FSM Project – Mode Selector

Objective

The objective of this project is to design and implement a Finite State Machine (FSM)-based Mode Selector using Arduino.

The system cycles through different operational modes each time a push button is pressed.

Theory

What is a Mode Selector?

A mode selector allows a user to switch between multiple predefined modes of operation using a single button or input source.
This is a very common concept in washing machines, ovens, embedded devices, drones, and robotics, where one input cycles through several modes.

FSM Approach

Instead of writing complex conditional logic, we use an FSM:

- States: Each mode is represented as a state.
- Transitions: Triggered by button presses.
- Output: LED patterns (or other actuators) represent the selected mode.

FSM Design

States (Modes)

We design 4 modes (expandable as needed):

- S0_IDLE: All LEDs OFF (system waiting).
- S1_BLINK: LED1 blinking.
- S2_RUN: LEDs chasing in sequence (like running light).
- S3_ON: All LEDs ON.

Input

- Button press → triggers state change (next mode).

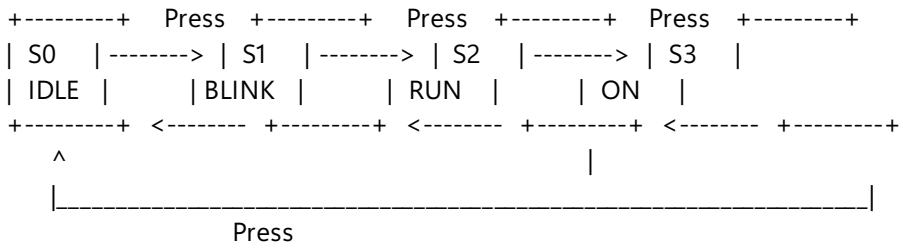
Outputs

- LEDs display the current mode pattern.

State Transition Table

Current State	Input (Button Press)	Next State	Output Action
S0_IDLE	Press	S1_BLINK	Start LED1 blinking
S1_BLINK	Press	S2_RUN	Start LED chase sequence
S2_RUN	Press	S3_ON	All LEDs ON
S3_ON	Press	S0_IDLE	All LEDs OFF

FSM State Diagram



Arduino Implementation

Hardware Required

- Arduino Uno (or compatible board)
- 3 LEDs + 220Ω resistors
- 1 Push Button + Pull-down resistor (10kΩ)
- Breadboard + Jumper wires

Pin Assignment

- LED1 → Pin 3
- LED2 → Pin 4
- LED3 → Pin 5
- Button → Pin 2

Arduino Code

```
// FSM Mode Selector using Arduino
// States: IDLE -> BLINK -> RUN -> ON -> back to IDLE

// Define states
typedef enum {S0_IDLE, S1_BLINK, S2_RUN, S3_ON} State;
State current_state = S0_IDLE;

// Pin configuration
const int ledPins[3] = {3, 4, 5};
const int buttonPin = 2;

// Variables
unsigned long lastUpdate = 0;
bool lastButtonState = HIGH;
int idx = 0;   // index for chasing LEDs

// Function: turn all LEDs OFF
void allOff() {
  for (int i = 0; i < 3; i++) digitalWrite(ledPins[i], LOW);
}

void setup() {
  for (int i = 0; i < 3; i++) pinMode(ledPins[i], OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); // button with internal pull-up
  Serial.begin(9600);
}

void loop() {
  // Check button press with debounce
  bool buttonState = digitalRead(buttonPin);
  if (lastButtonState == HIGH && buttonState == LOW) {
    // Button pressed -> change state
    current_state = (State)((current_state + 1) % 4);
    Serial.print("State Changed to: ");
    Serial.println(current_state);
```

```
    delay(200); // debounce delay
  }
  lastButtonState = buttonState;

  // FSM State Actions
  switch (current_state) {
    case S0_IDLE:
      allOff();
      break;

    case S1_BLINK:
      if (millis() - lastUpdate > 500) {
        digitalWrite(ledPins[0], !digitalRead(ledPins[0]));
        lastUpdate = millis();
      }
      break;

    case S2_RUN:
      if (millis() - lastUpdate > 300) {
        allOff();
        digitalWrite(ledPins[idx], HIGH);
        idx = (idx + 1) % 3;
        lastUpdate = millis();
      }
      break;

    case S3_ON:
      for (int i = 0; i < 3; i++) digitalWrite(ledPins[i], HIGH);
      break;
  }
}
```

Testing Procedure

1. Upload the sketch to Arduino.
2. Initially → All LEDs OFF (S0_IDLE).
3. Press button once → LED1 blinks (S1_BLINK).
4. Press again → LEDs chase in sequence (S2_RUN).
5. Press again → All LEDs ON (S3_ON).
6. Press again → Back to IDLE (all OFF).

Summary

- We implemented a Mode Selector using FSM principles.
- A single button cycles through four modes.
- Each mode has distinct LED behavior (OFF, Blink, Chase, All ON).
- The FSM approach makes the design structured, scalable, and easy to debug.
- Can be extended for real-world systems like fan speed controllers, robotic mode selection, washing machines, drones.
