

Escuela Académica Profesional de Ingeniería Informática



Curso : Arquitectura de computadores

Tema : Capítulo IV - V: Descripción del lenguaje de

programación y de la tarjeta Arduino y

explicación del programa control.

Docente : Dr. (c) Ing. David Arauco Cabrera

Integrantes :

Carhuavilca Maldonado, Diego.

Chirinos Huayna, Andrea.

Sihuas de la Torre, Angello.

Vergaray Carbajal, Katya.

2020 – I

CONTENIDO

[CAPÍTULO IV DESCRIPCION DEL LENGUAJE DE PROGRAMACION Y DE LA TARJETA ARDUINO QUE SE UTILIZA PARA ESTE TI 3](#_Toc47815677)

[**Sistema automatizado de casa inteligente** 3](#_Toc47815678)

[1. Lenguaje de programación Java 3](#_Toc47815679)

[2. Arduino Uno 8](#_Toc47815680)

[CAPÍTULO V EXPLICACION DEL PROGRAMA DE CONTROL Y EL CODIGO DE LA TARJETA ARDUINO 11](#_Toc47815681)

[1. Materiales 11](#_Toc47815682)

[2. Esquema 14](#_Toc47815683)

[3. Código de Arduino 15](#_Toc47815684)

[4. Programación JAVA 17](#_Toc47815685)

[5. Interfaces 20](#_Toc47815686)

[6. Funcionamiento 21](#_Toc47815687)

[CONCLUSIONES 27](#_Toc47815688)

[BIBLIOGRAFÍA 28](#_Toc47815689)

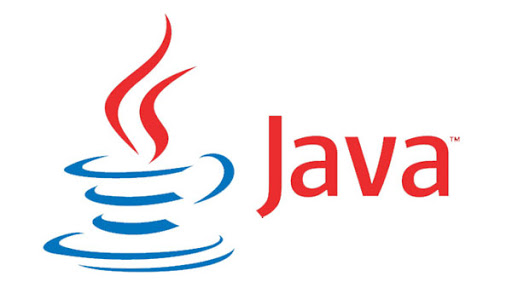
CAPÍTULO IV  
DESCRIPCION DEL LENGUAJE DE PROGRAMACION Y DE LA TARJETA ARDUINO QUE SE UTILIZA PARA ESTE TI

**Sistema automatizado de casa inteligente**

Este proyecto fue desarrollado para poder dar órdenes, el cual consiste del diseño e implementación una aplicación para el control de los artefactos electrónicos de una casa. La aplicación ha sido concebida para funcionar desde cualquier computadora por la cual se puede controlar la puerta principal, las luces de cada piso de la vivienda, las persianas y el aire acondicionado.

1. Lenguaje de programación Java

Java es una plataforma informática y a su vez un lenguaje de programación creado en 1995 por la empresa Sun Microsystem. El objetivo de este lenguaje es que los programadores sólo tuvieran que escribir el código de un programa una vez, y que éste, pudiese ejecutarse en cualquier dispositivo. Esto es posible gracias a la Máquina Virtual de Java (JVM), que brinda esa portabilidad necesaria. Con Java se pueden crear programas en una gran variedad de dispositivos, permitiendo ejecutar la misma aplicación en diversos sistemas operativos.



*Figura 01: Logo de Java*

Características:

Simple

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ no es un lenguaje conveniente por razones de seguridad, pero C y C++ son los lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que limita en mucho la fragmentación de la memoria.

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

Aritmética de punteros

No existen referencias

Registros (struct)

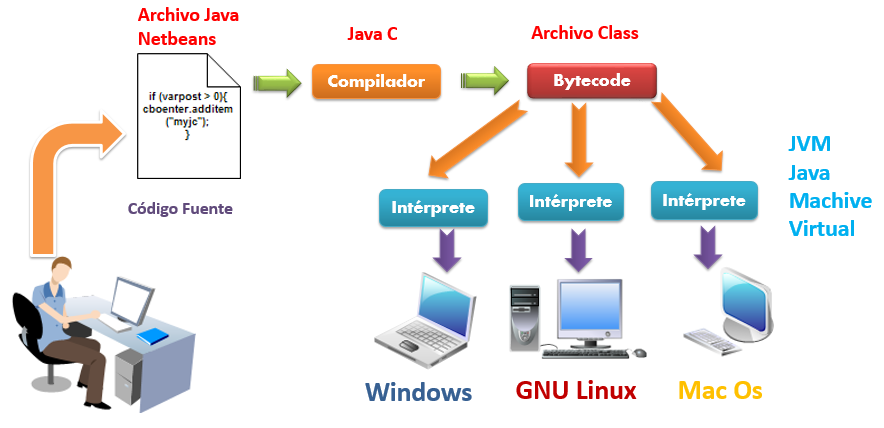
Definición de tipos (typedef)

Macros (#define)

Necesidad de liberar memoria (free)

Aunque, en realidad, lo que hace es eliminar las palabras reservadas (struct, typedef), ya que las clases son algo parecido.

Además, el intérprete completo de Java que hay en este momento es muy pequeño, solamente ocupa 215 Kb de RAM.



*Figura 02: Funcionamiento de Java*

Orientado a Objetos

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria. Incorpora funcionalidades inexistentes en C++ como, por ejemplo, la resolución dinámica de métodos. Esta característica deriva del lenguaje Objective C, propietario del sistema operativo Next. En C++ se suele trabajar con librerías dinámicas (DLLs) que obligan a recompilar la aplicación cuando se retocan las funciones de su interior.

Este inconveniente es resuelto por Java mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

Distribuido

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Robusto

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

Java proporciona, pues:

Comprobación de punteros

Comprobación de límites de arrays

Excepciones

Verificación de ByteCodes

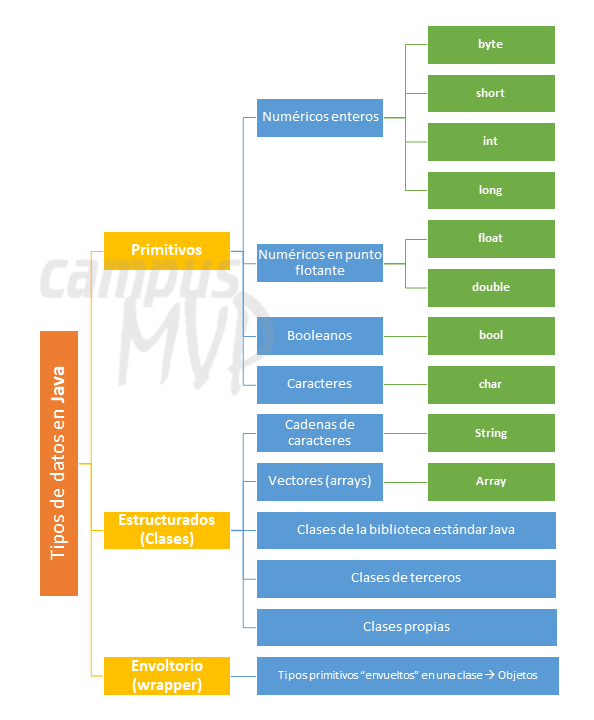
Arquitectura Neutral

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

El código fuente Java se "compila" a un código de bytes de alto nivel independiente de la máquina. Este código (ByteCode) está diseñado para ejecutarse en una máquina hipotética que es implementada por un sistema run-time, que sí es dependiente de la máquina.

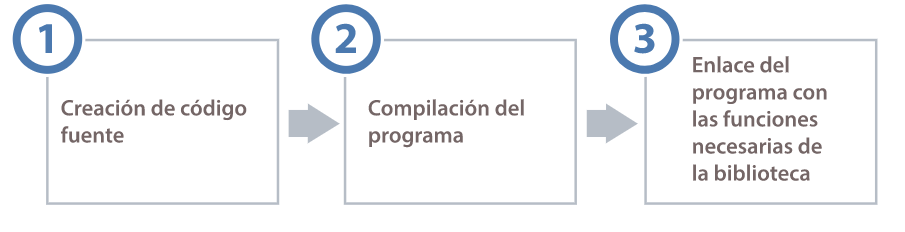
Programación en Java

Java es un lenguaje orientado a objetos, independiente de la plataforma hardware donde se desarrolla, y que utiliza una sintaxis similar a la de C++ pero reducida. Es un lenguaje con una curva de aprendizaje baja (se puede decir que es fácil de aprender) y que dispone de una gran funcionalidad de base (incrementada por la gran cantidad de código de terceros existente). Java, como lenguaje de programación, ofrece un código robusto, que ofrece un manejo automático de la memoria, lo que reduce el número de errores.



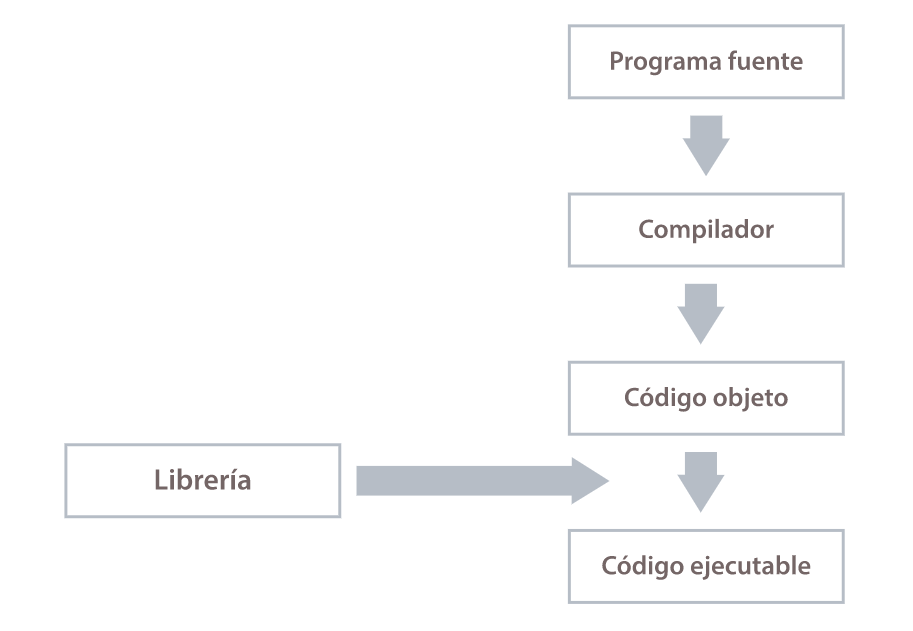
**Fases de la compilación**

La compilación permite crear un programa de computadora que puede ser ejecutado por ésta y comprende tres pasos:



Fases de compilación

Algunos procesos de compilación podrían presentar variaciones, pero en general se presenta así:

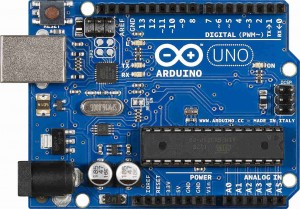


Los *compiladores* realizan la traducción en tiempo de desarrollo; es decir, el programa aún no se está ejecutando. El compilador recibe todo el código fuente, lo analiza, optimiza y traduce a lenguaje máquina dejando un programa completo listo para su ejecución. Por ejemplo, el C o el Pascal son lenguajes compilados. En cambio, los *intérpretes* realizan la traducción en tiempo de ejecución, o sea, a medida que el programa se va ejecutando, el intérprete traduce instrucciones al lenguaje máquina. Basic es un lenguaje interpretado.

Es importante mencionar que los lenguajes de programación son una herramienta principal para el desarrollo de *software* o sistema. Una de las funciones de la programación es llevar a cabo una comunicación entre la máquina y el usuario por medio de un *software*. Por ello es importante saber los tipos de lenguaje, características y, sobre todo, el tipo de plataforma que utilizan, ya que dependiendo del tipo de trabajo que se vaya a realizar, ya sea el usuario o empresa, se deberá elegir el lenguaje de programación más adecuado para agilizar y realizar tales actividades.

1. Arduino Uno

Es una placa electrónica basada en el microcontrolador ATmega328. Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM (Modulación por ancho de pulsos) y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera ICSP y un botón de reseteado. La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador.

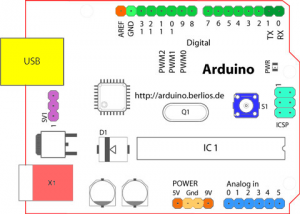


Características técnicas de Arduino Uno r3:

* Microcontrolador: ATmega328
* Voltage: 5V
* Voltage entrada (recomendado): 7-12V
* Voltage entrada (limites): 6-20V
* Digital I/O Pins: 14 (de los cuales 6 son salida PWM)
* Entradas Analogicas: 6
* DC Current per I/O Pin: 40 mA
* DC Current parar 3.3V Pin: 50 mA
* Flash Memory: 32 KB (ATmega328) de los cuales 0.5 KB son utilizados para el arranque
* SRAM: 2 KB (ATmega328)
* EEPROM: 1 KB (ATmega328)
* Clock Speed: 16 MHz

Introducción general a una placa Arduino

Mirando a la placa desde la parte de arriba, este es el esquema de lo que puedes ver (los componentes de la placa con los que puedes interactuar en su uso normal están resaltados):



Empezando según las agujas del reloj:

* Terminal de referencia analógica (naranja)
* Tierra digital (verde claro)
* Terminales digitales 2-13 (verde)
* Terminales digitales 0-1/ E/S serie – TX/RX (verde oscuro) – Estos pines no se pueden utilizar como e/s digitales (digitalRead() y digitalWrite()) si estás utilizando comunicación serie (por ejemplo Serial.begin).
* Botón de reinicio – S1 (azul oscuro)
* Programador serie en circuito «In-circuit Serial Programmer» o «ICSP» (azul celeste).
* Terminales de entrada analógica 0-5 (azul claro)
* Terminales de alimentación y tierra (alimentación: naranja, tierras: naranja claro)
* Entrada de alimentación externa (9-12VDC) – X1 (rosa)
* Selector de alimentación externa o por USB (coloca un jumper en los dos pines más cercanos de la alimentación que quieras) – SV1 (púrpura). En las versiones nuevas de Arduino la selección de alimentación es automática por lo que puede que no tengas este selector.
* USB (utilizado para subir programas a la placa y para comunicaciones serie entre la placa y el ordenador; puede utilizarse como alimentación de la placa) (amarillo).

Entradas y salidas digitales/analógicas

Un sistema electrónico es cualquier disposición de componentes electrónicos con un conjunto definido de entradas y salidas. Una placa Arduino, por tanto, puede pensarse de forma simplificada como un sistema que acepta información en forma de señal de entrada, desarrolla ciertas operaciones sobre ésta y luego produce señales de salida.

Justamente, una de las opciones que hacen más potente a Arduino son sus entradas/salidas digitales. ¿Entonces por qué hablamos de analógicas?

En los sistemas electrónicos, una magnitud física variable se representa generalmente mediante una señal eléctrica que varía de manera tal que describe esa magnitud. Por lo general, se hace referencia a las señales continuas como señales analógicas, mientras que asociamos las señales discretas a señales digitales: el ejemplo más claro es el de las señales binarias, donde la señal sólo puede tomar dos niveles, 0 o 1.

Arduino incorpora terminales digitales (señales discretas) pero de tal forma que tenemos un gran abanico de valores con los que trabajar (por ejemplo, 255 valores de luz en un fotosensor, siendo 0 ausencia de luz y 254 el máximo valor lumínico).

Terminales Digitales

Las terminales digitales de una placa Arduino pueden ser utilizadas para entradas o salidas de propósito general a través de los comandos de programación pinMode(), digitalRead(), y digitalWrite(). Cada terminal tiene una resistencia pull-up que puede activarse o desactivarse utilizando digitalWrite() (con un valor de HIGH o LOW, respectivamente) cuando el pin está configurado como entrada. La corriente máxima por salida es 40 mA.

* Serial: 0 (RX) y 1 (TX). Utilizado para recibir (RX) y transmitir (TX) datos serie TTL. En el Arduino Diacemila, estas terminales están conectadas a las correspondientes patas del circuito integrado conversor FTDI USB a TTL serie. En el Arduino BT, están conectados a las terminales correspondientes del módulo Bluetooth WT11. En el Arduino Mini y el Arduino LilyPad, están destinados para el uso de un módulo serie TTL externo (por ejemplo, el adaptador Mini-USB).
* Interruptores externos: 2 y 3. Estas terminales pueden ser configuradas para disparar una interrupción con un valor bajo, un pulso de subida o bajada, o un cambio de valor. Mira la función attachInterrupt() para más detalles.
* PWM: 3, 5, 6, 9, 10, y 11. Proporcionan salidas PWM de 8 bit con la función analogWrite(). En placas con ATmega8, las salidas PWM solo están disponibles en los pines 9, 10, y 11.
* Reset BT: 7. (solo en Arduino BT) Conectado a la línea de reset del módulo bluetooth.
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estas terminales soportan comunicación SPI. Aunque esta funcionalidad esta proporcionada por el hardware, no está incluida actualmente en el lenguaje Arduino.
* LED: 13. En el Diacemila y el LilyPad hay un led en placa conectado al pin digital 13. cuando el pin tiene valor HIGH, el LED está encendido, cuando el pin está en LOW, está apagado

Pines Analógicos

* Los pines de entrada analógicos soportan conversiones analógico-digital (ADC) de 10 bit utilizando la función analogRead(). Las entradas analógicas pueden ser también usadas como pines digitales: entrada analógica 0 como pin digital 14 hasta la entrada analógica 5 como pin digital 19. Las entradas analógicas 6 y 7 (presentes en el Mini y el BT) no pueden ser utilizadas como pines digitales.
* I2C: 4 (SDA) y 5 (SCL). Soportan comunicaciones I2C (TWI) utilizando la librería Wire (documentación en la página web de Wiring).

Pines de alimentación

* VIN (a veces marcada como «9V»). Es el voltaje de entrada a la placa Arduino cuando se está utilizando una fuente de alimentación externa (En comparación con los 5 voltios de la conexión USB o de otra fuente de alimentación regulada). Puedes proporcionar voltaje a través de este pin. Date cuenta que diferentes placas aceptan distintos rangos de voltaje de entrada, por favor, mira la documentación de tu placa. También date cuenta que el LilyPad no tiene pin VIN y acepta solo una entrada regulada.
* 5V. La alimentación regulada utilizada para alimentar el microcontrolador y otros componentes de la placa. Esta puede venir de VIN a través de un regulador en placa o ser proporcionada por USB u otra fuente regulada de 5V.
* 3V3. Una fuente de 3.3 voltios generada por el chip FTDI de la placa.
* GND. Pines de tierra.

Otros Pines

* AREF. Referencia de voltaje para las entradas analógicas. Utilizada con la función analogReference().
* Reset. Pon esta línea a LOW para resetear el microcontrolador. Utilizada típicamente para añadir un botón de reset a shields que bloquean el de la placa principal.

CAPÍTULO V  
EXPLICACION DEL PROGRAMA DE CONTROL Y EL CODIGO DE LA TARJETA ARDUINO

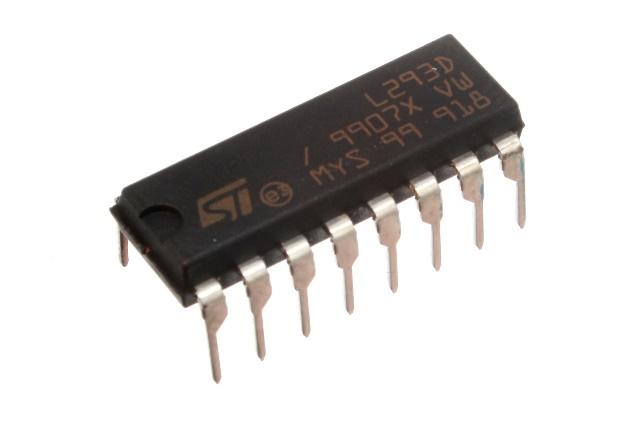
1. **Materiales**

L293D

El L293D es un C.I. de 4 canales capaz de proporcionar una corriente de salida de hasta 600mA por canal y puede soportar picos de hasta 1.2 A. Cada canal es controlado por señales TTL (los elementos de entrada y salida del dispositivo son transistores bipolares) y cada pareja de canales dispone de una señal de habilitación para conectar o desconecta las salidas de los mismos.

Tiene la disponibilidad de poder utilizar dos tensiones diferentes, una para el propio circuito integrado y otra para la alimentación del motor, cosa que nos facilita, al poder tomar la alimentación del Circuito Integrado (C.I.) del pin +5 v de Arduino y utilizar una batería auxiliar para la alimentación del motor o motores.

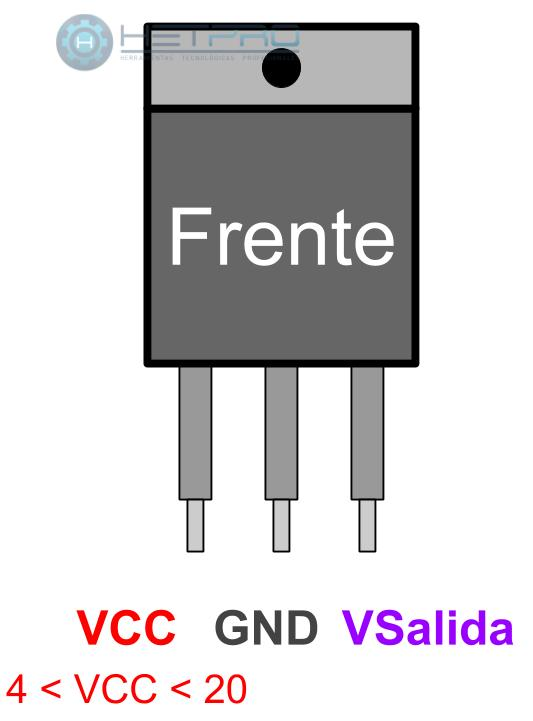
|  |  |  |
| --- | --- | --- |
| PIN | Nombre | Descripción |
| 1 | EN 1 y 2 | Habilitación de los Canales 1 y 2 |
| 2 | 1 A | Entrada Canal 1 |
| 3 | 1 Y | Salida Canal 1 |
| 4 | GND | Tierra |
| 5 | GND | Tierra |
| 6 | 2 Y | Salida Canal 2 |
| 7 | 2 A | Entrada Canal 2 |
| 8 | VCC 2 | Alimentación del Motor |
| 9 | EN 3 y 4 | Habilitación de los canales 3 y 4 |
| 10 | 3 A | Entrada Canal 3 |
| 11 | 3 Y | Salida Canal 3 |
| 12 | GND | Tierra |
| 13 | GND | Tierra |
| 14 | 4 Y | Salida Canal 4 |
| 15 | 4 A | Entrada Canal 4 |
| 16 | VCC 1 | Alimentación del Circuito |



LM35

El LM35 es un sensor de temperatura de buenas prestaciones a un bajo precio. Posee un rango de trabajo desde -55ºC hasta 150ªC. Su salida es de tipo analógica y lineal con una pendiente de 10mV/ºC. El sensor es calibrado de fábrica a una precisión de 0.5ºC.

Es un sensor muy popular por su fácil uso y variadas aplicaciones. No necesita de ningún circuito adicional para ser usado. Se alimenta directamente con una fuente de 5V y entrega una salida analógica entre 0V a 1.5V. Este voltaje analógico puede ser leído por el ADC de un microcontrolador como PIC o Arduino. Entre sus aplicaciones podemos encontrar termómetros, termostatos, sistemas de monitoreo y más.



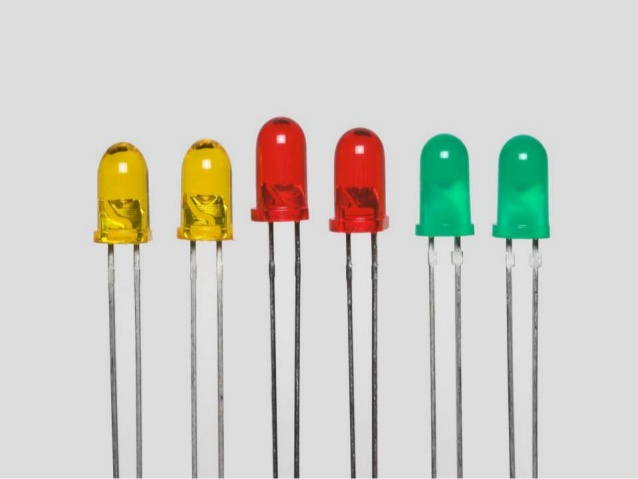
Arduino Uno



Motor 12V



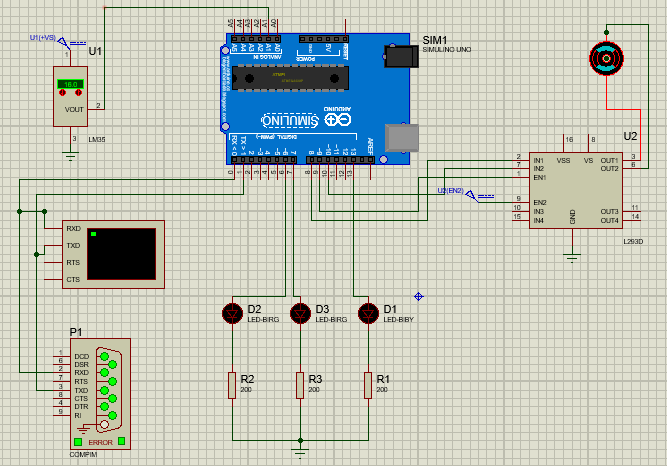
3 leds



3 resistencias 200 ohm



1. **Esquema**



1. **Código de Arduino**

int val;

#define Led 13

char dataIn = "";

int motor1 = 11; //conexión motora

DECLARACION

CONSTANTES

int pot1 =A0; // conexión potenciómetro

const int enPin = 8; // PWM se conecta al pin 1 del puente H

const int in1Pin =9; // Entrada 2 del puente H

const int in2Pin =4; // Entrada 7 del puente H

void setup(){

Serial.begin(9600); //La velocidad de comunicación serial en Arduino

digitalWrite(Led, LOW); //Valores lógicos digitales en un pin de Salida

pinMode (pot1 , INPUT) ;

DEFINICION DE PINES DE ENTRADA

pinMode (A1 , INPUT);

pinMode(Led, OUTPUT);

pinMode (6 , OUTPUT);

DEFINICION DE PINES DE SALIDA

pinMode (7 , OUTPUT);

pinMode (motor1 , OUTPUT);

pinMode (in1Pin , OUTPUT);

pinMode (in2Pin , OUTPUT);

}

void loop (){

int speed = map ('5' , '0' , '9' ,0,255); //Definir valores de la velocidad del motor

analogWrite (enPin ,speed);

if (Serial.available()) {

dataIn = Serial.read();

if (dataIn =='a'){

APAGADO MOTOR

digitalWrite (in1Pin , LOW);

digitalWrite (in2Pin , HIGH);

SISTEMA DE PERSIANA

}

if (dataIn =='b'){

digitalWrite (in1Pin , HIGH);

MOTOR GIRA A LA DERECHA

digitalWrite (in2Pin , LOW);

}

if (dataIn =='c'){

digitalWrite (in1Pin , HIGH);

MOTOR GIRA A LA IZQUIERDA

digitalWrite (in2Pin , LOW);

if (dataIn == '1') {

ENCENDIDO LUCES

digitalWrite(Led, HIGH);

SISTEMA LUCES

}

if (dataIn == '0') {

APAGADO LUCES

digitalWrite(Led,LOW);

}}

val = analogRead (A1); // Lectura de sensor de temperatura

float mv = (val/1024.0)\*5000;

float temp = mv/10; // calculo de temperatura en grados C

if (temp >20){

digitalWrite (7, HIGH);

VENTILACION Y LED DE ENCEDIDO

digitalWrite(6, HIGH);

}

SISTEMA

VENTILACION

else {

digitalWrite(6, HIGH);

delay(500);

VENTILACION APAGADA Y LED DE PARPADEANDO

digitalWrite(6, LOW);

delay(500);

digitalWrite (7, LOW);

}

delay (500);

}

1. **Programación JAVA**

package proyecto.arduino;

import com.panamahitek.ArduinoException;

import com.panamahitek.PanamaHitek\_Arduino;

import java.awt.BorderLayout;

import java.awt.Color;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import jssc.SerialPortEvent;

import jssc.SerialPortEventListener;

import jssc.SerialPortException;

public class GUI extends javax.swing.JFrame {

String Hora,Minutos,Segundos;

Thread Hilo;

public static float celsius = 0;

public static PanamaHitek\_Arduino Arduino = new PanamaHitek\_Arduino();

public GUI() {

initComponents();

ind\_1.setOpaque(true);

resetColor(new JPanel[]{btn\_2,btn\_3}, new JPanel[]{ind\_2,ind\_3});

home();

try {

Arduino.arduinoTX("COM6", 9600);

} catch (ArduinoException ex) {

// System.err.println("No hay arduino para comunicar");

Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);

JOptionPane.showMessageDialog(null, "No hay arduino para comunicar");} }

public void home(){

Home h = new Home();

h.setSize(721, 439);

h.setLocation(0,0);

PanelContenedor.removeAll();

PanelContenedor.add(h, BorderLayout.CENTER);

PanelContenedor.revalidate();

PanelContenedor.repaint();}

private void btn\_1MousePressed(java.awt.event.MouseEvent evt) {

setColor(btn\_1);

ind\_1.setOpaque(true);

resetColor(new JPanel[]{btn\_2,btn\_3}, new JPanel[]{ind\_2,ind\_3});}

private void btn\_2MousePressed(java.awt.event.MouseEvent evt) {

setColor(btn\_2);

ind\_2.setOpaque(true);

resetColor(new JPanel[]{btn\_1,btn\_3}, new JPanel[]{ind\_1,ind\_3});}

private void btn\_3MousePressed(java.awt.event.MouseEvent evt) {

setColor(btn\_3);

ind\_3.setOpaque(true);

resetColor(new JPanel[]{btn\_2,btn\_1}, new JPanel[]{ind\_2,ind\_1});}

private void btn\_1MouseClicked(java.awt.event.MouseEvent evt) {

Puerta p = new Puerta();

p.setSize(709, 435);

p.setLocation(0,0);

PanelContenedor.removeAll();

PanelContenedor.add(p, BorderLayout.CENTER);

PanelContenedor.revalidate();

PanelContenedor.repaint();}

private void btn\_2MouseClicked(java.awt.event.MouseEvent evt) {

Luces l = new Luces();

l.setSize(709, 435);

l.setLocation(0,0);

PanelContenedor.removeAll();

PanelContenedor.add(l, BorderLayout.CENTER);

PanelContenedor.revalidate();

PanelContenedor.repaint();}

private void btn\_InicioMouseClicked(java.awt.event.MouseEvent evt) {

Home h = new Home();

h.setSize(721, 439);

h.setLocation(0,0);

PanelContenedor.removeAll();

PanelContenedor.add(h, BorderLayout.CENTER);

PanelContenedor.revalidate();

PanelContenedor.repaint();}

private void btn\_3MouseClicked(java.awt.event.MouseEvent evt) {

Cortinas c = new Cortinas();

c.setSize(709, 435);

c.setLocation(0,0);

PanelContenedor.removeAll();

PanelContenedor.add(c, BorderLayout.CENTER);

PanelContenedor.revalidate();

PanelContenedor.repaint();}

private void setColor(JPanel pane){

pane.setBackground(new Color(23,35,51));}

private void resetColor(JPanel [] pane, JPanel [] indicators){

for(int i=0;i<pane.length;i++){

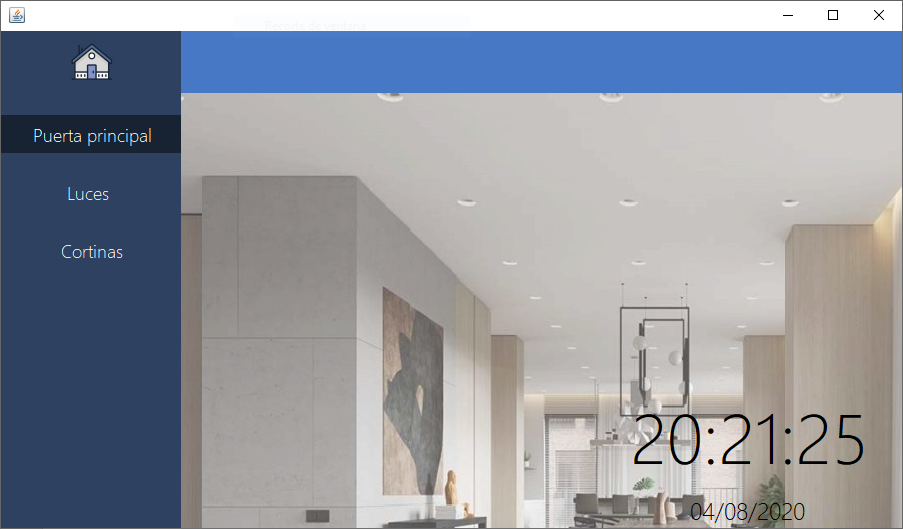
pane[i].setBackground(new Color(47,65,96));

} for(int i=0;i<indicators.length;i++){

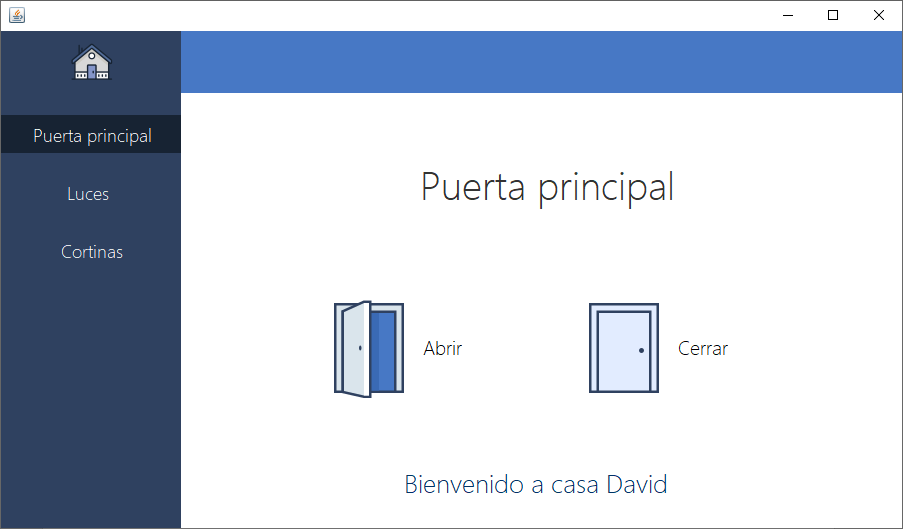
indicators[i].setOpaque(false);

} }

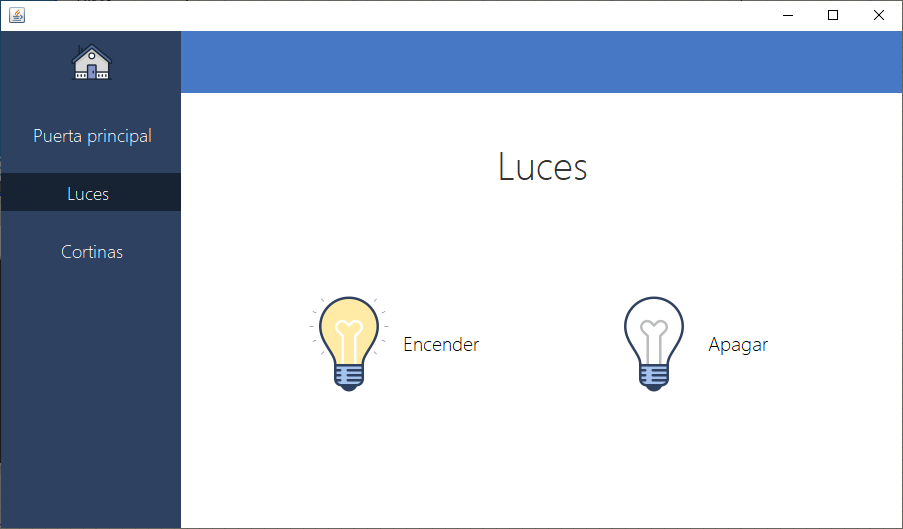
1. **Interfaces**



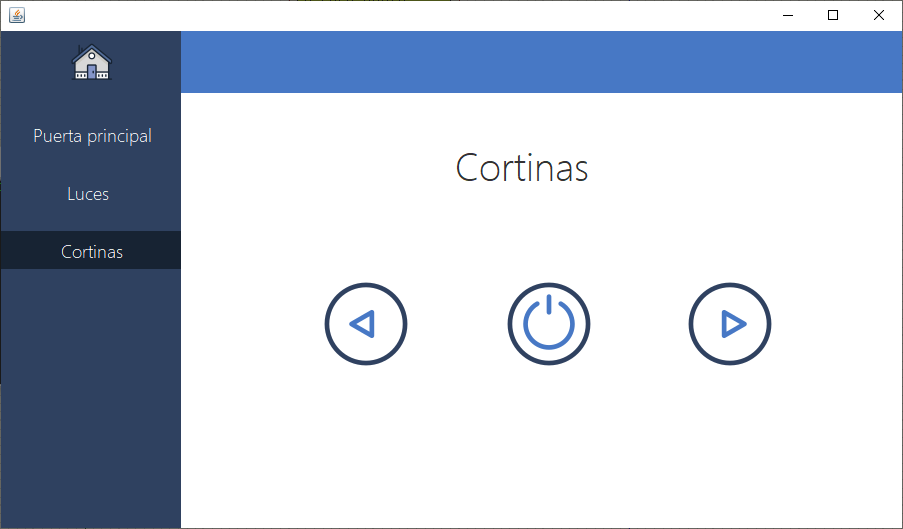
*Pantalla principal*



*Pantalla de la puerta principal*



*Pantalla de Luces*

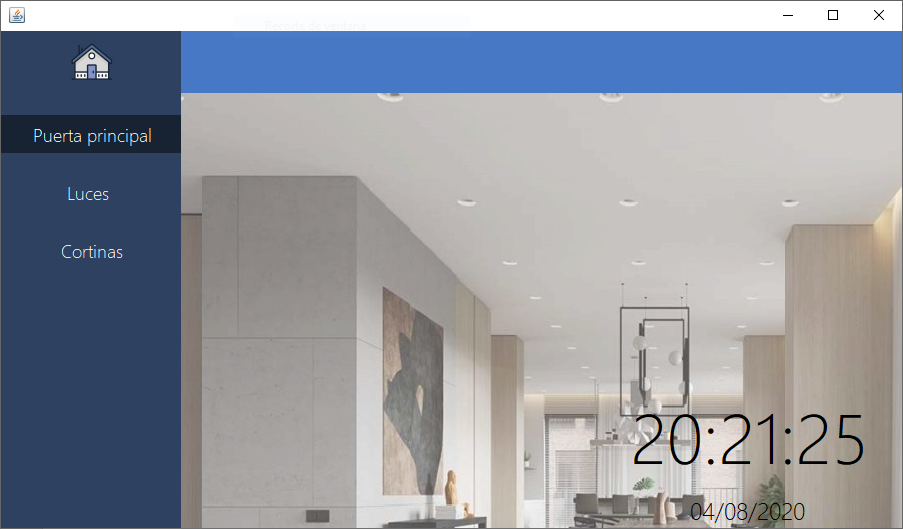


*Pantalla de Cortinas*

1. **Funcionamiento**

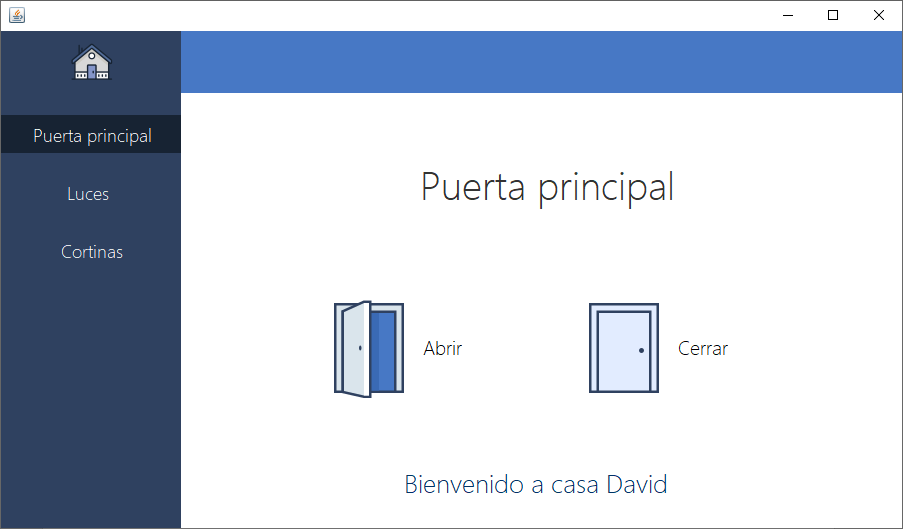
**Interfaz principal**

En ella nos muestra los diversos artefactos electrónicos que se pueden controlar. Además, no permite tener conocimiento de la hora y fecha del día.

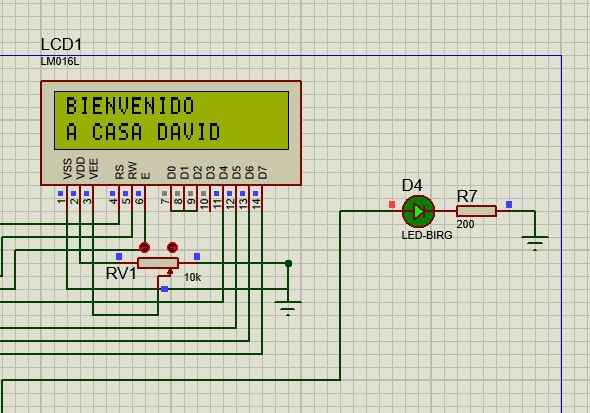


**Control Puerta**

Apertura de puerta



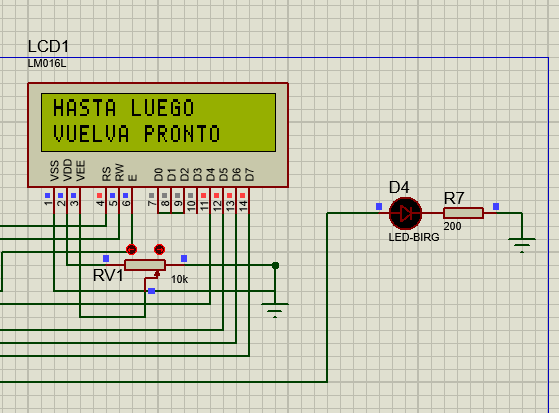
Al momento de accionar el botón de Abrir nos mostrara un mensaje de bienvenida y el led que simula el interruptor de la puerta se activara permitiendo ingresar a la casa.



Cerrar Puerta

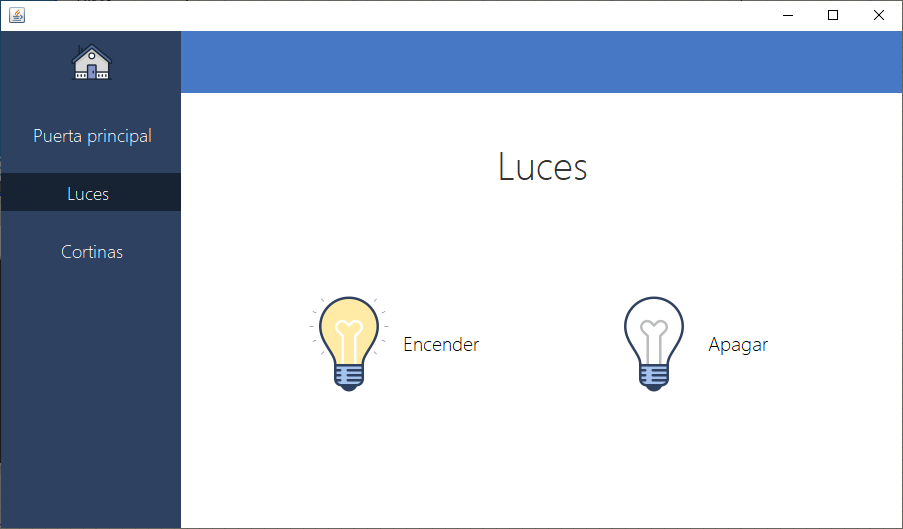


Al momento de accionar el botón de Cerrar nos mostrara un mensaje de despedida y el led que simula el interruptor de la puerta se desactivara indicando que las puertas se encuentran cerradas.

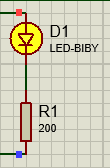


**Control luces**

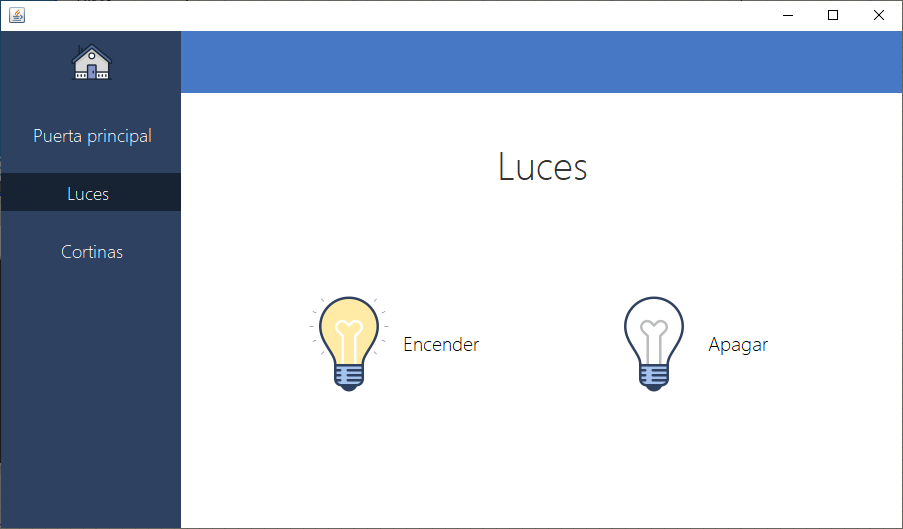
Encendido



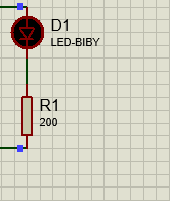
Al momento de presionar el botón Encender permitirá encender las luces de la casa simulada por el led.



Apagado

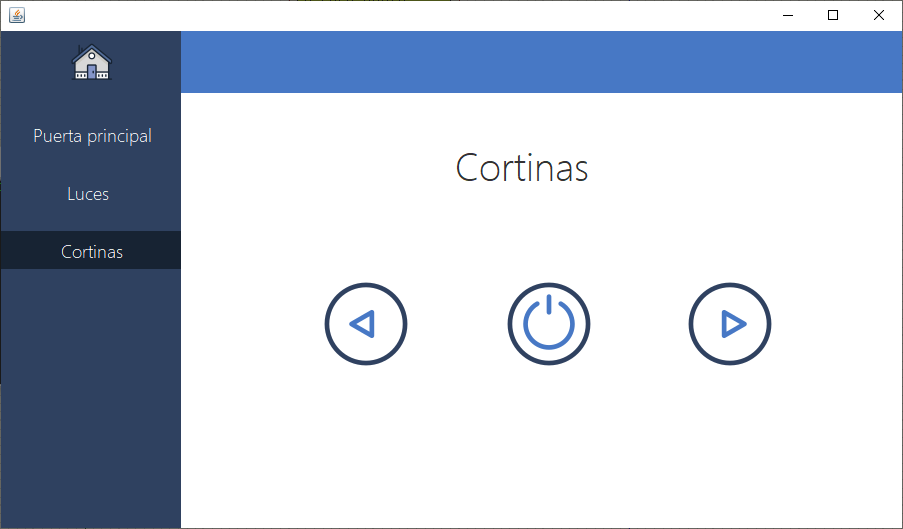


Al momento de presionar el botón Apagar permitirá apagar las luces de la casa simulada por el led

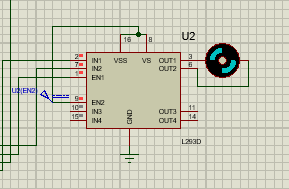


**Control Cortinas**

Nos permitirá cerrar y abrir las cortinas

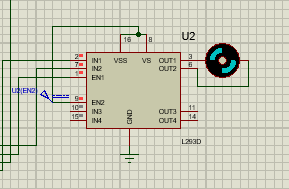


Apagar



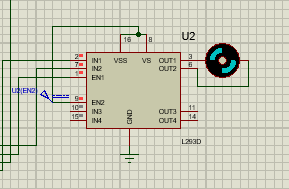
Detiene el movimiento de las cortinas

Cerrado de cortinas



Permite cerrar las cortinas y están se detendrán hasta que se presione el botón de apagado( )

Apertura Cortinas

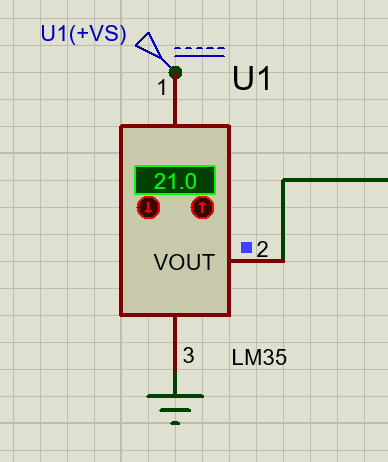
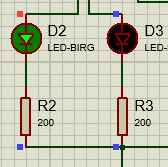


Permite abrir las cortinas y están se detendrán hasta que se presione el botón de apagado( )

**Sistema Automatizado de Ventilación**

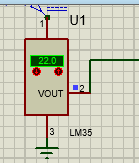
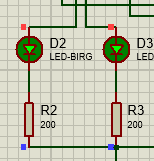
Ventilación Apagada

El led D2 continuara parpadeando y el led D3 que simula el ventilador permanecerá apagado mientras el sensor de temperatura se encuentre por debajo de los 22 grados



Ventilación Encendida

El led D2 se encenderá y el led D3 que simula el ventilador permanecerá encendido mientras el sensor de temperatura detecte una temperatura mayor o igual a los 22 grados



# CONCLUSIONES

La domótica es hoy en día desconocida para la mayoría de las personas, por lo que el mercado que ésta abarca es muy pequeño. El objetivo que se debería marcar, es dar a conocer todas las ventajas que ofrece, creando Interfaces fáciles de usar por parte de cualquier usuario, y que su instalación no provoque la necesidad de grandes obras en las viviendas.

La Interfaz Gráfica se ha creado pensando en que debe ser fácilmente utilizable por usuarios de diferentes edades y conocimientos. Por lo tanto, se optó porque fuera lo más visual posible, y que las órdenes a los distintos dispositivos domóticos, siempre que fuera posible, se hicieran haciendo tan sólo clic sobre los iconos que los representan.

La domótica ofrece muchas ventajas, no sólo de confort como la automatización de distintos sistemas como la iluminación, subida y bajada de persianas, gestión multimedia, etc. Sino también seguridad en el hogar, no sólo ante intrusiones, sino con una gran diversidad de detectores para controlar que no hay ninguna fuga o problema en la vivienda. Además, existe la posibilidad de un ahorro energético importante, gracias a programas de zonificación y gestión.

Al desarrollar en base a Arduino nos percatamos que es una herramienta en extremo útil para el desarrollo de proyectos que ofrece una gama de libertad tecnológica no encontrada en que otras herramientas y que de manera significativa contempla las siguientes ventajas.

* Está basado en dos sistemas abiertos, por lo que nos da la total libertad de entender el hardware y software.
* Su entorno nos permite que muchas personas sin experiencia opten por Arduino como herramienta de aprendizaje.
* Su circuito base ya se encuentra ensamblado lo que ahorra mucho tiempo a la hora de desarrollar el proyecto.

# BIBLIOGRAFÍA

<https://www.seas.es/blog/informatica/conoce-el-lenguaje-de-programacion-java/>

<https://www.arduino.cc>

<https://netbeans.org/downloads/8.0.1/?pagelang=es>

<http://panamahitek.com/arduino-java-facil-y-rapido/>

<https://icons8.com/icons>

<http://rhsoftperu2.blogspot.com/2016/11/programacion-java-con-netbeans.html>

<https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html>