REPUBLIC OF THE PHILIPPINES

SURIGAO DEL NORTE STATE UNIVERSITY

Narcisso Street, Surigao City 8400, Philippines

Caspe,Mark Angelo J

BSECE-3A

Laboratory Activity no.7

Button Pressing:Pushbuttons

**Introduction:**

Up to this point we have focused entirely on outputs, time to get our Arduino to listen, watch and feel. We'll start with a simple pushbutton. Wiring up the pushbutton is simple. There is one component, the pull up resistor, that might seem out of place.This is included because an Arduino doesn't sense the same way we do (ie button pressed,button unpressed). Instead it looks at the voltage on the pin and decides whether it is HIGH or LOW. The button is set up to pull the Arduino's pin LOW when it is pressed, however, when the button is unpressed the voltage of the pin will float (causing occasional errors). To get the Arduino to reliably read the pin as HIGH when the button is unpressed, we add the pull up resistor.

**Objectives:**

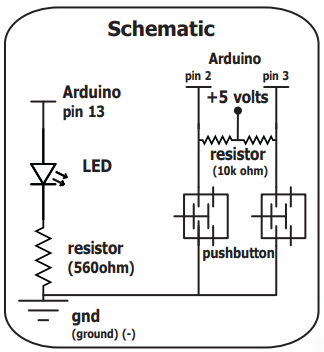
The objective is to learn how to connect a pushbutton to an Arduino so it can detect when the button is pressed or released. The pull-up resistor helps the Arduino understand when the button is not pressed by keeping the voltage stable, preventing errors.

**Materials:**

1.) Atmega328P 2.) Crystal 3.) Pushbutton x2 4.) 22pf Capacitor

5.) 10k Ohm Resistor 6.) 150 Ohm Resistor 7.) Red LED x1

**Schematic Diagram**:



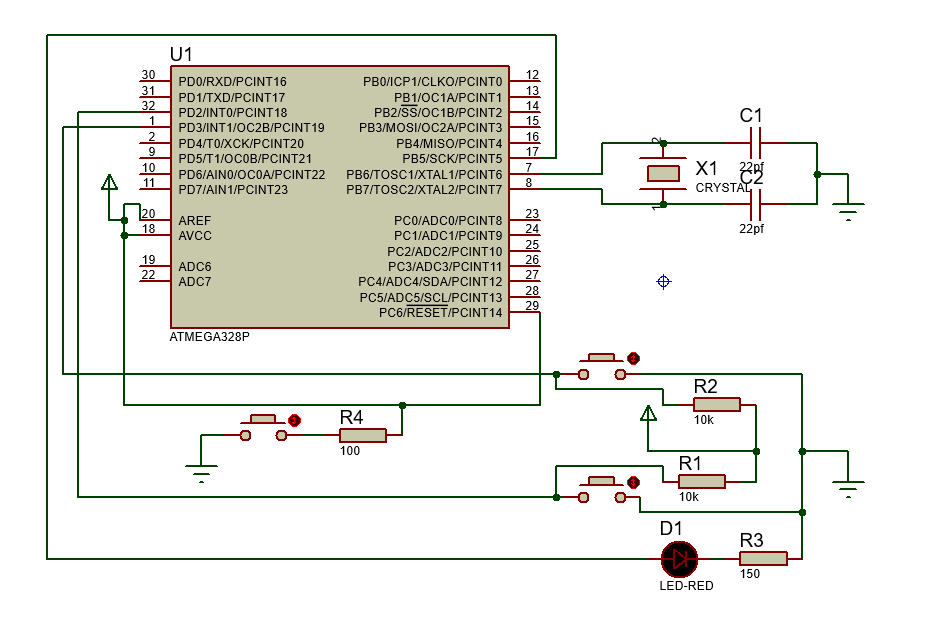
**Procedure:**

* Open Proteus and create a new project.
* Find the components needed for your activity in Proteus, including the ATmega328P microcontroller and a pushbutton.
* Refer to the schematic diagram for guidance on connecting the components. Wire up the pushbutton and add the pull-up resistor as per the instructions.
* Open the Arduino IDE and write the code for your project, which includes reading the state of the pushbutton and controlling an output based on its state.
* Save the code in the Arduino IDE.Go back to Proteus, click on the ATmega328P microcontroller in your project.Find the file you saved in step 5 (the .hex file) and choose it to upload your code to the microcontroller.
* Simulate or run the simulation in Proteus to see your project in action, observing how the Arduino responds to the pushbutton input.

**Results and Discussion:**

When the button is pushed, it activates the LED, and when released, it deactivates the LED, illustrating basic digital input and output interactions with Arduino.

**Circuit:**

****

**Program:**

int ledPin = 13;

int inputPin1 = 2;

int inputPin2 =3;

int val1 = 0;

int val2=0;

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(inputPin1, INPUT);

pinMode(inputPin2, INPUT);

}

void loop(){

val1 = digitalRead(inputPin1);

val2 = digitalRead(inputPin2);

if (val1 == LOW) {

digitalWrite(ledPin, HIGH);

}

else if (val2 == HIGH) {

digitalWrite(ledPin, LOW);

}

 else {

digitalWrite(ledPin, HIGH);

}

}

**Findings:**

Pushing the button makes the LED turn on, and releasing it makes the LED turn off.

**Recommendations:**

Double-check the connections and code to ensure reliable performance of the pushbutton and LED interaction.

**Conclusions:**

This project successfully demonstrates how to control an LED using a pushbutton with an Arduino, showcasing basic digital input and output interactions.