Spaceship Interface using Arduino

1. Detailed Project Description

This project demonstrates the design and implementation of a simple spaceship control interface using an Arduino microcontroller, three LED indicators, and one push button. The purpose of this exercise is to simulate a basic spacecraft status system, where different visual signals represent normal operation and alert conditions.

The system operates in two modes. In normal mode, a green LED remains active, indicating stable operation. When the push button is pressed, the system transitions into alert mode, where two red LEDs blink alternately, simulating a warning or emergency signal.

This project introduces fundamental concepts of digital input reading, output control, conditional logic, and time-based LED blinking using delay functions.

2. How the System Works

The push button is connected to a digital input pin on the Arduino. The system continuously reads the state of this input using the digitalRead() function.

When the button is not pressed, the input state is LOW, and the Arduino activates the green LED while keeping the red LEDs turned off. This represents the normal operational state of the spacecraft.

When the button is pressed, the input becomes HIGH. The Arduino then switches to alert mode. In this state, the green LED turns off, and the two red LEDs begin blinking alternately with a delay of 250 milliseconds between state changes. This creates a visual emergency effect.

The continuous execution of the loop() function ensures real-time system response.

3. Circuit Description

Arduino Board (Uno or Mega): The central controller that processes input and controls the LEDs.

Breadboard: Used for assembling the prototype without soldering.

Push Button: Acts as the control switch between normal and alert modes.

Three LED Diodes:

- One Green LED for normal operation.
- Two Red LEDs for alert signaling.

Resistors (220Ω): Used to limit current through each LED.

Pull-down Resistor (10kΩ): Ensures stable LOW reading when the button is not pressed.

Jumper Wires: Used for electrical connections.

USB Cable and Computer: Used for power supply and programming.

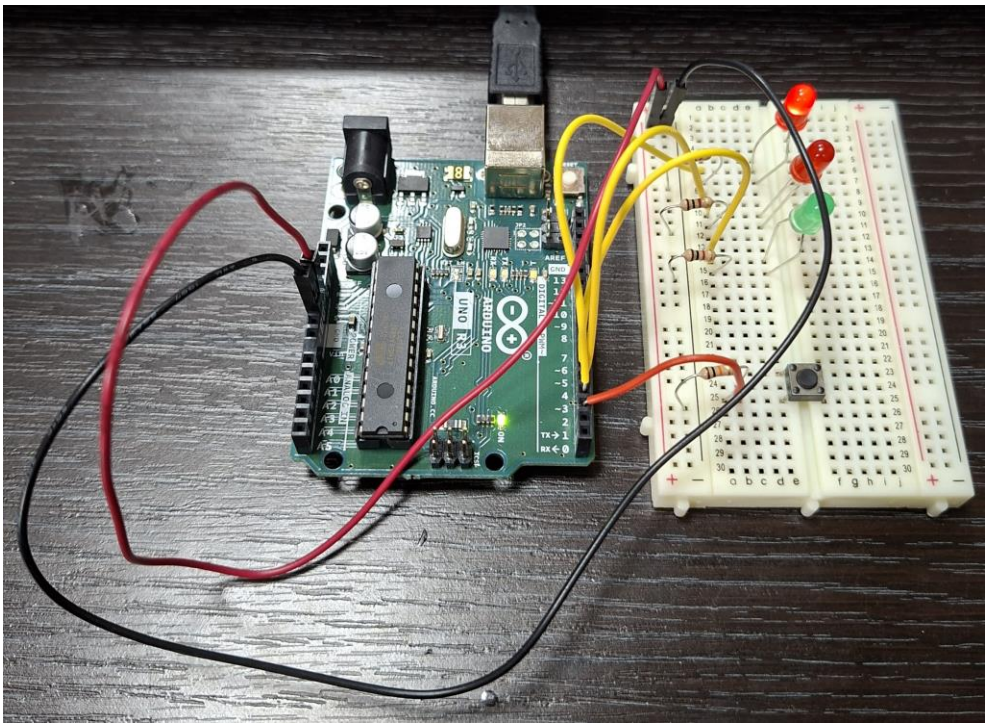

Figure 1. Implementation of the Spaceship Interface circuit on a breadboard using Arduino, push button, and three LEDs

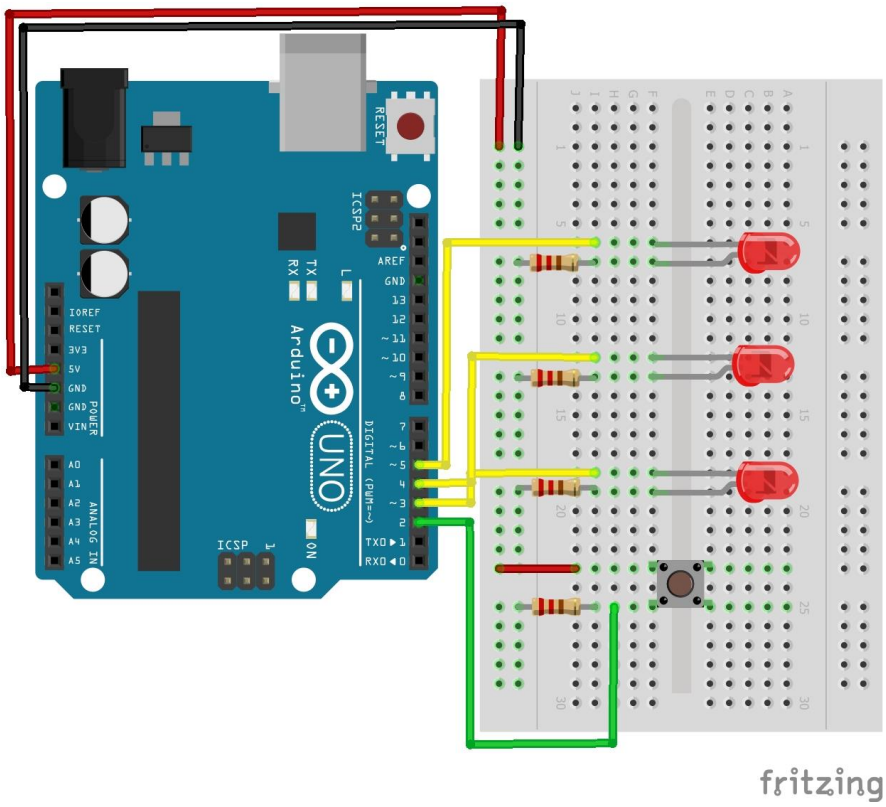

Figure 2. Circuit diagram of the Spaceship Interface created in Fritzing

4. Code

The following code controls the LED states based on the button input:

```
int switchState = 0;

void setup() {
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
}

void loop() {

  switchState = digitalRead(2);

  if (switchState == LOW) {
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
```

```
    digitalWrite(5, LOW);
  }
  else {
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);

    delay(250);

    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);

    delay(250);
  }
}
```

5. Software Implementation

The program uses a digital input to detect the state of the push button. Conditional statements determine which LEDs should be active.

In normal mode:

- Green LED is ON.
- Both red LEDs are OFF.

In alert mode:

- Green LED is OFF.
- Red LEDs blink alternately.
- delay(250) creates a visible blinking interval.

This implementation demonstrates event-driven behavior in embedded systems. The logic structure ensures that the system constantly monitors input and reacts immediately to user interaction.

6. Educational Value

Through this project, students learn:

- How to configure digital input and output pins in Arduino.

- How to use conditional statements to control system behavior.
- How to implement blinking effects using time delays.
- How to simulate real-world status systems using simple electronic components.
- The importance of structured programming in embedded systems.

This exercise strengthens understanding of digital logic and microcontroller-based control systems.

7. Conclusion

The Spaceship Interface project successfully demonstrates how a microcontroller can manage system states and visual indicators based on user input. By switching between normal and alert modes, students gain practical experience in digital control and embedded system design.

This laboratory exercise is fundamental for beginners in electronics and programming, as it combines theoretical knowledge with practical implementation. It also provides a foundation for more advanced projects involving sensors, alarms, and automated control systems.