
Desarrollo de sistema inmótico basado en plataforma Arduino

Subproyecto: 1. Diseño de sistema de control de accesos RFID
125 Khz y/o 13,56 Mhz (Mifare) autónomo

Autores:

Carlos Villora Fernández

Antonio Victoriano Bejarano Rísquez

Antonio Morcillo Ibañez

Índice de contenido

1.- Introducción.....	3
2.- Especificaciones.....	3
2.1.- Tarjetas RFID.....	3
2.2.- Tarjeta Electrónica.....	3
3.- Desarrollo de la tarjeta electrónica.....	4
4. Historia del RFID.....	5
5.- Funcionamiento del RFID.....	5
6.- Aplicaciones.....	6
7.- Reglamentación.....	7
8.- Descripción del prototipo.....	7
9.- Programación.....	11
10.- Lista de Material.....	25
11.- Presupuesto.....	26

1.- Introducción.

El objetivo de este proyecto es desarrollar un sistema de control de accesos con tecnología RFID, gobernado de forma autónoma, con el cual se pretende controlar los permisos de accesos a los usuarios.

Tarjeta Maestra.- El la tarjeta RFID programada para poder dar de alta a las distintas tarjetas de los usuarios.

Tarjeta Borrar.- El la tarjeta RFID programada para poder dar de baja a las distintas tarjetas de los usuarios.

Tarjetas de usuarios.- Serán las distintas tarjetas RFID que programaremos para darlas de alta/baja, con las tarjetas anteriormente mencionadas.

2.- Especificaciones.

Con las siguientes funciones: Compatible con cualquier abrepuertas eléctrico de mercado. Salida a relé 10 A. Alimentación 24 Vcc Capacidad para 200 usuarios por puerta Posibilidad de administrar el sistema sin necesidad de PC o accesorio de configuración/programación Integración estética. Instalable en cualquier caja para mecanismos estándar de mercado Altas y Bajas de tarjetas se realizan por medio de tarjetas maestras para administración de permisos

2.1.- Tarjetas RFID.

Se configurarán tres tipos de tarjetas para este proyecto. Tarjeta MAESTRA, encargada de dar de ALTA las tarjetas de usuarios, Tarjeta BORRAR encargada de dar de BAJA las tarjetas de usuarios y las Tarjetas USUARIOS, con las cuales podremos abrir la puerta si han sido autorizadas, a través, de la tarjeta maestro.

2.2.- Tarjeta Electrónica.

La tarjeta electrónica es la encargada de leer el código de las distintas tarjetas RFID y realizar el siguiente proceso:

- Si pasamos la tarjeta de usuario autorizada por el lector RFID activará el abrepuertas.
- Si pasamos la tarjeta Maestra el sistema se prepara para que pasaos una tarjeta virgen y, así, darle permiso para abrir la puerta (ALTA).
- Si pasamos la tarjeta Borrar el sistema se prepara para que pasemos una tarjeta, que ha sido dada de alta, para borrarla del sistema (BAJA).

Sus partes serán:

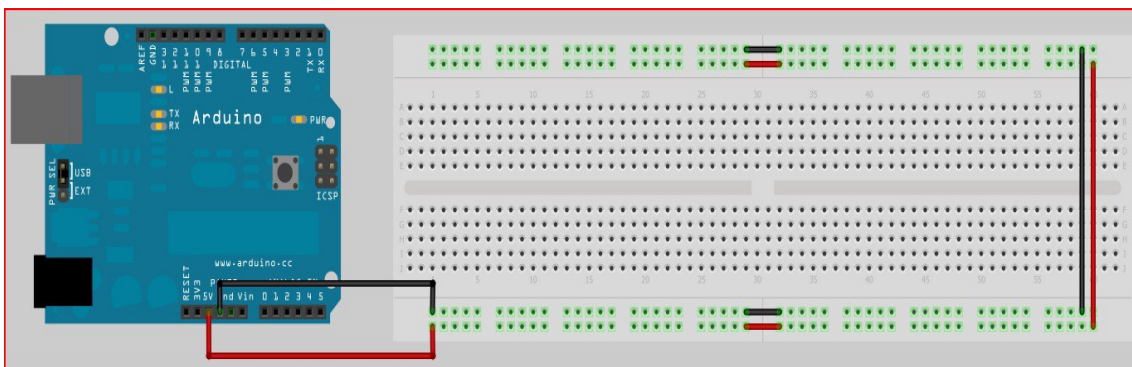
- 1.- Alimentación de 24 voltios externa.
- 2.- Tarjeta Arduino.

- 3.- Lector RFID.
- 4.- Led RGB.
- 5.- Resistencias, led, etc.

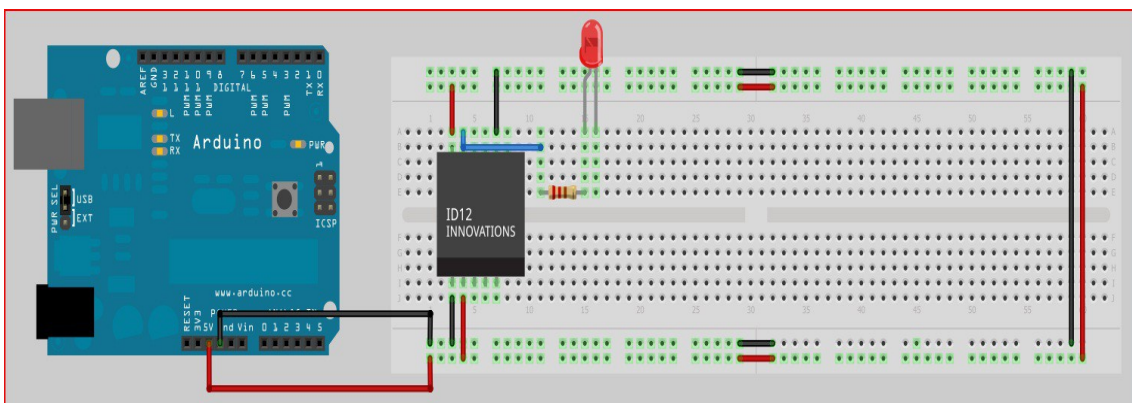
3.- Desarrollo de la tarjeta electrónica.

En este apartado se muestra el montaje del prototipo sobre una protoboard.

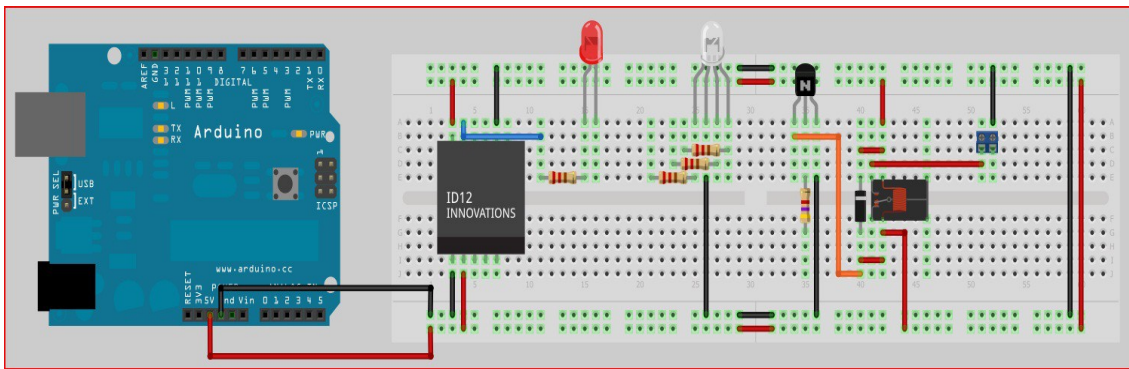
1.- Cableado de +5v y GND:



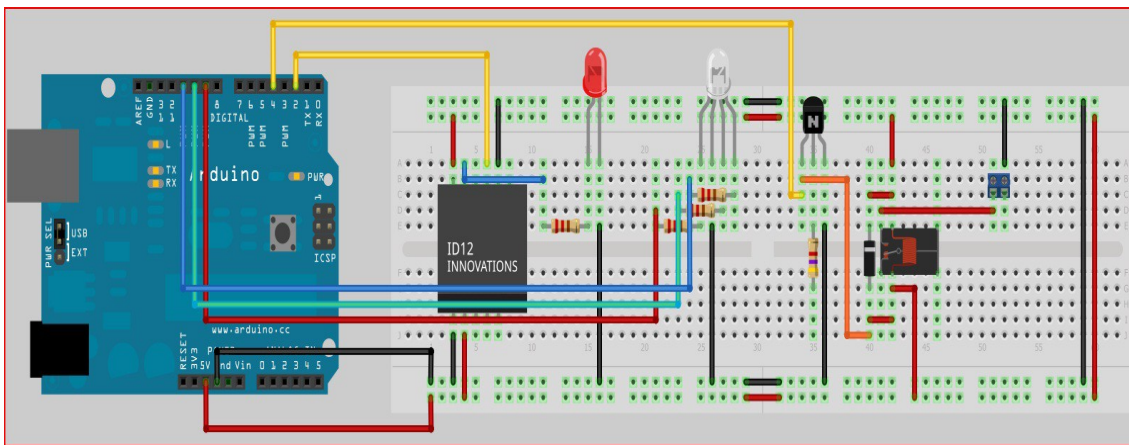
2.- Colocando el lector RFID y el LED de lectura:



3.- Led RGB, transistor bc547 y rele:



4.- Cableado final:



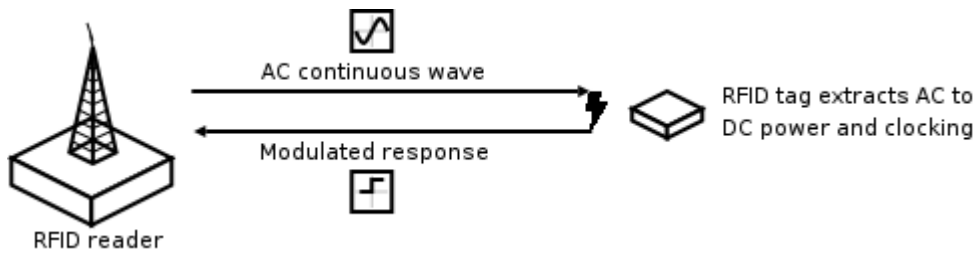
4. Historia del RFID.

Las siglas RFID proceden de Radio Frequency Identification (Identificación por Radio Frecuencia). Existen fuentes que indican que la tecnología RFID existe desde el 1920 en un proyecto desarrollado por el MIT, otros dicen que fueron los ingleses los que lo inventaron en 1945 durante el transcurso de la 2ª Guerra Mundial, esto se basaba en una tecnología llamada IFF y que utilizaban para identificar los aviones como amigos o enemigos.

Aunque el desarrollo de la RFID tal cual se conoce en estos mismos empieza en los años 90 con el abaratamiento de muchos de los componentes necesarios para que funcione.

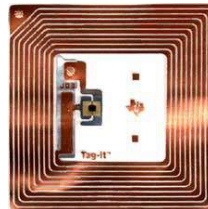
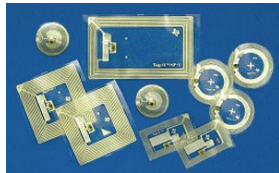
5.- Funcionamiento del RFID.

El funcionamiento de las RFID es muy simple. Los tags (o tarjetas) RFID, cuando reciben una pequeña corriente mediante radio frecuencia son capaces de emitir una onda con su número identificativo y esta onda es captada por un lector RFID que se encarga de pasar la información a un formato digital.



Existen una gran variedad de encapsulados para los tags. En tarjeta, encapsulados para implantaciones quirúrgicas, flexible para ponerlos sobre pegatinas, en llaveros y pulseras. Al ser un circuito muy pequeño se puede introducir en cualquier formato.

Los tags pueden ser, activos, semiactivos o pasivos. La principal diferencia es que los activos y los semiactivos necesitan una fuente externa de alimentación mientras que los pasivos no.

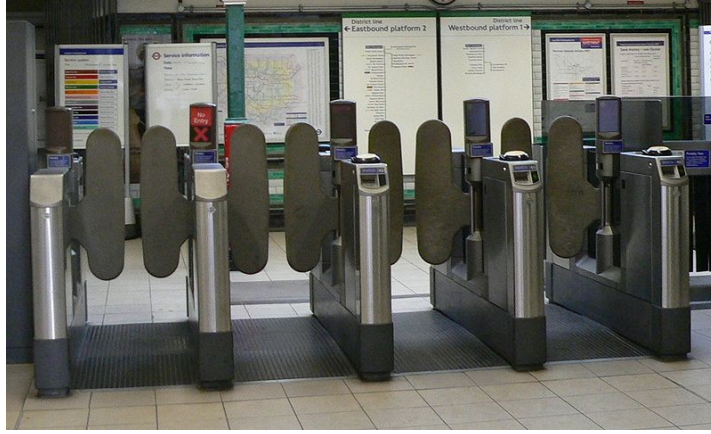


6.- Aplicaciones.

Los sistemas de control de acceso se usan muy frecuentemente en grandes empresas en las que hay una gran cantidad de trabajadores, los cuales no deben tener acceso a todas las zonas del edificio, como puede ser, un administrativo no debería entrar en un laboratorio ya que seguramente desconozca las normas de seguridad mínimas para estar allí.

No es necesario que para hacer uso de un sistema de control de accesos con RFID la empresa deba ser grande. También es aplicable para empresas pequeñas ya que aparte de controlar accesos, este sistema puede servir como sistema de fichaje para marcar las horas trabajadas.

Otra gran vía de aplicación para el sistema RFID es la gestión de pasajeros, siendo cada vez mayor el número de transportes públicos (metros, autobuses...) que utilizan este sistema, sustituyéndolo por los anticuados billetes. Con este sistema se permite una mayor personalización del control de zonas de viaje o de la administración del dinero.



Control de acceso RFID – Metro de Londres

Actualmente la tecnología RFID está muy extendida en el campo industrial, para el seguimiento de paquetes, palés y barricas. Se utiliza también en el mundo de la ganadería para identificar animales. Muchas autopistas tienen implementado un sistema RFID por el cual se paga con una tarjeta de crédito al pasar por el peaje, haciendo más fluido el tráfico por estos. Actualmente se está introduciendo en el sector textil.

7.- Reglamentación.

Existen varias legislaciones vigentes para el uso de la tecnología RFID, con el fin de hacer un uso cohesionado a nivel global.


En la norma ISO todas las tecnologías RFID han sido englobadas bajo el paraguas **ISO-18000**.


En nuestro caso bajo la ISO – 18000 – 2 que es la que engloba las RFID de menos de 135 KHz. A parte existen otras normas ISO que regulan las distancias de lectura. Otros organismos que regulan el manejo de la RFID son:



- ETSI
- EPC
- ROHS

8.- Descripción del prototipo.

A continuación se documenta los componentes utilizados en el desarrollo del prototipo. El montaje de dicho prototipo sobre la protoboard está explicado más adelante.

<p>Diodo LED</p> 	<p>¿Qué hace? Se enciende cuando una corriente pasa a través de él.</p> <p>¿Cómo identificarlo? Como una pequeña bombilla.</p>	<p>Número de patillas: 2, ánodo (larga) y cátodo (corta).</p> <p>A tener en cuenta: Solo funciona en una dirección Es necesaria una resistencia para limitar la corriente que lo atraviesa.</p>
---	--	---

Resistencia 	¿Qué hace? Reduce la corriente que puede pasar por un circuito. ¿Cómo identificarlo? Forma cilíndrica. El valor de la resistencia viene dado por un código de colores	Número de patillas: 2. A tener en cuenta: Comprobar bien los colores para no coger una resistencia con un valor equivocado.
Botón 	¿Qué hace? Cuando lo pulsamos hace que el circuito quede cerrado. ¿Cómo identificarlo? Forma cuadrada con un círculo negro en el centro.	Número de patillas: 4. A tener en cuenta: Revisar con un tester que patillas son las que se quedan NC y NO.
Condensador 	¿Qué hace? Es capaz de almacenar energía en su interior. ¿Cómo identificarlo? Forma cilíndrica, con dos patillas en la base.	Número de patillas: 2. Larga (+) y corta (-). A tener en cuenta: Muy importante cerciorarse de que se ha conectado con la polaridad correcta.
Cristal 	¿Qué hace? Convierte las vibraciones mecánicas en voltajes eléctricos a una frecuencia fija. ¿Cómo identificarlo? Forma ovalada, puede ser de perfil alto o bajo, se reconoce por su recubierta de chapa.	Número de patillas: 2. A tener en cuenta: Debemos colocarlo lo más cerca posible de las patillas a las que va conectado.
Transistor 	¿Qué hace? Permite o impide el paso de la corriente. ¿Cómo identificarlo? Puede tener varios encapsulados, en este caso TO92, una pequeña pieza de plástico, con tres patillas, lleva el nombre serigrafiado.	Número de patillas: 3. A tener en cuenta: Siempre comprobar el datasheet para conocer la posición de las patillas.
Lector RFID 	¿Qué hace? Leer un "tag" RFID y comunicarlo al microcontrolador. ¿Cómo identificarlo? De forma cuadrada, lleva serigrafiado el nombre en la parte superior.	Número de patillas: 11. A tener en cuenta: Comprobar el datasheet para conocer la posición de las patillas. Es un lector de 125 KHz por lo que hay que usar el mismo tipo de "tag"

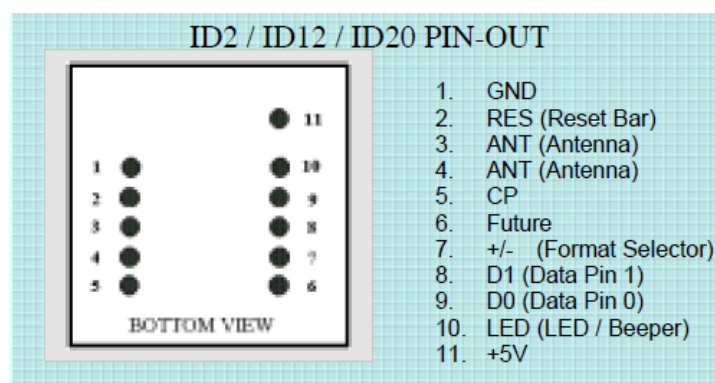
RELE 	¿Qué hace? Cuando se induce corriente en la bobina esta mueve un resorte y según sus patillas enciende o apaga un interruptor. ¿Cómo identificarlo? De forma rectangular, suelen ser transparentes por lo que podemos ver la bobina.	Número de patillas: 5 a 8. A tener en cuenta: El voltaje y amperaje soportados.
ATMEGA328P 	¿Qué hace? Un integrado con varios puertos analógicos y digitales, capaz de comunicarse vía serie. ¿Cómo identificarlo? Forma rectangular y alargada. Tiene serigrafiado el nombre en la parte superior.	Número de patillas: 28. A tener en cuenta: Lleva una pequeña muesca que indica cual que a su izquierda se encuentra el pin 1.

ID-12

El ID-12 es un lector RFID desarrollado por ID INNOVATIONS, este lector tiene una antena interior que puede llegar a los 12 cms de lectura. Este lector soporta los tipos de dato ASCII, Wiegand26 y Magnetic ABA Track2.

A continuación se muestra el patillaje y el conexionado con el integrado ATMEGA328P y con el mismo implementado en la plataforma ARDUINO.

ID-12	ATMEGA328P	ARDUINO
1	GND	GND
2	+5V	+5V
3	-	-
4	-	-
5	-	-
6	-	-
7	GND	GND
8	-	-
9	PD0(RX)	0 RX
10	-	-
11	+5v	+5V



Vista inferior del lector ID-12

Al poner la patilla 7 (+/- (Format Selector)) a masa hacemos que la salida de datos sea en forma de ASCII. La patilla 10 la conectamos a una resistencia de 220 ohms y esta a un LED el cual nos indicará cuando se ha leído una tarjeta.

Vamos a explicar lo que recibiremos por parte del lector al seleccionar ASCII. El lector envía una onda que imbuye al tag, con esa onda el tag es capaz de emitir un código que es leído por el ID12 y este lo manda al Arduino con el siguiente formato:

STX(02H)	DATA(10 ASCII)	CEHCKSUM(2 ASCII)	CR	LF	ETX(03H)
----------	----------------	-------------------	----	----	----------

Lo primero que recibimos en la transmisión es el STX o el carácter de inicio de transmisión (ASCII 02), seguido de 10 bytes que crean los HEX individuales del número. Los siguientes dos HEX son el checksum del número. Luego tenemos el CR o Carriage Return y el LF o Line Feed y por último el ETX o carácter de final de transmisión.

Lo que nosotros vamos a mostrar siempre por pantalla van a ser los 12 dígitos ASCII.

ATMEGA328P

El ATMEGA328P es un microcontrolador desarrollado por ATMEL. Es un integrado de 28 patillas. Tiene una memoria FLASH de 32K Bytes, una EEPROM de 1K Byte y una RAM de 2k Bytes.

A continuación se muestra el patillaje del integrado y la correspondiente función en la plataforma ARDUINO.

Atmega168 Pin Mapping

Arduino function			Arduino function		
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11 (PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

Digital Pins 11, 12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

9.- Programación.

El siguiente código nos sirve para hacer una demostración de las puertas a las que tiene acceso un tag. Al pasar por el lector, este recoge el código del tag y lo manda al ordenador, este devuelve las puertas que puede abrir, encendiendo el led de la puerta correspondiente.

```
#include <EEPROM.h>

#define powerPin 22 // Led Azul
#define failPin 26 // Led Rojo
#define passPin 24 // Led Verde
#define doorPin 30 // Rele
#define alarmPin 28 // Alarma

boolean programMode = false;
boolean deleteMode = false;
boolean wipeMode = false; //wipe limpiar memoria
boolean match = false; // Tarjeta Encontrada

byte storedCard[6]; // Guarda un ID leido desde la EEPROM
byte readCard[6]; // Gaurda un ID leido desde el lector ID-12
byte checksum = 0; // Guarda el checksum para comprobar que esta bien

int alarm;
int reed = 0;
```

```

void setup()
{
  pinMode(powerPin, OUTPUT);
  pinMode(passPin, OUTPUT);
  pinMode(failPin, OUTPUT);
  pinMode(doorPin, OUTPUT);
  pinMode(alarmPin, OUTPUT);
  alarm = 0;
  Serial.begin(9600);
}

void loop()
{
  byte val = 0;
  normalModeOn(); // Activo Modo Normal (Led Encendido ON, resto OFF)
  if (alarm==3)
  {
    digitalWrite(alarmPin, HIGH); // Suena el zumbador
    if (Serial.available())
    {
      if ((val = Serial.read()) == 2) // El primer byte debe se 2, que es el STX
      {
        getID(); // Funcion coger ID leido y lo coloco en readCard
        if (isMaster(readCard)) // Comprobamos si es la tarjeta maestro o la de borrar
        {
          digitalWrite(alarmPin, LOW);
          alarm = 0;
          checksum = 0;
        }
        else
        {
          checksum = 0;
        }
      }
      checksum = 0;
    }
  }
  else
  {
    if (programMode) // Modo programa para añadir una nueva tarjeta
    {
      programModeOn(); // Durante este modo los leds hace un ciclo
      if (Serial.available())
      {

```

```

if (val=Serial.read()==2)
{
  getID();
  if (isMaster(readCard) || isDelete(readCard) || isWipe(readCard))
  {
    programMode = false;
    if (isMaster(readCard))
    {
      openDoor(2);
    }
    else
    {
      failedWrite();
    }
    checksum = 0;
  }
  else
  {
    writeID(readCard); // Si no maestra o borrar, se guarda en la EEPROM
    programMode = false;
    checksum = 0;
  }
}
}
}
else if (deleteMode) // Modo Borrar, para desactivar alguna tarjeta
{
  deleteModeOn(); // Ciclo de leds verde y rojo
  if (Serial.available())
  {
    if (val=Serial.read()==2)
    {
      getID();
      if (isMaster(readCard) || isDelete(readCard) || isWipe(readCard))
      {
        deleteMode = false;
        checksum = 0;
        failedWrite();
      }
      else
      {
        deleteID(readCard); // Borrarnos la tarjeta de la EEPROM
        deleteMode = false;
        checksum = 0;
      }
    }
  }
}

```

```

    }
  }
}
else if (wipeMode) // Modo limpiar la memoria EEPROM
{
  Serial.end();
  wipeModeOn();
  for (int i=0; i < 512; i++)
  {
    EEPROM.write(i,0);
  }
  wipeMode = false;
  wipeModeOn();
}
// Operaciones normales
else
{
  if (Serial.available())
  {
    if (val=Serial.read()==2)
    {
      getID();
      byte bytesread = 0;
      for (int i = 0; i < 5; i++)
      {
        if (readCard[i] < 16) // imprimir un 0 si la tarjeta leida es menor que 16
        {
          Serial.print(0);
        }
        Serial.print(readCard[i],HEX); // Imprime el valor HEX leido
        Serial.print(" ");
      }
      Serial.println();
      Serial.print("Checksum: ");
      Serial.print(readCard[5], HEX); // Checksum lido de la tarjeta
      if (readCard[5] == checksum) // Comprobar si el 5 byte leido es el mismo al
calculado
      {
        checksum = 0;
        if (isMaster(readCard)) // Es la tarjeta maestro
        {
          programMode = true;
          alarm = 0;

```



```

byte bytesread = 0;
byte i = 0;
byte val = 0;
byte tempbyte = 0;
// 5 HEX Byte son 10 ASCII bytes
while (bytesread < 12) // lee 10 digitos + 2 del checksum
{
    if (Serial.available()) // Comprueba la llegada de datos por el puerto serial
    {
        val = Serial.read();
        if ((val==0x0D) || (val==0x0A) || (val==0x03) || (val==0x02))
        {
            // Si se detecta el STX o ETX se para
            break;
        }
        if ((val >= '0') && (val <= '9')) // Hacemos la conversion ASCII/HEX
        {
            val = val - '0';
        }
        else if ((val >= 'A') && (val <= 'F'))
        {
            val = 10 + val - 'A';
        }
        if (bytesread & 1 == 1) // Cada dos caracteres en ASCII = 1 BYTE en HEX
        {
            // Hacemos sitio para los digitos HEX, moviendo el digito anterior 4 veces a la
            // izquierda
            readCard[bytesread >> 1] = (val | (tempbyte << 4));
            if (bytesread >> 1 != 5) // si estamos en el byte del checksum
            {
                checksum ^= readCard[bytesread >> 1]; // calculamos el checksum haciendo
                // la XOR.
            }
        }
        else // Si es el primer caracter HEX
        {
            tempbyte = val;
        }
        bytesread++; // incrementamos el contador para seguir buscando
    }
}
bytesread = 0;
}

```



```

// Funcion: LEER UN ID DESDE LA EEPROM Y GUARDARLA EN
storedCard[6]
void readID(int number) // number = posicion en la EEPROM para coger los 5 bytes
{
    int start = (number * 5) - 4;
    for (int i=0; i<5; i++)
    {
        storedCard[i] = EEPROM.read(start+i); //Asignamos valores leidos desde la
EEPROM a un array
    }
}

// Funcion: ESCRIBIR UN ARRAY EN LA EEPROM EN EL SIGUIENTE SOLT
DISPONIBLE
void writeID(byte a[])
{
    if (!findID(a)) // Antes de escribir en la EEPROM miramos a veri si ya esta
guardada la tarjeta
    {
        int num = EEPROM.read(0); // La posicion 0 guarda el numero de tarjetas
almacenadas
        int start = (num*5) + 1; // Deducimos donde empieza el siguiente slot
EEPROM.write(0, num); // Actualizamos el contador de tarjetas
        for (int j=0; j < 5; j++)
        {
            EEPROM.write(start + j, a[j]); //Escribimos los valores del array en la EEPROM
en su posicion correcta
        }
        successWrite();
    }
    else
    {
        failedWrite();
    }
}

// Funcion: BORRAR UN ARRAY almacenado EN LA EEPROM del SOLT
DESIGNADO
void deleteID(byte a[])
{
    if (!findID(a)) // Antes de borrar, miramos a ver si tenemos esta tarjeta
almacenada
    {

```

```

    failedWrite(); // Si no esta
}
else
{
    int num = EEPROM.read(0); // Cargamos en num el numero de tarjetas
    almacenadas en la EEPROM
    int slot;
    int start;
    int looping; // Numero de veces a realizar el bucle
    int j;
    int count = EEPROM.read(0);
    slot = findIDSLOT(a); // Declaramos el numero de slots a borrar
    start = (slot*5) - 4;
    looping = ((num - slot)*5);
    num--;
    EEPROM.write(0,num); // Actualizamos el contador

    for (j=0; j<looping; j++)
    {
        EEPROM.write(start+j, EEPROM.read(start+5+j));
    }
    for (int k=0; k < 5; k++)
    {
        EEPROM.write(start+j+k,0);
    }
    successDelete();
}
}

// Funcion: ENCONTRAR EL NUMERO DEL SLOT DE LA ID A BORRAR
int findIDSLOT(byte find[])
{
    int count = EEPROM.read(0); // Leemos el numero de tarjetas guardadas
    for (int i=1; i <= count; i++)
    {
        readID(i); // Leemos una ID de la EEPROM que almacenamos en storedCard[6]
        if (checkTwo(find, storedCard))
        {
            return i; // Devuelve el numero del slot
            break;
        }
    }
}
}

```

// Funcion: COMPROBAR QUE DOS ARRAY SON IGUALES

boolean checkTwo(byte a[], byte b[])

```
{
  if (a[0] != NULL) // Aseguramos de que hay algo en el primer array
  {
    match = true; // Asumimos que coinciden desde el principio
  }
  for (int k=0; k<5; k++)
  {
    if (a[k] != b[k])
    {
      match = false;
    }
  }
  if (match) // Comprobamos si siguen coincidiendo
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

// Funcion: MIRAR EN LA EEPROM PARA COMPROBAR SI ALGUNA DE LAS
ID ALMACENADAS COINCIDE CON LA

// ULTIMA QUE HEMOS PASADO

boolean findID(byte find[])

```
{
  int count = EEPROM.read(0);
  for (int i = 1; i <=count; i++)
  {
    readID(i); // Leer una ID desde la EEPROM y almacenarla en storedCard[6]
    if (checkTwo(find, storedCard))
    {
      return true;
    }
  }
  return false;
}
```

// Funcion: ABRIMOS LA PUERTA Y ENCENDEMOS EL LED VERDE UN MOMENTO

void openDoor(int setDelay)

```
{
    setDelay *= 1000; // Ponemos el delay en segundos
    digitalWrite(powerPin, LOW);
    digitalWrite(failPin, LOW);
    digitalWrite(passPin, HIGH);
    digitalWrite(doorPin, LOW);
    delay(setDelay);

    do {
        digitalWrite(doorPin, HIGH);
        reed = analogRead(A1);
        digitalWrite(failPin, HIGH);
        delay(500);
        digitalWrite(failPin, LOW);
        delay(500);
    }
    while (reed > 60);

    delay(1000);
    digitalWrite(doorPin, LOW);
    delay(setDelay);
    digitalWrite(passPin, LOW);
}
```

// Funcion: FLASEAR EL LED ROJO SI FALLA AL PASAR LA TARJETA

void failed()

```
{
    digitalWrite(passPin, LOW);
    digitalWrite(powerPin, LOW);
    digitalWrite(failPin, HIGH);
    delay(1200);
}
```

// Funcion: COMPROBAR SI LA TARJETA QUE HEMOS PASADO ES LA MAESTRA

boolean isMaster(byte test[])

```
{
    byte bytesread = 0;
    byte i = 0;
```

```

byte val[10] = {
    '4','C','0','0','2','0','E','3','E','8' }; // Numero de la tarjeta MAESTRA
byte master[6];
byte checksum = 0;
byte tempbyte = 0;
bytesread = 0;

for (i = 0; i < 10; i++) // Lo primero es convertir el array en un array de 5 HEX
{
    if ((val[i] >= '0') && (val[i] <= '9')) // Convertimos un caracter en HEX.
    {
        val[i] = val[i] - '0';
    }
    else if ((val[i] >= 'A') && (val[i] <= 'F'))
    {
        val[i] = 10 + val[i] - 'A';
    }
    if (bytesread & 1 == 1) // Cada dos digitos HEX, añadimos uno al codigo
    {
        master[bytesread >> 1] = (val[i] | (tempbyte << 4));
        if (bytesread >> 1 != 5)
        {
            checksum ^= master[bytesread >> 1]; // Calculamos el checksum
        }
    }
    else
    {
        tempbyte = val[i];
    }
    bytesread++;
}
if (checkTwo(test, master)) // Comprobamos si coincide lo leído con el master
{
    return true;
}
else
{
    return false;
}
}

// Funcion: COMPROBAR SI LA TARJETA QUE HEMOS PASADO ES LA
BORRAR
boolean isDelete(byte test[])

```

```

{
byte bytesread = 0;
byte i = 0;
byte val[10] = {
    '4','C','0','0','2','1','0','B','F','B'    }; // Numero de la tarjeta BORRAR
byte master[6];
byte checksum = 0;
byte tempbyte = 0;
bytesread = 0;

for (i = 0; i < 10; i++) // Lo primero es convertir el array en un array de 5 HEX
{
    if ((val[i] >= '0') && (val[i] <= '9')) // Convertimos un caracter en HEX.
    {
        val[i] = val[i] - '0';
    }
    else if ((val[i] >= 'A') && (val[i] <= 'F'))
    {
        val[i] = 10 + val[i] - 'A';
    }
    if (bytesread & 1 == 1) // Cada dos digitos HEX, añadimos uno al codigo
    {
        master[bytesread >> 1] = (val[i] | (tempbyte << 4));
        if (bytesread >> 1 != 5)
        {
            checksum ^= master[bytesread >> 1]; // Calculamos el checksum
        }
    }
    else
    {
        tempbyte = val[i];
    }
    bytesread++;
}
if (checkTwo(test, master)) // Comprobamos si coincide lo leído con el master
{
    return true;
}
else
{
    return false;
}
}

```

```
// Funcion:
boolean isWipe(byte test[])
{

}

// Funcion: LEDS MODO NORMAL
void normalModeOn()
{
    digitalWrite(powerPin, HIGH);
    digitalWrite(passPin, LOW);
    digitalWrite(failPin, LOW);
    digitalWrite(doorPin, LOW);
}

// Funcion: LEDS MODO PROGRAMA
void programModeOn()
{
    digitalWrite(powerPin, LOW);
    digitalWrite(passPin, HIGH);
    digitalWrite(failPin, LOW);
    delay(200);
    digitalWrite(powerPin, HIGH);
    digitalWrite(passPin, LOW);
    digitalWrite(failPin, LOW);
    delay(200);
}

// Funcion: LEDS MODO BORRADO
void deleteModeOn()
{
    digitalWrite(powerPin, LOW);
    digitalWrite(passPin, LOW);
    digitalWrite(failPin, HIGH);
    delay(200);
    digitalWrite(powerPin, HIGH);
    digitalWrite(passPin, LOW);
    digitalWrite(failPin, LOW);
    delay(200);
}

// Funcion: LEDS MODO WIPE
void wipeModeOn()
{
```

```
}
```

```
// Funcion: FLASH DE 3 VECES DEL LED VERDE INDICA QUE HA SIDO  
BIEN ALMACENADO EN LA EEPROM
```

```
void successWrite()
```

```
{
```

```
    digitalWrite(powerPin, LOW);
```

```
    digitalWrite(passPin, LOW);
```

```
    digitalWrite(failPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(passPin, HIGH);
```

```
    delay(200);
```

```
    digitalWrite(passPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(passPin, HIGH);
```

```
    delay(200);
```

```
    digitalWrite(passPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(passPin, HIGH);
```

```
    delay(200);
```

```
}
```

```
// Funcion: FLASH DE 3 VECES DEL LED ROJO INDICA QUE HA SIDO MAL  
ALMACENADO EN LA EEPROM
```

```
void failedWrite()
```

```
{
```

```
    digitalWrite(powerPin, LOW);
```

```
    digitalWrite(passPin, LOW);
```

```
    digitalWrite(failPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(failPin, HIGH);
```

```
    delay(200);
```

```
    digitalWrite(failPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(failPin, HIGH);
```

```
    delay(200);
```

```
    digitalWrite(failPin, LOW);
```

```
    delay(200);
```

```
    digitalWrite(failPin, HIGH);
```

```
    delay(200);
```



```

}

// Funcion: FLASH DE 3 VECES DEL LED AZUL INDICA QUE HA SIDO
BORRADO DE LA EEPROM
void successDelete()
{
    digitalWrite(powerPin, LOW);
    digitalWrite(passPin, LOW);
    digitalWrite(failPin, LOW);
    delay(200);
    digitalWrite(powerPin, HIGH);
    delay(200);
    digitalWrite(powerPin, LOW);
    delay(200);
    digitalWrite(powerPin, HIGH);
    delay(200);
    digitalWrite(powerPin, LOW);
    delay(200);
    digitalWrite(powerPin, HIGH);
    delay(200);
}

```

10.- Lista de Material.

La siguiente tabla muestra todos los componentes utilizados para la realización del prototipo y el posterior diseño final.

COMPONENTE	VALOR	CANTIDAD
Resistencia	220 Ω	4
Resistencia	1 K Ω	1
Resistencia	4,7 K Ω	1
Condensador	22 pF	2
Condensador	10 μ F	2
Zócalo	28 patillas	1
Diodo	1N4007	1
Led	Rojo	1
Led	RGB	1
Transistor NPN	BC547	1
Rele	8 A / 220 V	1
Pulsador		1
Lector RFID		1
Atmega 328P		1

Cristal	16.000	1
Pines	6	1
Regleta	2 entradas	2
Regleta	3 entradas	1
Placa baquelita		1
	Total componentes	25

11.- Presupuesto.

A continuación se muestra una relación de precios por separado de cada componente, la suma por grupo y por ultimo el total del costo de materiales.

Resistencias:		
- 4 uds, 220 Ohms, ¼ W, +/- 5% tolerancia: 0,19 €		
- 1 uds, 1 KOhms, ¼ W, +/- 5% tolerancia: 0,19 €		
	Subtotal:	3,8 €
Diodos LED:		
- 1 uds, Led rojo, 5mm: 0,26 €		
- 1 uds, Led RGB, 5mm, catodo común: 1,47 €		
	Subtotal:	1,73 €
	Subtotal:	3,8 €
Diodos:		
- 1 uds, Diodo rectificador 1N4007: 0,11 €		
	Subtotal:	0,11 €
Condensadores:		
- 2 uds, condensador cerámico 22 pF, 200V: 0,19 €		
- 2 uds, condensador electrolítico 10 µF, 25V: 0,34 €		
	Subtotal:	1,06 €
Zócalos:		
- 1 uds, doble contacto, 28 pines, paso 2,54: 0,48€		
	Subtotal:	0,48 €
Transistores:		
- 1 uds, BC547, encapsulado To92: 0,10 €		
	Subtotal:	0,10 €
Rele:		
- 1 uds, 2 contactos 8 A, paso 5 mm, marca FINDER: 5,07 €		
	Subtotal:	5,07 €
Pulsadores:		
- 1 uds, Omron B3F: 0,26 €		

	Subtotal:	0,26 €
Regletas: - 2 uds, regleta 2 pines, paso 5 mm: 0,83 € - 1 uds, regleta 3 pines, paso 5 mm: 1,25 €		
	Subtotal:	2,91 €
Baquelita: - 1 uds, placa baquelita 1 cara virgen, 100x240mm: 4,62 €		
	Subtotal:	4,62 €
Lector RFID: - 1 uds, lector rfid(125 KHz): 22,57 €		
	Subtotal:	22,57 €
Atmega 328: - 1 uds, atmega328P, 28 pines: 3,24 €		
	Subtotal:	6,24 €
Cristal: - 1 uds, 16.000 MHz, perfil bajo/alto: 0,94 €		
	Subtotal:	0,94 €
Pines: - 6 uds, tira poste, paso 2,54: 0,07 €		
	Subtotal:	0,07 €
	TOTAL:	50,76 €