UNIT 9: RELAYS

AIMS

Purpose of this unit is to grasp the concept of a relay; we will examine their basic types, their principle of operation, and we will take a brief introduction on their usage with the Arduino microcontroller board.

THEORY SECTION

- WHAT IS A RELAY (PRINCIPLE OF OPERATION)?
- BASIC RELAY TYPES AND CHARACTERISTICS
  - Electromechanical relays
  - Solid-state relays
  - Relay contacts and terminals
  - Pole and throw
- THE DRIVING CIRCUIT
  - Power (current and voltage) requirements
  - The transistor switch
  - Protection features
  - Ready-made modules

PRACTISE SECTION

- EXAMPLE 1: Operating a buzzer through an electromechanical relay.
- EXAMPLE 2: Operating a heater coil through a solid-state relay.

REQUIRED MATERIALS

- A desktop or laptop computer.
- The Arduino IDE: this should include the supplementary material already installed and configured.
- An Arduino UNO microcontroller board.
- A USB cable.

Table of Contents

THEORY SECTION

1. WHAT IS A RELAY (PRINCIPLE OF OPERATION)

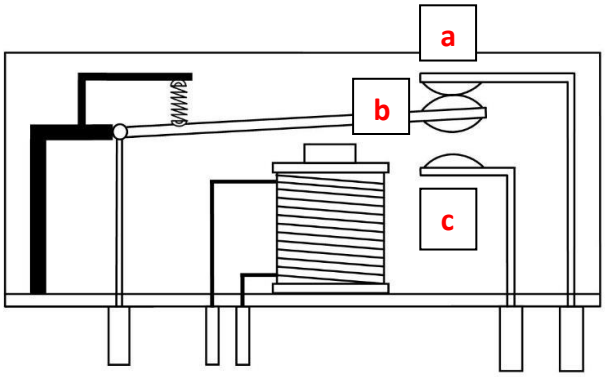
Relays are electrically operated switches. They are used where it is necessary to control a high-power circuit by a separate lower-power signal (typically isolating the two) or where several circuits must be controlled by one signal.

2. BASIC RELAY TYPES AND CHARACTERISTICS

Relays come in many forms and variations (e.g. “mini” –or PCB- relays, automotive relays, industrial relays, etc.) but the major families, characterized by the type of switch used, are only two: electromechanical relays, and solid-state relays.

A. Electromechanical relays

Electromechanical relays employ an electromagnet and a movable armature with one or more set of contacts. One contact is affixed to the armature, which allows it to move with it, whilst the other contact stays in a fixed position. See the figure below; parts (a) and (c) illustrate the fixed contacts, whilst part (b) is the contact that moves with the armature.



When there is adequate power applied to the electromagnet / coil, it generates a magnetic field which forces the armature to switch position, linking the contacts together, or breaking contact, depending on the relays’ construction.

When the electromagnet is de-energized, a spring that was tensioned in the previous phase moves the armature back to its’ initial position.

Due to the nature of electromechanical relays, (e.g. numerous moving parts) they have a rather restricted lifespan, (usually a few tens of thousands of ‘clicks’) and a rather big actuation latency, which makes them inappropriate for certain applications.

B. Solid-state relays

Solid-state relays, on the other hand, employ solid-state switches, like MOSFETs, TRIACs or SCRs, depending on the nature of the load (whether it is AC, or DC).

This lack of moving parts results in a greatly improved long-term reliability and lifespan, and there is also no contact bounce, which is common with electromechanical relays. A significant downside of this design is that semiconductor-based switching devices tend to produce heat. Excess heat can be damaging to the part itself; thus, heatsinking is usually required.

C. Relay contacts and terminals

Simple electromechanical relays that utilize a plain coil for their operation have no polarity requirements. As of this, their two coil terminals (typically labelled A1 and A2) can be used interchangeably. A few electromechanical relays, however, being more sophisticated, include additional circuitry on their inputs. These relays have polarity requirements that must be met.

On the loads' side, a "common" terminal exists (usually labelled 'COM', or '11') which is the terminal that is affixed to the armature. This terminal is always connected either to the load, or to the power source, depending on the circuit design.

The remaining terminals are described as "normally-open" (having no connection to the common terminal when there is no power running through the coil) and "normally-closed" (being connected to the common terminal when there is no power running through the coil). They are usually labeled as "NO", or "14" for the normally-open terminal, and "NC" or "12" for the normally-closed one.

Solid-state relays, on the other hand, since they are built upon semiconductor-based elements (such as LEDs or opto-couplers) have strict polarity requirements. On these relays, the low-side input may be labelled as plus (+) and minus (-), plus (+) and ground (GND), or similarly. On the load-side, they are very similar with their electromechanical counterparts.

Several standards (such as EN 50005) exist for labelling relay terminals, which try to standardize labelling.

A relay may have just two load-side terminals (either common and NC, or common and NO) or more. The latter is commonplace with relays having 'sets' of switches, instead of a single switch.

D. Pole and throw

With relays being switches, the terminology applied to switches can also be applied to relays; a relay switches one or more "poles", each of whose contacts can be "thrown" by energizing the coil.

A few common relay types include;

SPST-NO or SPST-NC	Single-pole, single-throw. These kinds of relays have a single normally-open (or normally-closed) contact.
SPDT	Single-pole, double-throw. This kind or relay has a common terminal that is connected to either of the other two terminals, but never to both at the same time.
DPST	Double-pole, single-throw. This is equivalent of two SPST switches actuated by a single coil.
DPDT	Double-pole, double-throw. This is equivalent of two SPDT switched actuated by a single coil.

The "S" (which stands for single) or "D" (which stands for double) notation for the pole count may be replaced with a number, indicating multiple contacts connected to a single actuator. For example, there are 3PDT relays available (with 3 poles and double throw).

3. THE DRIVING CIRCUIT

Relays have special driving requirements, and, unlike a LED or other similar discrete device, cannot be interfaced directly with a microcontroller board. This holds truer for electromechanical relays, with solid-state ones being easier to drive.

A. Power (current and voltage) requirements

The coil embedded inside electromechanical relays draws a considerable amount of power (voltage × current) to get energized and produce the magnetic field that 'throws' the switch.

Relays with coils rated for 5 V are the most common, whilst several others with even higher ratings (e.g. 12/24 V DV or 115/230 V AC) exist, for industrial, automation or automotive applications.

Apart from the voltage rating, the energy source must also be able to deliver to correct amount of current to the coil. Smaller, user-friendlier relays (like the 5V DC "mini" PCB-relays that we use with the Arduino) can be powered directly from the Arduino's voltage regulator, as they only require a handful of tens of milliamperes.

Typically, the datasheet of the relay will state the required current for the part to work. If that is not the case, just measure the coils' resistance with a multimeter and then use Ohm's law to calculate the required current (we already know the voltage rating, remember?).

B. The transistor switch

An NPN transistor is –typically- used in a common-emitter configuration to energize the relay. A small signal (most of the times less than 1mA of current) biases the base of the transistor and this allows for a larger current to pass through it.

For smaller relays that do not draw a significant amount of current a general-purpose small signal transistor (like the infamous 2N2222a) can be enough. For 'bulkier' and more 'power-hungry' relays, however, you may need to form a Darlington pair to drive them.

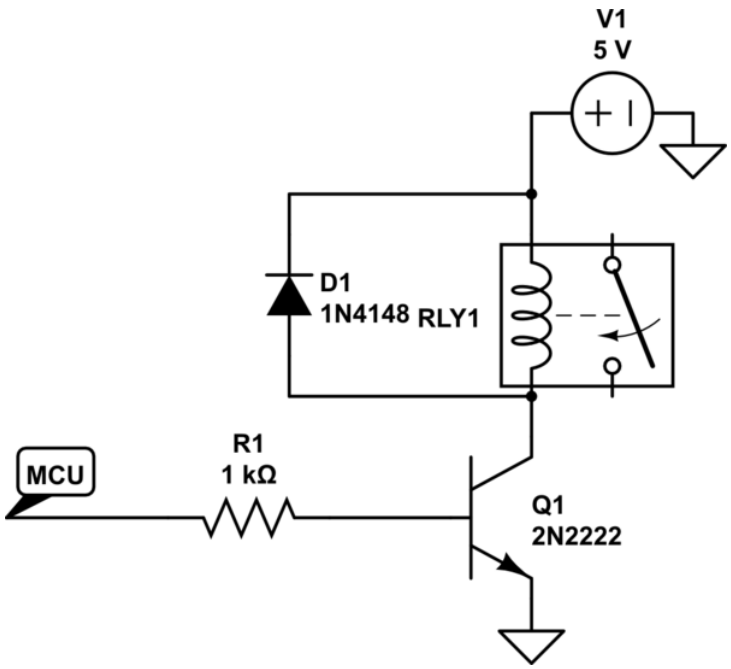
C. Protection features

A coil that deenergizes will use its inductance to try to maintain the current running. Adding a diode (called a "flyback" diode, or a "freewheeling" diode) will provide a path for the current to circulate until the relays' inductance has lost its energy.

This is especially important when driving the relay with a transistor, since transistors cannot handle the sudden voltage spike that is generated without taking damage to themselves.

An "off-the-shelf" diode (like the 1N4004, or the "beefier" 1N4148) is OK.

See the figure below for the complete circuit that drives the relay, making use of a NPN bipolar-junction transistor, as well as a freewheeling diode;



D. Ready-made modules

Luckily, an assortment of ready-made modules exists for driving relays with the Arduino that ease the process. These provide the necessary protection features (the diode) as well as handy LEDs and jumpers. In this course we will be using this kind of modules instead of implementing the transistor-diode driving circuit ourselves.

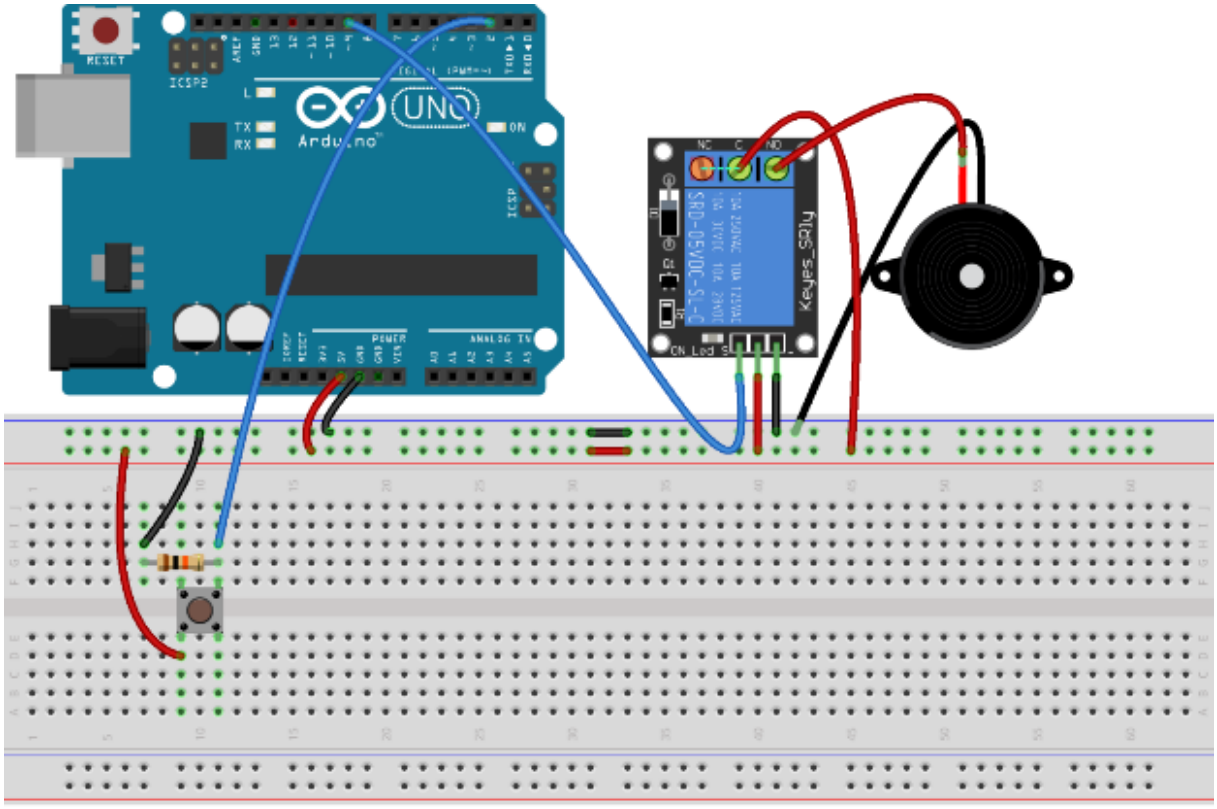
PRACTISE SECTION: EXAMPLES

4. EXAMPLE 1: Operating a buzzer through an electromechanical relay

In this example, we will operate a buzzer through an electromechanical relay. You have already used piezoelectrical buzzers in previous units, right?

Instead of directly interfacing the buzzer with the Arduino, we will place the relay in between.

Follow the circuit diagram below to implement the circuit;



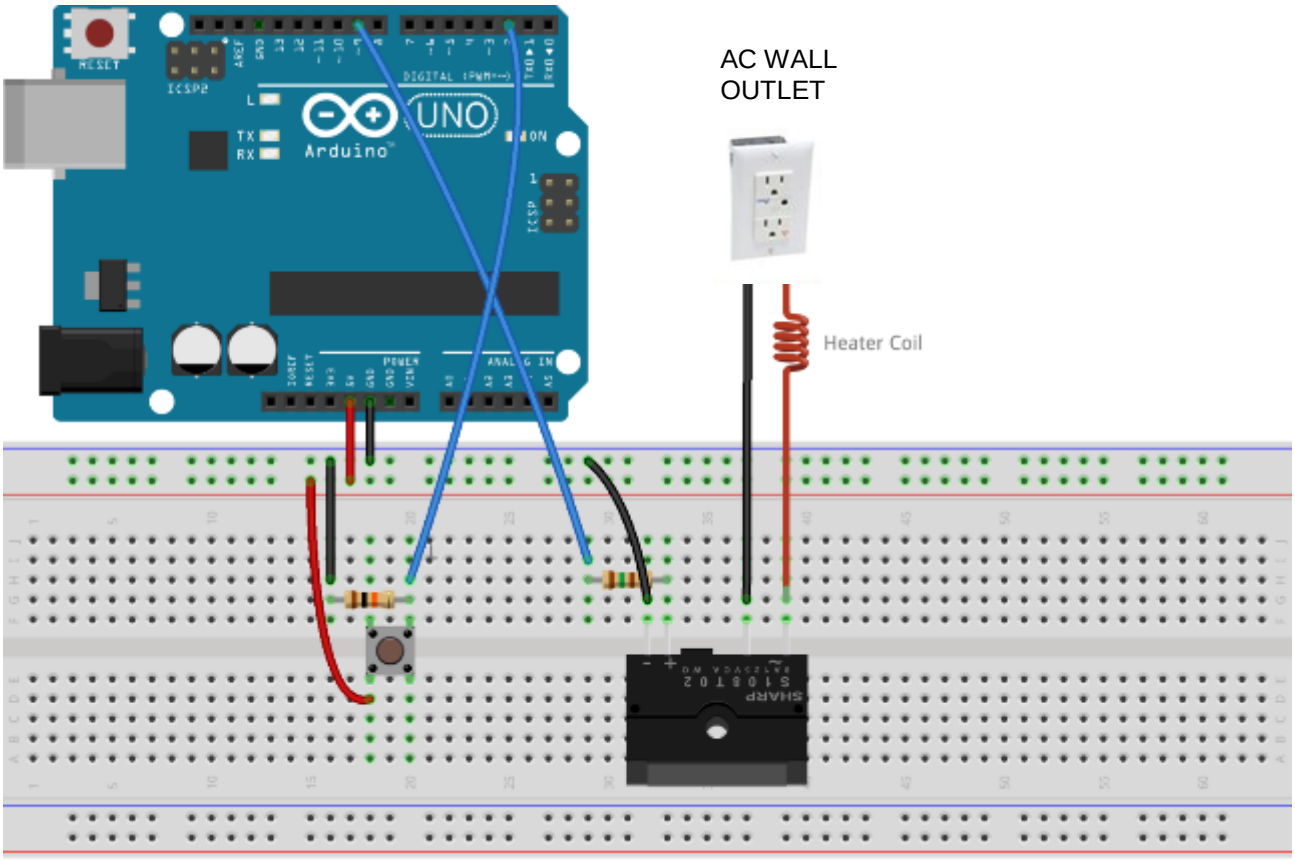
Pressing the button shall activate the relay, and the buzzer shall sound with its characteristic beeping sound. Interfacing the relay to the Arduino using the ready-made module illustrated above is as easy as interfacing an LED, (or even easier, since there are no resistors needed) and operating it is as easy as writing HIGH to the appropriate pin.

5. Example 2: Operating a heater coil through a solid-state relay.

We are now going to operate a high-powered heater coil (or whatever AC device you may have available) using a solid-state relay. The basic principle and circuit design remains the same, albeit with a slightly different driving circuit.

Be extra cautious around mains voltages, as they can be dangerous, or even lethal. That cannot be stressed out enough. Luckily, the solid-state relay has an embedded LED to give us feedback on its state (whether it is active or not). If you prefer you can avoid mains voltages and rely on this LED, instead.

As mentioned before, solid-state relays, due to their nature, can be interfaced with microcontrollers easily. Consult the figure below to implement your circuit;



Relays- Optocoupler

If you have used the same GPIO pins, there will be no code changes required to operate the solid-state relay. You should be able to trigger the relay simply by pressing the push-button.

REFERENCES

BOOKS

[1]. Practical Electronics for Inventors, Third Edition, Paul Scherz and Simon Monk

WEBSITES

[1]. https://www.arduino.cc/
[2]. https://electronics.stackexchange.com/
[3]. https://www.petervis.com/