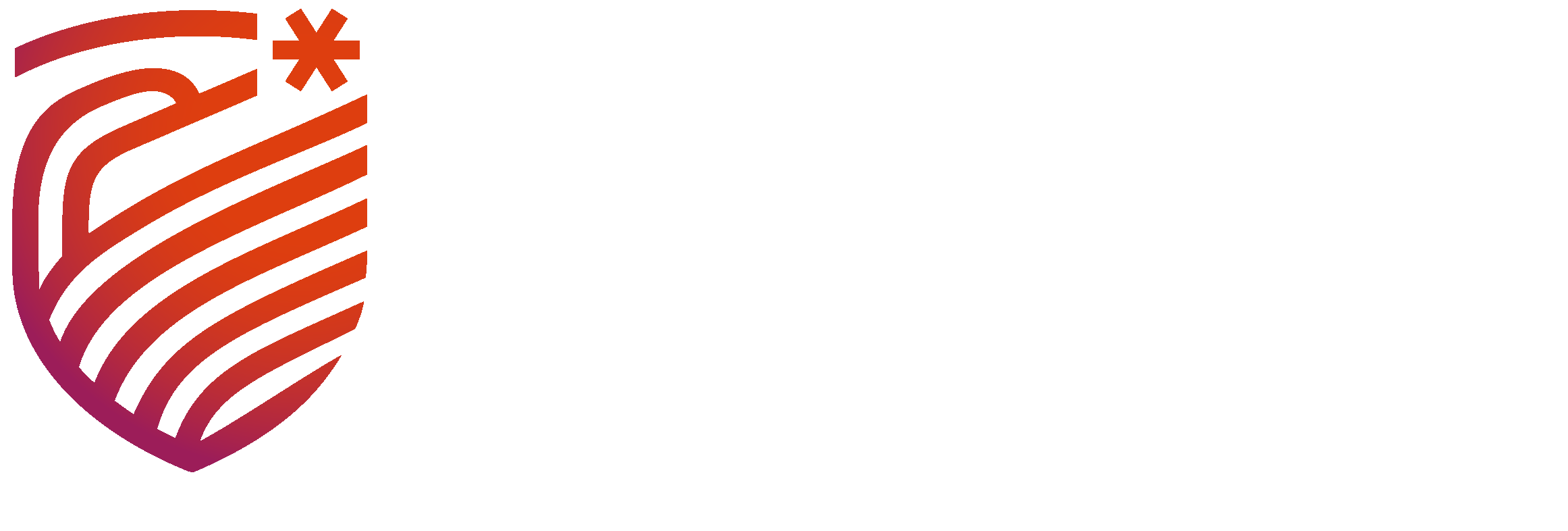
**Ramaiah Skill Academy**



**RAMAIAH**

**SKILL ACADEMY**

**TITLE OF THE PROJECT**

“Touch-Activated LED Control System Using Arduino with Register-Level Programming”

Submitted by:

**Mr.** **Harsha M S Reg no-RSAGWESD06**

# Interfacing a Touch Sensor with Arduino Uno Using Register-Level Programming

**Abstract**

This project focuses on the interfacing of a touch sensor with an Arduino Uno microcontroller using register-level programming. The main objective of the project is to make an LED glow when the touch sensor is activated. By bypassing high-level libraries and directly manipulating the microcontroller's registers, we gain a deeper understanding of how hardware-level operations are performed.

**1. Introduction**

Touch sensors have become integral in modern electronics, enabling touch-based controls in various applications. This project demonstrates the integration of a capacitive touch sensor with an Arduino Uno, utilizing register-level programming. Register-level programming provides greater control over the hardware and improves efficiency compared to high-level abstractions.  
  
The project’s aim is simple yet insightful: illuminate an LED when the touch sensor detects a touch input.

**2. Objectives**

- To interface a touch sensor with Arduino Uno using register-level programming.  
- To control an LED based on the sensor’s input.  
- To explore the advantages of low-level programming in microcontroller applications.

**3. Components Required**

1. Arduino Uno  
2. Capacitive Touch Sensor (TTP223 or equivalent)  
3. LED (Light Emitting Diode)  
4. Resistor (220Ω)  
5. Breadboard  
6. Jumper Wires

**4. Circuit Diagram**

- \*\*Connections:\*\*  
 - The VCC pin of the touch sensor connects to the 5V pin of the Arduino Uno.  
 - The GND pin of the touch sensor connects to the GND pin of the Arduino Uno.  
 - The output pin of the touch sensor connects to digital pin D2 of the Arduino Uno.  
 - The LED’s anode connects to digital pin D13 of the Arduino Uno through a 220Ω resistor, and the cathode connects to GND.

**5. Register-Level Programming Overview**

In Arduino Uno (ATmega328P microcontroller), registers are used to control the GPIO (General Purpose Input/Output) pins. Instead of using Arduino’s `digitalRead` and `digitalWrite` functions, registers like DDRx, PORTx, and PINx are manipulated for input and output control.  
  
- \*\*DDRx:\*\* Data Direction Register — Configures a pin as input or output.  
- \*\*PORTx:\*\* Port Register — Sets the logic level (HIGH or LOW) for output pins.  
- \*\*PINx:\*\* Pin Register — Reads the logic level of input pins.

**6. Implementation**

- \*\*Algorithm:\*\*  
 1. Configure the pin connected to the touch sensor as an input (D2).  
 2. Configure the pin connected to the LED as an output (D13).  
 3. Continuously read the state of the touch sensor.  
 4. If the sensor is activated, turn the LED on; otherwise, turn it off.

**Code:**

#include <avr/io.h>  
#include <util/delay.h>  
  
int main(void) {  
 // Configure pin D2 (PD2) as input  
 DDRD &= ~(1 << PD2); // Clear PD2 bit to set as input  
  
 // Configure pin D13 (PB5) as output  
 DDRB |= (1 << PB5); // Set PB5 bit to configure as output  
  
 while (1) {  
 // Read the state of the touch sensor on PD2  
 if (PIND & (1 << PD2)) {  
 // If touch sensor is activated, turn LED on  
 PORTB |= (1 << PB5); // Set PB5 high  
 } else {  
 // Otherwise, turn LED off  
 PORTB &= ~(1 << PB5); // Set PB5 low  
 }  
   
 \_delay\_ms(10); // Small delay to debounce the input  
 }  
}

**Conclusion:**

The project successfully demonstrates the implementation of a touch-activated LED control system using Arduino with register-level programming. By directly manipulating the microcontroller's registers, we achieved efficient and precise control over hardware components, eliminating the overhead of high-level libraries. This approach provided valuable insights into the low-level functioning of microcontrollers, offering a deeper understanding of input-output handling and hardware interaction.

The system effectively highlights the practical applications of touch sensors in embedded systems, paving the way for more complex touch-based control mechanisms in future projects. This hands-on experience reinforces the importance of mastering hardware-level programming for developing optimized and reliable embedded solutions