

HomeSecurity

David Ramos Perdomo

^a Escuela de Ingeniería de Vitoria, Nieves Cano Kalea, 12, 01006 Gasteiz, Araba, España.

Resumen

En este documento se va a desarrollar un proyecto utilizando el microcontrolador Arduino Mega 2560. En concreto se desarrolla un ejemplo que emula el funcionamiento de una “Puerta de seguridad” en un hogar, o cualquier otra instalación que lo precise. Se trata de un sistema de seguridad de dos pasos, completamente modular y con grandes posibilidades de ampliación de componentes, pudiendo incluir nuevos métodos de autenticación.

Palabras Clave:

Microcontrolador, IDE, Módulo

1. Introducción

Arduino es una plataforma de open-source que se utiliza para construir proyectos electrónicos. Consiste tanto en una placa denominada microcontrolador y en un IDE (Entorno de Desarrollo Integrado) que se ejecuta en un ordenador, utilizado para escribir y cargar código que se ejecutará en la placa física.

La plataforma Arduino se ha hecho bastante popular entre las personas que acaban de empezar con la electrónica, y por una buena razón. A diferencia de la mayoría de las placas de circuitos programables anteriores. Arduino se conecta por un cable USB a un ordenador y utiliza una versión simplificada de C++, lo que propicia el aprendizaje de la programación. Por último, nos proporciona un factor de forma estándar que desglosa las funciones del microcontrolador en un paquete más accesible.

Arduino ha pasado por numerosas versiones de sus microcontroladores, en concreto en este ejemplo se utilizará la versión “Mega 2560 Rev 3”

Autor en correspondencia.

Correos electrónicos: dramos014@ikasle.ehu.eus

URL: <https://www.ehu.eus/es/web/vitoria-gasteizko-ingeniaritza-eskola>

2. Arduino Mega 2560 Rev3

El Arduino Mega 2560 es una placa microcontroladora basada en el ATmega2560. Tiene 54 pines de entrada/salida digital (de los cuales 15 pueden ser usados como salidas PWM), 16 entradas analógicas, 4 UARts (puertos seriales de hardware), un oscilador de cristal de 16MHz, una conexión USB, un conector de alimentación, un cabezal ICSP y un botón de reinicio. Su conexión a un ordenador como bien se ha mencionado anteriormente es mediante un cable USB, también se puede alimentar con un adaptador de CA a CC o una batería.

La Mega 2560 es una actualización de la Mega de Arduino, a la que reemplaza.



Figura 1, Placa Arduino Mega 2560 Rev3

Más adelante se explicará en qué pines se conectan los componentes que vamos a usar.

3. HomeSecurity

Para la realización de este ejemplo, se ha optado por emular el funcionamiento de una “Puerta de Seguridad” que utiliza un proceso de autenticación de dos pasos. Los componentes utilizados son los siguientes:

RC522 RFID Módulo

Los lectores RFID (Radio Frequency IDentification) en la actualidad están teniendo bastante acogida en los sistemas de identificación, su uso abarca desde sistemas de seguridad, acceso de personal, identificación y logística de productos, como llaves de puertas eléctricas, entre otras aplicaciones.

En este caso se usará para leer las tarjetas que estén registradas en el sistema.

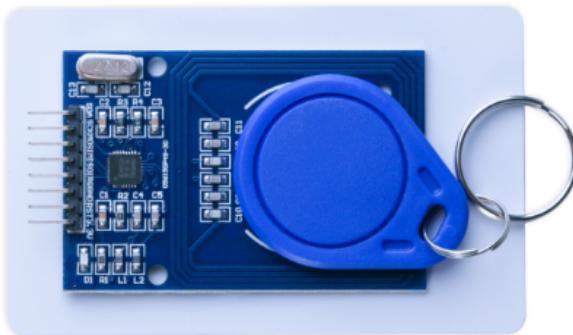


Figura 2: Módulo RC522 RFID

Keypad

Los teclados se utilizan en todo tipo de dispositivos. Así que saber cómo conectar un teclado a un microcontrolador como lo es la placa MEGA 2560 Rev3 es muy valioso para la construcción de muchos tipos diferentes de productos comerciales. Para este proyecto, el tipo de teclado que usamos es un teclado matricial. Este es un teclado que sigue un esquema de codificación que le permite tener mucho menos pines de salida. Por ejemplo, el teclado de matriz que estamos utilizando tiene 16 teclas (0-9, A-D, *, #), pero sólo 8 pines de salida. Con un teclado lineal, tendría que haber 17 pines de salida para trabajar. El esquema de codificación de matriz permite menos pines de salida y por lo tanto menos conexiones que tienen que hacer para que el teclado funcione. De esta manera, son más eficientes que los teclados lineales, ya que tienen menos cableado



Figura 3: KeyPad

Matriz de Leds (utilizado para obtener feedback)

La matriz de LED 8 x 8, utiliza un integrado MAX7219, disponible como un módulo precableados. Se podrá observar que el código y la conexión es muy sencilla ya que es facilitada en gran parte por este integrado. Especificación típica de este módulo de matriz de LED se muestra a continuación: Voltaje de funcionamiento: DC4.7V-5.3V Típica tensión: 5V Corriente de funcionamiento: 320mA Máxima corriente de funcionamiento:2A

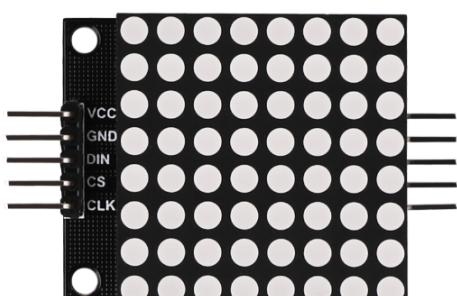


Figura 4: Módulo MAX7219

Sensor ultrasónico

El sensor HC-SR04 es un sensor de distancia de bajo costo, su uso es muy frecuente en la robótica, utiliza transductores de ultrasonido para detectar objetos. El rango de detección está entre los 2 cm - 400 cm, el margen de error es de 3mm. Para empezar a utilizar el sensor HC-SR04 solo necesitas una placa Arduino, en este tutorial utilizaremos un Arduino Uno R3, pero puedes utilizar cualquier placa de Arduino, el procedimiento es el mismo.

El módulo utiliza una señal de nivel alto de al menos 10us. Envía automáticamente 8 ciclos de ultrasonido a 40 kHz.

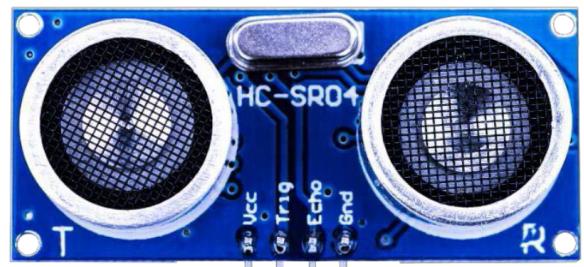


Figura 5: Módulo Sensor ultrasónico

Servo Motor (encargado de abrir y cerrar la puerta)

Servo es un tipo de motorreductor que sólo puede girar 180 grados. Se controla mediante el envío de impulsos eléctricos de la placa de MEGA2560 R3. Estos pulsos le dice al servo qué posición se debe mover a. El Servo tiene tres cables, de que el marrón es el cable a tierra y deben conectarse a GND puerto MEGA2560, el rojo es el cable de corriente y debe conectarse al puerto de 5v y la naranja es el alambre de señal y debe conectarse al puerto#9



Figura 6: Servo Motor

4. IDE

A la hora de construir el código que vamos a utilizar en el microcontrolador, se puede optar por varias herramientas que se ofrecen en la web oficial, <https://www.arduino.cc/>. En este caso se ha optado por ir un poco más allá y utilizar una herramienta más competente que se adapte a un nivel mayo de entendimiento del ámbito de la programación.

Se trata de **Visual Studio**, el cual mediante un plugin, **Visual Micro**, podemos programar el microcontrolador de arduino desde un entorno de desarrollo integrado con muchas potencialidades. Visual Micro es un software de pago que incluye una versión gratuita, con dicha versión para el ejemplo que se presenta en este documento es más que suficiente.

¿Cómo funciona Visual Micro?

El IDE proporcionado por arduino para windows, ofrece una funcionalidad simple e intuitiva para que el proceso de programación sea lo más rápido y fácil posible. El funcionamiento se describe perfectamente en la Figura X, donde el Arduino IDE se comunica con la librería gcc toolchain que mediante un driver usb logra la comunicación con el microcontrolador.

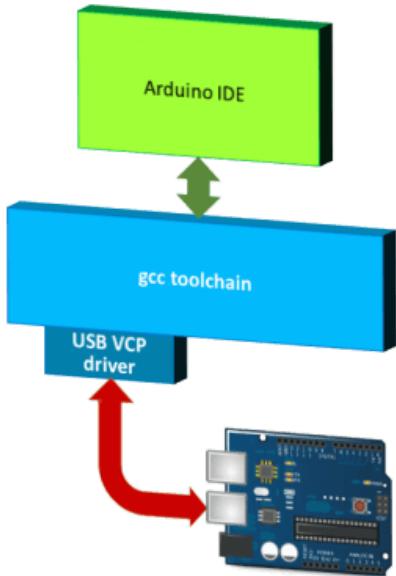


Figura 9: funcionamiento de arduino IDE

Al instalar el plugin Visual Micro en el Visual Studio el funcionamiento es similar, lo diferencia es que el Visual Studio se comunica con gcc toolchain a través de Visual Micro. Así es como se ve un entorno de desarrollo después de haber instalado Visual Studio/Atmel Studio y Visual Micro:

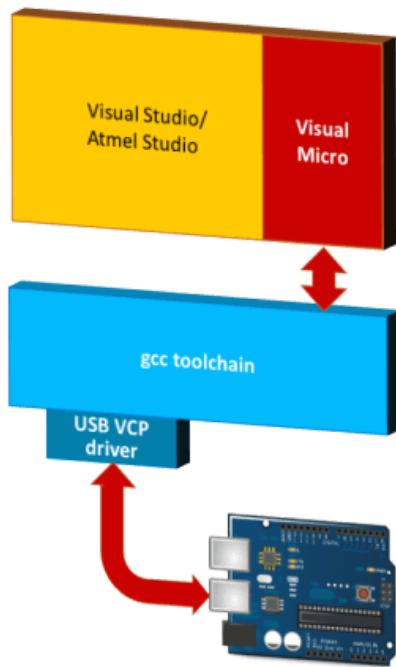


Figura 10: Funcionamiento de Visual Studio

La instalación es muy sencilla y se puede realizar desde el propio Visual Studio en la sección de plugins o si no siguiendo las indicaciones en la propia [página](#) de Visual Micro.

Al crear el proyecto la estructura es la mostrada en la figura 11.

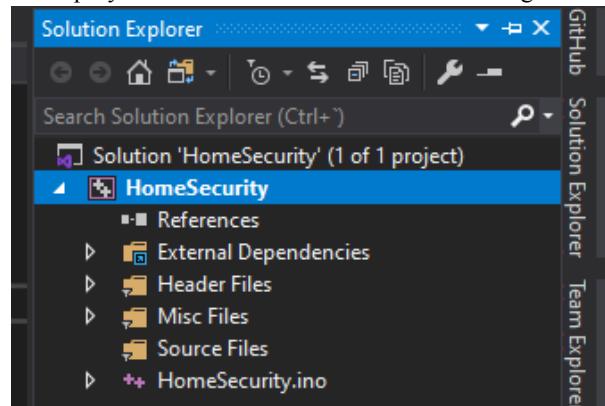


Figura 11: Aspecto de un proyecto en Visual Studio

La estructura del archivo **.ino**, donde vamos a programar es la misma que el tradicional IDE de arduino. Por lo que la retrocompatibilidad con el IDE tradicional de Arduino es completa.

```

/*
1  * Name:      HomeSecurity.ino
2  * Created:   12/9/2020 10:09:37 AM
3  * Author:    dramo
4  */
5

6
7 // the setup function runs once when you press reset or power the board
8 void setup() {
9
10}
11
12 // the loop function runs over and over again until power down or reset
13 void loop() {
14
15}
16

```

Figura 12: Estructura del archivo .ino

GitHub

Una de las posibilidades que nos ofrece Visual Studio es el uso de GitHub una plataforma que nos permite llevar un control de versiones de nuestro código basado en “Commit”, de esta forma podemos subir nuestro código a la plataforma y tener un registro de todas las versiones que se han subido, especialmente útil si se trabaja en un proyecto como este. Además de ofrecer la posibilidad de tener el código accesible desde cualquier parte, pudiendo clonarlo en cualquier equipo.

El repositorio del proyecto sería este: <https://github.com/dapro23/HomeSecurity>

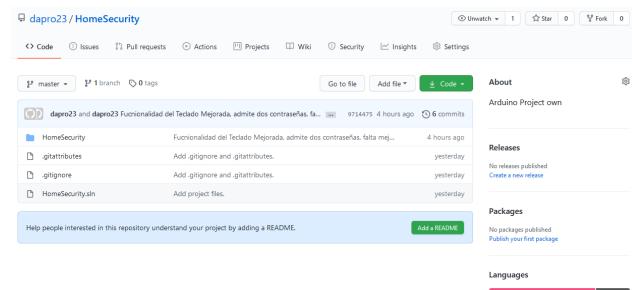


Figura 13: Imagen del proyecto subido a GitHub

5. Conexiones

En esta sección se detalla cómo estará conectado el circuito

En la figura 14, se puede apreciar una vista del circuito ya estructurado y con una representación muy cercana a su funcionamiento real. Los componentes que se pueden apreciar son los siguientes:

1. Sensor Ultrasónico, detecta cuando alguien intenta acercar una tarjeta RFID y activa el módulo RFID
2. Modulo RFID, encargado de comprobar si la tarjeta está registrada en el sistema.
3. Matriz de Led, se utiliza como “FeedBack” Pâra saber que comprobación se está realizando y si esta ha sido exitosa o no.
4. Teclado numérico, encargado de comprobar si la contraseña introducida está registrada en el sistema.
5. Microcontrolador Arduino Mega 2560 Rev3
6. BreadBoard
7. Tarjetas RFID, una de ellas no está registrada en el sistema, por lo que se podrá comprobar que el reconocimiento funciona correctamente.
8. Servo Motor que simula el mecanismo de abertura y cierre de la puerta.

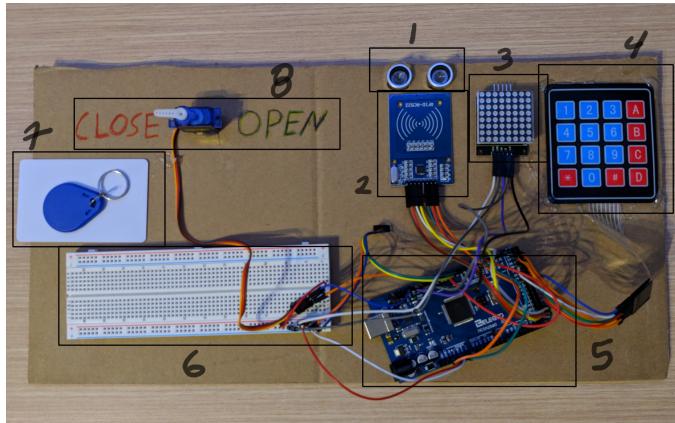


Figura 14: Proyecto montado en Arduino

Las conexiones de los módulos se han realizado de la siguiente manera.

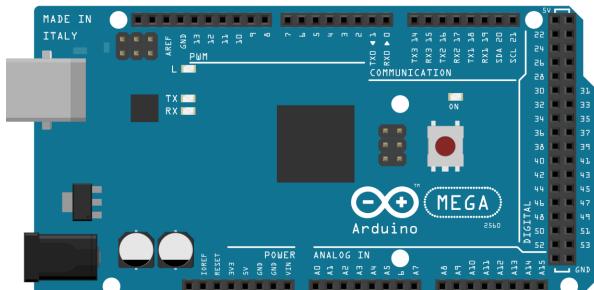


Figura 15: Arduino Mega 2560 Rev3

1. RC522 RFID Módulo

SDA	SCK	MOSI	MISO	IRQ	GND	RST	3.3V
D53	D52	D51	D50		gnd	5	3.3v

2. Membrane Switch Module (Keypad)

1	2	3	4	5	6	7	8
36	34	32	30	28	26	24	22

3. MAX7219 (utilizado para obtener feedback)

VCC	GND	DIN	CS	CLK
5V	GND	12	11	10

4. Servo motor, encargado de abrir y cerrar la puerta.

5V	GND	control
5v	gnd	7

5. Sensor ultrasónico

5V	Trig	Echo	GND
5v	4	3	gnd

6. Modularidad

A continuación se explicará el código utilizado para hacer funcionar a los componentes del sistema, incluyendo la explicación de la estructura, librerías y métodos usados.

Primero que todo se tienen que importar todas las librerías necesarias para que el código compile, serían las siguientes:

```
#include <time.h> //Incluimos la librería Time
//Servo Motor*****
#include <Servo.h>
Servo myservo;
//Keypad*****
#include <keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons on the keypads
char hexaKeys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};
byte rowPins[ROWS] = { 36, 34, 32, 30 }; //connect to the row pinouts of the keypad
byte colPins[COLS] = { 28, 26, 24, 22 }; //connect to the column pinouts of the keypad
//Initialize an instance of class NewKeypad
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
//RFID*****
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 5 // Configurable, see typical pin layout above
#define SS_PIN 53 // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
/* Set your new UID here! */
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
MFRC522::MIFARE_Key key;
//Matrix*****
#include "LedControl.h"
LedControl lc = LedControl(12, 10, 11, 1);
//Sensor Ultrasonico*****
#include "SR04.h"
#define TRIG_PIN 4
#define ECHO_PIN 3
SR04 sr04 = SR04(ECHO_PIN, TRIG_PIN);
```

Figura 16: Librerías y variables definidas antes de la ejecución del código

Además aquí es donde se le indica al microcontrolador que está conectado en cada puerto.

Primero que todo tenemos el método principal, llamado loop(), en el que se describe el proceso del programa, y como se le indica al usuario que está ocurriendo en cada momento. Los comentarios indican perfectamente en qué etapa se encuentra el programa.

```
void loop() {
    //Detecta la presencia de un usuario
    a = sr04.Distance();

    if (a<10)
    {
        //Le indica al Usuario que se tiene que autenticar con el RFID
        escribir("RFID");
        delay(500);
        if (comprobacionRFID())
        {
            escribir("OK");
            delay(500);

            //Le indica al Usuario que se tiene que autenticar con el teclado
            escribir("TECLADO");
            if (comprobacionTeclado())
            {
                escribir("OK");
                escribir("PUERTA");
                puerta(true);
                matrixOff();
            }
            else {
                //Le indica al Usuario que la autenticacion con el teclado ha fallado
                // y que se debe empezar de nuevo el proceso
                escribir("F");
                delay(250);
                escribir("RFID");
                delay(250);
                matrixOff();
            }
        }
        else {
            //Le indica al Usuario que la autenticacion con el RFID ha fallado
            escribir("F");
            matrixOff();
        }
    }
}
```

Figura 17: Loop principal del programa

Como se puede apreciar contamos con varias llamadas a distintos métodos, sea el escribir(), matrixOff(), etc.

El primer método a detallar será el que le da información al usuario sobre qué está ocurriendo en cada momento. En este caso, se utiliza la Matriz de leds de 8x8, cuya utilización es muy intuitiva.

Lo primero sería definir qué información o más bien, qué leds se deben encender y cuáles apagar. Para esto utilizamos un array de bytes de 8 posiciones, en el que detallaremos esa información.

```
byte flechaD[8] = {
    B00011000,
    B00011000,
    B00011000,
    B00011000,
    B10011001,
    B01011010,
    B00111100,
    B00011000
};
```

Figura 18: Array de Byte

Posteriormente utilizamos un bucle for para encender cada fila una por una, y aprovecha la incapacidad del ojo humano a percibir cambios muy rápidos en la matriz, creando la ilusión de imagen que se crea al instante.

```
for (int i = 0; i < 8; i++) {
    lc.setRow(0, i, flechaD[i]);
}
```

Figura 19: Bucle para encender los leds de la matriz

La cabecera de este método contendrá un parámetro que se usará para indicar que se debe mostrar en la matriz, “void escribir(String a)…” siendo “a” el parámetro de tipo String.

El siguiente método será el encargado de abrir y cerrar la puerta, recibe por parámetro un tipo de dato verdadero o falso, que indica si se tiene que abrir o cerrar la puerta.

```
void puerta(boolean a) {
    if (a == false) { //Cierra la puerta
        myservo.write(0); // move servos to center position -> 90°
    }
    else if (a == true) { //Abre la puerta
        myservo.write(180); // move servos to center position -> 90°
        delay(3000);
        puerta(false);
    }
}
```

Figura 20: Método que controla el servo motor

Una vez entendidos estos dos métodos, pasamos al proceso de autenticación en sí. Primero tenemos el proceso de verificación de la tarjeta RFID,

```
//Clave*****
byte validKey1[4] = { 0xB9, 0xB5, 0xB5, 0xB0 };

while(true) {
    if (mfrc522.PICC_IsNewCardPresent())
    {
        if (mfrc522.PICC_ReadCardSerial())
        {

            Serial.print(F("Card UID:"));
            printArray(mfrc522.uid.uidByte, mfrc522.uid.size);
            Serial.println();

            if (isEqualArray(mfrc522.uid.uidByte, validKey1, 4)) {
                Serial.println("Tarjeta valida");
                return true;
            }

            else {
                Serial.println("Tarjeta invalida");
                return false;
            }
        }
    }
}
```

Figura 21: Partes del Método comprobacionRFID()

Lo primero que aparece debajo de la cabecera es la clave de la tarjeta RFID, clave que se deberá comparar con la que el módulo lee del usuario. Posteriormente tememos el proceso de verificación, que utiliza el método isEqualArray para saber si la clave es idéntica a la almacenada.

```
boolean isEqualArray(byte* arrayA, byte* arrayB, int length)
{
    for (int index = 0; index < length; index++)
    {
        if (arrayA[index] != arrayB[index]) return false;
    }
    return true;
}
```

Figura 22: Método isEqualArray

Luego de la verificación del RFID, tenemos la del teclado, donde se compara la clave introducida con las almacenadas.

```
char customKey = customKeypad.getKey();

//Cancela el proceso de verificación del teclado si pulsamos *
if (customKey == '*') {
    return false;
}

if (customKey) {
    temp[pos] = customKey;
    pos++;
    Serial.println(customKey);
    escribir("DOT");
}

if (pos == 5) {
    pos = 0;
    break;
}
```

Figura 23: Parte interna del whilen, en el método comprobacionTeclado

Como se puede apreciar el proceso es simple, primero se almacena la clave introducida y luego se compara con las que ya están registradas utilizando el método comprobacionPassword, al cual se le pasa por parámetro la clave que el usuario ha introducido.

```
*****PASSWORD
char password1[5] = { '1','2','3','4','5' };

*****PASSWORD
char password2[5] = { '5','4','3','2','1' };

boolean out = false;
for (int i = 0; i < 5; i++) {

    //se agregan tantos if como contraseñas existan
    if(password[i] == password1[i]) {
        out = true;
    }else if(password[i] == password2[i]) {
        out = true;
    }
    else {
        out = false;
    }
}

return out;
```

Figura 24: Método comprobacionPassword

Tener en cuenta que si se desea agregar más claves, en este método es donde se deberían definir.

Por último, si en cualquier paso de la verificación falla algo, el bucle principal del programa reinicia el proceso de verificación.

7. Proyección de futuro

En todos los proyectos se debe buscar un equilibrio entre los recursos con los que se cuenta y el nivel exigido o auto exigido a lograr. Este proyecto no es una excepción de la regla, por lo que contiene aspectos que se pueden mejorar y por lo tanto hacer que sea un mejor producto de cara a una implementación real del mismo.

Algunos de los aspectos a mejorar serían los siguientes:

- Sería conveniente mejorar el proceso de autenticación de la contraseña, de tal forma que no sea necesario empezar todo el proceso otra vez
- También el proceso realizado en el método principal, loop(), puede ser mejorado, incluyendo nuevos componentes que ayudan a que el proceso de identificación sea aún más claro.

8. Conclusiones

Arduino como tal utiliza una versión simplificada de C++, que a su vez tiene una sintaxis prácticamente igual a Java, dos lenguajes muy usados y conocidos a nivel mundial. Esto le permite ser relativamente fácil de programar y usar, solo necesitando unas nociones básicas de conceptos de programación fundamentales, como son bucles, iteraciones, condicionales, declaración de variables, etc. Por otro lado, la gran comunidad con la que cuenta el concepto de Arduino, le ayuda mucho a llegar a casi cualquier tipo de público, puesto que la gran variedad de ejemplo hace que en mucho de los casos no se necesite entender que hacer exactamente una parte del código o un componentes, simplemente con saber que nos devuelve es más que suficiente, aunque desde luego lo más recomendado es entender en profundidad todo lo que se use.

En cuanto a este ejemplo se refiere, las intenciones no son crear un sistema real, sino emular el funcionamiento de uno, y sobretodo aplicar una metodología de programación que nos permita agregar componentes y actualizar el sistema sin mayor esfuerzo. De esta forma se asegura que la aplicación en un entorno real de uso, sea fácil y lo más importante, viables.

Cabe recalcar que los componentes usados representan conceptos, por ejemplo, la matriz de led representa el "FeedBack" del sistema, o como el usuario puede saber qué está ocurriendo y reaccionar a ello. Por lo tanto cada componente puede ser fácilmente sustituido por un homólogo real, o de mayores capacidades, si que representa una mayor problema.

English Summary

HomeSecurity

Abstract

In this document a project will be developed using the Arduino Mega 2560 microcontroller. In particular, an example is developed that emulates the operation of a "Security Door" in a home, or any other installation that requires it. It is a two-step security system, completely modular and with great possibilities of extension of components, being able to include new methods of authentication.

Keywords:

Microcontroller, IDE, Module,

9. Referencias

Página Oficial de Arduino

<https://www.arduino.cc/>

Visual Studio

<https://visualstudio.microsoft.com/es/>

Visual Micro

<https://www.visualmicro.com/>