UNIT III: Arduino Programming

Digital Pins as Input and Output, Reading Analog Quantities, PWM Pin- Arduino's Serial Port and Serial Communication. Interfacing of DC Motor and Relay

3.1 Digital Pins as Input and Output

3.1.1 Digital Pins as Output

When a digital pin is configured as an output, it can be used to send signals to external devices such as LEDs, motors, or any actuator that can be controlled electronically.

Key Steps:

1. Configuration:

- Use the `pinMode(pin, OUTPUT)` function to configure a pin as an output.
- This setup allows the pin to output either a HIGH or LOW signal.

2. Operation:

- HIGH Signal: Sends a voltage (usually 5V) to the connected device, effectively turning it on.
- LOW Signal: Sends no voltage (0V), turning the device off.
- Use the `digitalWrite(pin, HIGH/LOW)` function to set the pin's state.

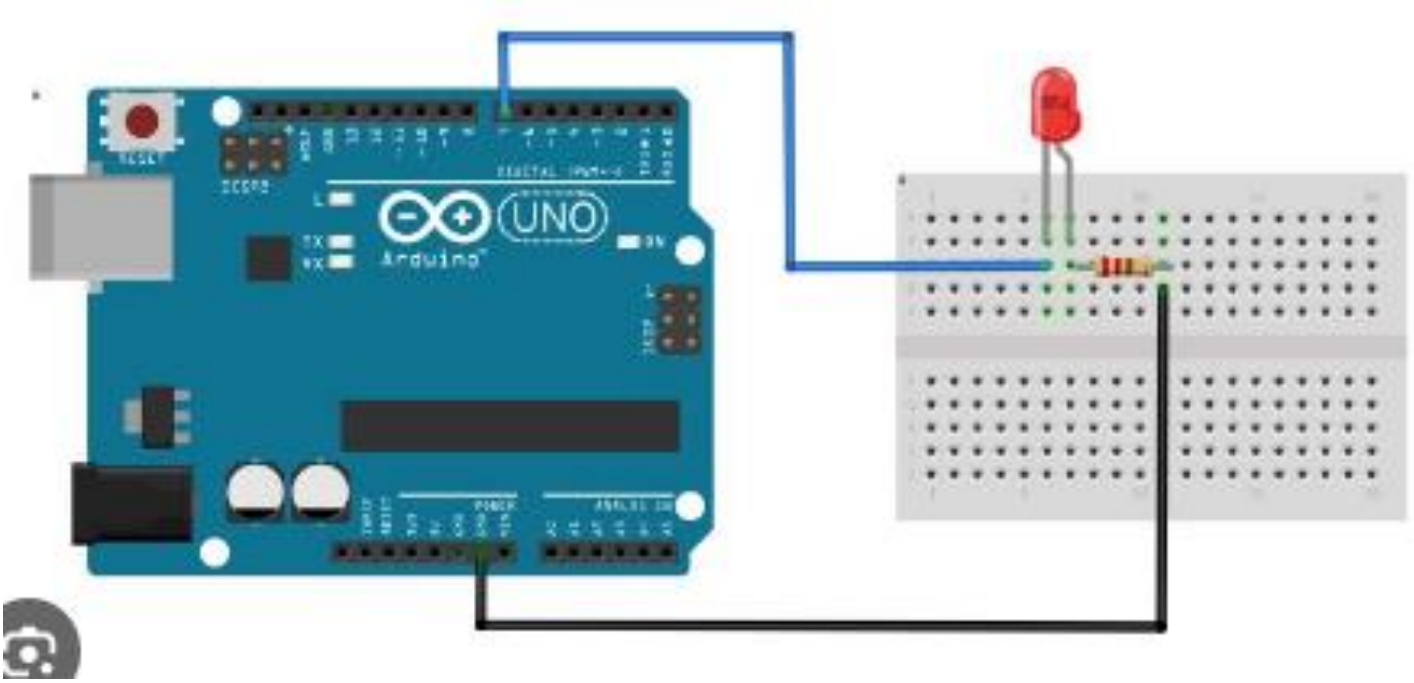
Example:

To control an LED connected to pin 13:

```
int ledPin = 13;  // Pin connected to LED
void setup() {
  pinMode(ledPin, OUTPUT);  // Set pin as output
}
void loop() {
  digitalWrite(ledPin, HIGH);  // Turn LED on
  delay(1000);                 // Wait for 1 second
  digitalWrite(ledPin, LOW);   // Turn LED off
  delay(1000);                 // Wait for 1 second
}
```

Circuit Diagram Description:

- **Arduino Board**: Any compatible Arduino board (e.g., Arduino Uno).
- **LED**: Connected to digital pin **13**.
- **Resistor (220Ω - 1kΩ)**: Connected in series with the LED to limit current.
- **Connections**:
  - **Anode (+) of LED → Pin 13** of Arduino.
  - **Cathode (-) of LED → Resistor → GND (Ground) of Arduino**.



3.1.2 Digital Pins as Input

When a digital pin is set as an input, it can read signals from external devices like sensors or switches.

Key Steps:

1. Configuration:

- Use the `pinMode(pin, INPUT)` function to configure a pin as an input.

- This setup allows the pin to detect a HIGH or LOW signal from the connected device.

2. Operation:

- HIGH Signal: The pin reads a voltage, indicating the device is active (e.g., switch pressed).

- LOW Signal: No voltage is detected, indicating the device is inactive.

- Use the `digitalRead(pin)` function to get the pin's state.

Example:

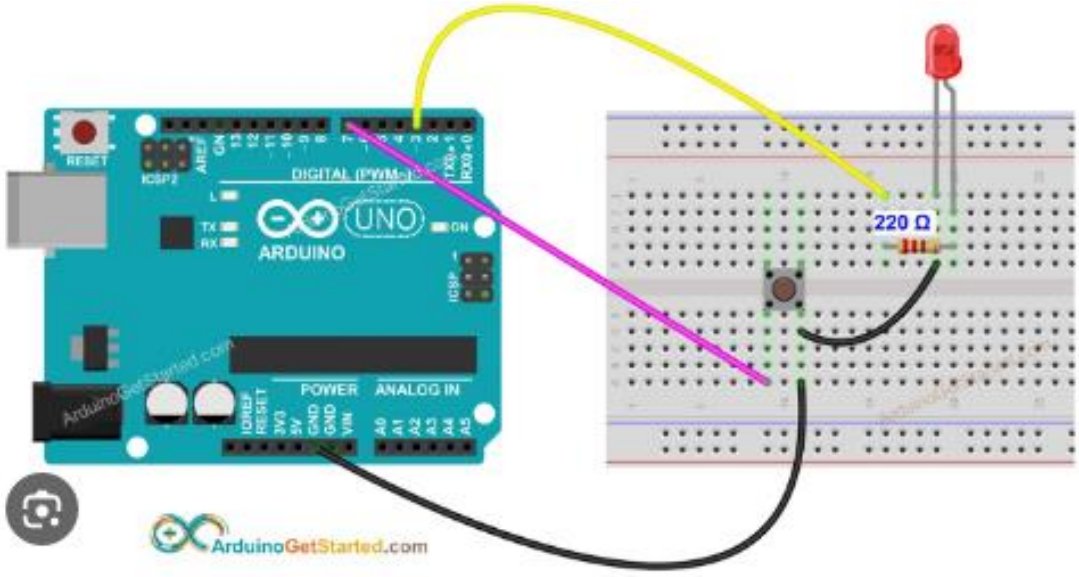
To read a button state connected to pin 2:

```
int buttonPin = 2;    // Pin connected to button
int buttonState = 0;  // Variable to store button state
void setup() {
  pinMode(buttonPin, INPUT);  // Set pin as input
}
void loop() {
  buttonState = digitalRead(buttonPin);  // Read the button state
  if (buttonState == HIGH) {
    // Perform action when button is pressed
  }
}
```

Circuit Diagram Description:

- **Arduino Board**: Any compatible Arduino board (e.g., Arduino Uno).
- **Push Button**: Connected to digital pin **2**.
- **Pull-down Resistor (10kΩ)**: Ensures the button reads LOW when not pressed.
- **Connections**:
  - **One terminal of button → Pin 2 (Arduino)**.
  - **Same terminal → 10kΩ resistor → GND (Ground of Arduino)**.
  - **Other terminal of button → +5V (Arduino VCC)**.

Circuit Diagram Representation:



Practical Applications

1. Output:

- LED Control: Turn LEDs on and off for visual indicators.

- Motor Control: Use digital signals to control motor drivers and hence the motors.

2. Input:

- Switches/Buttons: Detect when a user presses a button to trigger actions.

- Sensors: Read digital outputs from sensors to make decisions in your program.

Digital pins are crucial in interfacing with the physical world, enabling the Arduino to act as a bridge between software logic and hardware actions.

3.2 Reading Analog Quantities

Reading analog quantities with Arduino involves using the analog pins on the board to measure varying voltage levels from sensors. This process is crucial for interfacing with analog sensors, such as temperature sensors, light sensors, and potentiometers, which provide a range of outputs rather than simple on/off signals.

Steps to Read Analog Quantities:

1. Configuration:

- Connect the sensor to the analog pin (e.g., A0) of the Arduino.

- Ensure that the sensor's ground and power connections are properly set up.

2. Code Setup:

- Use the `analogRead(pin)` function to read the voltage level from the specified analog pin.

- This function returns a value between 0 and 1023, corresponding to the voltage level from 0V to 5V.

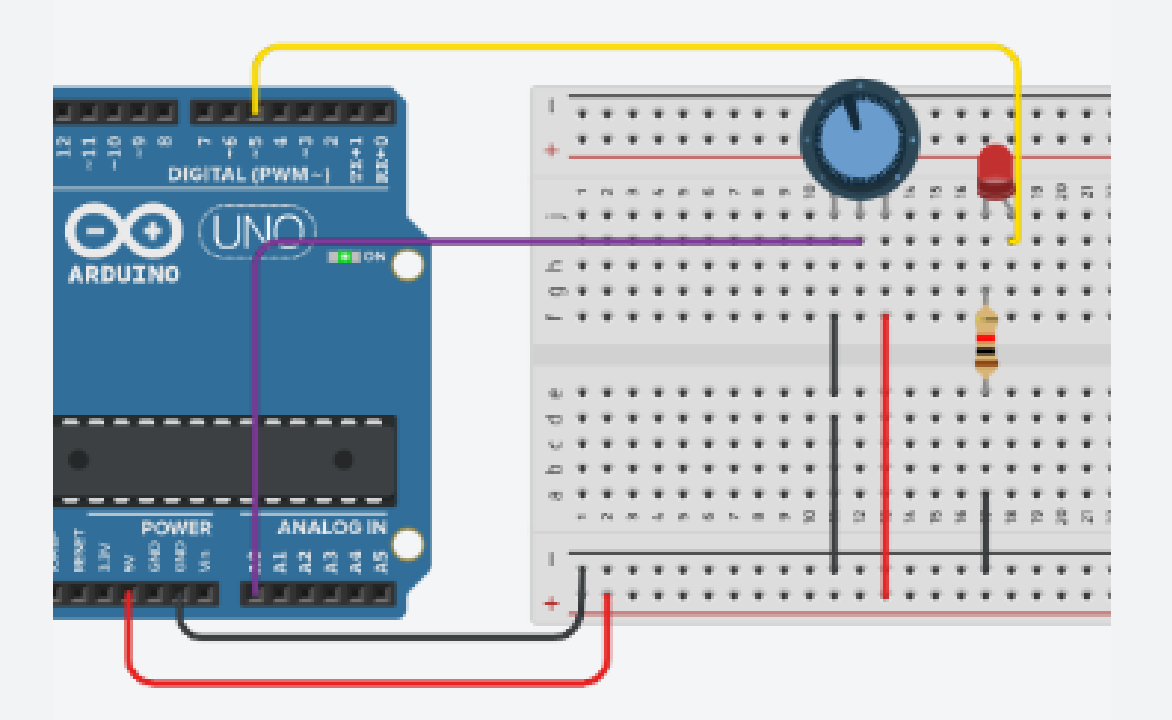
3. Example Code:

```
int sensorPin = A0; // Pin connected to the sensor

int sensorValue = 0; // Variable to store the sensor value

void setup() {

Serial.begin(115200); // Initialize serial communication

}

void loop() {

sensorValue = analogRead(sensorPin); // Read the sensor value

Serial.print("Sensor reading: ");

Serial.println(sensorValue); // Print the sensor value to the Serial Monitor

delay(500); // Wait for half a second

}
```

Circuit Diagram Description:

- **Arduino Board**: Any compatible Arduino board (e.g., Arduino Uno).
- **Analog Sensor**: Connected to **A0 (Analog Pin 0)**.
- **Power Connections**:
  - **VCC (5V or 3.3V from Arduino) → VCC of the sensor**.
  - **GND (Ground of Arduino) → GND of the sensor**.
- **Signal Connection**:
  - **Sensor Output (Analog) → A0 pin on Arduino**.

Circuit Diagram Representation:



4. Running the Code:

- Upload the code to your Arduino board.

- Open the Serial Monitor in the Arduino IDE to see the readings from the sensor.

5. Practical Application:

- As you change the input (e.g., adjusting a potentiometer or changing the light level on a light sensor), you'll see the sensor values change in the Serial Monitor. This demonstrates how the analog input translates into a digital representation that the Arduino can process.

By following these steps, we can effectively read and utilize analog signals from various sensors in our IoT projects.