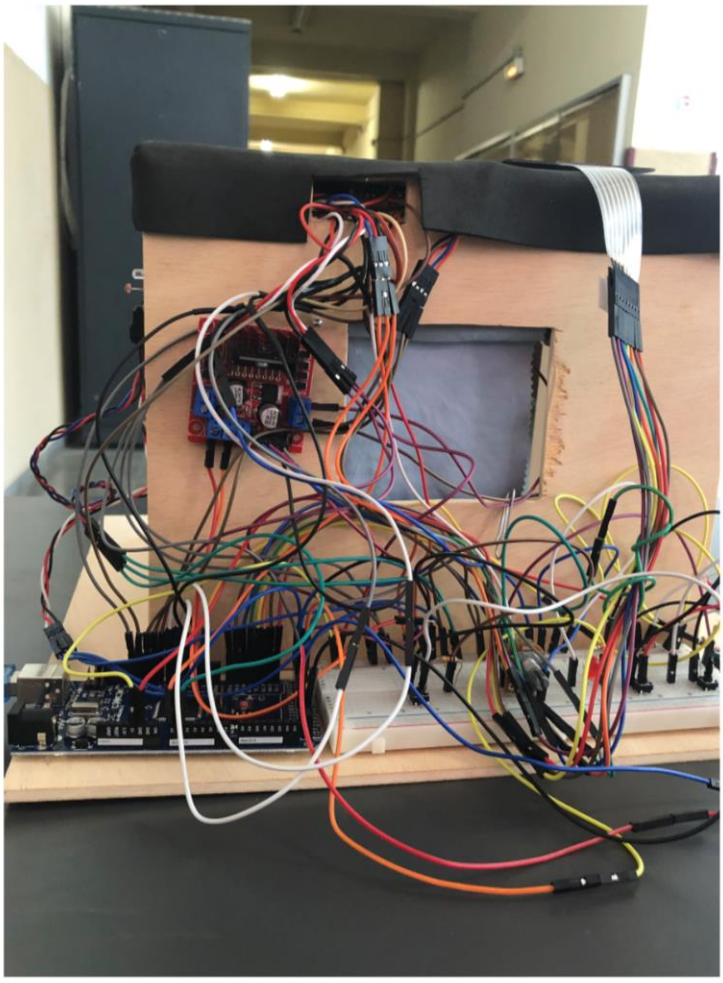


Trabajo informática

CURSO 18/19-GRUPO-A104

ETSIDI | UPM | 09 de mayo de 2019

Integrantes del grupo

- Irene Álvarez Pérez 54477
- Raquel García Franco 54615

Resumen del trabajo:

- **Domotización casa**

Automatizar los elementos básicos de una casa mediante sensores.

-Elevación de una persiana mediante pulsador y un motor.

-Timbre accionado mediante un pulsador.

-Luz exterior accionada mediante un sensor ldr.

-Luz interior accionada mediante pulsador.

-Termostato programable mediante el ordenador. Simulación de la temperatura mediante un potenciómetro y simulación de la calefacción mediante un led.

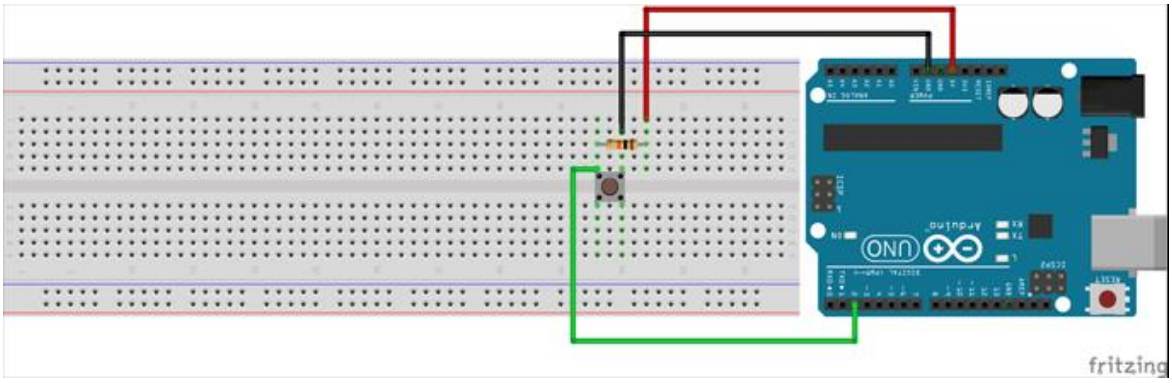
-Sistema de alarma accionado mediante un sensor de ultrasonidos y un zumbador. Controlado mediante un lcd y un teclado externo.

Hardware-Fundamentos técnicos

Pulsador PULLUP

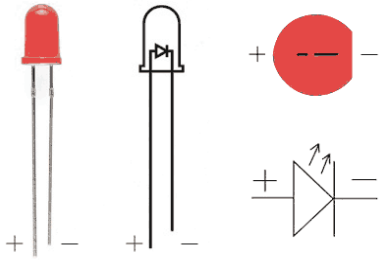
En la configuración pull up, cuando el circuito está en reposo, P1 sin pulsar, la caída de tensión es de 5V (HIGH), en cambio cuando pulsamos P1 se deriva toda la corriente a masa y la caída de tensión es 0V (LOW).

Normalmente las resistencias que se utilizan en estos casos son de 10K. Como hemos comprobado, estas dos configuraciones nos evitarán que en estado de reposo midamos un valor erróneo eliminando la influencia de factores externos sobre nuestras mediciones como el ruido eléctrico.



Esquema de conexión pulsador PULLUP

Led



Es un diodo que emite luz al ser atravesado por una corriente eléctrica.

El diodo está constituido por la unión de dos materiales semiconductores con dopados distintos, generando una barrera de potencial y haciendo que el paso de corriente solo sea posible en una dirección (tienen polaridad).

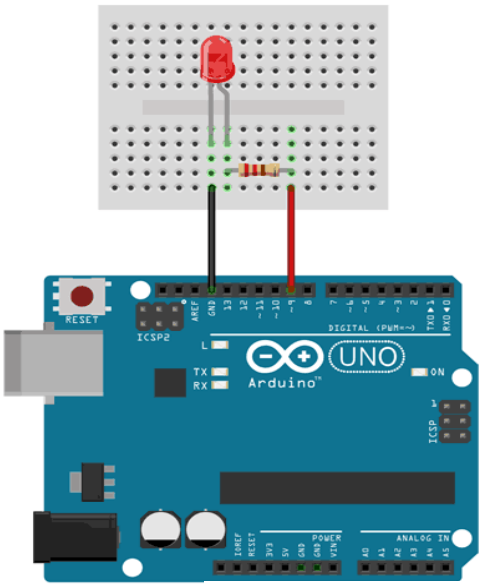
La corriente pasará por el diodo una vez que se alcance un valor de tensión llamado tensión de polarización directa (Vd). Para no romper el diodo se necesita una resistencia que limite la cantidad de corriente que circula.

```
#define led 9 //Asignamos el led al pin 9

void setup()
{
  Serial.begin(9600); //Iniciamos el puerto serie
  pinMode(led, OUTPUT); //Se define el pin como salida
}

void loop()
{
  digitalWrite(led, HIGH); //Se enciende el led
  delay(1000);             //Se espera un segundo
  digitalWrite(led, LOW);  //Se apaga el led
  delay(1000);             //Se espera un segundo
}
```

Ejemplo código sencillo led

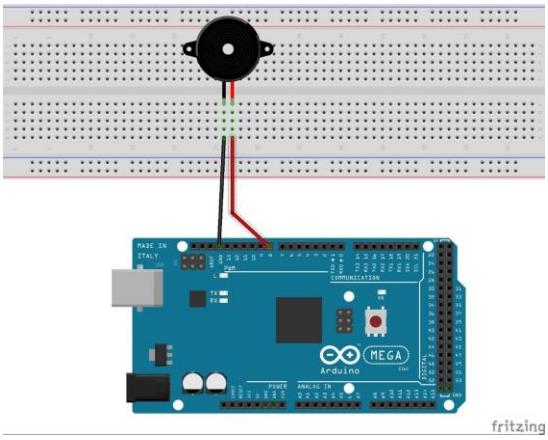


Esquema de conexión led

Zumbador

Un zumbador es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono (generalmente agudo).

Cuando se acciona, la corriente pasa por la bobina del electroimán y produce un campo magnético variable que hace vibrar la lámina de acero sobre la armadura, o bien, la corriente pasa por el disco piezoeléctrico haciéndolo entrar en resonancia eléctrica y produciendo ultrasonidos que son amplificados por la lámina.

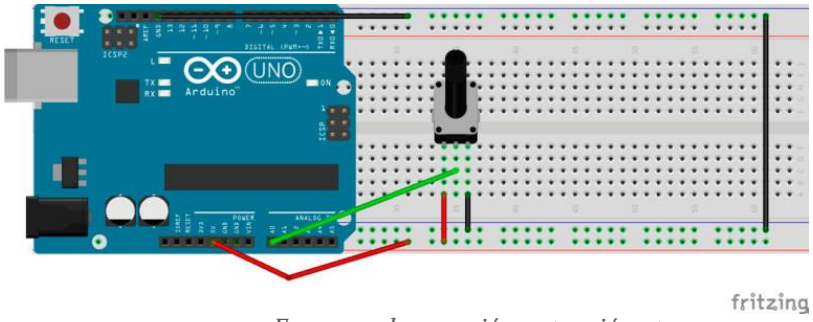


Esquema de conexión zumbador

Potenciómetro

Un potenciómetro es un dispositivo que permite variar su resistencia de forma manual, entre un valor mínimo Rmin, (normalmente ohmios) y un valor máximo Rmax. Valores habituales de Rmax son 5k, 10k o 20k ohmios.

El propio potenciómetro actúa como divisor de tensión.



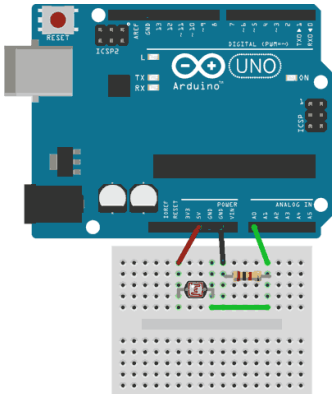
Esquema de conexión potenciómetro

LDR

Un fotorresistor, o LDR (light-dependent resistor) es un dispositivo cuya resistencia varia en función de la luz recibida. Podemos usar esta variación para medir, a través de las entradas analógicas, una estimación del nivel del luz.

Un fotorresistor está formado por un semiconductor, típicamente sulfuro de cadmio CdS. Al incidir la luz sobre él algunos de los fotones son absorbidos, provocando que electrones pasen a la banda de conducción y, por tanto, disminuyendo la resistencia del componente.

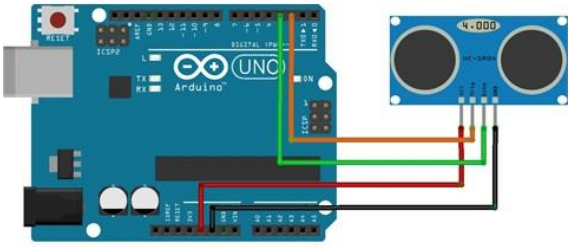
Por tanto, un fotorresistor disminuye su resistencia a medida que aumenta la luz sobre él. Los valores típicos son de 1 Mohm en total oscuridad, a 50-100 Ohm bajo luz brillante.



Esquema de conexión ldr

Ultrasonidos

Un sensor de ultra sonidos es un dispositivo para medir distancias. Su funcionamiento se base en el envío de un pulso de alta frecuencia, no audible por el ser humano. Este pulso rebota en los objetos cercanos y es reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia.

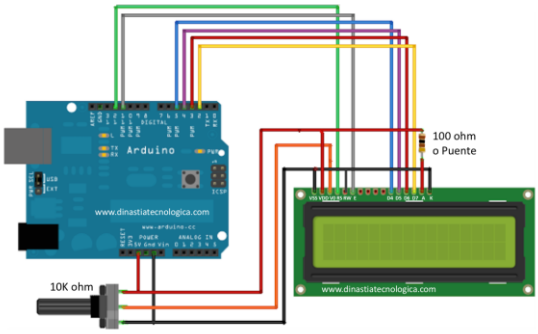


Esquema de conexión ultrasonidos

LCD

Una pantalla de cristal líquido o LCD (sigla del inglés Liquid Crystal Display) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente

de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.



Esquema de conexión lcd

Teclado matricial

Un teclado matricial es un dispositivo que agrupa varios pulsadores y permite controlarlos empleando un número de conductores inferior al que necesitaríamos al usarlos de forma individual. Podemos emplear estos teclados como un controlador para un autómata o un procesador como Arduino.

Estos dispositivos agrupan los pulsadores en filas y columnas formando una matriz, disposición que da lugar a su nombre. Es frecuente una disposición rectangular pura de NxM columnas, aunque otras disposiciones son igualmente posibles.

Esquema de conexión teclado matricial

Conexión Puerto serie – Arduino

En el código adjunto a continuación realizamos una conexión con el Arduino a través del puerto serie. Creamos un programa que nos permite controlar el led interno del Arduino como medida de comprobación del control de la placa por parte del puerto serie. También hemos creado un termostato con dos opciones. Una que nos permite leer el valor del potenciómetro (con el que simulamos la temperatura ambiente, aplicándole un factor de conversión) y mostrarlo por pantalla, y otra que nos permite configurar la temperatura a partir de la cual deseamos que se encienda la calefacción (simulada mediante un led).

```
/*
  Este ejemplo permite apagar, encender o poner a parpadear el led interno del Arduino.

  Solicita por pantalla al usuario que índice la acción que desea realizar (apagar,
encender o
  parpadear) y la envía por el puerto serie al Arduino.

  El programa necesita que se le pase como parámetro el puerto serie a utilizar.
  Por ejemplo:
       serie COM3

*/
```

```
#include <Windows.h>
#include <stdio.h>
#include <conio.h>
#include <locale.h>

/*

*/


int main(int argc, char* argv[])
{
	HANDLE hdPort;				// Manejador del puerto serie
	DCB paramsPort;					// Parámetros del puerto serie
	COMMTIMEOUTS tm;			// Tiempos de espera para el puerto serie
	char action;				// Acción a enviar al Arduino
	DWORD numBytes;					// Numero de bytes enviados al Arduino o
recibos del Arduino
	char serialName[20];  // Nombre del puerto serie
	char temperature[20]; // Temperatura enviada o recibida de Arduino. Se utiliza
una cadena de texto porque

	// hace más simple la programación aunque no sea lo más eficiente. Como máximo la
cadena puede tener 19 caracteres.

	// La cadena se reserva de 20 para poder contener los 19 caracteres máximos más

	// el byte de fin de cadena. Ese byte no se envía o recibe del Arduino.
	float faux;				// Variable flotante auxiliar. Se utiliza
sólo para comprobar que la cadena "temperature" tiene un
							// formato válido
	int res;				// Resultado de algunas funciones


	// Se establece que el idioma es el español. Esto es sólo para que por la
	// consola de Windows se representen bien los caracteres especiales del
	// español (tildes, ñ, etc.)
	setlocale(LC_ALL, "spanish");

	// Se comprueba que se haya llamado al programa con un parámetro.
	// argc contiene el número de parámetros + 1, por eso se compara con 2

	if (argc != 2)
	{
		printf("Error: Número de parámetros incorrecto.\n");
		return -1;
	}

	// Los puertos serie en Windows se nombran como "COM1", "COM2", "COM3", etc.
	// Pero solo los nueve primeros se pueden abrir usando directamente ese nombre.
	// A partir del 10, tiene que nombrarse como "\\.\COM10" "\\.\COM11" etc.
	// Por eso, al parámetro indicado por el usuario se le va a añadir la cadena
	// "\\." para completar el nombre.

	// Se almacena en serialName el nombre del puerto concatenando la cadena
	// inicial al nombre del puerto indicado por el usuario como parámetro.
	// Al utilizar sprintf_s se asegura que aunque el usuario introduzca un nombre
	// de puerto muy largo nunca se van a copiar más de 19 bytes y por tanto
	// no va desbordar el serialName.

	sprintf_s(serialName, 19, "\\\\.\\%s", argv[1]);
```

```
	// Se abre el puerto indicando que se va a utilizar para lectura y escritura
	hdPort = CreateFile(serialName, GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
	if (hdPort == INVALID_HANDLE_VALUE)
	{
		printf("Error: Imposible abrir el puerto serie\n");
		return 1;
	}

	// Una vez abierto hay que configurarlo para que su velocidad coincida con la que
se configura
	// en Arduino.

	// El puerto serie tiene bastantes parámetros de configuración. Para evitar tener
que indicar todos
	// lo más cómodo es leer la configuración que tiene actualmente, modificar sólo
lo que nos interesa
	// y volver a escribir la configuración

	// Se lee la configuración de los parámetros del puerto serie
	if (GetCommState(hdPort, &paramsPort) == 0)
	{
		printf("Error: Imposible configurar el puerto serie\n");
		return 2;
	}

	// Se modifican los parámetros necesarios para comunicar con Arduino

	paramsPort.BaudRate = CBR_9600;		// Velocidad: 9600 baudios
	paramsPort.Parity = NOPARITY;		// Sin paridad
	paramsPort.ByteSize = 8;		// Numero de bits por byte 8
	paramsPort.StopBits = ONESTOPBIT;	// 1 bit de stop

	// Se vuelve a escribir la configuración

	if (SetCommState(hdPort, &paramsPort) == 0)
	{
		printf("Error: Imposible configurar el puerto serie\n");
		return 2;
	}


	// Ahora también hay que configurar los tiempos de espera por la respuesta del
Arduino. Se va
	// a configurar un tiempo de 500 ms. Es decir, cuando se va a leer por el puerto
serie, se espera
	// como máximo 500 ms por si llega algo.

	// El proceso de configuración es similar al anterior. Primero se leen los
valores actuales de los
	// tiempos de espera, después se modifican los que hagan falta y por último se
vuelven a escribir.

	// Lectura de valores

	if (GetCommTimeouts(hdPort, &tm) == 0)
	{
		printf("Error: Imposible configurar el puerto serie\n");
		return 2;
	}
```

```
	// Se ajustan los valores
	tm.ReadIntervalTimeout = 0;
	tm.ReadTotalTimeoutMultiplier = 0;
	tm.ReadTotalTimeoutConstant = 500;

	// Se escriben los nuevos valores
	if (SetCommTimeouts(hdPort, &tm) == 0)
	{
		printf("Error: Imposible configurar el puerto serie\n");
		return 2;
	}

	// Bucle infinito de solicitud al usuario de la acción a realizar

	do {

		// Se le muestra al usuario las opciones que tiene
		printf("\nIndique la acción a realizar\n");
		printf("1 - Encender led\n");
		printf("2 - Apagar led\n");
		printf("3 - Forzar parpadeo led\n");
		printf("4 - Leer temperatura\n");
		printf("5 - Establecer temperatura deseada\n");
		printf("0 - Salir\n");
		printf("> ");

		// Se lee la acción indicada por el usuario.
		// _getch lee el caracter introducido en la consola pero no lo escribe,
por eso
		// después se hace el _putch. De esta manera el usuario si ve en pantalla
la tecla
		// pulsada

		action = _getch();
		_putch(action);

		// Según la acción...
		switch (action)
		{
		case '0':
			// Ha indicado salir. Se cierra el puerto y se finaliza
			CloseHandle(hdPort);
			return 0;

		case '1':
		case '2':
		case '3':
			// Para cualquiera de estas acciones se hace lo mismo, enviarla
por el puerto serie al
			// Arduino. Se escribe en el puerto, pasando como dirección del
buffer a escribir directamente
			// la dirección de la acción e indicando que sólo se va a escribir
1 byte.
			if (WriteFile(hdPort, &action, 1, &numBytes, NULL) == 0)
			{
				printf("\nError: Imposible enviar la acción.\n");
				return 4;
			}
			break;
```

```
		case '4':
			// Aquí también hay que enviar la acción al Arduino, pero después
hay que leer la respuesta
			// del Arduino con la temperatura
			if (WriteFile(hdPort, &action, 1, &numBytes, NULL) == 0)
			{
				printf("\nError: Imposible enviar la acción.\n");
				return 4;
			}

			// Se lee por el puerto la temperatura recibida. Se manda leer
como máximo 19
			// bytes, porque tiene que quedar al menos uno libre para poner el
final de la
			// cadena de texto
			if (ReadFile(hdPort, temperature, 19, &numBytes, NULL) == 0)
			{
				printf("\nError: Imposible leer la temperatura.\n");
				return 5;
			}

			// Se añade el final a la cadena de texto recibida. El final irá
en la posición
			// que corresponda a los bytes leídos. Por ejemplo, si se recibe
"12.35", el
			// valor de numBytes será 5 porque se han leído 5 caracteres y la
marca de final
			// de cadena habrá que ponerla en la posición de numBytes, porque
las cadenas en
			// C empiezan a contarse en 0. Es decir, quedará
			// temperature[0] = '1'
			// temperature[1] = '2'
			// temperature[2] = '.'
			// temperature[3] = '3'

			// temperature[4] = '5'
			// temperature[5] = '\0'
			temperature[numBytes] = '\0';

			// Se escribe el resultado en la pantalla
			printf("\nTemperatura: %s ºC\n", temperature);
			break;


		case '5':
			// Se le pide al usuario la temperatura deseada. Se mete en un
bucle porque
			// se va a pedir de forma continua hasta que el usuario
proporcione un valor
			// de coma flotante válido.
			res = 0;
			while (res != 1)
			{
				printf("\nIntroduzca la temperatura deseada en ºC: ");


				// Se recoge el valor en formato string
				res = scanf_s("%19s", temperature,
(unsigned)_countof(temperature));
				if (res == 1)
				{
```

```
                                        // Antes de aceptar el string indicado por el
usuario se comprueba que sea un
                                        // valor en como flotante válido. Para ello utilizo
la función sscanf_s.
                                        res = sscanf_s(temperature, "%f", &faux);
                                }
                                if (res != 1)
                                {
                                        printf("Error: valor de temperatura inválido.
Introduzca un valor de coma flotante válido\n");
                                }
                        }

                        // Se envía la acción al Arduino
                        if (WriteFile(hdPort, &action, 1, &numBytes, NULL) == 0)
                        {
                                printf("\nError: Imposible enviar la acción.\n");
                                return 4;
                        }

                        // Se envía el valor de temperatura deseado.
                        if (WriteFile(hdPort, temperature, strlen(temperature), &numBytes,
NULL) == 0)
                        {
                                printf("\nError: Imposible enviar la consigna de
temperatura.\n");
                                return 5;
                        }

                        break;
                case EOF:
                        printf("\nError: Fallo en la lectura desde consola\n");
                        return 3;

                default:
                        printf("\nError: '%c' no es una acción válida\n", action);
                }

        } while (TRUE);

        system("pause");
        return 0;

}
```

A continuación, adjunto el correspondiente código en Arduino.

```
const int action = 0;                   //Acción solicitada desde el PC
const int blinkLed = 0;                 //Indicador de que el led interno debe
parpadear
const int ledLevel = LOW;               //Valor para el led interno
const int PIN_CALDERA = 2;            // Pin donde estará conectado el led que indica
el arranque de la caldera por temperatura más baja de la deseada

const float SCALE_FACTOR = 0.05;    // Factor de escala para convertir el valor
leído del potenciómetro en grados centígrados.
float userTemperature = 0.0;         // Temperatura deseada por el usuario
```

```
void setup() {
	// Se abre el puerto serie con velocidad de 9600
	Serial.begin(9600);

	// Se espera de forma indefinida hasta que el puerto esté abierto
	while (!Serial);

	// Se configura el pin del led interno como salida
	pinMode(LED_BUILTIN, OUTPUT);

	// Se configura el pin 2 como salida.
	pinMode(PIN_CALDERA, OUTPUT);
}

void loop() {
	float temperature;    // Temperatura en grados obtenida del potenciómetro
	// Se realiza el tratamiento de la temperatura
	temperature = control_temperature(userTemperature);

	// Se lee del puerto serie
	// La función retorna en action el primer byte que haya en el puerto
	// pendiente de leer y si no hay ninguno -1
	action = Serial.read();

	switch (action)
	{
	case '1': {
		// Se indica que el led debe encenderse y no parpadear
		ledLevel = HIGH;
		blinkLed = 0;
		break;
	}

	case '2': {
		// Se indica que el led debe apagarse y no parpadear
		ledLevel = LOW;
		blinkLed = 0;
		break;
	}

	case '3': {
		// Se indica que el led debe parpadear
		blinkLed = 1;
		break;
	}

	case '4': {
		// Se envía el valor de temperatura
		Serial.print(temperature);
		break;
	}

	case '5': {
		// Se lee el valor deseado para la temperatura transformándolo en un
float
		userTemperature = Serial.parseFloat();

		break;
```

```
	}

                        // cualquier otra acción que se reciba, incluso que no se
reciba ninguna no altera el estado del led.
	}

	// Si hay que parpadear
	if (blinkLed)
	{
		// Se invierte la indicación para el led
		if (ledLevel == HIGH)
			ledLevel = LOW;
		else
			ledLevel = HIGH;

		// Se espera 500 ms
		delay(500);
	}

	// Se escribe el valor deseado en el led
	digitalWrite(LED_BUILTIN, ledLevel);

}


//-------------------------------------------------------------------------------
-------
// control_temperature
// Lee el valor del potenciómetro que simula el sensor de temperatura del pin A0.
// Lo convierte a grados centrígrados y compara ese valor con el valor deseado de
// temperatura indicado por el usuario. Si la temperatura es menor pone a 1 el pin
// donde esta conectado el arranque de la calefacción, sino lo pone a 0
//
// Necesita:
//  - userGradTemp: Valor de temperatura deseada por el usuario
//
// Retorna: el valor de temperatura leído del potenciómetro
//-------------------------------------------------------------------------------
-------
float control_temperature(float userGradTemp)
{
	int rawTemp = 0;	// Valor del potenciómetro
	float gradTemp = 0;	// Temperatura en grados


	// Se supone que el potenciómetro que simula el sensor de temperatura
	// está conectado al pin A0

	// Se lee el valor del potenciómetro
	rawTemp = analogRead(A0);

	// Se transforma el valor leído a grados usando el factor de escala
	// indicado
	gradTemp = rawTemp * SCALE_FACTOR;

	// Si la temperatura es menor que la deseada, se enciende la caldera
	if (gradTemp < userGradTemp)
		digitalWrite(PIN_CALDERA, HIGH);
```

```
        else
                digitalWrite(PIN_CALDERA, LOW);

        // Se devuelve el valor
        return gradTemp;
}
```

Código completo

En nuestro proyecto hemos tratado de crear una simulación de una casa domótica y autónoma. Para ello hemos programado desde funciones básicas, como encender una luz mediante un pulsador hasta una alarma. A continuación, detallamos cada elemento de nuestro proyecto:

Luz interior:

Para la luz interior hemos conectado un led y un pulsador a la placa de Arduino y hemos programado un sencillo código que lee el valor del pulsador y enciende el led si el pulsador está presionado.

Luz exterior:

Para la luz exterior de la casa hemos utilizado un sensor ldr y un led. En el código leemos el valor del sensor, y si este es bajo (detecta poca luz) se enciende el led. Por el contrario, si la luz incide sobre el ldr este se apaga. Consiguiendo así, que funciones como una farola automática.

Persiana:

Para la persiana hemos usado un motor conectado a un pulsador, que al rotar recoge la tela que simula la persiana. En el código hemos establecido la velocidad del motor y el sentido de giro del mismo. Además, mediante otro pulsador, y siguiendo el mismo procedimiento, el motor también desenrolla la tela simulando la bajada de la persiana.

Timbre:

Mediante un zumbador y un pulsador hemos creado un timbre. En el código se lee el valor del pulsador y si este está pulsado el zumbador emite sonido.

Alarma:

Este es la parte más extensa y complicada del proyecto. La alarma está compuesta de un lcd, un teclado matricial y un sensor de ultrasonidos. En el lcd se muestra un menú principal en el que nos da dos opciones: A-Activar la alarma y B-Cambiar la contraseña.

Si elegimos la opción A se inicia una cuenta atrás de 9 segundos hasta que se activa la alarma. Durante la cuenta atrás el sensor toma el valor de la distancia que tiene en ese instante, y cuando esta acaba, la compara con las distancias nuevas que va detectando. En el momento en que la nueva distancia sea menor que la detectada durante la cuenta atrás, salta la alarma y el zumbador comienza a emitir sonido. Para pararla debemos introducir la contraseña. Para ello en el código se llama a la función enterPassword que luego explicaremos con más detalle.

Si, en cambio, elegimos la opción B se nos pedirá la actual contraseña. Si presionamos '*' esta se compara con la declarada inicialmente, y si es correcta se pasa a la siguiente pantalla. Si por el

contrario no lo es, se introducen más dígitos de los que forman la contraseña o se pulsa la tecla '#' el programa borra la contraseña escrita y nos la vuelve a pedir. Además, aquí hemos añadido la asignación dinámica de memoria. Una vez que se pide cambiar la contraseña y se inserta la actual para comprobar que es el usuario el que quiere realizar la operación, el sistema nos pide el número de dígitos que queremos que formen parte de la contraseña. Inmediatamente después, la función realloc asigna el espacio correspondiente y se pide la nueva contraseña. Con esto, ampliamos mucho la elección de contraseña del usuario.

La función enterPassword es la encargada de recoger la contraseña introducida y comprobar si es correcta. En la pantalla nos aparece un mensaje para informarnos de que la alarma está activada y se nos pide la contraseña para desactivarla. Esta función también comprueba si la contraseña es correcta al presionar la tecla '*'. Si lo es la alarma se desactiva y si no lo es se muestra en pantalla un mensaje de error y se nos vuelve a pedir la contraseña. Por otro lado, si presionamos '#' o introducimos más dígitos de los que forman la contraseña, esta se nos vuelve a pedir.

Código completo Arduino

```
#include<LiquidCrystal.h>
#include<Keypad.h>
//declaro los pines a los que voy a conectar cada elemento
const int EchoPin = 10;
const int TriggerPin = 9;
const int PIN_CALDERA = 27;
#define LedPin 26
#define led_interior 28
#define pote A0
#define IN3 22
#define IN4 23
#define IN1 24
#define IN2 25
#define pulsador 2
#define pulsador_persiana_bajar 51
#define pulsador_persiana_subir 48
#define pulsador_luz 50
#define zumbador 8
#define ldr A1
#define DEBUG(a)

//declaro las variables que voy a utilizar en el código
int action = 0;
int blinkLed = 0;
int ledLevel = LOW;
//const int PIN_CALDERA = 2;
int dimension = 0;
const float SCALE_FACTOR = 0.05;
float userTemperature = 0.0;
long duration;
int distance, initialDistance, currentDistance, i;
int screenOffMsg = 0;
String password = "1234";
String tempPassword;
boolean activated = false; // estado de la alarma
boolean isActivated;
boolean activateAlarm = false;
boolean alarmActivated = false;
```

```
boolean enteredPassword; // estado de la contraseña para parar la alarma
boolean passChangeMode = false;
boolean passChanged = false;
const byte ROWS = 4; //4 filas
const byte COLS = 4; //4 columnas
char keypressed;
//defino los símbolos de las teclas del teclado
char keyMap[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = { 14, 15, 16, 17 }; //pines de las filas
byte colPins[COLS] = { 18, 19, 20, 21 }; //pines de las columnas
Keypad myKeypad = Keypad(makeKeymap(keyMap), rowPins, colPins, ROWS, COLS);
LiquidCrystal lcd(11, 12, 4, 5, 6, 7); // crea un objeto LCD. Parámetros: (rs,
enable, d4, d5, d6, d7). Indico donde conecto cada elemento del pin

void setup() {

	Serial.begin(9600);
	lcd.begin(16, 2);//lcd dividido en una matriz 16x2
	Serial.setTimeout(50);
	pinMode(pulsador_persiana_bajar, INPUT);
	pinMode(pulsador_persiana_subir, INPUT);
	pinMode(pulsador_luz, INPUT);
	pinMode(led_interior, OUTPUT);
	pinMode(LedPin, OUTPUT);
	pinMode(TriggerPin, OUTPUT);
	pinMode(EchoPin, INPUT);
	pinMode(IN4, OUTPUT);
	pinMode(IN3, OUTPUT);
	pinMode(IN1, OUTPUT);
	pinMode(IN2, OUTPUT);
	pinMode(pote, INPUT);
	pinMode(pulsador, INPUT);
	pinMode(zumbador, OUTPUT);
	pinMode(ldr, INPUT);
	pinMode(LED_BUILTIN, OUTPUT);
	pinMode(PIN_CALDERA, OUTPUT);
	int distancia(int, int); //prototipos funciones
	void pausa(unsigned int);
	void termostato(int);

}

void loop() {

	float temperature;
	char *ppas;
	ppas = &password[0];
	//termostato(analogRead(pote));
	luz(digitalRead(ldr));
	persiana_subir(digitalRead(pulsador_persiana_subir));
	persiana_bajar(digitalRead(pulsador_persiana_bajar));
	//alarma(digitalRead(pulsador));
	timbre(digitalRead(pulsador));
```

```
	temperature = control_temperature(userTemperature);
	action = Serial.read();
	conexionpc(action, temperature);
	luz_interior(digitalRead(pulsador_luz));



	//Código Alarma
	if (activateAlarm) {//si se selecciona la opción de activar la alarma
		lcd.clear();
		lcd.setCursor(0, 0);
		lcd.print("La alarma se");
		lcd.setCursor(0, 1);
		lcd.print("activara en");

		int countdown = 9; // 9 segundos de cuenta atrás antes de activar la
alarma
		while (countdown != 0) {
			lcd.setCursor(13, 1);
			lcd.print(countdown);
			countdown--;
			tone(zumbador, 700, 100);
			delay(1000);
		}
		lcd.clear();
		lcd.setCursor(0, 0);
		lcd.print("Alarma Activada");
		initialDistance = distancia(TriggerPin, EchoPin);//toma la distacia
inicial medida por el ultrasonidos
		activateAlarm = false;
		alarmActivated = true;
	}
	if (alarmActivated == true) {//si la alarma esta activada
		currentDistance = distancia(TriggerPin, EchoPin) + 10;//se comprueba
la distancia actual con la inicial calculada antes
		if (currentDistance < initialDistance) {//si la distancia es menor de
la inicial
			tone(zumbador, 1000); // Se activa el zumbador
			lcd.clear();
			enterPassword();//se llama a la función para introducir la
contraseña
		}
	}
	if (!alarmActivated) {//si la alarma no está activada
		if (screenOffMsg == 0) {
			lcd.clear();
			lcd.setCursor(0, 0);
			lcd.print("A - Activar");
			lcd.setCursor(0, 1);
			lcd.print("B - Cambiar contraseña");
			screenOffMsg = 1;
		}
		keypressed = myKeypad.getKey();
		if (keypressed == 'A') {	//Si se elige A se activa la alarma
			tone(zumbador, 1000, 200);
			activateAlarm = true;
		}
```

```
            else if (keypressed == 'B') { //si se elige la B se pide primero la
contraseña actual
                  lcd.clear();
                  int i = 1;
                  tone(zumbador, 2000, 100);
                  tempPassword = "";
                  lcd.setCursor(0, 0);
                  lcd.print("Contrasena:");
                  lcd.setCursor(0, 1);
                  lcd.print(">");
                  passChangeMode = true;
                  passChanged = true;
                  while (passChanged) {
                        keypressed = myKeypad.getKey();
                        if (keypressed != NO_KEY) {
                              if (keypressed == '0' || keypressed == '1' ||
keypressed == '2' || keypressed == '3' ||
                                    keypressed == '4' || keypressed == '5' ||
keypressed == '6' || keypressed == '7' ||
                                    keypressed == '8' || keypressed == '9') {
                                    tempPassword += keypressed;
                                    lcd.setCursor(i, 1);
                                    lcd.print("*");
                                    i++;
                                    tone(zumbador, 2000, 100);
                              }
                        }
                        if (i > 5 || keypressed == '#') {//Si excedemos el
número de dígitos o pulsamos # nos la vuelve a pedir
                              tempPassword = "";
                              i = 1;
                              lcd.clear();
                              lcd.setCursor(0, 0);
                              lcd.print("Contrasena:");
                              lcd.setCursor(0, 1);
                              lcd.print(">");
                        }
                        if (keypressed == '*') {
                              i = 1;
                              tone(zumbador, 2000, 100);
                              if (password == tempPassword) {//si la contraseña
es correcta se pide la nueva contraseña
                                    tempPassword = "";
                                    lcd.clear();
                                    lcd.setCursor(0, 0);
                                    //-----------------------------------
                                    lcd.print("Digitos:"); //nos pide el
número de dígitos que queremos en la nueva contraseña
                                    lcd.setCursor(0, 1);
                                    lcd.print(">");
                                    while (passChangeMode) {
                                          keypressed = myKeypad.getKey();
                                          if (keypressed != NO_KEY) {
                                                if (keypressed == '0' ||
keypressed == '1' || keypressed == '2' || keypressed == '3' ||keypressed == '4' ||
keypressed == '5' || keypressed == '6' || keypressed == '7' ||
keypressed == '8' || keypressed == '9')
```

```
{
              dimension += keypressed;
              lcd.setCursor(i, 1);
              lcd.print("*");
              i++;
              tone(zumbador, 2000, 100);
}

       lcd.clear();
       ppas = (char*)realloc(ppas, dimension * sizeof(char));
//-----------------------------------
       lcd.print("Nueva contrasena:");
       lcd.setCursor(0, 1);
       lcd.print(">");
       while (passChangeMode) {
       keypressed = myKeypad.getKey();
       if (keypressed != NO_KEY) {

       if (keypressed == '0' || keypressed == '1' || keypressed == '2' ||
       keypressed == '3' ||keypressed == '4' || keypressed == '5' || keypressed ==
       '6' || keypressed == '7' ||keypressed == '8' || keypressed == '9')
       {

       tempPassword += keypressed;

       lcd.setCursor(i, 1);

       lcd.print("*");
       i++;

       tone(zumbador, 2000, 100);
       }
}
if (i > dimension + 1 || keypressed == '#') {//se vuelve a pedir la nueva
contraseña

       tempPassword = "";

       i = 1;

       tone(zumbador, 2000, 100);

       lcd.clear();

       lcd.setCursor(0, 0);

       lcd.print("Nueva contrasena:");

       lcd.setCursor(0, 1);

       lcd.print(">");
}
if (keypressed == '*') {//se guarda la nueva contraseña
i = 1;
tone(zumbador, 2000, 100);
password = tempPassword;
passChangeMode = false;
passChanged = false;
```

```
screenOffMsg = 0;
                                                                  }
                                                            }
                                                      }
                                                }
                                          }
                                    }
                              }
                        }
                  }
            }
}

//-------------------------------------------------------------------
void enterPassword() { //funcion para introducir la contraseña por teclado
      int k = 5;
      tempPassword = "";
      activated = true;
      lcd.clear();//limpiamos pantalla
      lcd.setCursor(0, 0);//cursor en la parte superior izq de la pantalla
      lcd.print(" *** ALARMA *** ");
      lcd.setCursor(0, 1);//cursor en la parte inferior izq de la pantalla
      lcd.print("Cont>");
      while (activated) {
            keypressed = myKeypad.getKey();
            if (keypressed != NO_KEY) {
                  if (keypressed == '0' || keypressed == '1' || keypressed ==
'2' || keypressed == '3' ||
                        keypressed == '4' || keypressed == '5' || keypressed ==
'6' || keypressed == '7' ||
                        keypressed == '8' || keypressed == '9') {
                        tempPassword += keypressed;
                        lcd.setCursor(k, 1);
                        lcd.print("*");
                        k++;
                  }
            }
            if (k > dimension + 1 || keypressed == '#') { //si la contraseña
supera los dígitos o se presiona # se vuelve a pedir la contraseña
                  tempPassword = "";
                  k = dimension;
                  lcd.clear();
                  lcd.setCursor(0, 0);
                  lcd.print(" *** ALARMA *** ");
                  lcd.setCursor(0, 1);
                  lcd.print("Cont>");
            }
            if (keypressed == '*') { //si se presiona * entonces se comprueba si
la contraseña introducida es correcta
                  if (tempPassword == password) {
                        activated = false;
                        alarmActivated = false;
                        noTone(zumbador);
                        screenOffMsg = 0;
                  }
                  else if (tempPassword != password) {
                        lcd.clear();
                        lcd.setCursor(0, 1);
                        lcd.print("Error");
```

```
                        delay(2000);
                        lcd.clear();
                        lcd.setCursor(0, 0);
                        lcd.print(" *** ALARMA *** ");
                        lcd.setCursor(0, 1);
                        lcd.print("Cont>");
                    }
                }
            }
}
//--------------------------------------------------------------------------------
-------------------------------------------------------
int distancia(int TriggerPin, int EchoPin) //funcion distancia del ultrasonidos
{
        long duration, distance;
        digitalWrite(TriggerPin, LOW);
        delayMicroseconds(4);
        digitalWrite(TriggerPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(TriggerPin, LOW);

        duration = pulseIn(EchoPin, HIGH); //calculo el tiempo
        distance = duration * 10 / 292 / 2;  // calculo la distancia en cm
        return distance;
}
//--------------------------------------------------------------------------------
-----------------------------------------------------------
void luz(int valor)//enciende la luz en funcion de la lectura del ldr
{
        //Serial.println(valor);
        if (valor == HIGH)
        {
                digitalWrite(LedPin, LOW);
        }
        else
        {
                digitalWrite(LedPin, HIGH);

        }
}
//--------------------------------------------------------------------------------
-----------------------------------------------
void luz_interior(int valor)//función que enciende la luz en función de la lectura
del pulsador
{
        if (valor == LOW)
        {
                digitalWrite(led_interior, HIGH);
        }
        else
        {
                digitalWrite(led_interior, LOW);
        }
}
//--------------------------------------------------------------------------------
-------------------------------------------------------
void persiana_subir(int valorPulsador)//función que sube la persiana si el valor
del pulsador es 1
```

```
{
	//Serial.println(valorPulsador);//Mirar los pines según sentido del motor
	if (valorPulsador == LOW)
	{
		digitalWrite(IN3, LOW);
		digitalWrite(IN4, HIGH);
	}
	else
	{
		digitalWrite(IN4, LOW);
	}
}
//--------------------------------------------------------------------------------
----------------------------------------------------------------
void persiana_bajar(int valorPulsador)//función que baja la persiana si el valor
del pulsador es 1
{
	if (valorPulsador == LOW)
	{
		digitalWrite(IN3, HIGH);
		digitalWrite(IN4, LOW);
	}
	else
	{
		digitalWrite(IN3, LOW);
	}
}
//--------------------------------------------------------------------------------
----------------------------------------------------------------
void alarma(int valorPulsador)//
{
	int cm = distancia(TriggerPin, EchoPin);
	if (valorPulsador == LOW)
	{
		if (cm <= 19)
		{
			analogWrite(zumbador, 300);
			pausa(5000);
			analogWrite(zumbador, LOW);
		}
	}
	else
	{
		analogWrite(zumbador, LOW);
	}
}
//--------------------------------------------------------------------------------
----------------------------------------------------------------
void timbre(int valorPulsador)//función que activa el timbre al pulsar el pulsador
{
	if (valorPulsador == HIGH)
	{
		analogWrite(zumbador, 884);
	}
	else
	{
		digitalWrite(zumbador, LOW);
	}
```

```
}
//--------------------------------------------------------------------------------
-------------------------------------------------------------------
void pausa(unsigned int milisegundos)//función para crear una pausa sin usar el
delay
{
	volatile unsigned long compara = 0;
	volatile int contador = 0;
	do
	{
		if (compara != millis())
		{
			contador++;
			compara = millis();
		}
	} while (contador <= milisegundos);
	return;
}
//--------------------------------------------------------------------------------
-------
// control_temperature
// Lee el valor del potenciómetro que simula el sensor de temperatura del pin A0.
// Lo convierte a grados centrígrados y compara ese valor con el valor deseado de
// temperatura indicado por el usuario. Si la temperatura es menor pone a 1 el pin
// donde está conectado el arranque de la calefacción, sino lo pone a 0
//
// Necesita:
//  - userGradTemp: Valor de temperatura deseada por el usuario
//
// Retorna: el valor de temperatura leído del potenciómetro
//--------------------------------------------------------------------------------
-------
float control_temperature(float userGradTemp)
{
	int rawTemp = 0;	// Valor del potenciómetro
	float gradTemp = 0;	// Temperatura en grados


	// Se supone que el potenciómetro que simula el sensor de temperatura
	// está conectado al pin A0

	// Se lee el valor del potenciómetro
	rawTemp = analogRead(A0);

	// Se transforma el valor leído a grados usando el factor de escala
	// indicado
	gradTemp = rawTemp * SCALE_FACTOR;

	// Si la temperatura es menor que la deseada, se enciende la caldera
	if (gradTemp < userGradTemp)
		digitalWrite(PIN_CALDERA, HIGH);
	else
		digitalWrite(PIN_CALDERA, LOW);

	// Se devuelve el valor
	return gradTemp;
}
```

```
//-------------------------------------------------------------------------------
-----------------------
void conexionpc(int action, float temperature)//función que conecta el puerto serie
al Arduino
{
	switch (action)
	{
	case '1': {
		// Se indica que el led debe encenderse y no parpadear
		ledLevel = HIGH;
		blinkLed = 0;
		break;
	}

	case '2': {
		// Se indica que el led debe apagarse y no parpadear
		ledLevel = LOW;
		blinkLed = 0;
		break;
	}

	case '3': {
		// Se indica que el led debe parpadear
		blinkLed = 1;
		break;
	}

	case '4': {
		// Se envia el valor de temperatura
		Serial.print(temperature);
		break;
	}

	case '5': {
		// Se lee el valor deseado para la temperatura transformándolo en un
float
		userTemperature = Serial.parseFloat();

		break;
	}
	}
	if (blinkLed)
	{
		// Se invierte la indicación para el led
		if (ledLevel == HIGH)
			ledLevel = LOW;
		else
			ledLevel = HIGH;

		// Se esperan 500 ms
		delay(500);
	}

	// Se escribe el valor deseado en el led
	digitalWrite(LED_BUILTIN, ledLevel);
}
```