## CÓDIGO FUENTE C_ARDUINO

```
#include <EEPROM.h> // Libreria EEPROM incluida.

#include <LiquidCrystal.h> // Libreria LiquidCrystal incluida.
LiquidCrystal lcd(5, 4, 3, 2, A5, A4); // Asignación de pines de arduino para LiquidCrystal.

#include <Keypad.h> // Libreria Keypad incluida
char* secretCode = "1234"; // Variable que contiene el número PIN.
char* secretCode2 = "12345678"; // Variable que contiene el número PUK.
int position = 0; // Posición de inicio de la memoria EEPROM donde están guardados los números secretos.
const byte rows = 4; // Número de filas del teclado.
const byte cols = 3; // Número de columnas del teclado.
char keys[rows][cols] = { // Array de filas y columnas del teclado.
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[rows] = {12, 11, 10, 9}; // Asignación de pines de arduino para las filas del teclado.
byte colPins[cols] = {8, 7, 6}; // Asignación de pines de arduino para las columnas del teclado.
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, rows, cols);

byte switchs = A0; // Pin de entrada para interruptores de puerta.
byte pir = A3; // Pin de entrada para sensor de presencia.
byte leds = A1; // Pin de salida para los leds de estado de la alarma.
byte sirena = A2; // Pin de salida para el relé que activa la sirena.
byte altavoz = 13; // Pin de salida para el altavoz.
byte estado = 0; // Variable donde se almacena el estado de la alarma.
byte cntPul = 0; // Contador de pulsaciones de botón del teclado.
byte cntInt = 0; // Contador de intentos para el código PIN y PUK.
char recibirDato; // Variable para almacenar el dato recibido por el puerto serie.
long tiempo; // Variable para guardar tiempo.
char key;  // Variable de lectura de teclado.
boolean claveBien = false; // Variable para verificar si la clave es correcta.
boolean claveMal = false; // Variable para verificar si la clave es errónea.
```

```
void setup()
{
  Serial.begin(19200); // Se establece la velocidad de transmisión en 19200
                          Bauds.
  lcd.begin(16, 2); // Se inicializa la pantalla del LCD con 16 columnas y 2 filas.
  pinMode(pir, INPUT);  // pir es una entrada.
  pinMode(switchs, INPUT); // switchs es una entrada.
  pinMode(leds, OUTPUT); // leds es una salida.
  pinMode(sirena, OUTPUT); // sirena es una salida.
  pinMode(altavoz, OUTPUT); // altavoz es una salida.
  Serial.print("0"); // Inicializo imprimiendo "0" en el puerto serie.
}


void loop()
{

/*
//// En el siguiente estado (estado '0') la alarma está apagada y necesita una
clave para poderse activar ////

     El programa  comienza su ejecución en este estado y realizará de forma
secuencial las siguientes acciones:
     - Comprobar si hay un dato en el puerto serie (a través de la subrutina
       puertoSerie()).
     - Poner a nivel bajo las salidas leds, sirena y altavoz.
     - Indicar en la pantalla LCD qué la alarma está inactiva.
     - Comprobar si se ha marcado algún número (a través de la subrutina
       clavePin()).
     - Verificar si el número marcado es el correcto (claveBien o claveMal).

     Después de la verificación del número marcado, el programa irá al
estado que le corresponda  y notificará el cambio a la aplicación de c#
enviando un dato a través del puerto serie.
*/
 while(estado == 0)
 {
    puertoSerie();
    digitalWrite(leds, LOW);
    digitalWrite(sirena, LOW);
    digitalWrite(altavoz, LOW);
    lcd.setCursor(0, 0);
    lcd.print("Alarm OFF, press");
    lcd.setCursor(0, 1);
```

```
    lcd.print("key to activate ");
    clavePin();
    if(claveBien)
    {
      estado = 1;
      Serial.print("1");
      claveBien = false;
    }
    else if(claveMal)
    {
      estado = 5;
      Serial.print("5");
      claveMal = false;
    }
 }

  /*
//// En el siguiente estado (estado '1'), la alarma acaba de activarse, pero no estará operativa hasta que pasen 20 segundos ////

      En este estado, el programa realizará de forma secuencial las siguientes acciones:
        - Poner a nivel alto el indicador de alarma activa (led verde).
        - Indicar en la pantalla LCD qué la alarma se conectará pasados 20 segundos.
        - El programa espera 20 segundos.
        - Posteriormente, el programa pasa a estado '2'.
        - Notifica el cambio a través del puerto serie.
*/
 while(estado == 1)
 {
    digitalWrite(leds, HIGH);
    digitalWrite(sirena, LOW);
    digitalWrite(altavoz, LOW);
    lcd.setCursor(0, 0);
    lcd.print("Alarm ON, 20 sc ");
    lcd.setCursor(0, 1);
    lcd.print("to activate     ");
    delay(20000);
    estado = 2;
    Serial.print("2");
 }

/*
```

//// En el siguiente estado (estado '2') la alarma está operativa  e inspecciona si un intruso ha entrado en casa ////

En este estado, el programa realizará de forma secuencial las siguientes acciones:

- Comprobar si hay un dato en el puerto serie (a través de la subrutina puertoSerie()).
- Poner a nivel alto solo el indicador de alarma activa (led verde).
- Indicar en la pantalla LCD qué la alarma está operativa.
- Verificar si el sensor pir o los interruptores de puerta han detectado algún intruso.

En el caso de detectar a algún intruso, el programa irá al estado que le corresponda y notificará el cambio a la aplicación de c# enviando un dato a través del puerto serie.

*/

```
 while(estado == 2)
 {
    puertoSerie();
    digitalWrite(leds, HIGH);
    digitalWrite(sirena, LOW);
    digitalWrite(altavoz, LOW);
    lcd.setCursor(0, 0);
    lcd.print("ON, press key   ");
    lcd.setCursor(0, 1);
    lcd.print("to desactivate  ");
    if(digitalRead (switchs) == HIGH || digitalRead (pir) == HIGH)
    {
      estado = 3;
      Serial.print("3");
      delay(1000);
    }
 }
```

/*

//// En el siguiente estado (estado '3') la alarma acaba de detectar un intruso y pondrá en funcionamiento la sirena si en 20 segundos no se introduce el código PIN ////

En este estado y durante 20 segundos, el programa realizará de forma secuencial las siguientes acciones:

- Comprobar si hay un dato en el puerto serie (a través de la subrutina puertoSerie()).
- Poner a nivel alto solo el indicador de alarma activa (led verde).

- Indicar en la pantalla LCD qué la alarma debe desactivarse en menos de 20 segundos.
- Comprobar si se ha marcado algún número (a través de la subrutina clavePin()).
- Verificar si el número marcado es el correcto (claveBien o claveMal).

Después de la verificación del número marcado, el programa irá al estado que le corresponda y notificará el cambio a la aplicación de c# enviando un dato a través del puerto serie.
*/

```
 while(estado == 3)
 {
   tiempo = millis() + 20000;
   do
   {
     digitalWrite(leds, HIGH);
     digitalWrite(sirena, LOW);
     digitalWrite(altavoz, LOW);
     lcd.setCursor(0, 0);
     lcd.print("Desactivate in  ");
     lcd.setCursor(0, 1);
     lcd.print("20 seconds      ");
     clavePin();
     if(claveBien || claveMal || (Serial.available()>0))
     {
       break;
     }
   }while(tiempo > millis());

   puertoSerie();
   if(claveBien)
   {
      estado = 0;
      Serial.print("0");
      claveBien = false;
   }
   else if(claveMal)
   {
      estado = 6;
      Serial.print("6");
      claveMal = false;
   }
   else if (estado!= 0)
   {
       estado = 4;
```

```
      Serial.print("4");
  }
}

/*
//// En el siguiente estado (estado '4') la sirena de la alarma está en funcionamiento y se necesita el código PIN para detener la alarma ////

      En este estado, el programa realizará de forma secuencial las siguientes acciones:
        - Comprobar si hay un dato en el puerto serie (a través de la subrutina puertoSerie()).
        - Poner a nivel alto el indicador de alarma activa (led verde).
        - Poner a nivel alto el relé que activa la sirena.
        - Indicar en la pantalla LCD qué la alarma qué la alarma está en funcionamiento y necesita clave para desactivarse.
        - Comprobar si se ha marcado algún número (a través de la subrutina clavePin()).
        - Verificar si el número marcado es el correcto (claveBien o claveMal).

      Después de la verificación del número marcado, el programa irá al estado que le corresponda  y notificará el cambio a la aplicación de c# enviando un dato a través del puerto serie.
*/
 while(estado == 4)
 {
   puertoSerie();
   digitalWrite(leds, HIGH);
   digitalWrite(sirena, HIGH);
   digitalWrite(altavoz, LOW);
   lcd.setCursor(0, 0);
   lcd.print("ON, press key   ");
   lcd.setCursor(0, 1);
   lcd.print("to desactivate  ");
   clavePin();
   if(claveBien)
   {
     estado = 0;
     Serial.print("0");
     claveBien = false;
   }
   else if(claveMal)
   {
     estado = 6;
```

```
    Serial.print("6");
    claveMal = false;
  }

}
```

/*
//// En el siguiente estado (estado '5') el código PIN está bloqueado y se necesita el código PUK para activar la alarma ////

En este estado, el programa realizará de forma secuencial las siguientes acciones:

- Comprobar si hay un dato en el puerto serie (a través de la subrutina puertoSerie()).
- Poner a nivel bajo las salidas leds, sirena y altavoz.
- Indicar en la pantalla LCD qué el código PIN está bloqueado y se necesita introducir el código PUK para activar la alarma.
- Comprobar si se ha marcado algún número (a través de la subrutina clavePuk()).
- Verificar si el número marcado es el correcto (claveBien).

Después de la verificación del número marcado, el programa irá al estado que le corresponda y notificará el cambio a la aplicación de c# enviando un dato a través del puerto serie.
*/

```
 while(estado == 5)
 {
   puertoSerie();
   digitalWrite(leds, LOW);
   digitalWrite(sirena, LOW);
   digitalWrite(altavoz, LOW);
   lcd.setCursor(0, 0);
   lcd.print("PIN locked      ");
   lcd.setCursor(0, 1);
   lcd.print("Enter PUK code  ");
   clavePuk();
   if(claveBien)
   {
     estado = 1;
     Serial.print("1");
     claveBien = false;
   }
 }
```

/*
//// En el siguiente estado (estado '6') el código PIN está bloqueado y se necesita el código PUK para desactivar la alarma ////

En este estado, el programa realizará de forma secuencial las siguientes acciones:

- Comprobar si hay un dato en el puerto serie (a través de la subrutina puertoSerie()).
- Poner a nivel alto el indicador de alarma activa (led verde).
- Poner a nivel alto el relé que activa la sirena.
- Indicar en la pantalla LCD qué el código PIN está bloqueado y se necesita introducir el código PUK para desactivar la alarma.
- Comprobar si se ha marcado algún número (a través de la subrutina clavePuk()).
- Verificar si el número marcado es el correcto (claveBien).

Después de la verificación del número marcado, el programa irá al estado que le corresponda y notificará el cambio a la aplicación de c# enviando un dato a través del puerto serie.
*/

```
 while(estado == 6)
 {
   puertoSerie();
   digitalWrite(leds, HIGH);
   digitalWrite(sirena, HIGH);
   digitalWrite(altavoz, LOW);
   lcd.setCursor(0, 0);
   lcd.print("ON, enter PUK   ");
   lcd.setCursor(0, 1);
   lcd.print("to desactivate  ");
   clavePuk();
   if(claveBien)
   {
     estado = 0;
     Serial.print("0");
     claveBien = false;
   }
  }
}
```

/*
//// Subrutina puertoSerie() ////

En esta subrutina el programa realizará de forma secuencial las siguientes acciones:

- Comprobar si el puerto serie está disponible.
- En el caso de que esté disponible, verificar que dato se ha recibido.
- Si el dato recibido es ‘1’ el programa irá al estado 1.
- En cambio, si es ‘0’ el programa irá al estado 0.
- En cualquier caso, se notificará del cambio efectuado a la aplicación c# vía puerto serie.

*/

```
void puertoSerie()
{
 if(Serial.available()>0)
 {
   recibirDato = Serial.read();
   if(recibirDato == '1')
   {
     estado = 1;
     Serial.print("1");
     digitalWrite(altavoz, HIGH);
     delay(500);
     digitalWrite(altavoz, LOW);
     delay(500);
   }
   else if (recibirDato == '0')
   {
     estado = 0;
     Serial.print("0");
     digitalWrite(altavoz, HIGH);
     delay(500);
     digitalWrite(altavoz, LOW);
     delay(500);
   }
 }
}

/*
```

//// Subrutina clavePin() ////

En esta subrutina el programa realizará de forma secuencial las siguientes acciones:

- Detectar qué teclas han sido pulsadas.
- Comprobar si el número pulsado coincide con el número secreto de la clave:

Si esto es así, la variable booleana claveBien será verdadera, el LCD pondrá ‘clave correcta’ y el altavoz emitirá un bip corto.

- Si por el contrario, el número no coincide con la clave secreta y se ha intentado introducirlo más de tres veces:
  La variable booleana claveMal será verdadera, el LCD pondrá ‘Pin bloqueado’ y el altavoz emitirá un bip largo.
- Las teclas ‘*’ y ‘#’ se usarán para borrar las teclas presionadas y poder pulsar el código de nuevo.

*/

```
void clavePin()
{
  key = keypad.getKey();
  if (key == '*' || key == '#')
  {
    position = 0;
    cntPul = 0;
    delay(100);
  }
  if (key == secretCode[position])
  {
    position ++;
    delay(100);
  }
  if (key >= '0')
  {
    cntPul ++;
    delay(100);
  }
  if (cntPul == 4)
  {
    if (position == 4)
    {
      claveBien = true;
      cntPul = 0;
      cntInt = 0;
      position = 0;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Correct key      ");
      digitalWrite(altavoz, HIGH);
      delay(500);
      digitalWrite(altavoz, LOW);
      delay(500);
    }
```

```
    else
    {
      cntPul = 0;
      cntInt ++;
      position = 0;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Wrong key      ");
      digitalWrite(altavoz, HIGH);
      delay(2000);
    }
  }
  if (cntInt == 3)
  {
    cntInt = 0;
    claveMal = true;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("PIN locked       ");
    lcd.setCursor(0, 1);
    lcd.print("Enter PUK code   ");
    digitalWrite(altavoz, HIGH);
    delay(2000);
  }
}

/*
//// Subrutina clavePuk() ////

      En esta subrutina el programa realizará de forma secuencial las siguientes acciones:
        - Detectar qué teclas han sido pulsadas.
        - Comprobar si el número pulsado coincide con el número secreto de la clave:
            Si esto es así, la variable booleana  claveBien será verdadera, el LCD pondrá 'clave correcta' y el altavoz emitirá un bip corto.
        -  Si por el contrario, el  número no coincide con la clave secreta:
            El LCD pondrá 'Clave errónea' y el altavoz emitirá un bip largo.
        - Las teclas  '*'  y  '#' se usarán para borrar las teclas presionadas y poder pulsar el código de nuevo.
*/
void clavePuk()
{
  key = keypad.getKey();
```

```
 if (key == '*' || key == '#')
 {
   position = 0;
   cntPul = 0;
   delay(100);
 }
 if (key == secretCode2[position])
 {
   position ++;
   delay(100);
 }
 if (key >= '0')
 {
   cntPul ++;
   delay(100);
 }
 if (cntPul == 8)
 {
   if (position == 8)
   {
     claveBien = true;
     cntPul = 0;
     position = 0;
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Correct key     ");
     digitalWrite(altavoz, HIGH);
     delay(500);
     digitalWrite(altavoz, LOW);
     delay(500);
   }
   else
   {
     cntPul = 0;
     position = 0;
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Wrong key       ");
     digitalWrite(altavoz, HIGH);
     delay(2000);
    }
  }
}
```