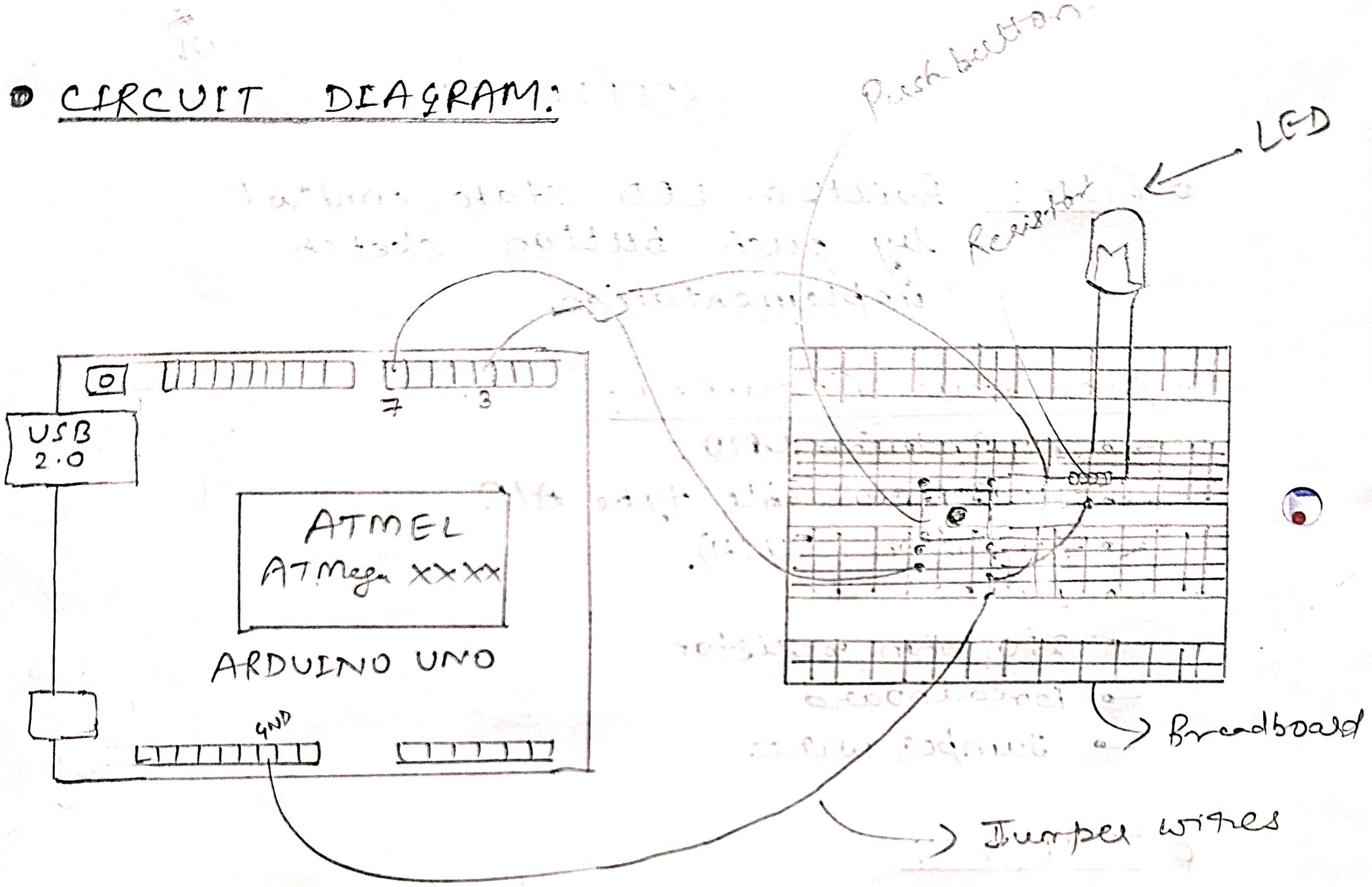
- **AIM:** Built-in LED state control by push button sketch implementation.

- **APPARATUS REQUIRED:**
  - 1x Arduino UNO
  - 1x USB 2.0 cable type A/B
  - 1x Button (PUSH).
  - 1x LED
  - 220 ohm resistor
  - Breadboard
  - Jumper wires

- **THEORY:**

  - Here, Arduino Board will handle the blinking of LED by using a PUSH Button.
  - It will work when it is wired and programmed and well connections is performed by the user.
  - It is programmed using the suitable code for blinking an LED via push button.

- **CIRCUIT DIAGRAM:**



Topic : EXPERIMENT-1

PROCEDURE :

Step1 : Connect the LED, PUSH BUTTON, to UNO Board using jumper wires.

Step 2 : Connect the board to computer using USB 2.0 cable.

Step 3 : Write the code in sketch.

Step 4 : Select the port and board in the ARDUINO IDE.

Step 5 : Upload the code to Arduino UNO.

Arduino Code :

```
const   int   BUTTON_PIN = 7;
const   int   LED_BUILTIN = 3;
  int  buttonState = 0;

 void setup () {

      pinMode (LED_BUILTIN, OUTPUT);
      pinMode (BUTTON_PIN, INPUT_PULLUP);
   }

void loop () {

      buttonState = digitalRead (BUTTON_PIN);
      if (buttonState == LOW)
        digitalWrite(LED_BUILTIN, HIGH);
      else
        digitalWrite (LED_PIN, LOW);
      }
```

Topic : Experiment - 2

- **AIM:** Built-in LED blinking sketch implementation.

- **APPARATUS REQUIRED:**
  - ARDUINO UNO
  - USB 2.0 cable type A/B
  - LED
  - 220 ohm resistor
  - Breadboard
  - Jumper wires

- **THEORY:**

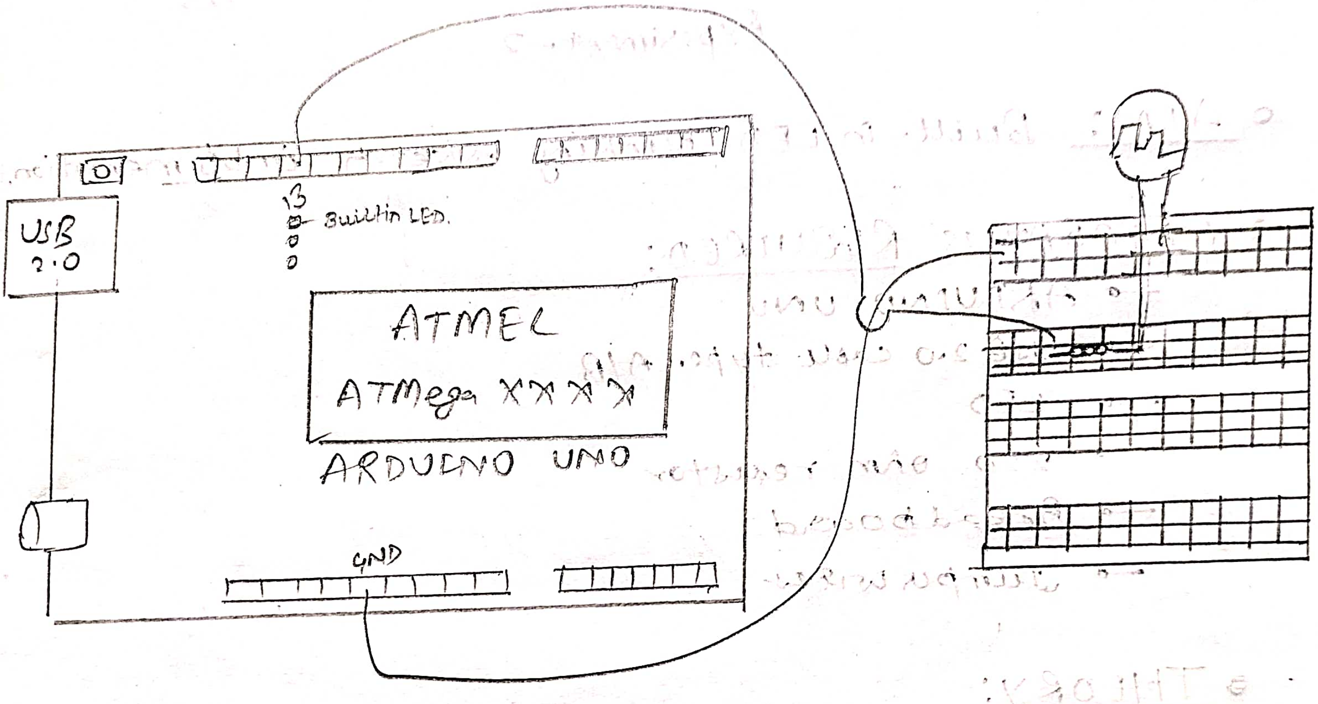
LED includes two pins:
→ cathode (−) pin : needs to be connected to GND (or).
→ Anode (+) pin : is used to control LED's state.

If generating a PWM signal to the anode (+), the brightness of LED is changed according to PWM value.

When an Arduino's pin is configured as a digital output, the pins voltage can be programically set to GND or VCC value.

By connecting the Arduino's pin to LED's anode (+) pin (via a resistor), we can programmically control LED's state.

CIRCUIT DIAGRAM:



The built-in LED is connected to digital pin 13 of Arduino UNO board.

We can also light an external LED by joining the LED in corresponding pin 13.

PROCEDURE:

Step1: Connect the external LED to pin 13 of Arduino UNO and other terminal at GND.
Step 2: connect the board to computer using the USB 2.0 cable.
Step3: write the code in sketch.
Step4: select the post and board in the tools of Arduino IDE.
Step5: Upload the code to Arduino UNO.

Arduino Code:

```
Void setup(){
    pinMode(LED_BUILTIN, OUTPUT);
}
Void loop(){
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

Topic : EXPERIMENT-3.

- **AIM:** Built-in LED blinking by toggling states based on binary operation.

- **APPARATUS REQUIRED:**
  - → 1x Arduino UNO
  - → 1x USB 2.0 cable type A/B.

- **THEORY:**

Toggling the state of a built-in LED by using binary operation is a common technique in embedded systems programming.
The concept involves using bitwise operations, such as XOR (^), to manipulate specific bits within a variable that controls the state of the LED.

Binary operations for LED Toggling.

(i) Bitwise XOR (^) operation.

– XOR toggles bits to change the state without affecting the other bits. When you XOR a value with another, the results is '1' if the bits are different and '0' if they are the same.

(ii) Controlling LEDs with Bits:
- LEDs are often controlled by single pin, which can be in an " on " or "off" state (1 or 0). We can represent the state the of LED using a single bit within a variable.

PROCEDURE:

Step 1: Connect the arduino with PC as using USB 2.0 cable.

Step 2: Write the code.

"CODE"

```
const int ledPin = 13;
void setup() {
      pinMode (ledPin, OUTPUT);
   }
void loop () {
      static unit 8_t toggle = 0b00000001;
      digitalWrite (ledPin, toggle & 0b00000001);
      toggle ^ = 0b00000001;
      delay (1000);
   }
```

CONCLUSION:

After doing this experiment, we see our project on LED works satisfactorily.

Topic : EXPERIMENT-4

AIM: Built in-LED state control by user interface through serial port.

REQUIREMENTS:

(i) Arduino Board
(ii) LED
(iii) Arduino IDE
(iv) 1x USB 2.0 cable type A/B.

THEORY:

Controlling the state of a built-in LED through a user interface via the serial port involves establishing a communication link between a microcontroller (such as an Arduino) and a computer or external device.
This allows users to send commands or instructions via serial terminal interface (like the Arduino Serial Monitor) to control the LED's state.

PROCEDURE:

Step 1: Connect the Arduino UNO to PC by using USB 2.0 cable.
Step 2: Write the code on the sketch then upload.

code

```
const int ledPin =13;
void setup () {
    pinMode (ledPin, OUTPUT);
    Serial.begin (9600);
 }
void loop () {
     if (Serial.available () > 0) {
     String command = Serial. readStringUntil ('\n');
     if (command == "ON" || command == "on") {
      digitalWrite (ledPin , HIGH) ;
      Serial. println( "LED is ON") ;
     } else if (command == "OFF" || command == "off") {
      digitalWrite (ledPin ; LOW);
       Serial. println ("LED is OFF") ;
     } else {
       serial. print ln ("Invalid command. Enter 'ON' or
                          OFF' to controll the LED.");
  }
  }
 }
```

Step 3 : Open serial monitor. and type your desired code ordate to turn on or off the LED as ON and OFF.

CONCLUSION:

Our practical work successfully after giving the instructions in serial monitor.

Topic : EXPERIMENT - 5.

◎ **AIM:** User interface for Boolean operation and bit-wire operation through serial port.

◎ **APPARATUS:**

→ 1X Arduino UNO

→ 1X USB 2.0 cable type A/B

◎ **THEORY:**

Creating a user interface to perform Boolean and bitwise operations via the serial port involves setting up a communication link between a microcontroller (like Arduino) and a computer or terminal software that allows users to input commands to perform these operations.

Boolean or Bitwise operations and display the results back to the user interface.

◎ **PROCEDURE:**

Step 1: Connect the Arduino to PC using USB 2.0 type cable.

Step 2: Upload the sketch.

" Sketch is the written code ".

"code"

```
void setup(){
    Serial.begin(9600);
    while (!Serial) {
        ;
    }
    Serial.println("Enter two integers and choose an operation:");
    Serial.println("1. AND");
    Serial.println("2. OR");
    Serial.println("3. XOR");
}

void loop(){
    if (Serial.available()){
     int operand1 = Serial.parseInt();
     int operand2 = Serial.parseInt();
     int operation = Serial.parseInt();
   if (operation == 1){
     int result = operand1 & operand2;
     Serial.print("Result of AND operation:");
     Serial.println(result);
   } else if (operation == 2){
      int result = operand1 | operand2;
     Serial.println("Result of OR operation:");
     Serial.println(result);
   } else if (operation== 3){
```

```
int result = operand1 ^ operand 2;
Serial.print ("Result of XOR operation:");
} else {
   Serial.println ("Invalid operation choice.");
}
   Serial.println ("Enter two integers and choose an operation:");
   Serial.println ("1. AND");
   Serial.println ("2. OR");
   Serial.println ("3. XOR");
}
}
```

Step 3: Open the serial monitor and give the command which asked.

- **CONCLUSION:**

After uploading the code and using serial monitor, after processing the input, it prompts the user to perform the bitwise and boolean operations.

Topic : EXPERIMENT- 6

- **AIM:** Looping mechanism to check the state of pin and if change print it status on serial port.

- **APPARATUS REQUIRED:**
  - → Arduino UNO
  - → Jumper wires
  - → External devices/Sensors (push button).
  - → Breadboard.

- **THEORY:**

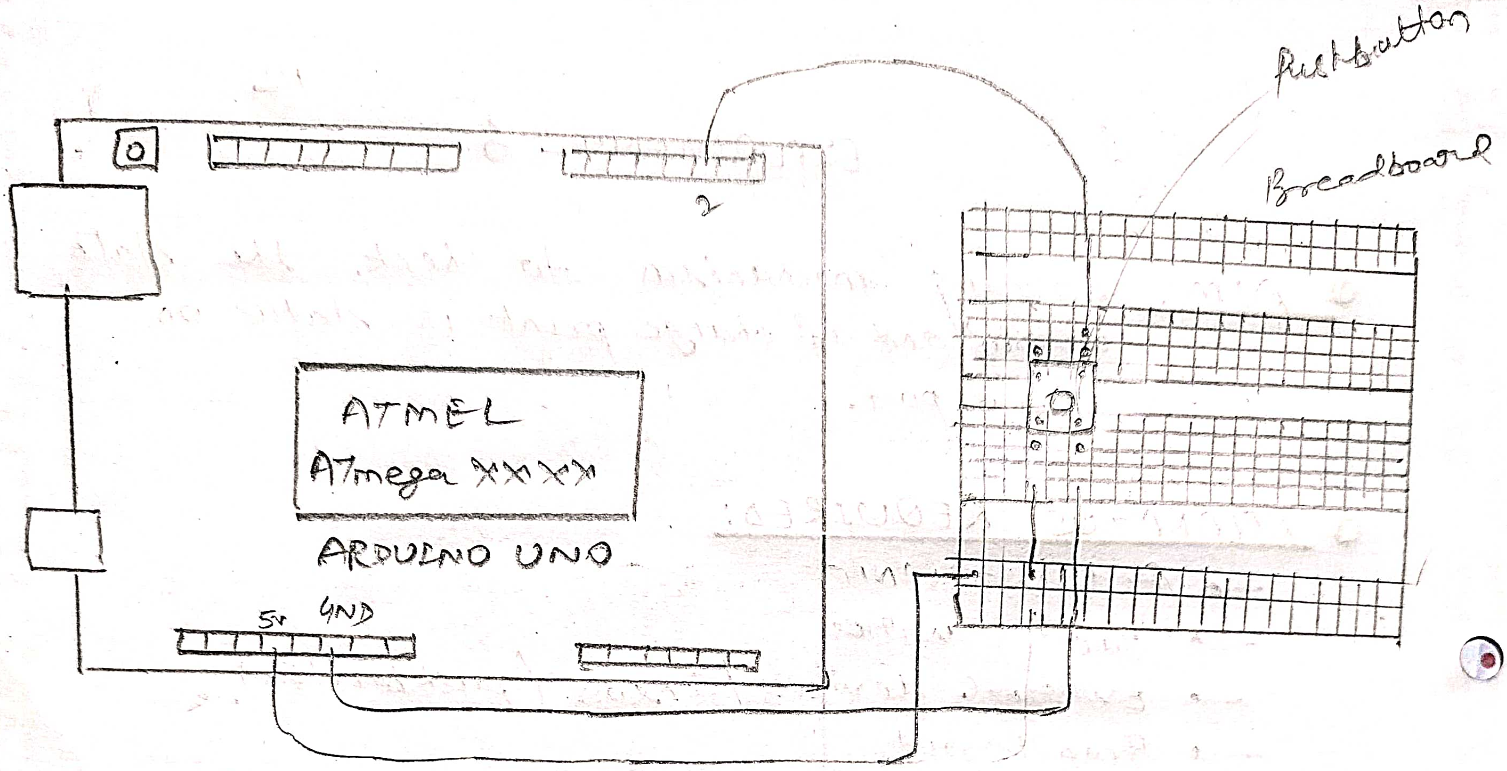
This practical implementation involves using an IOT devices, such as an Arduino, to continuously monitor the state of a pin and report any changes in state to a connected serial port.

- **PROCEDURE:**

Step 1: Connect the arduino to PC and do the connection of a Pushbutton to arduino.

Step 2: Write the sketch for this project. ~~codes~~ and then upload.

CONNECTION :


Pushbutton
Breadboard
2
ATMEL
ATmega XXXX
ARDUINO UNO
5v
GND

Step 3: Code for program.

```
int pushButton = 2;
void setup ( ) {
  Serial.begin (9600);
  pinMode (pushButton, INPUT);

 void loop ( ) {
   int buttonstate = digitalRead (pushbutton);
   Serial.println (buttonState);
   delay (1000);
 }
```

Step 4: Open the serial monitor, see the state of pushbutton.

Step 5: Push the button and the result should vary in the serial monitor.

- **CONCLUSION:**

Hence, we demonstrated the state of a pin and also printed their state in serial port (serial monitor) after changing the state of push Button.

Topic : EXPERIMENT- 7

- AIM: Controlling multiple LEDs with a loop and an array.

- APPARATUS REQUIRED:

→ Arduino UNO
→ Breadboard
→ 5 LEDs.
→ 10k Ohm resistor.

- THEORY:

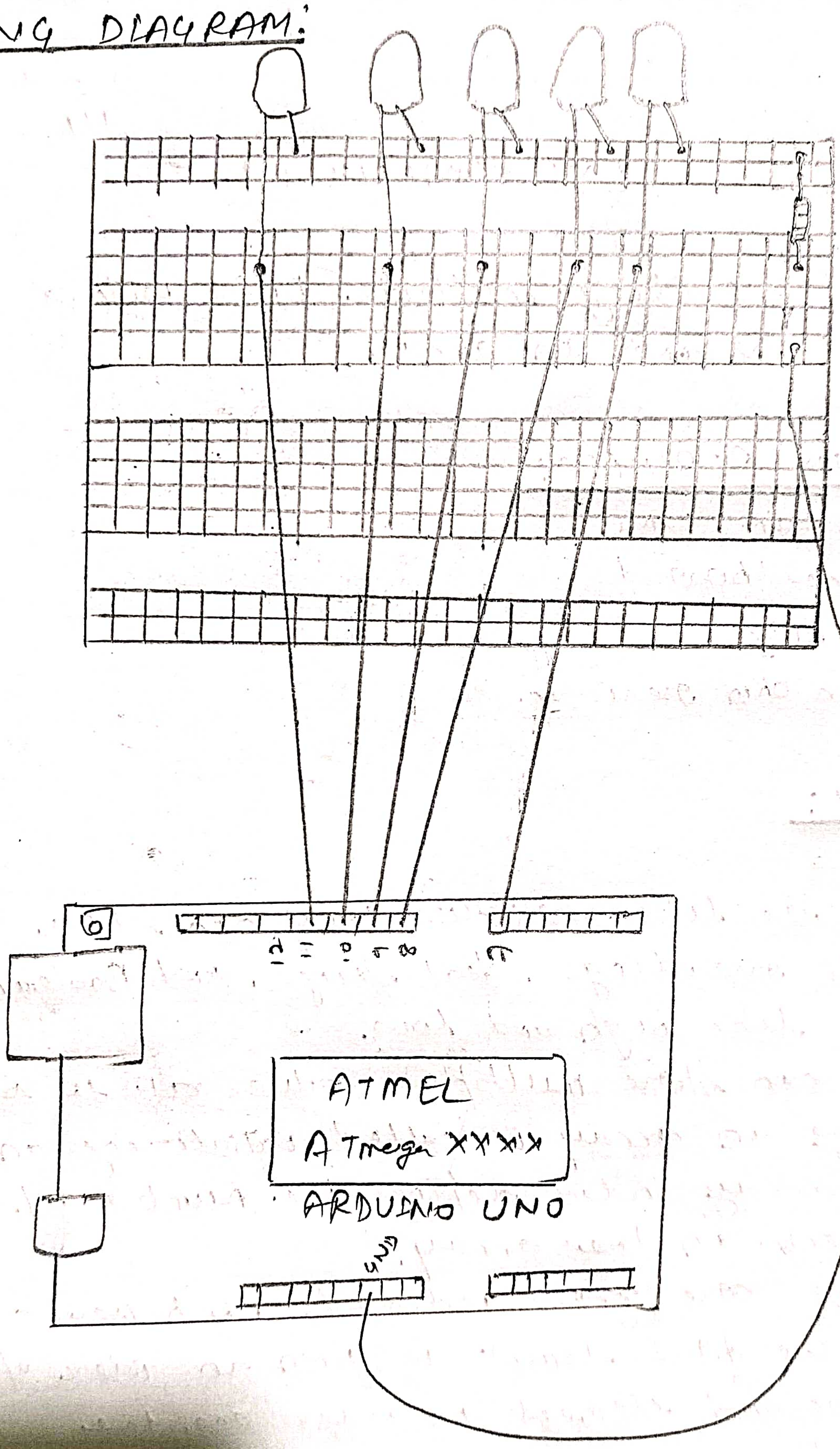
→ Arrays are like variables- they can store sensor readings, text strings, and Boolean values like high and low.

→ Arrays can store multiples values at the same time. Creating an array is called initializing an array.

→ The array index defines the number of elements in the array.

→ Arrays are zero indexed, which means that the first element is given an idex of zero, the second element is index one, the third element is index two, and so on.

WIRING DIAGRAM:



PROCEDURE:

Step1 : Do well connection as shown alongside.

Step2: Upload the sketch.

"code"

```
int Ledpins[5] = {11,10,9,8,7};
 void setup(){
      for (int i = 0; i<5 ; i++) {
       pinMode (Ledpin [i], OUTPUT);
     }
   }
    void loop( ){
    for (int j = 0; j<5; j++) {
     digitalwrite (LedPins [j], HIGH);
     delay (500);
     digitalwrite (LedPins[j], LOW);
     delay (500);
    }
   }
```

CONCLUSION:

Afte performing the steps we see our LED's are glowing in form of array.

Topic : EXPERIMENT- 8

AIM: Use a potentiometer to control the blinking of an LED.

APPARATUS REQUIRED:

- Arduino UNO
- Potentiometer
- LED
- Resistor
- Jumper wires

THEORY:

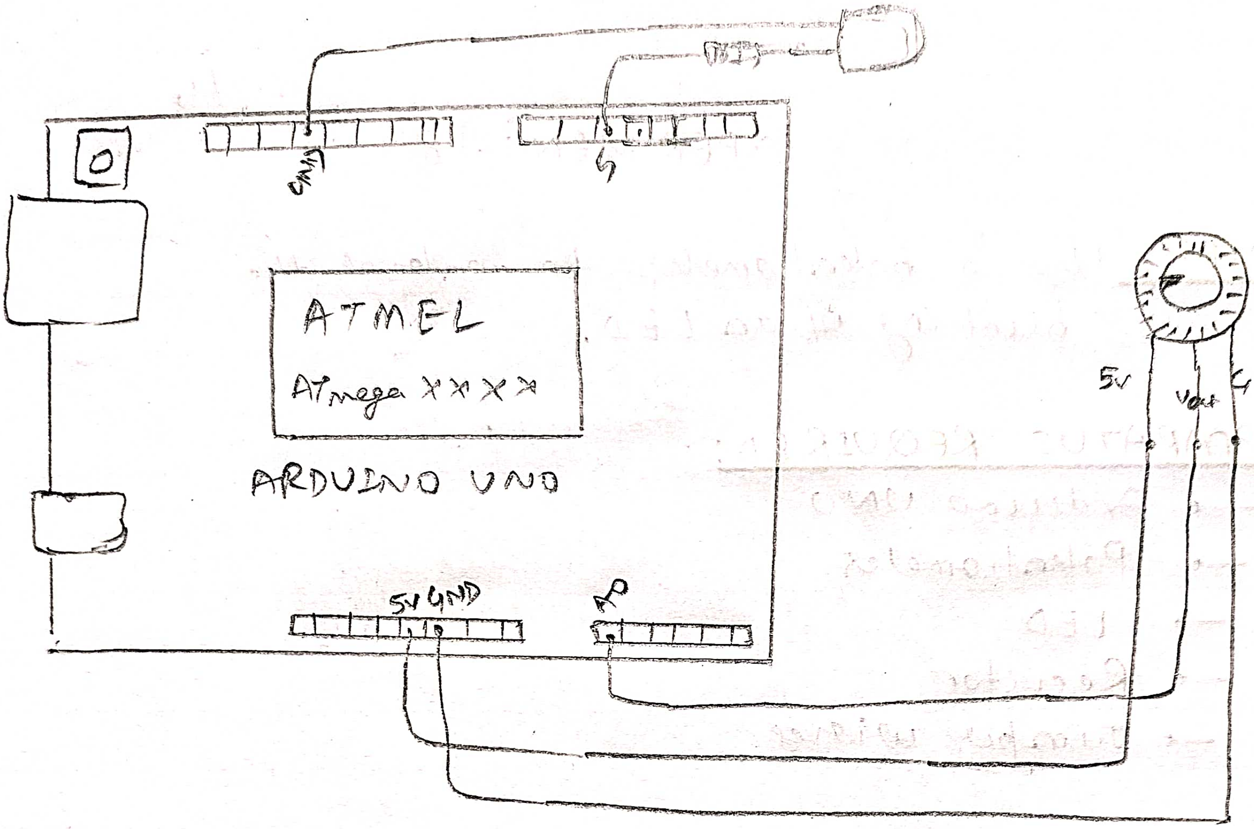
A potentiometer, also known as a variable resistor, can be used to control the blinking speed of an LED in a simple electronic ckt.

Arduino read Analog value by analogRead function. The voltage is adjusted by the potentiometer.

PIN CONNECTIONS

Potentiometer and LED Pins	Arduino Pins.
5V or Vin	5V
GND	GND
Vout	A0
LED anode	D5
LED cathode	GND

CONNECTION :



Topic : Experiment - 8

PROCEDURE:

Step1: Do the well connections as shown alongside.

Step2: Upload the sketch.

"code".

```
const int potPin = A0;
const int ledPin = 5;
int val = 0;
void setup(){
    pinMode (ledPin, OUTPUT);
    Serial.begin (9600);
}
void loop(){
    val = analogRead (potPin);
    digitalWrite (ledPin, HIGH);
    delay (val);
    digitalWrite (ledPin, LOW);
    delay (val);
    Serial.println (val);
}
```

Step 3: Adjust the potentiometer as your desire and you will see the LED will be blinking on the potentiometer value changes.

CONCLUSION:

Thus, we made the blinking LED by potentio meter value adjusting.

Topic : EXPERIMENT - 9

AIM: Use an analog output (PWM pin) to fade an LED.

APPARATUS:

→ Arduino UNO

→ LED

→ Jumper wires

→ Resistor

→ Breadboard.

THEORY:

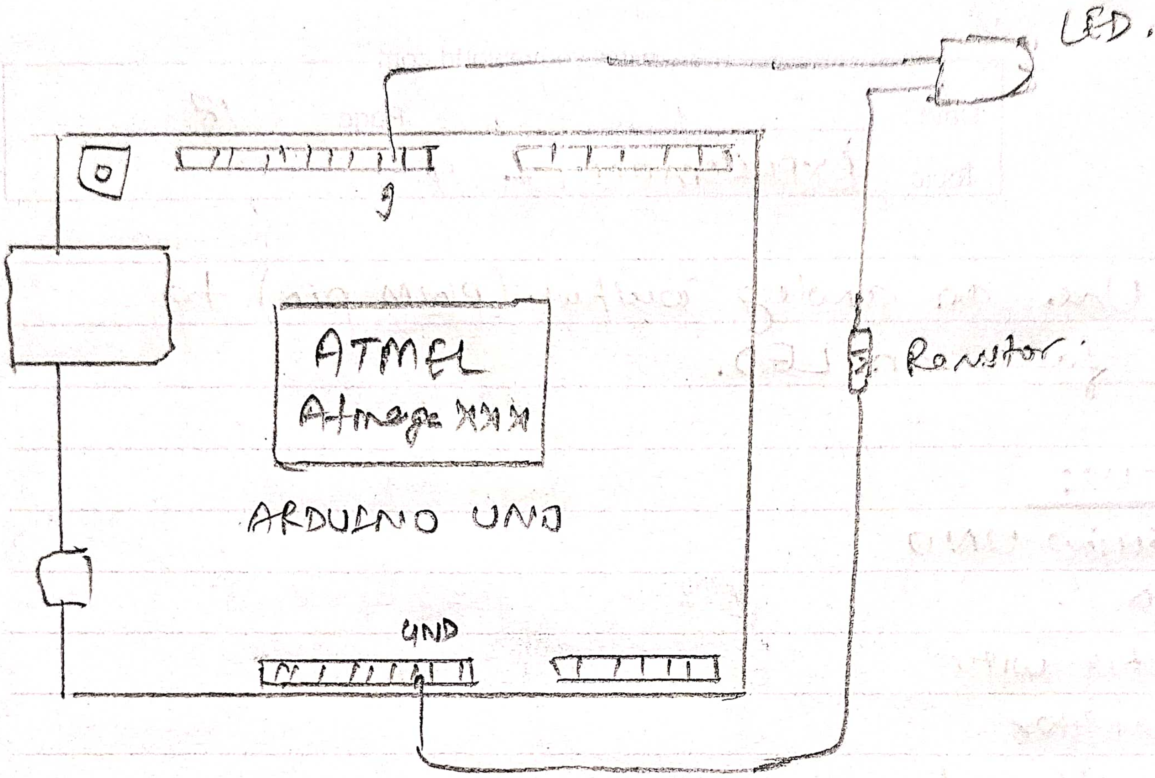
PWM is a technique used to stimulate an analog output using digital means. With PWM, the microcontroller produces a square wave with a variable duty cycle — the ratio of time the signal is high (ON) to the time it is LOW (OFF) within each period. By changing this ratio, we can effectively control the average voltage applied to the LED, altering its brightness.

PROCEDURE:

Step 1: Connect the LED to the arduino as your desired PWM pins.

Step 2: Upload the sketch.

• CONNECTION:



Topic : Experiment - 9

"code"

```
const int ledPin = 9;
void setup(){
    pinMode(ledPin , OUTPUT);
  }
void loop(){
    for( int brightnese =0; brightness <=255;
            brightness ++){
    analogwrite (ledPin, brightness);
    delay (10);
  }
    for(int brightness =255; brightness >=0;
            brightness --){
    analogWrite(ledPin , brightness);
    delay(10);
  }
 }
```

Step 3 : after uploading the sketch we can see our LED is fading.

- CONCLUSION :

we successfully get the result of fading of LED using PWM pin.

Topic : EXPERIMENT- 10

- AIM: Servo Motor Control using PWM.

- APPARATUS :
  - → Arduino Board UNO.
  - → Servo Motor
  - → Jumper wires.

- THEORY:

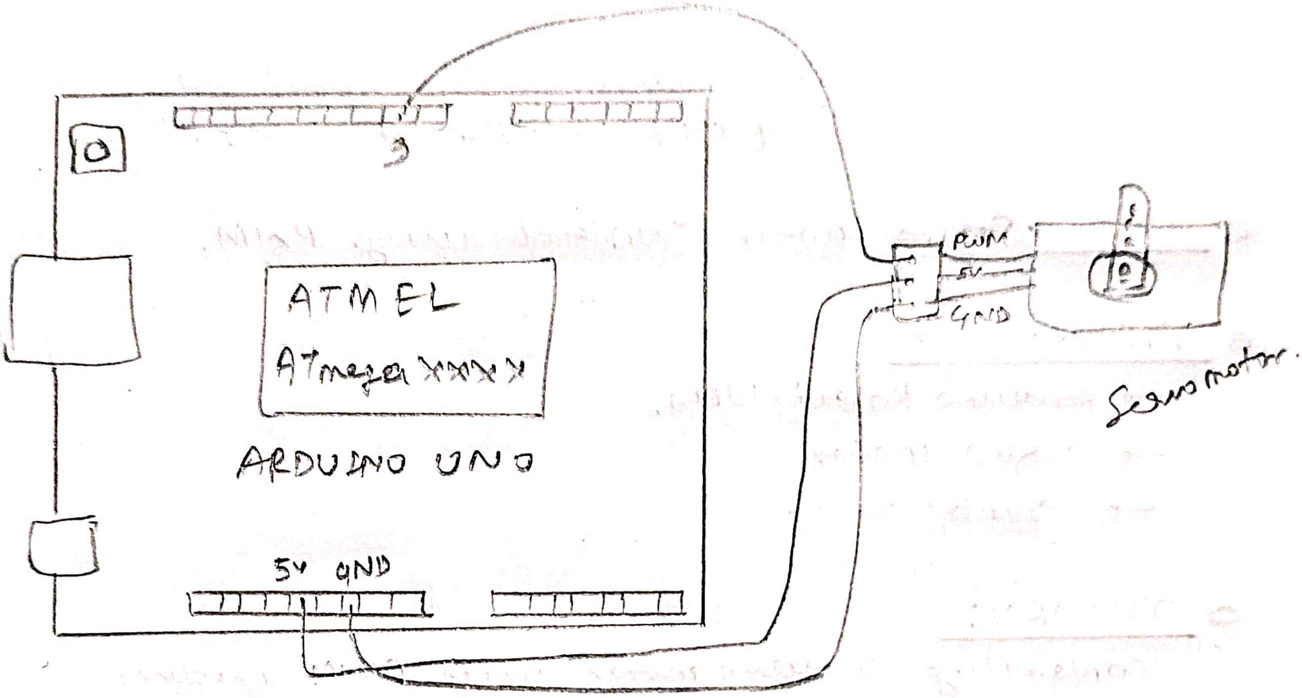
controlling a servo motor using PWM involves generating specific pulse signals to position the servo's shaft accurately. Servo motors are widely used for requiring precise angular positioning. The pulse width determines the servo motor's position, with a typical range of 1000 to 2000 microseconds.

Servo motor consist of a motor, feedback mechanism, and control circuitry. They can rotate to a specific angle within a given range, typically 0 to 180 degrees, based on the PWM signal's pulse width.

- PROCEDURE:

Step1: connect the servo motor to arduino as given next page connection.

- CONNECTION:



Topic : Experiment - 10

Step 2: Upload the sketch.

"code"

```
#include <servo.h>
  Servo   myServo;
  int servoPin = 9;
  void setup(){
   myServo. attach (servoPin);
  }
  void loop(){
   for (int angle = 0; angle <= 180; angle ++){
   myServo.write (angle);
   delay (15);
   }
   delay (1000);
   for (int angle = 180; angle >= 0; angle--){
   myServo.write (angle);
   delay (15);
   }
   delay (1000);
   }
```

- **CONCLUSION :**

After performing this experiment, we see our servo motor is controlled to a angle between 0° to 180°.