Memoria del trabajo

Carlota Pinedo

Inés López-Boado

Sara Khiar Souri

El objetivo del trabajo era desarrollar y programar un sistema de seguridad dirigido para prisiones que limita el acceso al exterior exclusivamente para los guardias. La identificación mediante algo único de cada persona que no puede ser reproducido o falsificado permite una mayor seguridad por lo que la identificación mediante huella es la opción que hemos considerado más sensato usar.

Los guardias antiguos de la prisión se encuentran registrados mediante su huella que les permite salir.

Los guardias nuevos aún no están registrados pero disponen de una contraseña que les permite registrarse en el sistema y así usar su huella las próximas veces.

El resto son los prisioneros que al no estar registrados ni disponer de la contraseña para registrarse tienen la salida denegada.

- Componentes utilizados:

- **Módulo ultrasonidos HC-SR04**

Mide la distancia a la que se encuentra el objeto más próximo.

El EchoPin va conectado al pin 8 y el TriggerPin al pin 9.

- **LED**

De color naranja y va conectado al pin 5

- **Resistencia 10k**

Va conectado al sensor de huellas

- **Resistencia 220**

Va conectada al LED

- **Servo SG90**

Va al pin 6

- **Jumpers M-M**

Unen todos los componentes con la Protoboard y la Placa Arduino

- **Jumpers H-M**

Unen el sensor de huellas con la Protoboard

- **Protoboard MB-102**
- **Placa Arduino UNO**
- **Módulo de huellas dactilares JM-101**

El trabajo dispone de dos códigos: el de la aplicación Visual Studio y Arduino.

El código de Visual:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <conio.h>
#include "SerialClass/SerialClass.h"

#define MAX_BUFFER 200
#define PAUSA_MS 200
#define TAM 20
#define MAX_REGIS 20

typedef struct
{
        char nombre[TAM];
        int id;
}identificacion;

// Funciones prototipo
int leer_fichero(identificacion[]);
void guardar_fichero(identificacion*, int);
char menu1(void);
int menu2(void);
int menu3(void);

float leer_sensor_distancia(Serial*);
int Comprobar_huella(Serial*, identificacion[]);
int leer_huella(Serial*);
void AbrirPuerta(Serial*);
void CerrarPuerta(Serial*);
int Comprobar_contrasena(void);
void ActivarAlarma(Serial* Arduino);
int registrar_huella(Serial*, identificacion*, int);

int Enviar_y_Recibir(Serial*, const char*, char*);
int Recibir(Serial*, char*);
float float_from_cadena(char* cadena);


int main(void)
{
        Serial* Arduino;
        char puerto[] = "COM4"; // Puerto serie al que está conectado Arduino

        int opcion_menu2 = 0;
        char opcion_letra;
        int numero;
        int opc = 0;

        identificacion registros[MAX_REGIS];
        int i, numregistros = 0;
        float distancia;

        setlocale(LC_ALL, "es-ES");
        Arduino = new Serial((char*)puerto);
        numregistros = leer_fichero(registros);

        do
        {
                distancia = leer_sensor_distancia(Arduino);
                if (distancia == -1)
                        printf("Atención: No se ha podido leer el sensor de distancia\n");
                else
                if (distancia <= 10) // Se detecta una persona
                {
                opcion_letra = menu1(); //
                if (opcion_letra == 'S' || opcion_letra == 's')
                {
```

```
                    opcion_menu2 = menu2();

                   switch (opcion_menu2)
                   {
                   case 1:
                   numero = Comprobar_huella(Arduino, registros);
                   if (numero < 0)
                   {
                   printf("Usted no está registrado.¿Quiere registrarse?\n");
                   opc = menu3();
                   if (opc == 1)
                   {

                   if (Comprobar_contrasena() == 0)
                   {
                   printf("Acceso denegado\n");
                   ActivarAlarma(Arduino);
                   }
                   else
                   {
                   printf("Contraseña correcta\n");
                   numregistros = registrar_huella(Arduino, registros, numregistros);
                   guardar_fichero(registros, numregistros);
                   AbrirPuerta(Arduino);
                   CerrarPuerta(Arduino);
                   }
                   }
                   }

                   else
                   {
                   AbrirPuerta(Arduino);
                   CerrarPuerta(Arduino);
                   }
                   break;

                  case 2:
                   {

                  if (Comprobar_contrasena() == 0)
                   {
                   printf("Acceso denegado\n");
                   ActivarAlarma(Arduino);
                   }
                   else
                   {
                   printf("Contraseña correcta\n");
                   numregistros = registrar_huella(Arduino, registros, numregistros);
                   guardar_fichero(registros, numregistros);
                   AbrirPuerta(Arduino);
                   CerrarPuerta(Arduino);
                   }

                   }
                   break;

                   case 3:
                  printf("Los datos registrados actualmente son:\n");
                  for (i = 0; i < numregistros; i++)
                  printf("%s - %d\n", registros[i].nombre, registros[i].id);
                  break;
                   case 4:
                   break;
                   default: printf("\n Opción incorrecta\n");
                   }
                    }
                    }
          } while (opcion_menu2 != 4);
          return 0;


int leer_fichero(identificacion registros[])
{
          FILE* fichero;
          errno_t e;
          int numregistros = 0;

          e = fopen_s(&fichero, "Personas_registradas.txt", "rt");
          if (fichero == NULL)
          {
                    printf("El archivo de registros no existe\n");
          }
          else
          {
                    fscanf_s(fichero, "%s", registros[numregistros].nombre, TAM);
```

```
                while (!feof(fichero))
                {
                        fscanf_s(fichero, "%d\n", &registros[numregistros].id);
                        numregistros++;
                        fscanf_s(fichero, "%s", registros[numregistros].nombre, TAM);
                }
                fclose(fichero);
        }
        return numregistros;
}

void guardar_fichero(identificacion* i, int n)
{
        FILE* fichero;
        errno_t e;
        int j;

        e = fopen_s(&fichero, "Personas_registradas.txt", "wt");
        if (fichero == NULL)
        {
                printf("Hay algún error a la hora de grabar el archivo\n");
        }
        else
        {
                for (j = 0;j < n;j++)
                        fprintf(fichero, "%s %d\n", i[j].nombre, i[j].id);
                fclose(fichero);
        }
}

char menu1(void)
{
        char letra;
        char intro;
        printf("==============\n");
        printf("Si desea salir pulse S\n");
        printf("==============\n");
        scanf_s("%c", &letra, 1);
        scanf_s("%c", &intro, 1);
        return letra;
}

int menu2(void)
{
        int opcion;
        char intro;
        printf("\n");
        printf("==============\n");
        printf("1 - Guardia antiguo\n");
        printf("2 - Registrarse (Nuevo guardia)\n");
        printf("3 - Consultar lista de guardias registrados\n");
        printf("4 - Salir de la aplicación\n");
        printf("==============\n");
        scanf_s("%d", &opcion);
        scanf_s("%c", &intro, 1);
        return opcion;
}
int menu3(void)
{
        int opc;
        printf("\n");
        printf("==============\n");
        printf("1 - Sí\n");
        printf("2 - No\n");
        printf("==============\n");
        scanf_s("%d", &opc);
        return opc;
}

float leer_sensor_distancia(Serial* Arduino)
{
        float distancia;
        int bytesRecibidos;
        char mensaje_recibido[MAX_BUFFER];

        bytesRecibidos = Enviar_y_Recibir(Arduino, "GET_DISTANCIA\n", mensaje_recibido);

        if (bytesRecibidos <= 0)
                distancia = -1;
        else
                distancia = float_from_cadena(mensaje_recibido);
        return distancia;
}


int Comprobar_huella(Serial* Arduino, identificacion* i)
```

```
{
	int numero;
	numero = leer_huella(Arduino);
	if (numero >= 0)
	{
		printf("Usted se ha identificado como %s. Acceso otorgado\n", i[numero].nombre);
	}
	return numero;
}

int leer_huella(Serial* Arduino)
{
	int id;
	int bytesRecibidos;
	char mensaje_recibido[MAX_BUFFER];

	bytesRecibidos = Enviar_y_Recibir(Arduino, "HUELLA_EXISTENTE\n", mensaje_recibido);
	do
	{
		if (bytesRecibidos > 0)
			printf("%s\n", mensaje_recibido);
		bytesRecibidos = Recibir(Arduino, mensaje_recibido);
	} while (strstr(mensaje_recibido, "Match") == NULL);

	if (strstr(mensaje_recibido, "No Match") != NULL)
		id = -1;
	else
		id = (int)float_from_cadena(mensaje_recibido);
	return id;
}
void AbrirPuerta(Serial* Arduino)
{
	int bytesRecibidos;
	char mensaje_recibido[MAX_BUFFER];

	bytesRecibidos = Enviar_y_Recibir(Arduino, "ABRE_LA_PUERTA\n", mensaje_recibido);
	if (bytesRecibidos <= 0)
		printf("\nNo se ha recibido confirmación\n");
	else
		printf("\n%s\n", mensaje_recibido);
}
void CerrarPuerta(Serial* Arduino)
{
	int bytesRecibidos;
	char mensaje_recibido[MAX_BUFFER];

	bytesRecibidos = Enviar_y_Recibir(Arduino, "CIERRA_LA_PUERTA\n", mensaje_recibido);

	if (bytesRecibidos <= 0)
		printf("\nNo se ha recibido confirmación\n");
	else
		printf("\n%s\n", mensaje_recibido);
}
int Comprobar_contrasena(void)
{
	long int contrasena;
	long int clave;
	char intro;
	int n = 0;
	int intento = 3;
	clave = 906151612;
	printf("Introduzca la contraseña: ");
	scanf_s("%d", &contrasena);
	scanf_s("%c", &intro, 1);
	while (intento > 1)
	{
		if (contrasena == clave)
		{
			n = 1;
			intento = 0;
		}
		else
		{
			intento--;
			printf("\nContraseña incorrecta. Le quedan %d intentos\n", intento);
			printf("Introduzca la contraseña: ");
			scanf_s("%d", &contrasena);
			scanf_s("%c", &intro, 1);
		}
	}
	return n;
}

void ActivarAlarma(Serial* Arduino)
{
	int bytesRecibidos;
```

```
        char mensaje_recibido[MAX_BUFFER];

        bytesRecibidos = Enviar_y_Recibir(Arduino, "ALARMA\n", mensaje_recibido);
}


int registrar_huella(Serial* Arduino, identificacion* i, int id)
{
        int bytesRecibidos;
        char mensaje_recibido[MAX_BUFFER];
        char mensaje_enviado[MAX_BUFFER];
        char intro;

        bytesRecibidos = Enviar_y_Recibir(Arduino, "REGISTRAR\n", mensaje_recibido);
        printf("%s\n", mensaje_recibido);
        i[id].id = id;
        printf("Introduzca su nombre: ");
        scanf_s("%s", i[id].nombre, TAM);
        scanf_s("%c", &intro, 1);
        sprintf_s(mensaje_enviado, "%d\n", id);
        bytesRecibidos = Enviar_y_Recibir(Arduino, mensaje_enviado, mensaje_recibido);
        do
        {
                if (bytesRecibidos > 0)
                        printf("%s\n", mensaje_recibido);
                bytesRecibidos = Recibir(Arduino, mensaje_recibido);
        } while (strstr(mensaje_recibido, "Match") == NULL);
        printf("%s\n", mensaje_recibido);
        if (strstr(mensaje_recibido, "No Match") == NULL)
                id++;
        return id;
}



int Enviar_y_Recibir(Serial* Arduino, const char* mensaje_enviar, char* mensaje_recibir)
{
        int bytes_recibidos = 0, total = 0;
        int intentos = 0, fin_linea = 0;

        Arduino->WriteData((char*)mensaje_enviar, (unsigned int)strlen(mensaje_enviar));
        Sleep(PAUSA_MS);

        bytes_recibidos = Arduino->ReadData(mensaje_recibir, sizeof(char) * MAX_BUFFER - 1);

        while ((bytes_recibidos > 0 || intentos < 5) && fin_linea == 0)
        {
        if (bytes_recibidos > 0)
        {
        total += bytes_recibidos;
        if (mensaje_recibir[total - 1] == 13 || mensaje_recibir[total - 1] == 10)
        fin_linea = 1;
        }
        else
        intentos++;
        Sleep(PAUSA_MS);
        bytes_recibidos = Arduino->ReadData(mensaje_recibir + total, sizeof(char) * MAX_BUFFER - 1);
        }
        if (total > 0)
                mensaje_recibir[total - 1] = '\0';
        return total;
}

int Recibir(Serial* Arduino, char* mensaje_recibir)
{
        int bytes_recibidos = 0, total = 0;
        int intentos = 0, fin_linea = 0;

        bytes_recibidos = Arduino->ReadData(mensaje_recibir, sizeof(char) * MAX_BUFFER - 1);

        while ((bytes_recibidos > 0 || intentos < 2) && fin_linea == 0)
        {
        if (bytes_recibidos > 0)
        {
        total += bytes_recibidos;
        if (mensaje_recibir[total - 1] == 13 || mensaje_recibir[total - 1] == 10)
        fin_linea = 1;
        }
        else
        intentos++;
        Sleep(PAUSA_MS);
       bytes_recibidos = Arduino->ReadData(mensaje_recibir + total, sizeof(char) * MAX_BUFFER - 1);
        }
        if (total > 0)
        mensaje_recibir[total - 1] = '\0';
```

```
        return total;
}


float float_from_cadena(char* cadena)
{
        float numero = 0;
        int i, divisor = 10, estado = 0;

        for (i = 0; cadena[i] != '\0' && estado != 3 && i < MAX_BUFFER; i++)
                switch (estado)
                {
                case 0:// Antes del número
                        if (cadena[i] >= '0' && cadena[i] <= '9')
                        {
                                numero = (float)cadena[i] - '0';
                                estado = 1;
                        }
                        break;
                case 1:// Durante el número
                        if (cadena[i] >= '0' && cadena[i] <= '9')
                                numero = numero * 10 + cadena[i] - '0';
                        else
                                if (cadena[i] == '.' || cadena[i] == ',')
                                        estado = 2;
                                else
                                        estado = 3;
                        break;
                case 2: // Parte decimal
                        if (cadena[i] >= '0' && cadena[i] <= '9')
                        {
                                numero = numero + (float)(cadena[i] - '0') / divisor;
                                divisor *= 10;
                        }
                        else
                                estado = 3;
                        break;
                }
        return numero;
}
```

Funciones:

- `int Enviar_y_Recibir(Serial*, const char*, char*);`

Esta función se encarga de enviarle a Arduino (Serial*) un mensaje del tipo const char y recibe uno de vuelta. La función devuelve el valor de los bytes recibidos.

- `int Recibir(Serial* Arduino, char* mensaje_recibir)`

Esta función se encarga de recibir mensajes de Arduino y devuelve los bytes recibidos.

- `float float_from_cadena(char* cadena)`

Esta función se encarga de extraer un numero con sus decimales de una cadena de texto.

Devuelve el numero extraído.

- `float leer_sensor_distancia(Serial* Arduino)`

Utiliza la función de Enviar_y_recibir para enviar el mensaje de “GET_DISTANCIA” que le indica al programa de Arduino que debe de medir la distancia a la que se encuentra el objeto más próximo en ese instante. Esa velocidad es enviada de vuelta en una cadena, se extrae usando la función de float_from_cadena. El valor de la velocidad en centímetros es lo que devuelve.

- `char menu1(void), int menu2(void), int menu3(void)`

Son funciones que se limitan a mostrar en pantalla las diferentes opciones que ofrecen estos menús y guardar la opción seleccionada por el usuario. Devuelven la opción.

El menu1 aparece en pantalla cuando alguien se aproxima a la salida, este menú ofrece una única opción: salir de la prisión.

El menu2 muestra las distintas acciones relacionadas con la salida.

-Guardia antiguo (previamente registrado)

-Registrarse

- Mostrar lista de registrados

El menu3 aparece en caso de que al intentar salir utilizando una huella no registrada anteriormente. Este menú ofrece la posibilidad de registrarse (opción 2 del menu2) o simplemente salir del programa

- `int leer_huella(Serial* Arduino)`

Envía a Arduino el mensaje de “HUELLA_EXISTENTE”. Utiliza la función Recibir para imprimir en pantalla todos los mensajes provenientes del Arduino relacionados con el proceso de tomar la huella hasta llegar a la palabra clave MATCH.

Cuando el mensaje es MATCH significa que la huella existe y la función devuelve el id asociado a esa huella.

Cuando el mensaje es NO MATCH significa que esa huella no está registrada por lo que la función devuelve un id= -1.

- `int Comprobar_huella(Serial* Arduino, identificacion* i)`

La estructura identificacion se compone de dos valores: un entero que hace referencia al numero del id asociado y una cadena que es el nombre del guardia registrado.

Esta función recibe por referencia una estructura el que están guardados los guardias registrados. Utiliza la función leer_huella para comprobar el id.

Si el id es un numero mayor o igual que cero significa que esa persona se encuentra en el vector asociado a los registrados. Se imprime en pantalla el nombre de la persona correspondiente sabiendo que el valor de id dentro de la estructura es el que devuelve la función leer_huella.

En cambio si el id es -1 la persona no está registrada.

La función devuelve el id leído

- `int Comprobar_contrasena(void)`

Cuando alguien quiere registrarse se debe verificar que es un guardia y no un prisionero por lo que se le solicita una contraseña. Esta función se encarga de comprobar esto.

Partimos de 3 intentos, la contraseña se encuentra guardada en el programa y tiene formato long int. Se le solicitará al usuario por pantalla introducir la contraseña y si esta coincide tiene el acceso permitido y la función acaba devolviendo un 1.

Si la contraseña introducida no coincide se resta un intento y se vuelve a solicitar introducirla. Si los intentos se agotan, la función acaba y devuelve un 0.

- `int registrar_huella(Serial* Arduino, identificacion* i, int id)`

Se le envía a Arduino el mensaje de REGISTRAR y se le solicita al usuario introducir su nombre que se guardará en la posición que indique el valor de id en ese momento. Se imprimen en pantalla todos los mensajes provenientes de Arduino hasta que el mensaje incluya la palabra clase MATCH.

El valor inicial de la id es 0, cada vez que se ejecuta correctamente esta función y se registra alguien nuevo se incrementa el valor de la id en uno.

La función devuelve el valor del id.

- `void ActivarAlarma(Serial* Arduino)`

  Esta función se encarga meramente de enviarle a Arduino el mensaje de ALARMA.

  Esto es cuando el usuario a la hora de registrarse agota los intentos de poner la contraseña correcta por lo que no es un guardia.

- `void AbrirPuerta(Serial* Arduino)`

  Le envía a Arduino el mensaje de ABRE_LA_PUERTA.

  Esta función será solicitada cada vez que el acceso haya sido aprobado ya sea mediante la huella o haber introducido bien la contraseña.

- `void CerrarPuerta(Serial* Arduino)`

Le envía a Arduino el mensaje de CIERRA_LA_PUERTA.

Esta función será solicitada después de la función AbrirPuerta para cerrar la puerta una vez el guardia ya ha salido.

- `int leer_fichero(identificacion registros[])`

Esta función imprime en pantalla los datos registrados en el fichero “Personas_registradas.txt”.

Imprime el nombre de las personas registradas junto con su id.

- `void guardar_fichero(identificacion* i, int n)`

Guarda en el fichero “Personas_registradas.txt” los datos que están en la estructura del tipo identificacion que recibe por referencia. Recibe también el número datos que contiene el vector estructura.

El código de Arduino:

```
#include <Adafruit_Fingerprint.h>
#include <Servo.h>
SoftwareSerial mySerial(2, 3);
Servo motor;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
String mensaje_entrada;
String mensaje_salida;
const int EchoPin = 8;
const int TriggerPin = 9;
uint8_t id;



void setup()
{
 Serial.begin(9600);
 finger.begin(57600);
 pinMode(5, OUTPUT);
 pinMode(12, OUTPUT);
 pinMode(TriggerPin,OUTPUT);
 pinMode(EchoPin,INPUT);
 motor.attach(12);
 if (finger.verifyPassword())
 {
  //Serial.println("Sensor de huella activo");
 }
 finger.getParameters();
 finger.getTemplateCount();


}

void loop()
{
 int myid;
 if( Serial.available()> 0)
```

```
{
 mensaje_entrada = Serial.readStringUntil('\n');
 if (mensaje_entrada.compareTo("GET_DISTANCIA")==0)
 {
  float cm = ping(TriggerPin, EchoPin);
  mensaje_salida=String("DISTANCIA="+String(cm,3));
  Serial.println (mensaje_salida);
 }
 else
 if (mensaje_entrada.compareTo("REGISTRAR")==0)
 {
  Serial.println("Registrando...");
  while (Serial.available()<=0);
   mensaje_entrada = Serial.readStringUntil('\n');
   id=mensaje_entrada.toInt();
   Serial.println("Le corresponde el id="+String(id));
   while (! getFingerprintEnroll() );
 }
 else
  if (mensaje_entrada.compareTo("HUELLA_EXISTENTE")==0)
  {
    Serial.println("Acerque su dedo");

  do{

  myid=getFingerprintID();
  }
  while (myid==255);
   delay(50);
 }
 else
  if (mensaje_entrada.compareTo("ABRE_LA_PUERTA")==0)
  {
   Serial.println("Abriendo puerta");
   AbrirPuerta();
  }
  else
```

```
      if (mensaje_entrada.compareTo("CIERRA_LA_PUERTA")==0)
      {
       Serial.println("Cerrando puerta");
       Cerrar_Puerta();
      }
      else
       if (mensaje_entrada.compareTo("ALARMA")==0)
       {
        ActivarAlarma();
       }
    }
}

uint8_t getFingerprintEnroll()
{
 int p = -1;
 Serial.println("Acerque su dedo");
 while (p != FINGERPRINT_OK )
 {
   p = finger.getImage();
   if(p == FINGERPRINT_OK )
     Serial.println("Huella tomada");
 /*  else
     Serial.println("Reintentando tomar la huella"); */
 }

 p = finger.image2Tz(1);
 if (p == FINGERPRINT_OK )
 {

     Serial.println("Huella convertida");
 }
 else
 {
  /*Serial.println("Problemas convirtiendo la huella");*/
  return false;
 }
```

```
 Serial.println("Aleje su dedo");
 delay(2000);
 p = 0;
 while (p != FINGERPRINT_NOFINGER)
 {
   p = finger.getImage();
 }
 p = -1;
 Serial.println("Vuelva a acercar su dedo");
 while (p != FINGERPRINT_OK)
 {
  p = finger.getImage();
  if(p == FINGERPRINT_OK )
    Serial.println("Huella tomada");
  else
  {
     /*Serial.println("No ha podido tomarse la huella");*/
  }
   }

p = finger.image2Tz(2);
if(p== FINGERPRINT_OK)
    Serial.println("Huella convertida");
else
{
   /*Serial.println("Problemas convirtiendo la huella");*/
   return false;
}
p = finger.createModel();
if (p == FINGERPRINT_OK)
{
  Serial.println("Huellas coinciden");
}
else
{
  Serial.println("Las huellas no coinciden");
  return false;
```

```
  }

  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK)
  {

    Serial.println("Match");
  }
  else
  {
     Serial.println("No Match");
    return false;
  }
  return true;
}

uint8_t getFingerprintID()
{
  uint8_t p = finger.getImage();
  // Paso 1: Captura la huella como imagen
  if(p == FINGERPRINT_OK)
  {
    Serial.println("Huella tomada");
  }
  else
  {
     //Serial.println("No ha tomado la Huella");
     return 255;
  }
  // Paso 2: Transforma la imagen
  p = finger.image2Tz();
  if(p== FINGERPRINT_OK)
  {
     delay(100);
    Serial.println("Huella convertida");
  }
  else
```

```
 {
  //Serial.println("No ha convertido la huella");
  return 255;
 }
 // Paso 3: Busca entre las huellas ya registradas
 p = finger.fingerSearch();
 if (p == FINGERPRINT_OK)
 {
  // found a match!
  Serial.println("Match "+String(finger.fingerID));
  //Serial.println("Match"+String(1));
  return finger.fingerID;
 }
 else
 {  // Not found a match!
  Serial.println("No Match");
  return 254;
 }
}

int getFingerprintIDez() {
 uint8_t p = finger.getImage();
 if (p != FINGERPRINT_OK)  return -1;

 p = finger.image2Tz();
 if (p != FINGERPRINT_OK)  return -1;

 p = finger.fingerFastSearch();
 if (p != FINGERPRINT_OK)  return -1;


 Serial.print(finger.fingerID);
 return finger.fingerID;
}

void AbrirPuerta(void)
{
```

```
  motor.write(9);
  delay(500);
  motor.write(40);
  delay(500);
  motor.write(80);
  delay(500);
  motor.write(130);
  delay(1000);
}

void Cerrar_Puerta(void)
{
  delay(2000);
  motor.write(130);
  delay(500);
  motor.write(80);
  delay(500);
  motor.write(40);
  delay(500);
  motor.write(9);
  delay(1000);
}

void ActivarAlarma(void)
{
  digitalWrite(5, HIGH);
  delay(500);
  digitalWrite(5, LOW);
  delay(500);
  digitalWrite(5, HIGH);
  delay(500);
  digitalWrite(5, LOW);
  delay(500);
  digitalWrite(5, HIGH);
  delay(500);
  digitalWrite(5, LOW);
```

```
  delay(500);
}

float ping(int TriggerPin, int EchoPin)
 {
   long duration, distanceCm;

   digitalWrite(TriggerPin, LOW);  //para generar un pulso limpio ponemos a LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);  //generamos Trigger (disparo) de 10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);

   duration = pulseIn(EchoPin, HIGH);  //medimos el tiempo entre pulsos, en microsegundos

   distanceCm = duration * 10 / 292/ 2;   //convertimos a distancia, en cm
   return distanceCm;
}
```

Funcionamiento

El código de Arduino se basa en recibir un mensaje de Visual y dependiendo de este mensaje llevar a cabo una serie de acciones.

Los diferentes mensajes recibidos determinan la respuesta y son:

- GET_DISTANCIA

El sensor de movimiento detecta la distancia, se transforma a centímetros y se envía a visual este valor

- REGISTRAR

Se hace una llamada a la función getFingerprintEnroll()

Una vez recibido este mensaje, Arduino espera recibir otro mensaje más consecutivo que indicará el id bajo el que se guardará la huella a registrar.

Una vez recibido el id el programa le irá enviando a Visual las diferentes instrucciones que se requiere que el usuario realice (“acerque su dedo”,”aleje su dedo”,etc). Si hay algún error en el proceso de registro de huella el programa vuelve a iniciarse hasta que se consigue registrar correctamente.

El último mensaje contiene la palabra “MATCH” que es la que le indica a Visual que ya no llegarán más mensajes

- HUELLA_EXISTENTE

Se hace una llamada a la función getFingerprintID()

Nuevamente esta función se encargará de enviar a visual todas las instrucciones que se espera que el usuario realice. Una vez finalizado el proceso la función envía el último mensaje que le indicará a visual si la huella ha sido reconocida o no. Este último mensaje podría ser MATCH o NO MATCH.

- ABRE_LA_PUERTA

  La puerta tiene viene conectada a un servomotor. Este mensaje hace una llamada a la función AbrirPuerta() que hace que el servomotor gire desde los 9 a los 130 grados.

- CIERRA_LA_PUERTA

  Se hace una llamada a la función Cerrar_Puerta() que hace que el servomotor gire desde el ángulo 130 a los 9 grados.

- ALARMA

  Al recibir este mensaje se hace una llamada a la función ActivarAlarma() que hace parpadear un led varias veces en señal de advertencia y se vuelve a apagar.

Funcionamiento del programa general

Cuando alguien se aproxima a la puerta el sensor de distancia lo detecta y ofrece en pantalla el mensaje de “Salir”. Si pulsa la s indica que quiere salir.

Si se pulsa la s (tanto mayúscula como la minúscula) aparece en pantalla un menú con 4 opciones:

-Guardia antiguo

- Registrarse (Nuevo guardia)

- Consultar lista de guardias registrados

- Salir de la aplicación

- Si se elige Guardia antiguo

Se solicita acercar el dedo al sensor y comienza el proceso de comprobación de la huella.

Si la huella existe se imprime en pantalla el mensaje de “Usted se ha identificado como….” junto al nombre del guardia.

A continuación, aparece el mensaje de “Acceso otorgado” y se abre y se cierra la puerta permitiendo la salida.

Si la huella no figura como registrada se ofrece la opción de registrarse.

- Si se elige Registrarse (Nuevo guardia)

Se solicita introducir la contraseña que solo tienen los guardias disponiendo de 3 intentos.

Si la contraseña es correcta:

Se solicita el nombre bajo el que se quiere registrar y se solicita acercar el dedo al sensor hasta completar el registro. Una vez completado se permite la salida abriendo y cerrando la puerta.

Si la contraseña es incorrecta tras agotar los intentos:

Se enciende la luz de alarma (led) y aparece el mensaje de “Acceso denegado”

- Consultar lista de guardias registrados

Se imprimen en pantalla los datos guardados en el fichero “Personas_registradas.txt”

Aparecen los nombres de los guardias registrados junto con su id correspondiente.

- Salir de la aplicación

  Cierra el programa.