**1. Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to blink LED. c. Write down the observations on Input and Output d. Write down the Result and Conclusion**

**a. Block Diagram / Pin Diagram**

You need to draw a block diagram or pin diagram of the **Arduino Uno** interfacing with one of the following sensors:

* **IR Sensor** (for object detection)
* **Temperature Sensor** (e.g., LM35, DHT11)
* **Camera** (e.g., OV7670)

Each diagram should include:

* Power connections (VCC, GND)
* Communication interface (Analog/Digital/PWM/I2C/SPI/UART)
* Connection to Arduino Uno pins

**b. C++ Program to Blink an LED**

Here is a simple **C++ (Arduino)** program to blink an LED connected to **pin 13** of the Arduino Uno:

// Define LED pin

const int ledPin = 13;

void setup() {

pinMode(ledPin, OUTPUT); // Set LED 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

}

**c. Observations on Input and Output**

* **Input:** No external input is required; the Arduino runs the code independently.
* **Output:** The LED blinks **ON** and **OFF** with a 1-second delay.

**d. Result and Conclusion**

* **Result:** The LED successfully blinks at a 1-second interval, confirming the correct working of the Arduino Uno and the uploaded C++ program.
* **Conclusion:** The experiment demonstrates the basic working of Arduino Uno’s **digital output** functionality using **pinMode()** and **digitalWrite()** functions. This can be extended for controlling other electronic devices.

Here’s a structured response for your requirement:

**2. Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to turn ON/OFF buzzer. c. Write down the observations on Input and Output d. Write down the Result and Conclusion**

**a. Block Diagram / Pin Diagram**

Since the **Internal Examiner will assign one option**, I can provide general interfacing details for each sensor with **Arduino Uno**:

1. **IR Sensor (Object Detection)**
   * **VCC** → 5V (Arduino)
   * **GND** → GND (Arduino)
   * **OUT** → Digital Pin (e.g., D2 on Arduino)
2. **Temperature Sensor (LM35/DHT11)**
   * **VCC** → 5V (Arduino)
   * **GND** → GND (Arduino)
   * **OUT (Analog Signal)** → Analog Pin (e.g., A0 on Arduino)
3. **Camera (OV7670)**
   * Uses **I2C and SPI Communication**
   * **SCL & SDA** → I2C Pins (A4, A5 on Arduino)
   * **VSYNC, HREF, PCLK** → Digital I/O Pins
   * **D0-D7 (Data Bus)** → Digital Pins D2-D9

Let me know which **specific sensor** you want the block diagram for! 😊

**b. C++ Program to Turn ON/OFF Buzzer**

Here’s a simple **Arduino C++ code** to turn ON/OFF a buzzer connected to **Pin 8**:

const int buzzer = 8; // Define buzzer pin

void setup() {

pinMode(buzzer, OUTPUT); // Set buzzer pin as output

}

void loop() {

digitalWrite(buzzer, HIGH); // Turn ON buzzer

delay(1000); // Wait 1 second

digitalWrite(buzzer, LOW); // Turn OFF buzzer

delay(1000); // Wait 1 second

}

**c. Observations on Input and Output**

* **Input:** No external input required; the program runs independently.
* **Output:**
  + The **buzzer turns ON** for **1 second**, then **OFF** for **1 second**, repeating continuously.
  + Audible **beeping sound** is heard in a 1-second interval.

**d. Result and Conclusion**

* **Result:** The buzzer successfully produces a **1-second ON/OFF sound cycle**, confirming the correct working of the Arduino Uno and the program.
* **Conclusion:** This experiment demonstrates the control of a **buzzer (digital output device)** using **Arduino Uno**. The same logic can be applied to control other actuators like motors or relays.

1. **Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to toggle two LED’s. c. Write down the observations on Input and Output d. Write down the Result and Conclusion**

**a. Block Diagram / Pin Diagram**

Since the **Internal Examiner will assign one option**, I’ll describe the interfacing of each sensor with **Arduino Uno**. Let me know if you need a diagram for a specific sensor!

**1. IR Sensor (Object Detection)**

* **VCC** → **5V (Arduino)**
* **GND** → **GND (Arduino)**
* **OUT** → **Digital Pin (e.g., D2 on Arduino)**

**2. Temperature Sensor (LM35/DHT11)**

* **VCC** → **5V (Arduino)**
* **GND** → **GND (Arduino)**
* **OUT (Analog Signal)** → **Analog Pin (e.g., A0 on Arduino)**

**3. Camera (OV7670 - Image Processing)**

* **VCC** → **3.3V (Arduino)**
* **GND** → **GND (Arduino)**
* **SCL & SDA** → **I2C Pins (A4, A5 on Arduino)**
* **D0-D7 (Data Bus)** → **Digital Pins D2-D9**

**b. C++ Program to Toggle Two LEDs**

This **Arduino C++ code** toggles two LEDs connected to **pins 9 and 10**:

const int led1 = 9; // Define first LED pin

const int led2 = 10; // Define second LED pin

void setup() {

pinMode(led1, OUTPUT); // Set LED1 pin as output

pinMode(led2, OUTPUT); // Set LED2 pin as output

}

void loop() {

digitalWrite(led1, HIGH); // Turn ON LED1

digitalWrite(led2, LOW); // Turn OFF LED2

delay(1000); // Wait 1 second

digitalWrite(led1, LOW); // Turn OFF LED1

digitalWrite(led2, HIGH); // Turn ON LED2

delay(1000); // Wait 1 second

}

**c. Observations on Input and Output**

* **Input:** No external input is required; the Arduino runs the code automatically.
* **Output:**
  + **LED1 turns ON while LED2 turns OFF**, then they **toggle** after **1 second**.
  + The LEDs **alternate their states** continuously in a **1-second interval**.

**d. Result and Conclusion**

* **Result:** The two LEDs successfully **toggle alternately** every **1 second**, proving that the Arduino Uno is correctly programmed.
* **Conclusion:** This experiment demonstrates how to control **multiple digital outputs** using **Arduino Uno**. The same logic can be extended for controlling **motors, buzzers, or other electronic components**.