Programmed Control with Arduino

Activity 10: The Automated System with Data Return

You have reached the final stop of our learning journey! In this final activity, we will integrate practically everything you have learned so far. We will design an intelligent safety system. To structure the code flawlessly, we will create a diagnostic function that evaluates the state of a pushbutton and returns a logical value (`return`) to the main loop, determining whether to activate a power relay and warning LEDs.

Learning Objectives

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

- Design custom functions that process information and return logical values using the `return` statement.
- Understand data return types in C++ (specifically, the boolean type `bool`).
- Integrate multiple input and output components (pushbutton, signaling LEDs, and a power relay) into a single complex circuit.
- Develop the logic for an industrial emergency stop safety system.

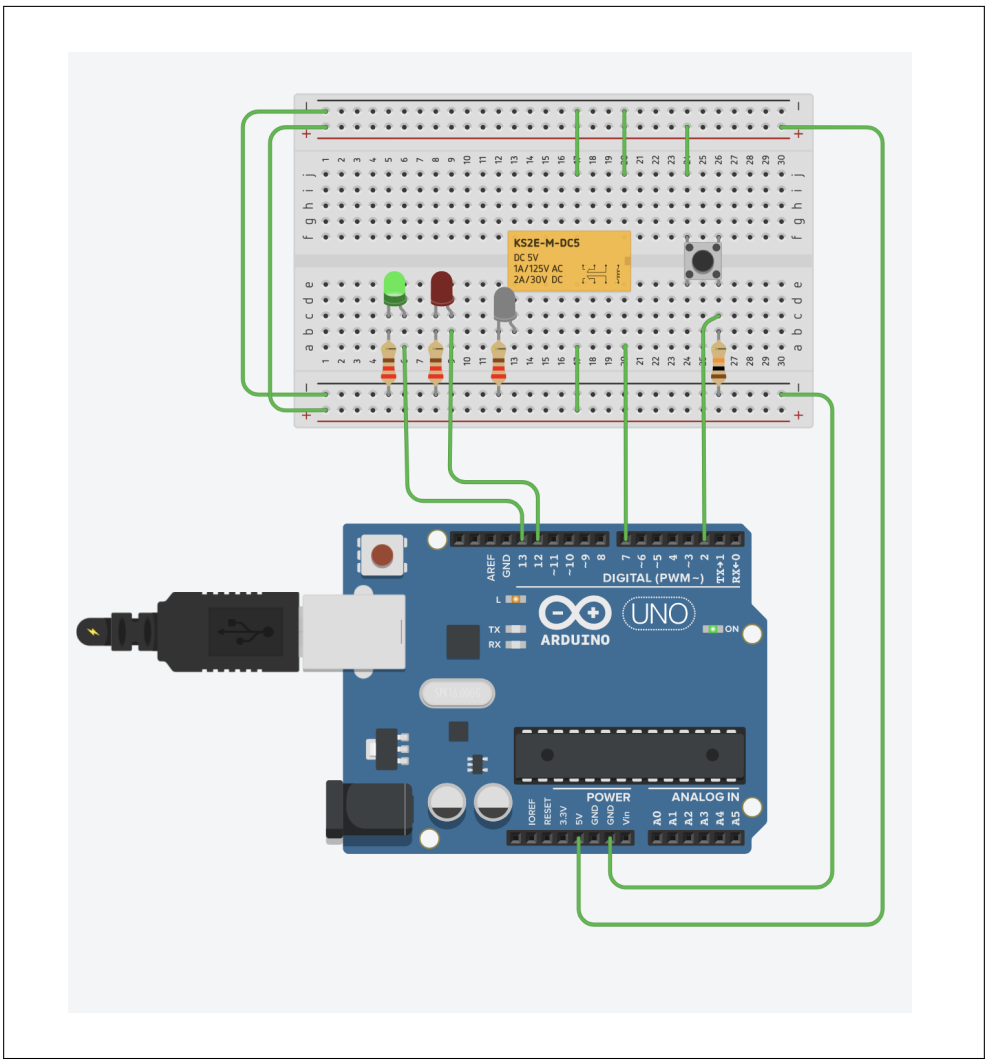
Required Components

This is the most comprehensive assembly of the course. Place the following components on your TinkerCad simulator:

- 1 Arduino Uno board.
- 1 Small breadboard.
- 2 LEDs (1 Green for "System Safe" and 1 Red for "Danger").
- 2 $220\,\Omega$ resistors (to protect the LEDs).
- 1 Four-pin pushbutton (Emergency button).
- 1 $10\,\text{k}\Omega$ resistor (for the button's Pull-Down configuration).
- 1 5V SPDT Relay (to simulate cutting off power to industrial machinery).
- Virtual jumper wires.

Breadboard Assembly Diagram

Organize the wiring patiently. Remember that the relay coil is activated via an Arduino pin, while its contacts govern power completely independently.



Base Code: Functions with a 'bool' Return Type

Write the following program in TinkerCad. Notice that the data type of our function is no longer `void`, but `bool`, which means it will return a value of true (`true`) or false (`false`) using the `return` command.

```
// Activity 10: Industrial safety system with status return
int buttonPin = 2;
int relayPin = 7;
int greenLed = 13;
int redLed = 12;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(redLed, OUTPUT);
}

void loop() {
  // 1. FUNCTION CALL: We evaluate the button and save the result
  bool alarmActive = evaluateSafety(buttonPin);

  // 2. ACTIONS BASED ON THE FUNCTION'S RETURN VALUE
  if (alarmActive == true) {
    // If the alarm is active, trigger the relay and turn on the red LED
    digitalWrite(relayPin, HIGH);
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
  } else {
    // If everything is safe, the relay rests and the green LED shines
    digitalWrite(relayPin, LOW);
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
  }
}

% 3. DEFINITION OF THE FUNCTION WITH A RETURN VALUE
% We state 'bool' at the beginning because the function must return a boolean
bool evaluateSafety(int sensorPin) {
  int reading = digitalRead(sensorPin);

  // If a press is detected (danger detected)
  if (reading == HIGH) {
    return true;  // Return the value 'true' to the loop()
  } else {
    return false; // Return the value 'false' to the loop()
  }
}
```

Listing 1: Code using a function with a data return to evaluate system status.

How does the code and the `return` statement work?

Until now, all of our custom functions began with the word `void` because they only performed physical actions. However, in programming, it is vital for functions to perform calculations or readings and "return" a response.

- **Return type (`bool`)**: In the function header, we replace `void` with `bool`. This forces the function to return a boolean value (true or false) before exiting.
- **The `return` statement**: This is the exit door of the function. It sends the chosen value back to the exact line where the function was called. As soon as the program reads `return`, the function terminates immediately.
- **Receiving the data**: In the line `bool` , the Arduino executes the function and replaces that entire block with the returned result (`true` or `false`), saving it directly into the local variable .

Activity 10 Challenge: The Final Desafío!

Consolidate your knowledge by completing this ultimate structured programming safety challenge:

1. **Timed Acoustic Alarm**: Add an emergency blinking pattern. If the function returns `true`, make the Red LED blink rapidly and continuously while the Green LED remains completely off.
2. **Numeric Return Function**: Design a new diagnostic function named `checkErrors()` that does not return a boolean value, but an integer (`int`). If the system is safe, it should return `0`. If the system is triggered, it must count the seconds it remains active and return that numeric value to the `loop` to track the response delay time.
3. *Question for reflection*: Congratulations on completing all 10 activities! Write a brief personal reflection on how your programming skills have evolved from Activity 1 (where everything was purely sequential) to this Activity 10 (using functions with parameter passing and return values). How does this structure help you think like a software engineer?