Programmed Control with Arduino

Activity 8: The Relay (Magnetic Power Control)

Up until now, all of our activities have controlled small LEDs that consume very little current. However, in the real world, automation systems must activate motors, heaters, or light bulbs that operate at high voltages. Today, we will learn how to use a **relay**, a mechanical switch controlled by magnetism that allows us to govern large electrical loads safely and in an isolated manner.

Learning Objectives

By the end of this session, you will be able to:

- Understand the electromechanical operating principle of a relay.
- Physically differentiate the terminals of a relay: Coil, Common (COM), Normally Open (NO), and Normally Closed (NC).
- Design a circuit with galvanic isolation, separating the control stage (low power) from the power stage.
- Program timing sequences for the activation and deactivation of a magnetic relay.

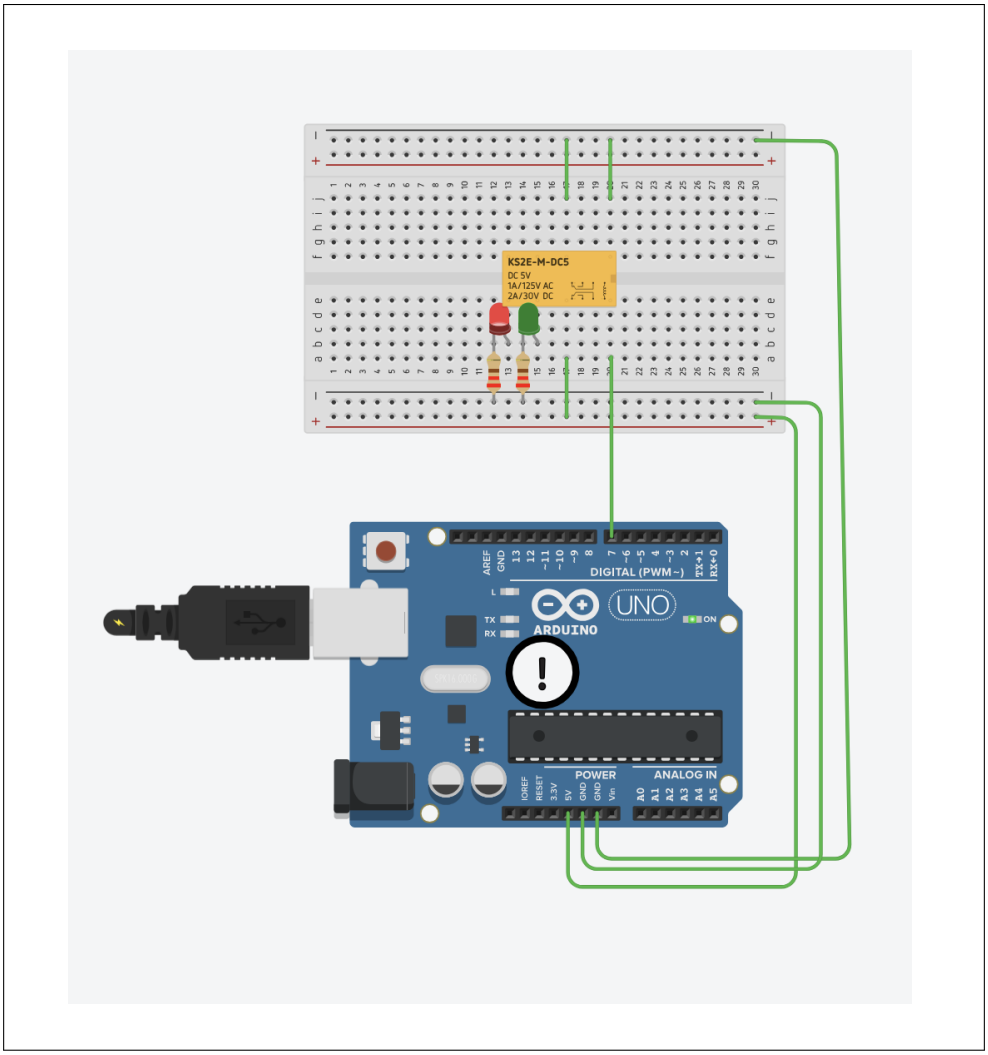
Required Components

Find and place the following components on your TinkerCad workspace:

- 1 Arduino Uno board.
- 1 Small breadboard.
- 1 5V SPDT Relay (listed as “Relay SPDT” in TinkerCad).
- 1 LED (to simulate the load we are going to turn on).
- 1 $220\,\Omega$ resistor (LED protection).
- Virtual jumper wires.

Breadboard Assembly Diagram

The relay acts as a physical switch. The current coming out of the Arduino is only used to power an internal electromagnet (coil). The circuit that lights up the LED is completely independent.



Base Code: Intermittent Activation

Write the following program in your TinkerCad text editor. You will see that programming a relay is identical to programming an LED: we just need to send a HIGH or LOW digital signal through the corresponding pin.

```
// Assign Digital Pin 7 to control the relay coil
int relayPin = 7;

void setup() {
  % The relay pin works as an OUTPUT (supplies power to the coil)
  pinMode(relayPin, OUTPUT);
}

void loop() {
  // STATE 1: Activate the relay (attracts the mechanical contact toward 'NO')
  digitalWrite(relayPin, HIGH);
  delay(3000); // Keep the relay active for 3 seconds

  // STATE 2: Deactivate the relay (the internal spring returns the contact to '
    NC')
  digitalWrite(relayPin, LOW);
  delay(3000); // Keep the relay inactive for 3 seconds
}
```

Listing 1: Code to cycle a relay on and off.

How does the code work?

When you start the simulation in TinkerCad, you will hear a visual click and see the internal needle of the relay change position:

- **Relay Logic**: When executing `digitalWrite HIGH`, pin 7 sends 5V to the relay coil. This generates a magnetic field that attracts the metallic contact of the **Common (COM)** terminal, mechanically connecting it to the **Normally Open (NO)** terminal. At that instant, current flows to the LED and it turns on.
- **The Return Spring**: When executing `digitalWrite LOW`, the coil is deprived of electricity, the magnetism disappears, and an internal spring pushes the metallic contact back to the **Normally Closed (NC)** terminal, instantly turning off the LED.

Electrical Protection in the Real World

Even though we connect the relay directly to the Arduino pin in the TinkerCad simulator, in real circuits you must **never** power a relay coil directly from a microcontroller pin. Coils consume more current than an Arduino can safely provide and generate high-voltage spikes when turned off. In a real assembly, a transistor is used as an intermediate switch, along with a protective diode connected in parallel (known as a flyback or freewheeling diode).

Activity 8 Challenge

Master the use of electromechanical switches by solving the following practical challenges:

1. **Dual Status Indicator**: Add a second LED (for example, a Red one) to the circuit. Make the necessary connections so that the Green LED is wired to the **NO** terminal and the Red LED is wired to the **NC** terminal. Without changing the base code, observe what happens when you start the simulation. How do the LEDs behave?
2. **Remote Power Switch**: Integrate a pushbutton (like the one used in Activity 3) into your TinkerCad circuit. Modify the code so that the relay remains inactive (off) by default and **only activates continuously while you hold the pushbutton down**.
3. *Question for reflection*: Why do you think this component is associated with the concept of "galvanic isolation"? What would happen if the power stage circuit suffered a severe short circuit? Would the Arduino board be affected?