Part A

Class B Tech CSE 4th Year **Sub: Internet of Things Lab**

Aim: Introduction to Arduino platform and programming

Prerequisite: Basics of programming, microcontrollers and basic electronics

Outcome: Understanding of basic IoT framework and programming with Arduino IDE

Theory:

1. Study of fundamental IoT and its components

IoT refers to the network of physical objects (devices) embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet.

Core Components of IoT

a. **Devices/Sensors:** Devices or sensors are the physical components that collect data from the environment. They can measure temperature, humidity, light, motion, and more.
b. **Connectivity:** Connectivity is essential for devices to communicate with each other and with central systems. This can be achieved through various communication protocols such as Wi-Fi, Bluetooth, Zigbee, and cellular networks.
c. **Data Processing:** Collected data needs to be processed to extract meaningful insights. This can be done locally on the device (edge computing) or sent to centralized servers (cloud computing).
d. **User Interface:** Users interact with IoT systems through user interfaces, which can be mobile apps, web interfaces, or voice commands.
e. **Actuators:** Actuators are devices that perform actions based on processed data. For example, a thermostat adjusting the temperature or a smart lock securing a door
f. **Cloud and Data Storage:** The cloud provides scalable storage and processing power for IoT data. It enables remote access and management of IoT devices.
g. **Security:** Security is crucial in IoT systems to protect data integrity, privacy, and prevent unauthorized access.

2. Understanding of Arduino boards, Arduino IDE, and Serial monitor.

Arduino Boards: Arduino boards are open-source microcontroller platforms designed for easy use in various electronic projects. They consist of both hardware (the board) and software (the Arduino IDE).

Common Arduino Boards:

- **Arduino Uno:** One of the most popular boards, featuring an ATmega328P microcontroller, 14 digital I/O pins, and 6 analog input pins.
- **Arduino Mega:** Offers more I/O pins and memory, suitable for larger projects.
- **Arduino Nano:** A smaller, breadboard-friendly version of the Uno.
- **Arduino Leonardo:** Features a microcontroller with built-in USB communication, allowing it to be recognized as a mouse or keyboard.

Components:

- **Microcontroller:** The brain of the board that executes the code.
- **Digital I/O Pins:** Used for digital input and output operations.
- **Analog Input Pins:** Used to read analog signals from sensors.
- **Power Supply:** Can be powered via USB or an external power source.

- **USB Port:** For uploading code from the Arduino IDE and serial communication.
- **Reset Button:** Resets the microcontroller.

Arduino IDE: The Arduino IDE is an open-source software platform used to write, compile, and upload code to Arduino boards. It provides a user-friendly interface and various tools to help developers create and debug their projects.

Features

- **Code Editor:** A simple text editor for writing sketches (Arduino programs).
- **Library Manager:** Allows users to include and manage libraries that provide additional functionalities.
- **Serial Monitor:** A built-in tool for debugging and communicating with the Arduino board.
- **Board and Port Selection:** Options to select the specific Arduino board and communication port being used.
- **Examples and Tutorials:** Preloaded example sketches and tutorials to help users get started.

Serial Monitor: The Serial Monitor is a tool within the Arduino IDE that allows for real-time communication between the Arduino board and a computer. It is useful for debugging and monitoring the output from the board.

Features:

- **Input Field:** Allows users to send data to the Arduino board.
- **Output Window:** Displays data sent from the Arduino board to the computer.
- **Baud Rate Selection:** Allows users to set the communication speed (e.g., 9600 baud).
- **Newline Characters:** Options to automatically append newline characters to the input.

Part B

1. LED Blinking Program

Components:

- **LED (Light Emitting Diode):** An LED is a semiconductor device that emits light when current flows through it. LEDs have two terminals: the anode (positive) and the cathode (negative).
- **Digital Pins:** The Arduino board has several digital input/output (I/O) pins that can be programmed to control various components, such as LEDs.
- **Microcontroller:** The Arduino microcontroller is programmed to control the timing and state (ON/OFF) of the LED.
- **Delay Function:** The delay() function in Arduino pauses the program for a specified number of milliseconds, creating the on-off blinking effect.

Steps:

a. Connect the LED:

- Place the LED on the breadboard.
- Connect the anode (longer leg) of the LED to a digital pin (e.g., pin 13) on the Arduino using a jumper wire.

- Connect the cathode (shorter leg) of the LED to one end of the resistor.
- Connect the other end of the resistor to the GND (ground) pin on the Arduino.

b. Double-check connections:

- Ensure that the LED is connected correctly: anode to the digital pin and cathode to ground through the resistor.

Program:

```
int ledPin = 13;  // Write pin of output device

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  Serial.println("Hello Arduino!!!");
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH);
  Serial.println("LED ON");
  delay(500);
  digitalWrite(ledPin, LOW);
  Serial.println("LED OFF");
  delay(500);
}
```

2. LED control using serial monitor and buttons.

Steps:

Connect the LED:

- Place the LED on the breadboard.
- Connect the anode (longer leg) of the LED to a digital pin (e.g., pin 13) on the Arduino using a jumper wire.
- Connect the cathode (shorter leg) of the LED to one end of the resistor.
- Connect the other end of the resistor to the GND (ground) pin on the Arduino.

Connect the Button:

- Place the push button on the breadboard.
- Connect one leg of the button to a digital pin (e.g., pin 2) on the Arduino.
- Connect the opposite leg of the button to the GND pin on the Arduino.
- Connect a pull-up resistor (10k ohms) between the button leg connected to the digital pin and the 5V pin on the Arduino. Alternatively, you can use the internal pull-up resistor in the Arduino code.

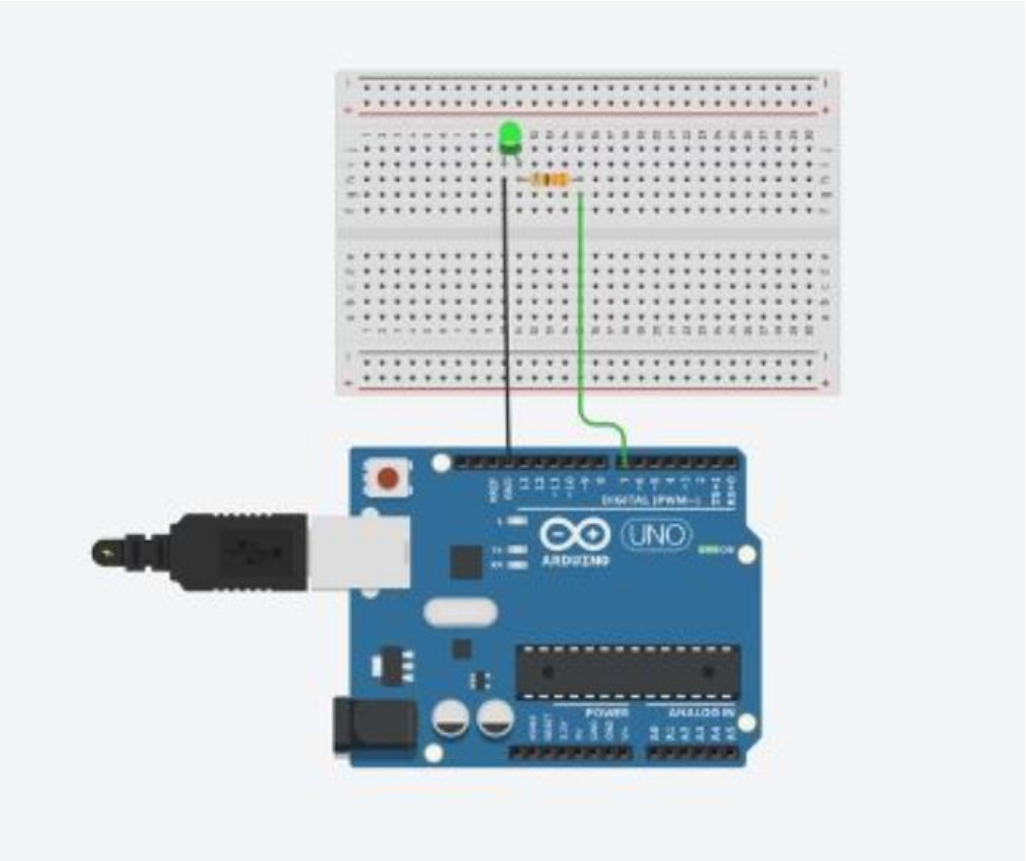
Program:

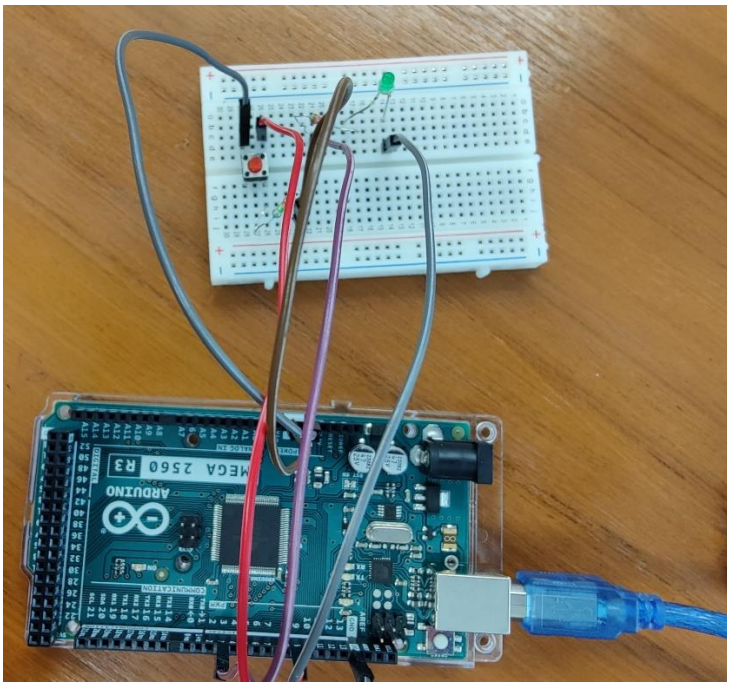
```
int ledPin = 3;
int buttonPin = 9;
int buttonStatus = 0;
```

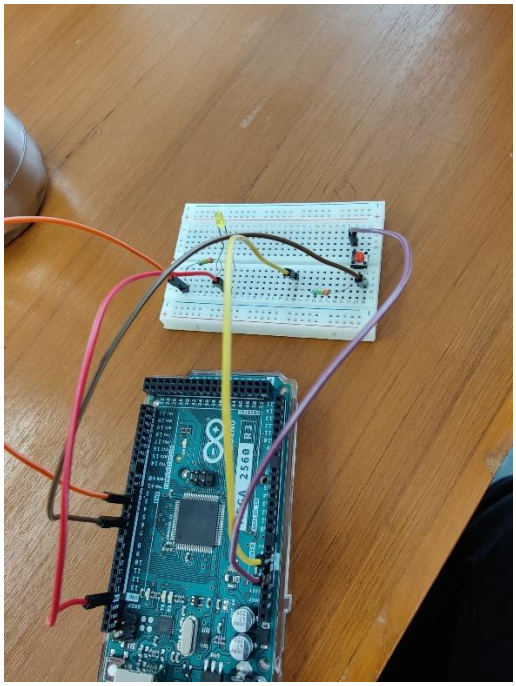
```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT); // Button in input device
  Serial.println("Hello Arduino!!!");
}

void loop() {
  // put your main code here, to run repeatedly:
  buttonStatus = digitalRead(buttonPin);
  if (buttonStatus == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Output:







Observation & Learning:

The LED blinking program effectively demonstrates fundamental Arduino programming concepts, such as controlling digital outputs and using delay functions to create timed effects. This basic example highlights the simplicity and power of Arduino for developing and testing microcontroller-based projects.

The combined use of the Serial Monitor and a physical button to control the LED showcases the Arduino's capability to handle multiple input methods and real-time communication. This approach emphasizes the versatility of Arduino in integrating manual and digital controls, providing a robust foundation for more complex projects involving user interactions and serial communication.

Conclusion:

The LED blinking program and the LED control experiment using the Serial Monitor and a button both illustrate key aspects of Arduino functionality. The blinking program demonstrates basic digital output and timing, while the combined control methods highlight the flexibility of Arduino in integrating multiple input sources. Together, these experiments showcase the simplicity and versatility of Arduino for developing interactive and responsive electronic projects.