TRABAJO INFORMÁTICA

ALARMA DE INCENDIOS

CURSO 2018/19

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Sergio Pérez Cubedo 54155

Brais Rodríguez Marigil 54186

FECHA: 16/05/2019

GRUPO: A109

PROFESOR: Joaquín González Gigosos.

ALARMA ARDUINO

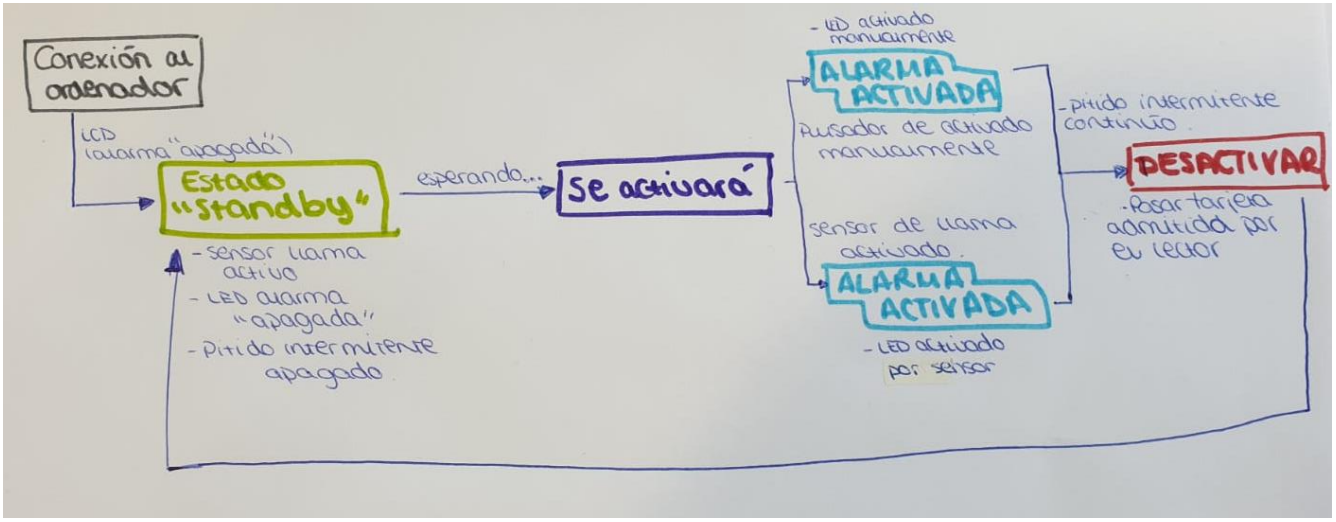
DESCRIPCIÓN DEL PROYECTO

Hemos realizado una alarma de incendios que nos permitirá detectar cuando el sitio en el que esté instalada este ardiendo o si alguna persona detecta el fuego antes que el sensor, la active manualmente mediante un pulsador.

Una vez activada la alarma empezará a sonar un pitido (zumbador) de manera intermitente, a su vez en la pantalla LCD pondrá si la alarma ha sido activada por el sensor de llama o manualmente por el pulsador.

Una vez solucionado el problema con el incendio, ya sea falsa alarma o sea verdad, para su desactivación se necesita una tarjeta autorizada que mediante un sensor desactivará el sonido del zumbador y volverá a estar la alarma esperando a que se vuelva a activar la alarma de las dos formas que hemos comentado anteriormente.

ESQUEMA FUNCIONAL DEL SISTEMA



HARDWARE Y FUNDAMENTOS TÉCNICOS

PLACA ARDUINO MEGA

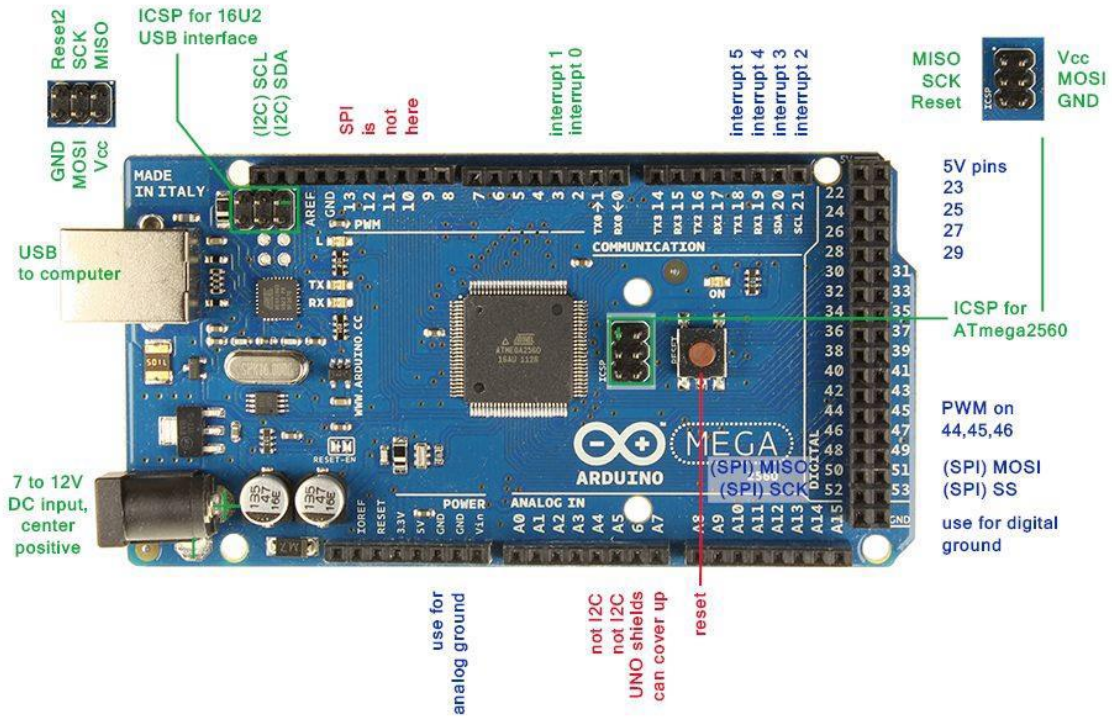
El Arduino Mega es un microcontrolador. Posee 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un boton de reset y una entrada para la alimentación de la placa.

La comunicación entre la computadora y Arduino se produce a través del Puerto Serie. Posee un convertidor usb-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB como el que utilizan las impresoras.

Arduino Mega posee las siguientes especificaciones:

- • **Microcontrolador:** ATmega2560

- • Voltaje Operativo: 5V
- • Voltaje de Entrada: 7-12V
- • **Voltaje de Entrada(límites)**: 6-20V
- • **Pines digitales de Entrada/Salida:** 54 (de los cuales 15 proveen salida PWM)
- • Pines análogos de entrada: 16
- • **Corriente DC por cada Pin Entrada/Salida:** 40 mA
- • **Corriente DC entregada en el Pin 3.3V:** 50 mA
- • **Memoria Flash:** 256 KB (8KB usados por el bootloader)
- • **SRAM:** 8KB
- • **EEPROM:** 4KB
- • **Clock Speed:** 16 MHz



PANTALLA LCD (16X2)

Es un dispositivo que nos permite escribir en la pantalla lo que nosotros pongamos en el código, utiliza las propiedades de la luz polarizada para mostrarnos la información en una pantalla. A partir de una serie de filtros, se consigue mostrar la información gracias a la iluminación de fondo.

ZUMBADOR (REPRODUCTOR DE SONIDOS)

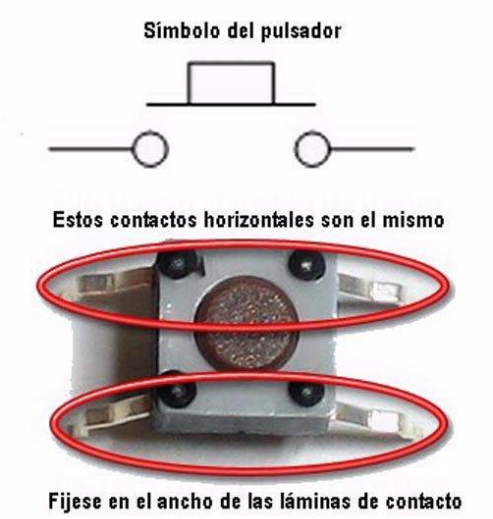
Un buzzer pasivo o un altavoz son dispositivos que permiten convertir una señal eléctrica en una onda de sonido. Estos dispositivos no disponen de electrónica interna, por lo que tenemos que proporcionar una señal eléctrica para conseguir el sonido deseado.

Técnicamente tanto buzzers como altavoces son transductores electroacústicos, es decir, dispositivos que convierten señales eléctricas en sonido. La diferencia entre ambos es el fenómeno en el que basan su funcionamiento.

El zumbador lo hemos utilizado para simular el pitido constante e intermitente de una alarma de incendios cuando es activada.

PULSADOR

Se trata de un mecanismo simple, constituido por un par de contactos eléctricos que se unen o separan por medios mecánicos. El pulsador lo hemos utilizado para el activado manual de la alarma.



LED

Los LED son dispositivos eléctricos que emiten luz cuando nosotros queremos, es decir emiten luz cuando nosotros hacemos que pase corriente a través de el LED.

Nosotros los hemos utilizado para indicar si la alarma está desactivada, si ha sido activada por el sensor o si ha sido activada por el sensor de llama.

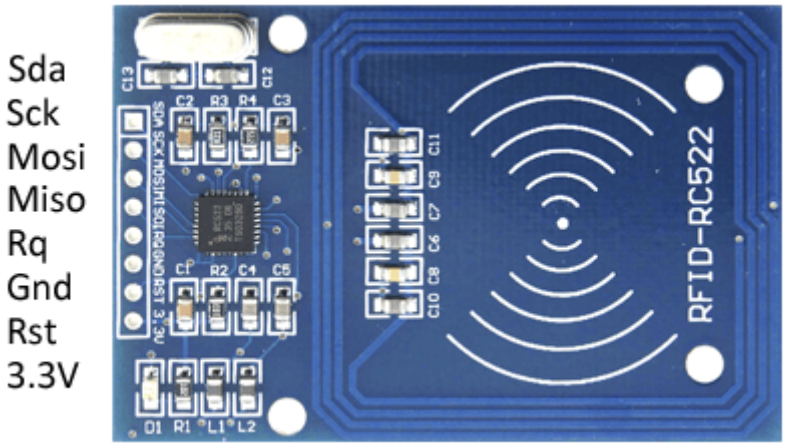
LECTOR DE TARJETAS

El RFID (Identificador por radiofrecuencia) es un conjunto de tecnologías diseñadas para leer etiquetas (tags) a distancia de forma inalámbrica. Los lectores RFID pueden ser conectados a un autómata o procesador como Arduino.

Los RFID son ampliamente empleados, por ejemplo, en sistemas de alarma, aplicaciones comerciales en sustitución de códigos de barras, cerraduras electrónicas, sistemas de pago, tarjetas personales, control de accesos recintos como gimnasios o piscinas, fichaje en empresas, entre otras muchas aplicaciones.

En nuestro caso hemos utilizado el lector de tarjetas para la desactivación de la alarma de incendios. Únicamente hay una tarjeta autorizada. Para ello primero hemos tenido que

diseñar un programa para saber las claves de la tarjeta y después autorizar en el programa principal las claves de la tarjeta.



SENSOR DE LLAMA

Nosotros vamos a utilizar un sensor infrarrojo de llamas. Funcionan detectando una longitud de onda especifica (de unos 760 nm) que son características de las llamas, aunque son relativamente fáciles de engañar y pueden dar falsos positivos con ciertas luces.

CÓDIGO

CÓDIGO PARA SABER LAS CLAVES DE LA TARJETA NFC

Para poder saber las claves de la tarjeta nos hemos tenido que descargar la librería MFRC522.h ya que no venía con la aplicación Arduino. Una vez introducido este código en el Arduino, en la pantalla serial nos dirá las claves de la tarjeta para poder autorizarla posteriormente en el programa principal.

```
#include <SPI.h>

#include <MFRC522.h>


const int RST_PIN = 5;          // Pin 9 para el reset del RC522

const int SS_PIN = 53;          // Pin 10 para el SS (SDA) del RC522

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Crear instancia del MFRC522
```

```
void printArray(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}

void setup()
{
  Serial.begin(9600);     //Inicializa la velocidad de Serial
  SPI.begin();        //Función que inicializa SPI
  mfrc522.PCD_Init();    //Función  que inicializa RFID
}

void loop()
{
  // Detectar tarjeta
  if (mfrc522.PICC_IsNewCardPresent())
  {
    if (mfrc522.PICC_ReadCardSerial())
    {
      Serial.print(F("Card UID:"));
      printArray(mfrc522.uid.uidByte, mfrc522.uid.size);
      Serial.println();

      // Finalizar lectura actual
      mfrc522.PICC_HaltA();
    }
  }
  delay(250);
```

}

CÓDIGO PRINCIPAL

Con este código hemos realizado el programa principal una vez que hemos sabido las claves de la tarjeta que utilizaremos para desactivar la alarma

```
/*
 * LOW es BAJO, y en el caso de los le3d lo usamos para mandar a GND la corriente y que se encienda
 * HIGH es ALTO,
 */

#include <LiquidCrystal.h>
#include <SPI.h>
#include <MFRC522.h>


 // Ejemplo de clave valida


//configuracion al crear el objeto lcd
//1º parametro = RS
//2º parametro = E
//3º parametro = D4
//4º parametro = D5
//5º parametro = D6
//6º parametro = D7

LiquidCrystal lcd(23,25,27,29,31,33);
```

```
//VARIABLES GENERICAS
const int RST_PIN = 5;        // Pin 9 para el reset del RC522
const int SS_PIN = 53;   // Pin 10 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Crear instancia del MFRC522
byte validKey1[4] = { 0xE0, 0x4D, 0xD0, 0x57 };
bool alarmaStandby=true;
bool esperandoClave=false;
int pinZumbador=22;
int pinLedApagado=34;
int pinLedActivoSensor=30;
int pinLedActivoPulsador=38;
int pinPulsadorEncendido=42;
int pinLlama=0;



//Esta funcion emite pitidos tantas veces como deseemos y con una duracion
//determinada
void emitirPitidos(int numVeces, int duracion){
  for(int cont=0; cont<numVeces; cont++){
    digitalWrite(pinZumbador, HIGH);
    delay(duracion);
    digitalWrite(pinZumbador, LOW);
    delay(duracion);
  }
}
bool isEqualArray(byte* arrayA, byte* arrayB, int length)
{
  for (int index = 0; index < length; index++)
  {
    if (arrayA[index] != arrayB[index]) return false;
```

```
  }
  return true;
}


//Configuramos los pines
//Inicializamos los estados de los pines
//configuramos pantalla
void setup() {
  Serial.begin(9600);
  SPI.begin();        // Iniciar SPI
  mfrc522.PCD_Init(); // Iniciar MFRC522
  pinMode(pinLedApagado, OUTPUT);
  pinMode(pinLedActivoSensor, OUTPUT);
  pinMode(pinLedActivoPulsador, OUTPUT);
  //pinMode(pinLlama, INPUT);
  pinMode(pinPulsadorEncendido, INPUT);
//  pinMode(pinPulsadorApagado, INPUT);
  pinMode(pinZumbador, OUTPUT);
  pinMode(ENABLE,OUTPUT);
  pinMode(DIRA,OUTPUT);
  pinMode(DIRB,OUTPUT);
//  pinMode(pinMotor, OUTPUT);


  digitalWrite(pinLedApagado, HIGH);
  digitalWrite(pinLedActivoSensor, LOW);
  digitalWrite(pinLedActivoPulsador, LOW);
```

```
  lcd.begin(16,2); // Inicializamos la interfaz del LCD y especificamos las dimensiones de la pantalla


  emitirPitidos(3, 150);


  lcd.print("Alarma apagada"); // Imprime inicializando en el LCD
  lcd.setCursor(0,1);



}


void loop() {
  int valida1=0;
  int valida2=0;
  int i;
  pinLlama = analogRead(A0);
  Serial.println(pinLlama);
  //SI LA ALARMA ESTA "APAGADA"


  if(alarmaStandby){
    digitalWrite(pinLedApagado, HIGH);
    digitalWrite(pinLedActivoSensor, LOW);
    digitalWrite(pinLedActivoPulsador, LOW);
  }



  if(digitalRead(pinPulsadorEncendido)==HIGH)  //PARA ACTIVAR LA ALRAMA
  {
    while(valida1!=1)
    {
    lcd.clear();
```

```
  lcd.setCursor(0,0);
  lcd.print("Alarma activada");
  lcd.setCursor(0,1);
  lcd.print("manualmente");
  digitalWrite(ENABLE,HIGH); // enable on
  for (i=0;i<5;i++) {
  digitalWrite(DIRA,HIGH); //one way
  digitalWrite(DIRB,LOW);
  }

// Detectar tarjeta
 if (mfrc522.PICC_IsNewCardPresent())
 {
  //Seleccionamos una tarjeta
  if (mfrc522.PICC_ReadCardSerial())
  {
    // Comparar ID con las claves válidas
    if (isEqualArray(mfrc522.uid.uidByte, validKey1, 4))
      valida1=1;

    // Finalizar lectura actual
    mfrc522.PICC_HaltA();
  }
 }
 delay(250);


  digitalWrite(pinLedActivoSensor, HIGH);
  digitalWrite(pinLedApagado, LOW);
  digitalWrite(pinLedActivoPulsador, LOW);
  emitirPitidos(1,200);
```

```
  }
     lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alarma apagada");
  lcd.setCursor(0,1);
  lcd.print(" ");
  digitalWrite(ENABLE,LOW);
  digitalWrite(pinLedActivoSensor, LOW);
  digitalWrite(pinLedApagado, HIGH);
  digitalWrite(pinLedActivoPulsador, LOW);

}
  if(pinLlama < 500) //PARA ACTIVAR ALARMA
{
  while(valida2!=1){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alarma activada");
  lcd.setCursor(0,1);
  lcd.print("por sensor");
     digitalWrite(ENABLE,HIGH); // enable on
  for (i=0;i<5;i++) {
  digitalWrite(DIRA,HIGH); //one way
  digitalWrite(DIRB,LOW);
  }
   // Detectar tarjeta
if (mfrc522.PICC_IsNewCardPresent())
{
  //Seleccionamos una tarjeta
  if (mfrc522.PICC_ReadCardSerial())
  {
```

```
    // Comparar ID con las claves válidas
    if (isEqualArray(mfrc522.uid.uidByte, validKey1, 4))
      valida2=1;

    // Finalizar lectura actual
    mfrc522.PICC_HaltA();
  }
}
delay(250);
  digitalWrite(pinLedActivoSensor, LOW);
  digitalWrite(pinLedApagado, LOW);
  digitalWrite(pinLedActivoPulsador, HIGH);
  emitirPitidos(1,200);

  }
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alarma apagada");
  lcd.setCursor(0,1);
  lcd.print(" ");
    digitalWrite(ENABLE,LOW);
  digitalWrite(pinLedActivoSensor, LOW);
  digitalWrite(pinLedApagado, HIGH);
  digitalWrite(pinLedActivoPulsador, LOW);

 }
 }
```

PROBLEMAS ENCONTRADOS A LA HORA DE REALIZAR EL TRABAJO

- No hemos podido introducir el motor o bomba de agua por falta de corriente ya que si lo metíamos en el Arduino por ejemplo la pantalla a penas lucía o los LED se iluminaban muy poco.

- Nos costó mucho realizar la desactivación de la Alarma por el método de la tarjeta NFC ya que no es un código fácil de entender o programar.
-