Ramaiah Skill Academy

RAMAIAH
SKILL ACADEMY

PROJECT REPORT

"TOUCH CONTROL BASED LED BLINKING USING ARDUINO UNO"

Submitted by:

Mr. Shivabasayya **Reg no-RSAGWPESD18**

1. **Introduction**
   The Touch Control LED Blinking project demonstrates the use of a touch sensor to control the on/off state of an LED. Unlike traditional Arduino programming using high-level functions like digitalRead() and digitalWrite(), this project utilizes register-level programming for precise control and better understanding of the underlying hardware. Register-level programming provides deeper insight into how microcontrollers operate and allows for faster execution compared to high-level abstractions.

2.Objectives

- To understand the concept of register-level programming on the Arduino UNO.
- To control an LED using input from a touch sensor.
- To utilize the ATmega328P's internal registers for port manipulation.

3. Components Required

- Arduino UNO
- Touch Sensor (e.g., TTP223)
- LED (Light Emitting Diode)
- Resistor (220 ohms) for the LED
- Breadboard and Connecting Wires

4. Pin Connections

- **Touch Sensor**: Connect the sensor's output (DO) to pin PD2 (digital pin 2) of the Arduino.
- **LED**: Connect the LED anode to pin PB0 (digital pin 8) of the Arduino through a 220-ohm resistor. Connect the LED cathode to GND.

- **Power Supply**: Connect the VCC and GND of the touch sensor to the 5V and GND pins of the Arduino UNO, respectively.

5. Register-Level Concepts To operate at the register level, it is important to understand the key registers of the ATmega328P used in the Arduino UNO. The main registers used in this project are:

- **DDRx (Data Direction Register)**: Configures the pin as an input (0) or output (1).
- **PORTx**: Sets the HIGH (1) or LOW (0) state of an output pin.
- **PINx**: Reads the logic state of an input pin.

Register	Purpose	Example	
DDRB	Set pin direction	`DDRB = 0x01;` (Set PB0 as output)	
PORTB	Set output logic level	`PORTB	= 0x01;` (Set PB0 HIGH)
PINB	Read input logic level	`if (PINB & 0x01)` (Check if PB0 is HIGH)	

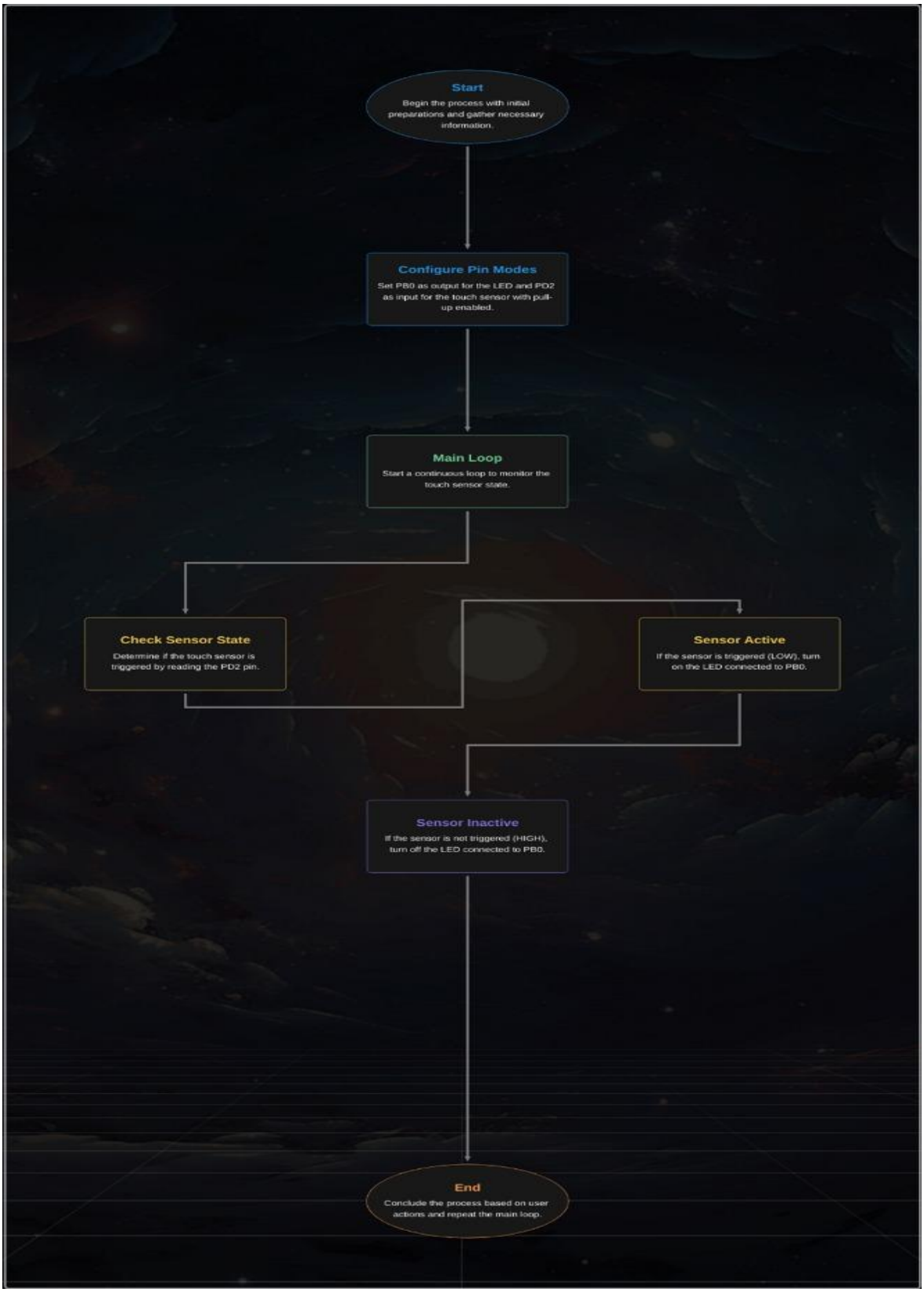
6. Register-Level Program

```
#define F_CPU 16000000UL // Define clock frequency
#include <avr/io.h>      // Include AVR IO header
#include <util/delay.h>  // Include delay function

int main(void) {
   // Step 1: Configure pin modes
   DDRB |= (1 << PB0);  // Set PB0 (digital pin 8) as output for LED
   DDRD &= ~(1 << PD2); // Set PD2 (digital pin 2) as input for Touch Sensor
   PORTD |= (1 << PD2); // Enable pull-up resistor on PD2

   while (1) {
      // Step 2: Check if the touch sensor is triggered
      if (!(PIND & (1 << PD2))) {  // Check if touch sensor output is LOW (active)
         PORTB |= (1 << PB0);      // Turn on LED connected to PB0
      } else {
         PORTB &= ~(1 << PB0);     // Turn off LED connected to PB0
      }
   }
```

Explanation of the Code

1. **Clock Frequency**: The F_CPU macro sets the clock frequency to 16 MHz, which is the default for Arduino UNO.
2. **Pin Configuration**:
    - **DDRB** is used to configure PB0 as an output (for the LED).
    - **DDRD** is used to configure PD2 as an input (for the touch sensor).
    - A pull-up resistor is enabled for PD2 to ensure a stable high logic state when the sensor is not touched.
3. **Infinite Loop**: The program continuously checks the state of the touch sensor.
4. **Touch Sensor Logic**: If the sensor is touched (logic 0 on PD2), the LED on PB0 is turned on. If not touched, the LED is turned off.

7.FLOWCHART:


Start
Begin the process with initial preparations and gather necessary information.
Configure Pin Modes
Set PB0 as output for the LED and PD2 as input for the touch sensor with pull-up enabled.
Main Loop
Start a continuous loop to monitor the touch sensor state.
Check Sensor State
Determine if the touch sensor is triggered by reading the PD2 pin.
Sensor Active
If the sensor is triggered (LOW), turn on the LED connected to PB0.
Sensor Inactive
If the sensor is not triggered (HIGH), turn off the LED connected to PB0.
End
Conclude the process based on user actions and repeat the main loop.

8. Testing and Results

1. **Initial Setup**: Connect the components as per the circuit diagram.
2. **Power On**: Once the Arduino UNO is powered on, the program starts running.
3. **Touch Control**: Touching the sensor will turn on the LED, and releasing the touch will turn it off. This confirms that the touch sensor is working as intended.

Conclusion:

This project demonstrates how to control an LED using a touch sensor with register-level programming on the Arduino UNO. By manipulating the Data Direction Registers (DDRx), Port Registers (PORTx), and Pin Input Registers (PINx), users gain greater insight into low-level microcontroller operations. Although more complex than standard Arduino sketches, register-level programming offers faster execution and more efficient use of resources.