

# **Universidad Autónoma de Baja California**



**Proyecto Final | Control de Apertura de  
Persianas  
Programable**

**Programación de Sistemas Empotrados**

**2024-1 Alumno: Norman O. Guzmán Chaboya**

**Profesora: Dra. Eloísa García Canseco**

**29 de mayo de 2024**

# Objetivo

Aplicar los conocimientos, técnicas y herramientas aprendidos y utilizados a lo largo del curso de Programación de Sistemas Empotrados para crear un dispositivo de fácil uso que permita programar la apertura y cierre de una persiana, empleando un motor de pasos.

## Introducción

La automatización del hogar es una tendencia en crecimiento debido a los múltiples beneficios que ofrece, como la comodidad, la eficiencia energética y la seguridad. El presente proyecto tiene como principal motivo implementar un sistema empotrado cuyo propósito sea automatizar algún elemento de un hogar, en este caso particular una persiana vertical, ofreciendo una alternativa de bajo costo relativo al de otros sistemas de automatización o de internet de las cosas que se encuentran actualmente en el mercado. Ultimadamente, el sistema ofrece una solución de comodidad al permitir que el usuario programe la hora de apertura y cierre de persianas, y que se controla mediante un reloj de tiempo real integrado al circuito.

## Materiales

- Tarjeta ELEGOO UNO R3: plataforma de desarrollo basada en la Arduino Uno

## Software

- Arduino IDE
- Bibliotecas especializadas para el *hardware* y actuadores utilizados en la presente práctica:
  - Wire.h
  - DS321
  - Adafruit
  - Stepper.h

## **Hardware**

- Reloj de tiempo real AT24C32
- Controlador-puente H L298 para motor de pasos
- Pantalla OLED de  $128 \times 32$  pixeles
- Adaptador-fuente de 7.5 V y 2000 mA
- Cables jumper
- Jumpers hembra-macho
- Switches de Push
- Adaptador DC
- Protoboard
- Resistencias

## **Actuadores**

- Motor de pasos NEMA 17

Tabla 1. Lista de costos para los componentes de hardware y actuadores utilizados para la elaboración del proyecto

Componente	Núm. utilizado	Precio unitario	Total por Comp.
Tarjeta ELEGOO UNO R3	1	\$ 339.00	\$ 339.00
Motor de pasos NEMA 17	1	\$ 249.00	\$ 249.00
Controlador-puente H L298 para motor de pasos	1	\$ 125.00	\$ 125.00
Reloj de tiempo real AT24C32	1	\$ 135.00	\$ 135.00
Pantalla OLED de $128 \times 32$ pixeles	1	\$ 89.00	\$ 89.00
Adaptador-fuente de 7.5 V y 2000 mA	1	\$ 170.00	\$ 170.00
Adaptador DC	1	\$ 22.00	\$ 22.00
Switches de Push	2	\$ 3.00	\$ 6.00

Componente	Núm. utilizado	Precio unitario	Total por Comp.
Resistencias	2	\$ 1.00	\$ 2.00
Cables Jumper	18	\$ 1.50	\$ 27.00
Jumpers Macho-Hembra	10	\$ 3.00	\$ 30.00
<b>Total:</b>		\$ 1,194.00	

## Funcionamiento

### Tarjeta Arduino Uno/ELEGOO UNO R3

Arduino consiste en una tarjeta para desarrollo de proyectos de electrónica y que sirve como una herramienta de fácil uso para personas principiantes en el área. Este dispositivo fue creado en 2005 por los ingenieros italianos David Cartelless y Massino Manzi.

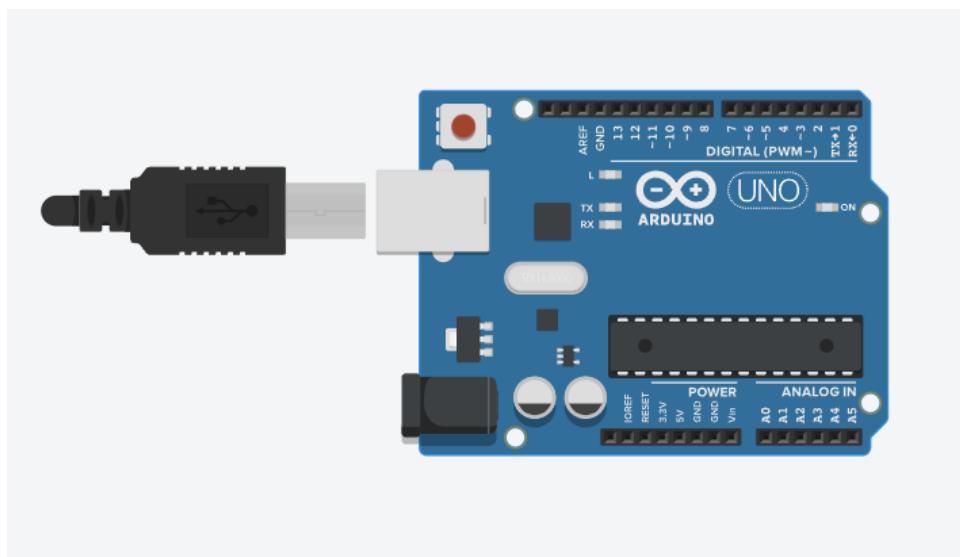


Figura 1. Tarjeta Arduino Uno

Los principales componentes de un Arduino son los que listan a continuación:

- **Microcontrolador ATMega328p:** Se trata del componente más importante de un Arduino. Es un microcontrolador de la familia AVR de aquellos desarrollados por Atmel. Este cuenta con registros y bus de datos de 8 bits, lo

que significa que se pueden enviar datos de forma paralela a través de 8 señales digitales diferentes.

- **Memoria:** 3 dispositivos diferentes a su disposición:

- **Memoria flash no volátil:** con una capacidad de 32 KB, se utiliza para almacenar los programas provistos a la Arduino, lo que permite almacenar código localmente.
- **Memoria RAM:** se utiliza para almacenar las variables utilizadas durante la sesión en ejecución.
- **Memoria EEPROM no volátil:** tiene como propósito almacenar los datos que necesitan estar disponibles tras conectar de nuevo el Arduino, sumando un total de 1KB de almacenamiento.

## Motor de pasos NEMA 17

Se trata de un motor de pasos híbrido y de 4 fases que tiene una frecuencia de giro de 1.8° o de 200 pasos por revolución. Requiere una alimentación con un voltaje nominal de 3.1 V y una corriente de 1700 mA.



Por esta razón, se optó por una fuente con una corriente de 2000 mA para asegurar el correcto funcionamiento del motor.

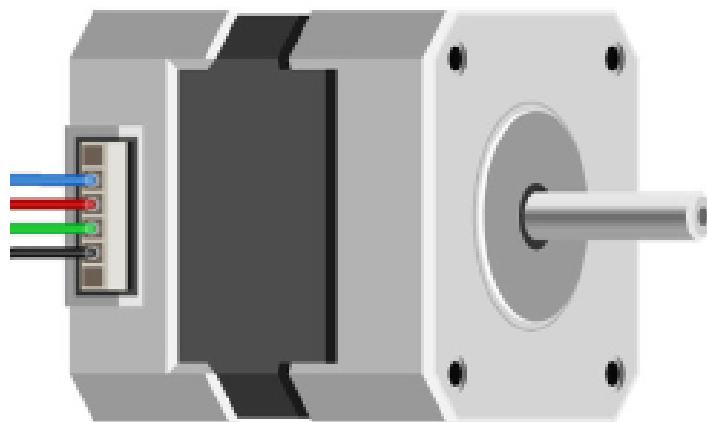


Figura 2. Motor de pasos NEMA 17

Se trata de un actuador eléctrico compuesto por dos bobinas electromagnéticas que controlan con precisión el movimiento del eje y engrane que sostiene al primero. El motor cuenta con 4 pines que se dividen en dos pares que controlan el encendido de la bobinas y el sentido de rotación de cada uno de los motores al cambiar la polaridad del circuito que posee cada una de ellas, como se describe en la Tabla 2.

Tabla 2. Lógica de bobinas en un motor de pasos

A1 o B1	A2 o B2	Dirección de Giro
0	0	Motor Apagado
1	0	Sentido de Reloj
0	1	Contrarreloj
1	1	Motor Apagado

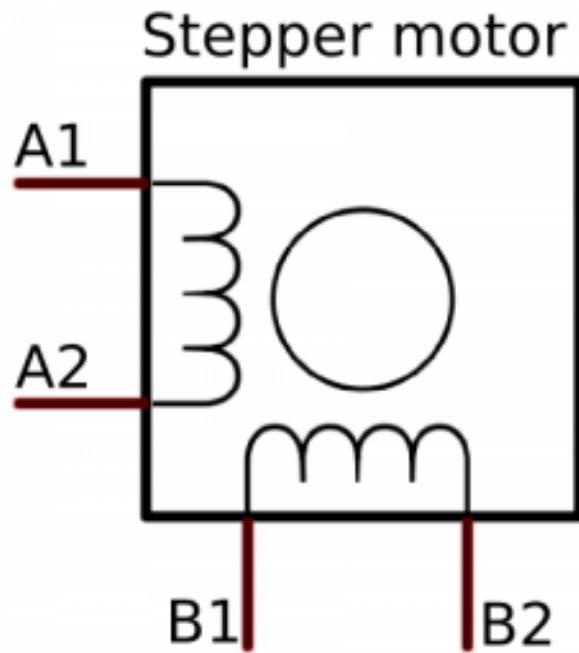


Figura 3. Diagrama de las conexiones y bobinas que componen a un motor de pasos

Para facilitar el proceso de programación del motor, se recurre al uso de controladores que centralizan la comunicación y alimentación del dispositivo.

## Controlador-puente H L298 para motor de pasos

Se trata de un *driver* que puede administrar de forma independiente dos motores de 2 fases o un solo motor de 4 fases, y que permite una alimentación de entre 5 y 35 V.

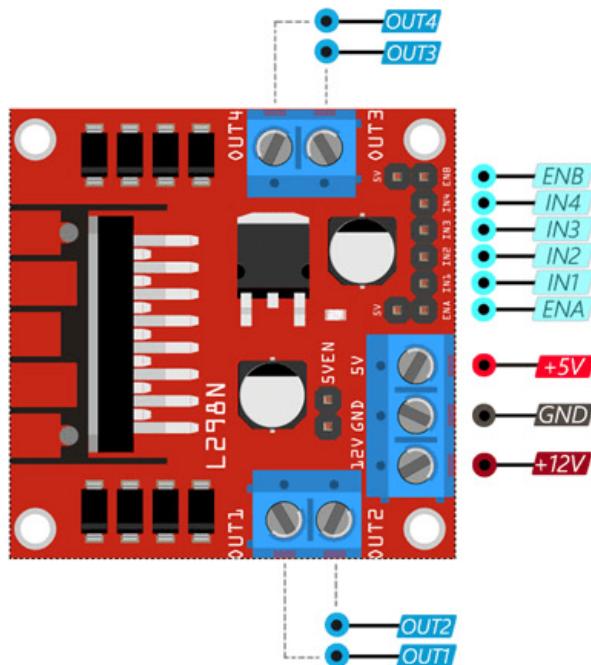


Figura 4. Diagrama de un controlador H L298 y sus nodos de conexión

Este módulo incluye un regulador de voltaje para intensidades de más de 5 V, y los conectores para entradas mayores de 12 V y para tierra.

## Reloj de tiempo real AT24C32

Es un circuito integrado que mantiene control del tiempo, basándose en la frecuencia de oscilación de un cristal contenido en el circuito. Además, utiliza una batería CR2032 como respaldo de energía.

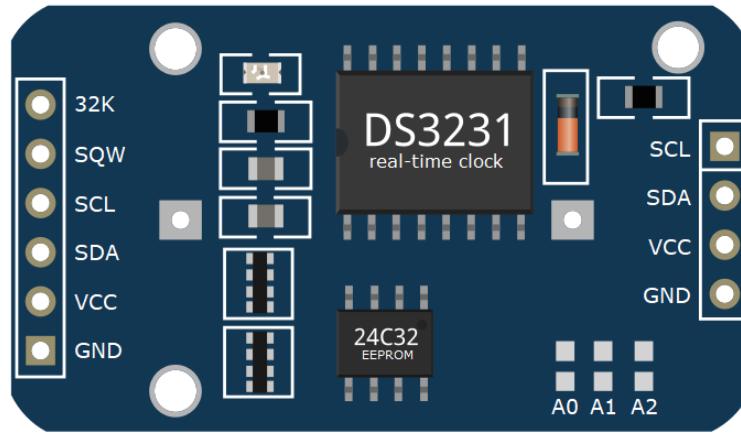


Figura 5. RTC AT24C32

El cirucito cuenta con un total de 4 pines que siguen el procolo I2C:

- **GND**: conexión a tierra del circuito.
- **VCC**: nodo de alimentación.
- **SCL (System Clock)**: nodo que mantiene al sistema sincronizado.
- **SDA (System Data)**: el puerto de flujo de datos.

## Pantalla OLED de $128 \times 32$ pixeles

Se trata de un *display* con una resolución de  $128 \times 32$  pixeles y que utiliza el protocolo IC2.



Figura 7. Display OLED de 128×32

## Desarrollo del Proyecto

Como primer paso para la elaboración del proyecto, se dispuso como prioridad integrar el módulo encargado de mantener control del tiempo real, para ello se añadió al circuito el RTC, y para inicializarlo, se utilizó el siguiente código que toma las variables de entorno de tiempo de la computadora para sincronizar el tiempo interno del RTC.

```
#include <RTClib.h>
#include <Wire.h>
RTC_DS3231 rtc;
char t[32];
void setup()
{
    Serial.begin(9600);
    Wire.begin();
    rtc.begin();
```

```

    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    //rtc.adjust(DateTime(2019, 1, 21, 5, 0, 0));
}
void loop()
{
    DateTime now = rtc.now();
    sprintf(t, "%02d:%02d:%02d %02d/%02d/%02d", now.hour(), now.minute(),
    Serial.print(F("Date/Time: "));
    Serial.println(t);
    delay(1000);
}

```

Este código utiliza la biblioteca `RTClib` que sirve para el manejo de circuitos de reloj de tiempo real, y `Wire.h` que permite la comunicación con dispositivos que utilizan el protocolo I<sup>C</sup>2.

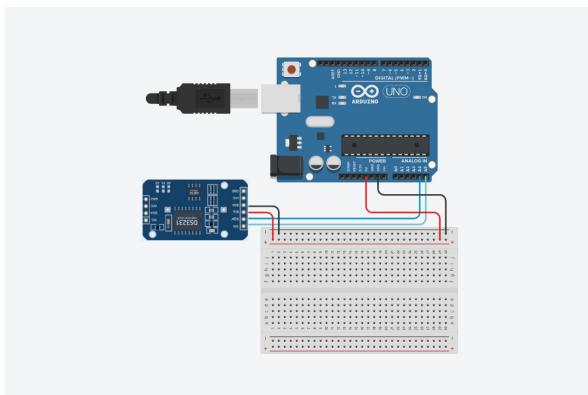


Figura 8. Diagrama del circuito que integra un RTC con una tarjeta Arduino

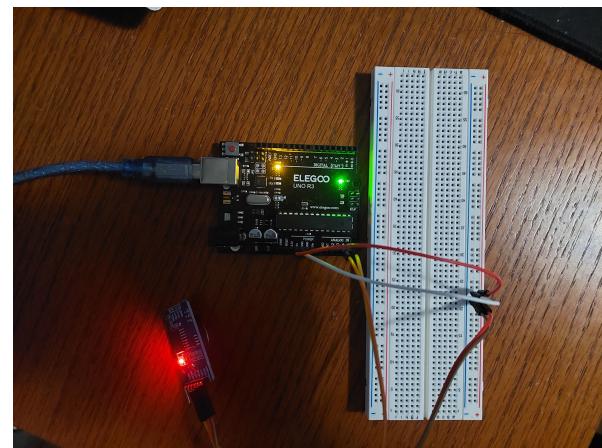


Figura 9. Circuito con RTC armado sobre una protoboard

Para integrar este módulo, se utilizaron las entradas análogas A4 y A5 para recibir el flujo de datos SDA y SCK del RTC, respectivamente, y se alimentó utilizando el pin de salida de 5 V de la tarjeta.

El siguiente módulo integrado al sistema fue la pantalla OLED, en la que se despliega la fecha actual y las horas respectivas a cierre y apertura de la

persiana. El código que se muestra a continuación muestra la función encargada de concatenar e imprimir la fecha en el *display*, para lo cual utiliza las bibliotecas de `adafruit` para controlar el despliegue en pantalla y la biblioteca `DS3231.h` para acceder a las funciones del RTC.

```
String readRTC( ) {
    String s = "";

    int hour = Clock.getHour(h12, PM);
    if (hour < 10) s += "0";
    s += String(hour, DEC) + ":";

    int min = Clock.getMinute();
    if (min < 10) s += "0";
    s += String(min, DEC) + ":";

    int sec = Clock.getSecond();
    if (sec < 10) s += "0";
    s += String(sec, DEC) + " | ";

    int date = Clock.getDate();
    if (date < 10) s += "0";
    s += String(date, DEC) + "/";

    int month = Clock.getMonth(Century);
    if (month < 10) s += "0";
    s += String(month, DEC) + "/";

    s += "20" + String(Clock.getYear(), DEC);

    return s;
}
```

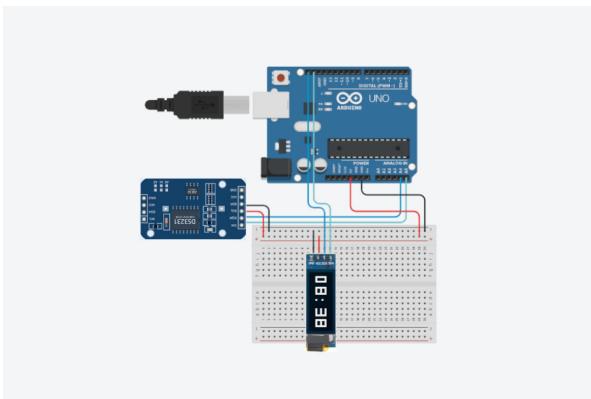


Figura 10. Diagrama del circuito que permite visualizar fecha y hora, así como los horarios de apertura y cierre de la persiana

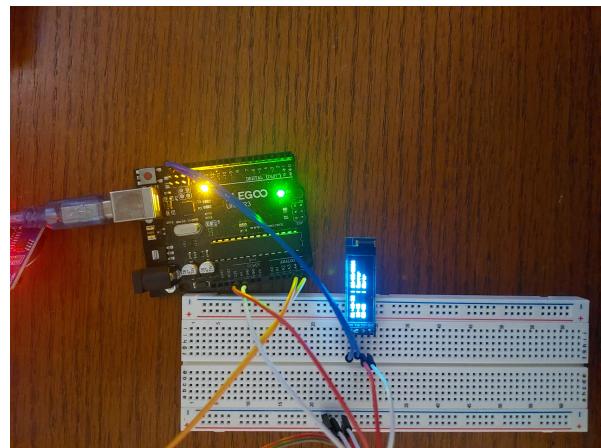


Figura 11. Circuito implementado en una protoboard

Para poder controlar la hora de cierre y apertura, se vale de dos switches de push para aumentar, en intervalos de 10 minutos, los respectivos horarios. Para la lectura de estas entradas, se utilizaron los pines 3 y 4 para apertura y cierre, respectivamente.

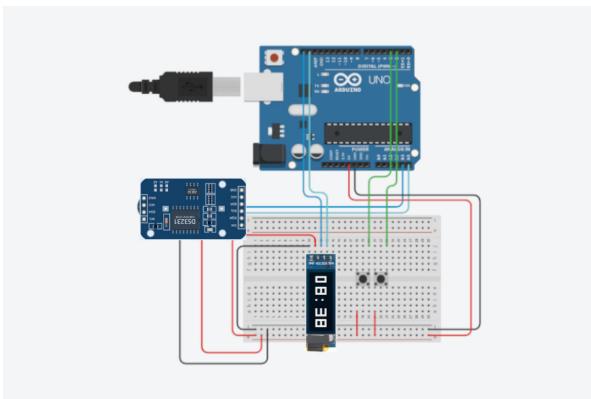


Figura 12. Diagrama del circuito que ahora permite indicar los horarios de apertura y de cierre.

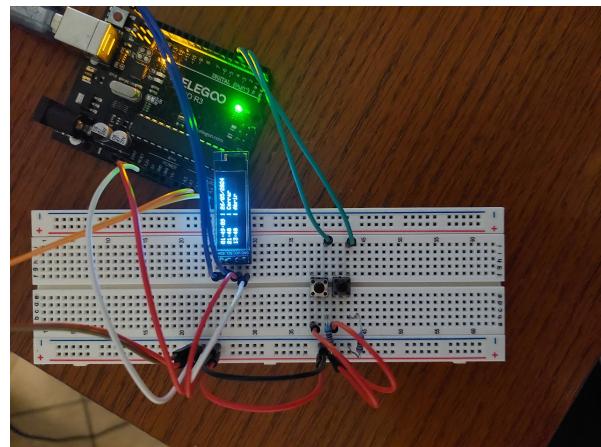


Figura 13. El mismo circuito implementado

El último módulo por integrar fue el motor de pasos y su controlador. Para hacer que el motor funcione, una fuente de 7.5 V y 2000 mA se conectó al circuito de tal manera que alimentara directamente al controlador y así obtener más de los

1700 mA de corriente necesarios para hacer funcionar al motor. Al mismo tiempo, se conectaron las entradas del motor a los nodos indicados dentro del controlador para controlar las fases de cada una de la bobinas, y se utilizaron los pines 8, 10, 11 y 12 de la tarjeta Arduino para puntear dichas entradas a través del controlador.

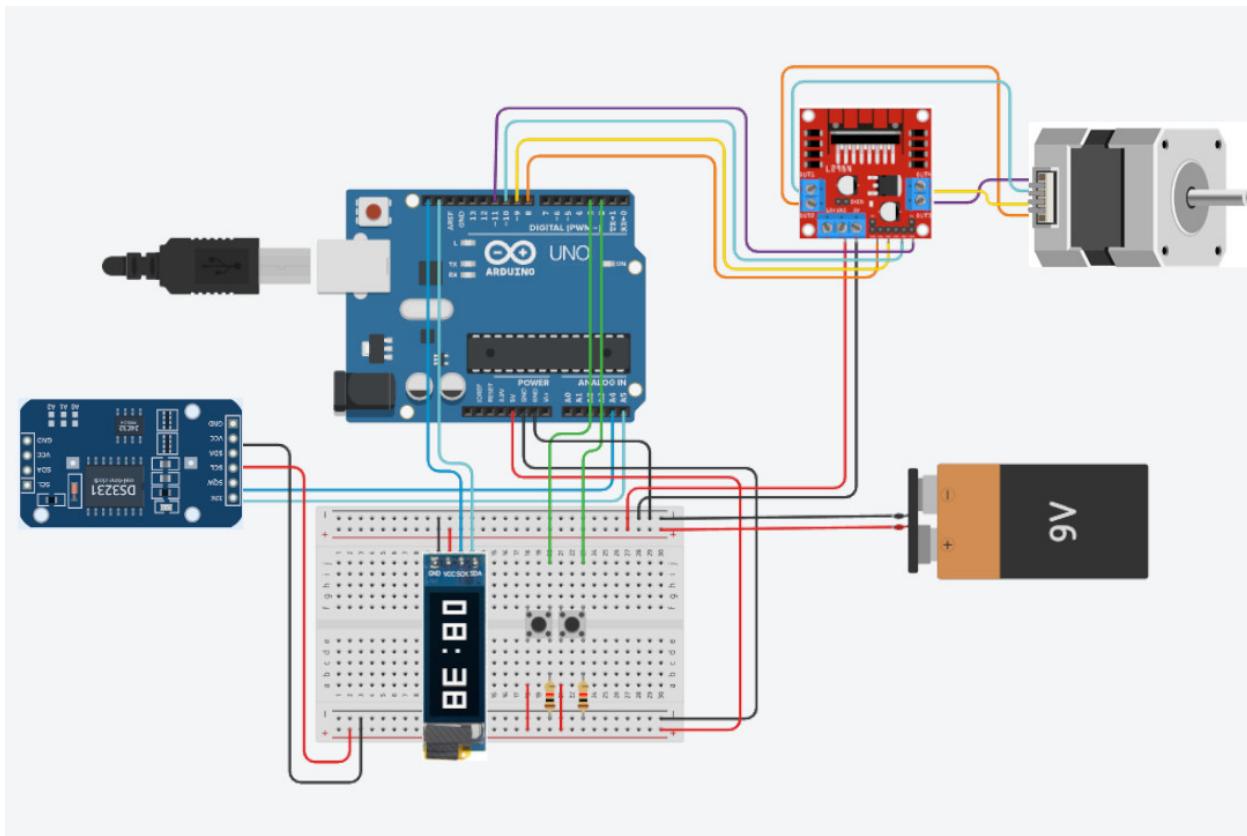


Figura 14. Diagrama del sistema empotrado que permite programar la apertura y cierre de una persiana

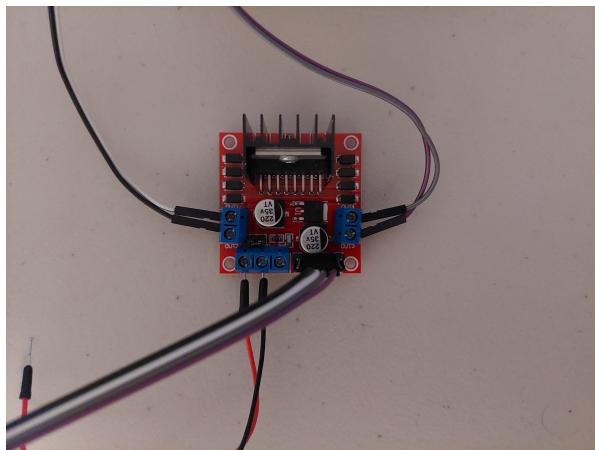


Figura 15. Conexiones del controlador del motor de pasos

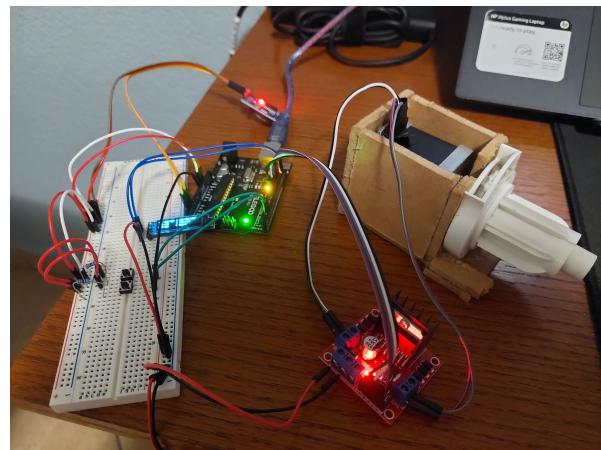


Figura 16. Controlador y motor unidos al circuito principal

Por último, la alimentación de la trajeta y el resto de circuitos se conectó a la de la fuente de 7.5 V, lo que permite al sistema empotrado funcionar autónomamente sin tener que mantener vínculo con la computadora.

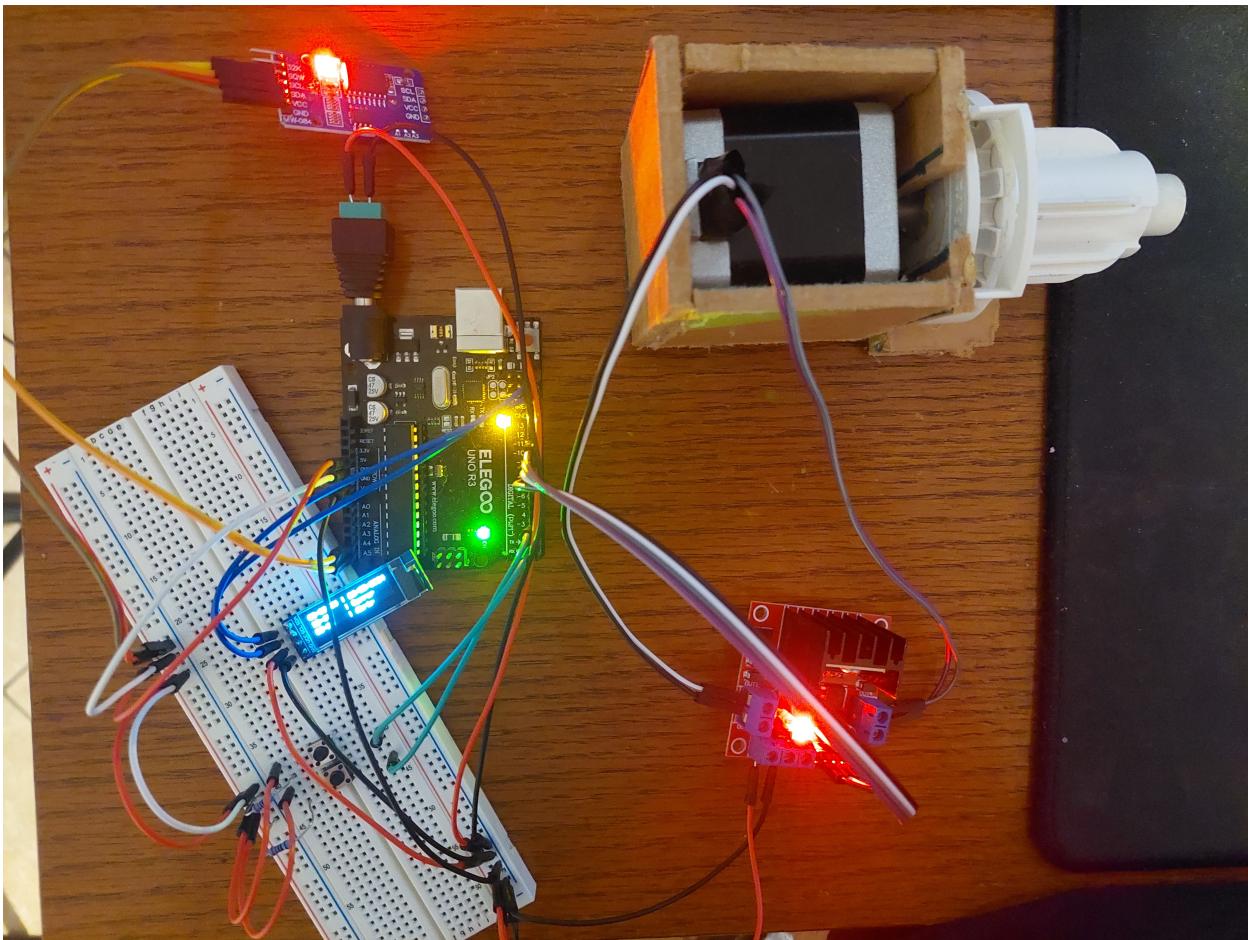


Figura 17. Vista final del sistema empotrado dónde ya se encuentra alimentado por una fuente de 7.5 V y 2000 mA

## Conclusiones

El presente proyecto fue un reto que permitió poner a prueba lo aprendido a lo largo del curso, pero también fue un impulso para conocer nuevas herramientas y métodos para aplicar en la resolución del problema aquí propuesto.

Como se mencionó reiteradamente y con anterioridad, la fuente necesaria para alimentar el circuito requería de una corriente de al menos 1.7 A, por lo que en un principio, al intentar alimentar el sistema con una batería de 9 V, la corriente resultaba insuficiente para mantener al motor en funcionamiento. De la mano con esto, también surgieron problemas con la pantalla OLED, ya que ocasionalmente esta terminaba congelandose indefinidamente tras unos minutos de ejecución. Es muy probable que el error haya surgido por la falta de una fuente de alimentación

adecuada, ya que al cambiar la fuente de 5 V utilizada en un principio por la de 7.5 V, el error ya no volvió a aparecer.

Por otro lado, tras terminar el proyecto y analizar de nuevo las ofertas presentes en el mercado de características similares a aquella propuesta aquí, se hizo evidente que el costo final de esta resultó ser demasiada elevado en comparación con las ya existentes. La principal razón de esto es el hecho de que el circuito funciona con una Arduino y no una tarjeta o circuito dedicado para controlar los módulos del sistema. Además, el motor utilizado excede por mucho el torque/fuerza necesaria para hacer girar el eje que permite el cierre/apertura de la persiana; un motor más débil y de menor costo sería suficiente para este sistema.

---

## Referencias

1. HiBit (2020). *How to use the L298N motor driver module*. Recuperado de: <https://www.hibit.dev/posts/89/how-to-use-the-l298n-motor-driver-module>
2. Mallari, J. (2022). *HOW TO USE A REAL-TIME CLOCK MODULE WITH THE ARDUINO*. Recuperado de: <https://www.circuitbasics.com/how-to-use-a-real-time-clock-module-with-the-arduino/>
3. Components 101. *Nema 17 Stepper Motor*. Recuperado de: <https://components101.com/motors/nema17-stepper-motor>
4. Zmabetti, N., Soderby, K., Hylén, J. (2024). *Inter-Integrated Circuit (I2C) Protocol*. Recuperado de: <https://docs.arduino.cc/learn/communication/wire/>