalWrite(Trig,LOW);
     delayMicroseconds(4);
     digitalWrite(Trig,HIGH);
     delayMicroseconds(10);
     digitalWrite(Trig,LOW);

     //Medir distancia
     distancia=MedirDistancia(Echo);
     delay(50);

     //Deteccion de paso
     if(distancia<distancialimite)
      {
        Serial.println("s"); //Si detecta presencia, manda una 's' al
puerto serie para que la lea el programa
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("      PASO     ");
        lcd.setCursor(0,1);
        lcd.print("   DETECTADO!!     ");
        while(active!='o')
        {
          digitalWrite(Led[0],LOW);
          delay(40);
          digitalWrite(Led[0],HIGH);
          delay(40);
```

```
          digitalWrite(Led[0],LOW);
          SonarAlarma(Zumbador);
          if (Serial.available()>0)
          {
            active=Serial.read();
          }
        }
        Serial.println(0);
        noTone(Zumbador);
      }

    if(active!='o')
    {
      if (Serial.available() > 0)
      {
        active=Serial.read();
      }
    }
  }

  if(active=='o')  //Si recibe del puerto serie una 'o' desactiva la
alarma
  {
   lcd.clear();
   digitalWrite(Led[0],LOW);
   delay(500);
  }
 }
}


//Funciones
void InicioLcd()
{
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   SISTEMAS    ");
  lcd.setCursor (0,1);
  lcd.print("KEEP'N YOU SAFE");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   BIENVENIDO  ");
  delay(2000);
  lcd.clear();
}

void SonarAlarma(int zumb)
{
  int i;
  float sinVal;
  int toneVal;

   for(i=0;i<180;i++)
  {
    sinVal=(sin(i*(3.1412/180))); //Señal senoidal para el zumbador.
Solo valores positivos
    toneVal=2000+(int)(sinVal*1000);
    tone(zumb,toneVal); //Tono zumbador
```

```
    delay(2);
  }
}

int MedirDistancia(int ech)
{
  long duracion;
  long distancia;

  duracion=((pulseIn(ech,HIGH))/2); //La señal sonora va y vuelve, por
eso se divide por 2
  distancia=duracion/29.2; //Velocidad del sonido= 1cm/29.2
microsegundos

  return distancia;
}
```

Funcionamiento del programa

En primer lugar, se han definido las variables “echo”, “trig”, “zumbador”, “led” con los valores de los pines en los que se encuentra conectado cada elemento. Esto lo hemos hecho simplemente para facilitar la programación de cada elemento, de manera que, si cambiamos algún pin, solamente deberemos modificar el valor de la variable.

Hemos implementado la librería “LiquidCrystal_I2C.h” para poder hacer funcionar el LCD con el módulo l2C, que permite conectarlo a Arduino de forma más sencilla que conectando todos los pines del puerto paralelo de la pantalla.

Dentro de la función “setup” iniciamos la comunicación serial y los pines de todos los elementos que tenemos conectados, y realiza un pequeño juego de luces con los leds.

Luego dentro de la fusión “loop” se ejecuta en bucle el programa propiamente dicho.

Al inicio, parte con la alarma desactivada, es decir imprimimos en la LCD “Alarma desactivada”, activamos el led verde, y hacemos que el zumbador no suene. Si el programa principal ha enviado al puerto serie una “a”, la alarma se activa, cambia el LCD e imprime “Alarma activa”, activa el led rojo y desactiva el verde, y activa el sensor de ultrasonidos.

Si el sensor detecta un objeto a una distancia menor que la que el usuario indica al principio del programa en “#define distancialimite” (un tope), envía una señal al programa principal a través del puerto serie para indicar que se ha detectado paso, hace sonar al zumbador, hace parpadear al led rojo y cambia el LCD para que escriba “Paso detectado!”. Esto lo realiza hasta que no detecte una señal proveniente del programa principal indicando que la alarma debe desactivarse.

Si no detecta paso y llega una señal por puerto serie, la alarma se desactiva y vuelve al inicio.

Funciones utilizadas

- void InicioLcd()

  Función que inicia la conexión con la pantalla LCD e imprime en ella el mensaje que aparece al inicio del programa.

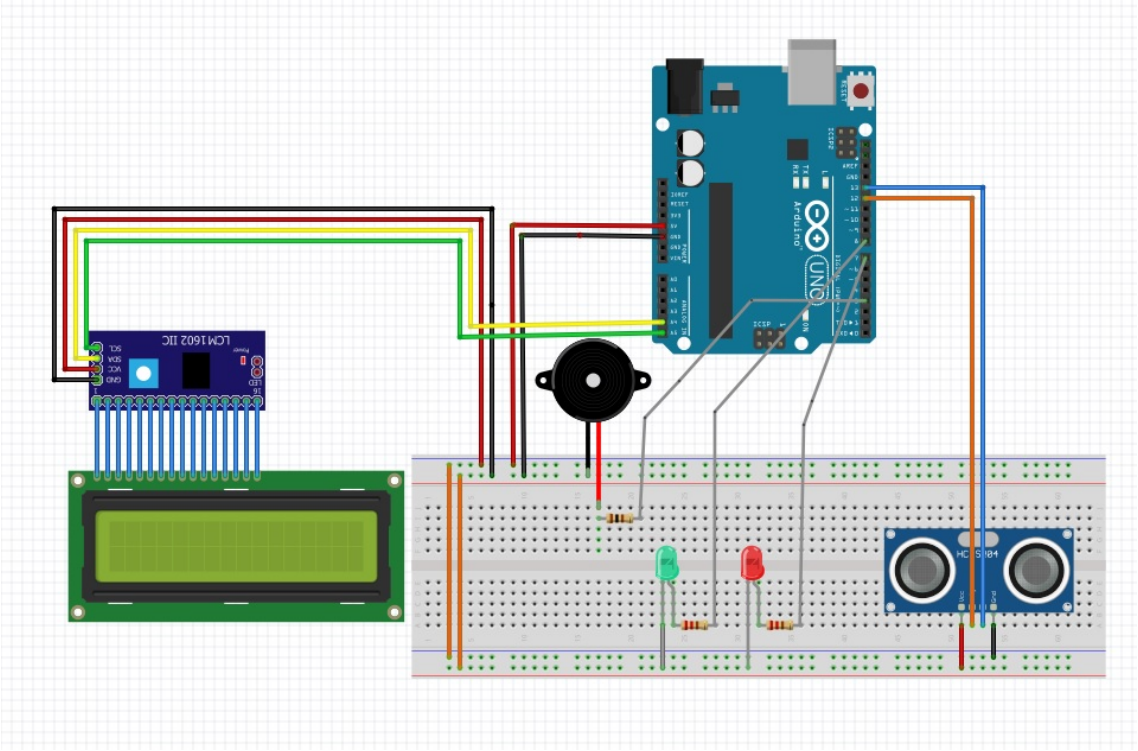
- void SonarAlarma (int Zumb)

  Función que hace que el zumbador suene con una frecuencia senoidal que simula el sonido de una alarma. Para que funcione correctamente debes indicarle el pin en el que has conectado el zumbador.

- Int MedirDistancia (int ech)

  Función que realiza los cálculos necesarios para devolver la distancia a la que el sensor de ultrasonidos detecta un obstáculo. Debes indicarle el pin “echo” del sensor.

Esquema del montaje de Arduino



Conclusiones

Por medio de este proyecto hemos aprendido a manejar Arduino, hemos adquirido conocimientos básicos sobre su utilización y familiarizándonos con el hardware. También a comunicar Visual Studio con Arduino con unas funciones por medio de pruebas e investigación y con la ayuda de una carpeta y unos ejemplos que había en el Moodle.
El proyecto nos ha servido para repasar lo visto a lo largo de la asignatura dado que hemos usado casi todo lo visto en la realización de este y a verle la utilidad a temas que hemos dado que parecían inútiles hasta que nos hemos encontrado un problema que se resolvía por medio de la utilización de estos mismos.
Nos ha sido de gran ayuda la herramienta de Github con la que cada uno podía trabajar desde casa y desde la que es muy cómodo detectar el problema al ver fácilmente lo último editado y que sea necesario explicar el cambio.
También hemos tenido que buscar nuevas formas de plantear el código, tanto cuando se nos planteaba un problema o no compilaba. Hemos tenido que pensar cada cosa que editábamos porque podía conducir a un problema de compilación. Pero hemos llegado al resultado final del que estamos muy satisfechos y orgullosos.