SAPTHAGIRI NPS UNIVERSITY

UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

INTERNET OF THINGS (24BECSE307)

LAB MANUAL

DEPT OF CSE

SAPTHAGIRI NPS UNIVERSITY

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

List of Experiments

Introduction to IoT Development Boards: Arduino IDE Software, ESP32, and Raspberry Pi(micro python)

PART A (Simulation Experiments)

1. Develop a program to illustrate the working of LED with a push button.
2. Write a program to blink an LED with different times and duration using the concept of user defined function.
3. Develop a program to rotate servo motor both in clockwise and anticlockwise direction.
4. Develop a program to simulate the working of LCD and print the given text on LCD.
5. Develop a program to calculate the distance of an object using ultrasonic sensor.
6. Develop a program to interface temperature sensor to read the room temperature, humidity and heat index and print the readings on the serial monitor.

PART B (Hardware experiments)

7. Develop a program to illustrate the working of LED with a push button using ESP32/Aurdino UNO.
8. Write a program to blink an LED with different times and duration using the concept of user defined function using ESP32/Aurdino UNO.
9. Develop a program to simulate the working of LCD and print the given text on LCD using ESP32.
10. Develop a program to interface temperature sensor to read the room temperature, humidity and heat index and print the readings on the serial monitor using ESP32.

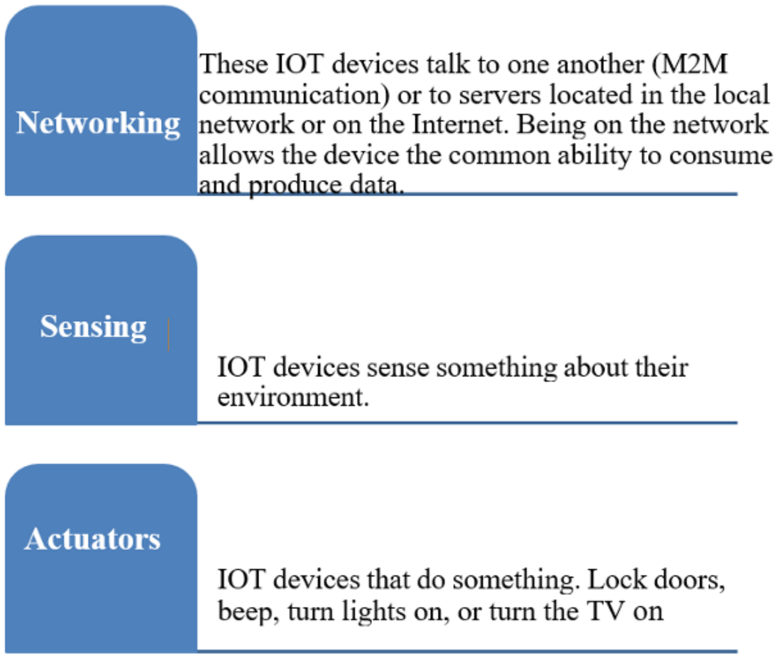
Introduction to Internet of Things (IOT)

IOT stands for “Internet of Things”. The IOT is a name for the vast collection of “things” that are being networked together in the home and workplace (up to 20 billion by 2020 according to Gardner, a technology consulting firm). The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Internet of things has been considered a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable.

Applications:

- Smart home
- Medical and healthcare
- Transportation
- Industrial applications
- Agriculture
- Building and home automation
- Environmental monitoring
- Military applications

Characteristics of the IOT



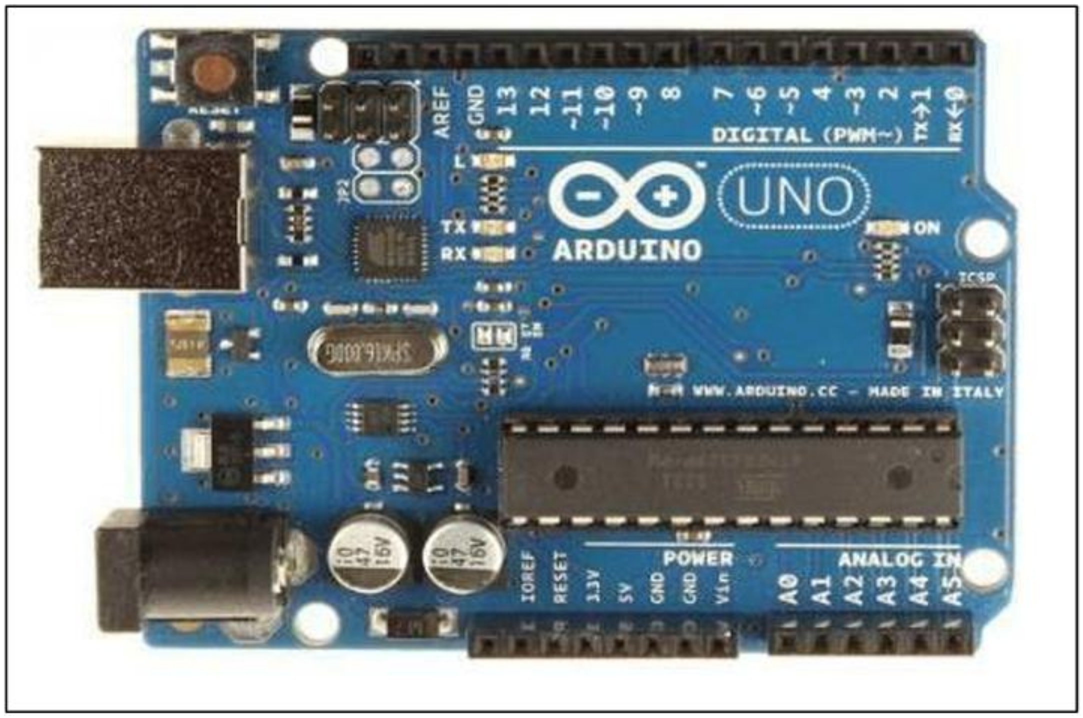
Networking
These IOT devices talk to one another (M2M communication) or to servers located in the local network or on the Internet. Being on the network allows the device the common ability to consume and produce data.

Sensing
IOT devices sense something about their environment.

Actuators
IOT devices that do something. Lock doors, beep, turn lights on, or turn the TV on

Arduino

In IoT applications, the Arduino is used to collect the data from the sensors/devices to send it to the internet and receives data for purpose of control of actuators. Arduino Uno The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software.



Aurdino UNO

Features of the Arduino

1. Arduino boards can read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
2. The board functions can be controlled by sending a set of instructions to the microcontroller on the board via Arduino IDE.
3. Arduino IDE uses a simplified version of C++, making it easier to learn to program.
4. Arduino provides a standard form factor that breaks the functions of the micro- controller into a more accessible package.

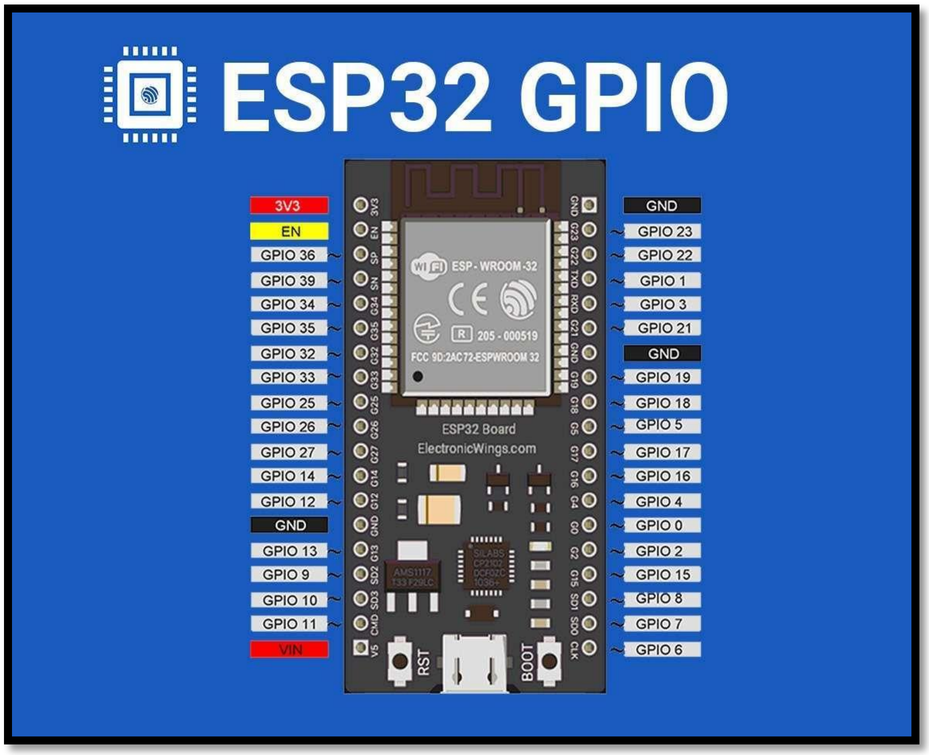
PIN Functions

• LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off. • VIN: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. • 5V:

This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board. • V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA. • GND: Ground pins. • IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V. • Reset: Typically used to add a reset button to shields that block the one on the board.

ESP32

- **Dual-core processing:** Most ESP32 variants feature two 32-bit Xtensa LX6 microprocessor cores that can be controlled and powered independently. This provides more processing power for demanding tasks compared to its predecessor, the ESP8266.
- **Wireless connectivity:** It supports both 2.4 GHz Wi-Fi (802.11 b/g/n) and dual-mode Bluetooth (Classic and BLE), allowing for seamless wireless communication.
- **Peripherals:** It features a rich set of peripherals, including multiple GPIOs, 12-bit Analog-to-Digital Converters (ADCs), 8-bit Digital-to-Analog Converters (DACs), I²C, SPI, UART, PWM, and capacitive touch sensors.
- **Low power consumption:** The ESP32 is designed for energy efficiency, with an ultra-low-power co-processor and multiple power modes, including a deep sleep mode that extends battery life for portable devices.
- **Security:** It offers advanced security features like secure boot, flash encryption, and hardware-accelerated cryptographic algorithms for a more secure connection.
- **Memory:** It typically includes 520 KiB of SRAM and 4 MiB of flash memory, with support for additional external memory.

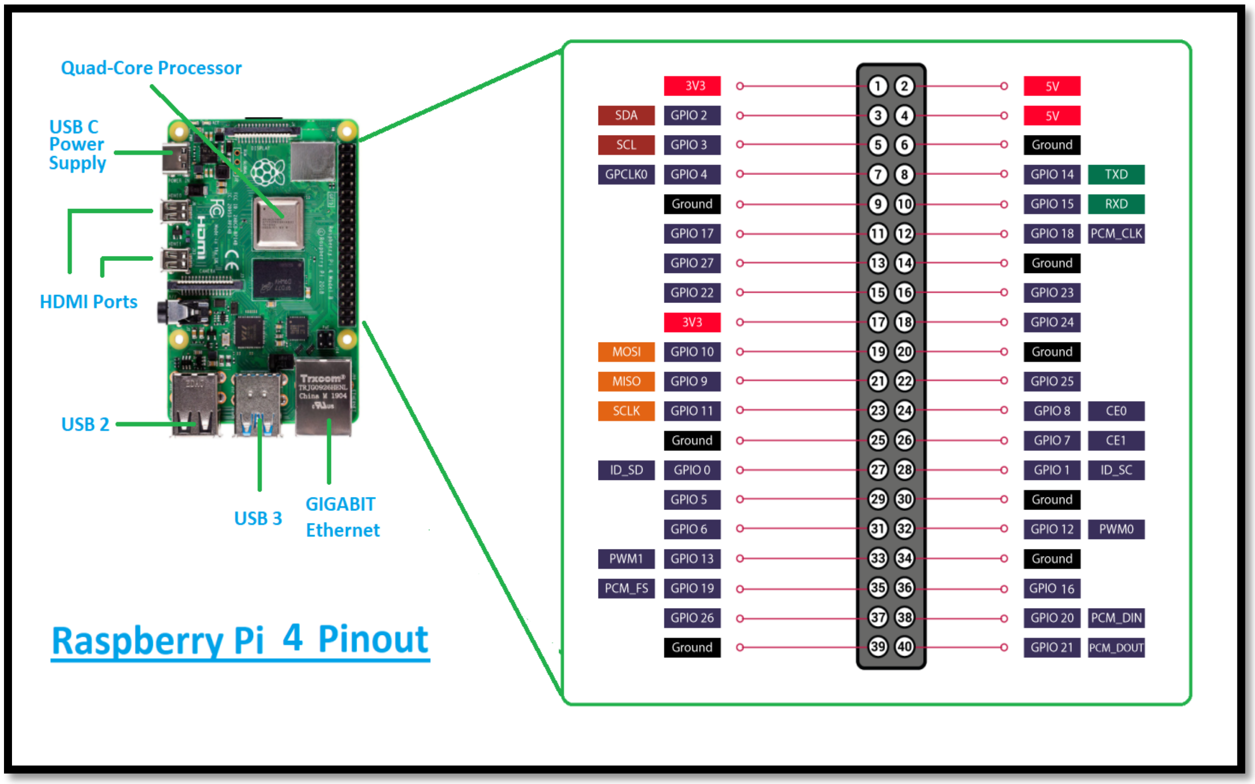


ESP32

Raspberry Pi 4

- **Processor:** Broadcom BCM2711, a 64-bit quad-core ARM Cortex-A72 SoC clocked at 1.5 GHz (or 1.8 GHz in later versions).
- **RAM:** Available in 1GB, 2GB, 4GB, and 8GB LPDDR4 variants, offering significantly improved performance over the Pi 3.
- **Connectivity:**
- ➢ Wireless: Dual-band 2.4 GHz and 5.0 GHz IEEE 802.11ac Wi-Fi and Bluetooth 5.0.
- ➢ Wired: True Gigabit Ethernet, with no bottleneck from a shared USB connection like on previous models.
- ➢ **USB:** Two USB 3.0 ports for faster data transfer and two USB 2.0 ports.
- **Video and Multimedia:**
- ➢ Dual Micro-HDMI: Supports up to two 4K displays at 60 frames per second.

- ➢ Hardware Decode: Capable of H.265 (4Kp60) and H.264 (1080p60) video decoding.
- **Storage:** A micro SD card slot is used for the operating system and data storage.
- **GPIO:** A standard 40-pin GPIO header provides backwards-compatibility with older Pi boards and allows interaction with electronics.
- **Power:** Uses a USB-C connector for power and requires a 5V/3A power supply.


Quad-Core Processor
USB C Power Supply
HDMI Ports
USB 2
USB 3
GIGABIT Ethernet
Raspberry Pi 4 Pinout
3V3 1 2 5V
SDA GPIO 2 3 4 5V
SCL GPIO 3 5 6 Ground
GPCLK0 GPIO 4 7 8 GPIO 14 TXD
Ground 9 10 GPIO 15 RXD
GPIO 17 11 12 GPIO 18 PCM_CLK
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 17 18 GPIO 24
MOSI GPIO 10 19 20 Ground
MISO GPIO 9 21 22 GPIO 25
SCLK GPIO 11 23 24 GPIO 8 CE0
Ground 25 26 GPIO 7 CE1
ID_SD GPIO 0 27 28 GPIO 1 ID_SC
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 PWM0
PWM1 GPIO 13 33 34 Ground
PCM_FS GPIO 19 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 PCM_DIN
Ground 39 40 GPIO 21 PCM_DOUT

GPIOs OF RASPBERRY PI 4

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Experiment 1

A program to illustrate the working of LED with a push button.

Introduction:

Push-button is a very simple mechanism which is used to control electronic signal either by blocking it or allowing it to pass. This happens when mechanical pressure is applied to connect two points of the switch together. Push buttons or switches connect two points in a circuit when pressed. When the push-button is released, there is no connection between the two legs of the push-button. Here it turns on the built-in LED on pin 11 when the button is pressed. The LED stays ON as long as the button is being pressed.

LED

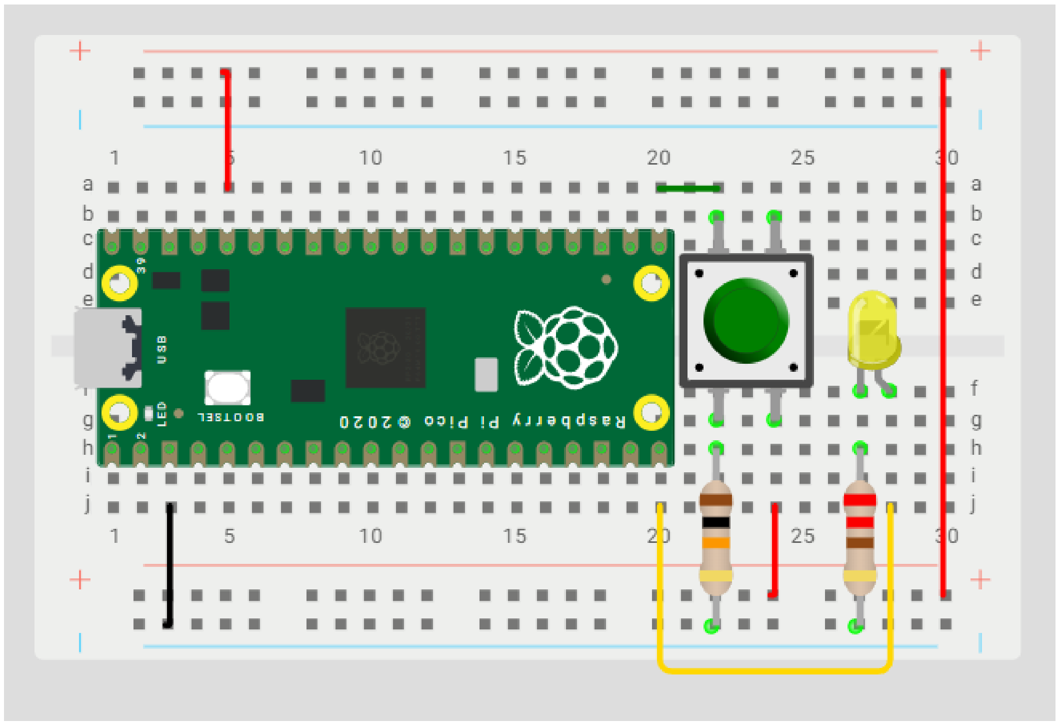
Pin definition	
Long pin	+5V
Short pin	GND

Push Button

Hardware Required:

Component Name	**Quantity**
Raspberry Pi	1
LED	1
Push Button	1
220Ω resistor	1
10KΩ resistor	1
USB Cable	1
Breadboard	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Insert the push button into your breadboard and connect it to the digital pin GPIO Pin 16 which act as INPUT.
2. Insert the LED into the breadboard. Attach the negative leg to GPIO pin 15 of the Raspberry Pi, and the positive leg via the 220-ohm resistor to GND. The pin 15 is taken as OUTPUT.
3. Upload the code as given below.
4. Press the push-button to control the ON state of LED. This sketch works by setting pin 16 as for the push button as INPUT and pin 15 as an OUTPUT to power the LED. The initial state of the button is set to OFF. After that the code run a loop that continually reads the state from the pushbutton and sends that value as voltage to the LED. The LED will be ON accordingly.

Code

```
# modules
from machine import Pin
from time import sleep

LED = Pin(15, Pin.OUT)      # creating LED object, setting it as OUT
BUTTON = Pin(16, Pin.IN)    # creating Push Button object, setting it as IN

# continuously read the signal from the push button while the board has power
while True:
    # if the signal from the button is HIGH (button is pressed), turn LED on
    if BUTTON.value() == 1:
        LED.on()
        sleep(0.1)

    # otherwise, turn the LED off
    else:
        LED.off()
```

Observation Table:

Sl No.	Push button State	LED State
1		
2		

Result

The simulation of hardware connections were made and the program was executed to control of LED using Push button.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Experiment 2

A program to blink an LED with different times and duration using the concept of user defined function.

Introduction:

- The GPIO pin connected to the LED is initialized as an output using the Pin object.
- A user-defined function blink_led is written to make the LED blink a specified number of times, with customizable ON and OFF durations.
- The blinking action is managed using a for loop, which repeatedly turns the LED ON and OFF with delays in between.

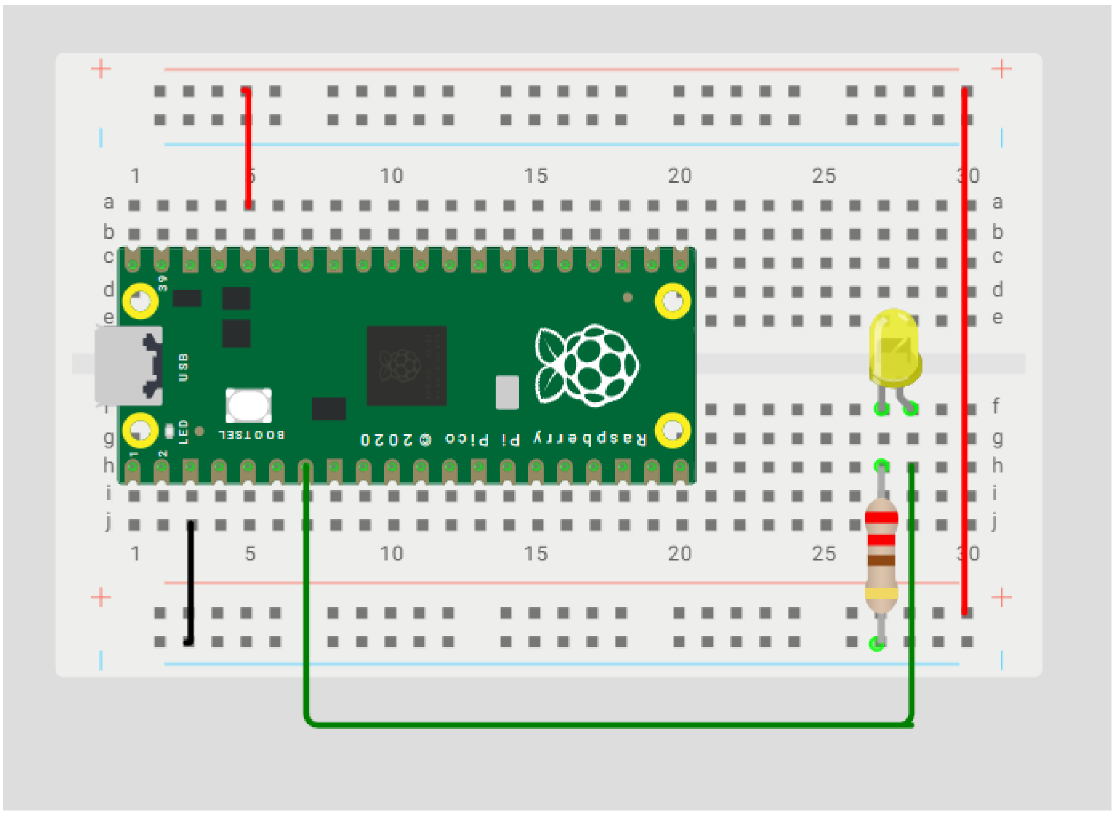
LED

Pin definition	
Long pin	+5V
Short pin	GND

Hardware Required:

Component Name	**Quantity**
Raspberry Pi	1
LED	1
220Ω resistor	1
USB Cable	1
Breadboard	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Insert the LED into the breadboard. Attach the negative leg to GPIO pin 5 of the Raspberry Pi, and the positive leg via the 220-ohm resistor to GND. The pin 5 is taken as OUTPUT.
2. Upload the code as given below.
3. The LED will be ON according to the input parameters.

Code

```
from machine import Pin
from utime import sleep

# Initialize the LED pin
led = Pin(5, Pin.OUT)  # Replace 5 with the GPIO pin number connected to the
LED

# User-defined function to blink the LED
def blink_led(times, duration_on, duration_off):
    """
    Blink the LED a specified number of times with custom on/off durations.

    :param times: Number of times to blink the LED
    :param duration_on: Duration (in seconds) the LED stays ON
    :param duration_off: Duration (in seconds) the LED stays OFF
    """
    for _ in range(times):
        led.value(1)  # Turn LED ON
        sleep(duration_on)
        led.value(0)  # Turn LED OFF
        sleep(duration_off)

# Example usage of the function
while True:
    blink_led(3, 0.5, 0.5)  # Blink 3 times with 0.5s ON and 0.5s OFF
    sleep(2)  # Pause before the next sequence
    blink_led(5, 0.2, 0.8)  # Blink 5 times with 0.2s ON and 0.8s OFF
    sleep(3)  # Pause before the next sequence
```

Result

The simulation of hardware connections were made and the program was executed to control the blinking of LED for various pattern.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

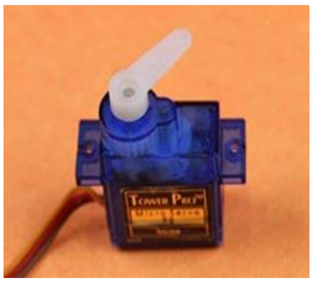
Experiment 3

A program to rotate servo motor both in clockwise and anticlockwise direction.

Introduction:

A servo motor is a compact and precise actuator widely used in robotics, automation, and control systems. It is capable of rotating its shaft to a specified angular position, typically between 0° and 180°. Controlling a servo motor requires a Pulse Width Modulated (PWM) signal, where the duty cycle of the pulse determines the angle of rotation.

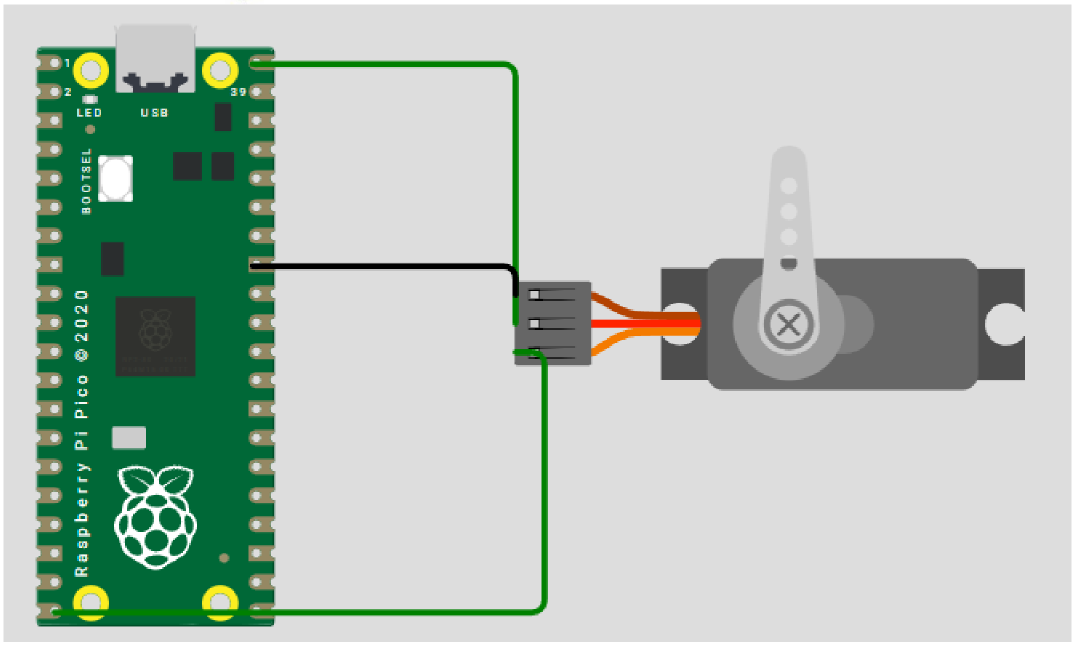
In this experiment, we use a Raspberry Pi Pico, a microcontroller board based on the RP2040 chip, to control a servo motor in both clockwise and anticlockwise directions. The Pico generates PWM signals on its GPIO pins, which makes it well-suited for servo applications. By writing a MicroPython program, we can sweep the servo from 0° to 180° and back, thereby achieving bidirectional rotation.



Hardware Required:

Component Name	Quantity
Raspberry Pi	1
Servomotor	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Connect the **signal (PWM)** wire of the servo motor to **GPIO15** of Raspberry Pi Pico.
2. Connect the **VCC pin** of the servo motor to the **VBUS (5V)** pin of the Pico.
3. Connect the **GND pin** of the servo motor to the **GND** pin of the Pico.
4. Upload the code as given below.
5. Run the code and observe the rotation of servomotor.

Code

```
from machine import Pin, PWM
from time import sleep

# Initialize servo on GPIO pin 15
servo = PWM(Pin(15))
servo.freq(50)   # Standard servo frequency: 50 Hz

# Function to set angle
def set_angle(angle):
    duty = int(((angle / 180) * 5000) + 1000)  # Convert angle to duty cycle (1000–8000 range)
    servo.duty_u16(duty)

while True:
    # Clockwise rotation (0° → 180°)
    for angle in range(0, 181, 10):
        set_angle(angle)
        sleep(0.05)
    sleep(1)

    # Anticlockwise rotation (180° → 0°)
    for angle in range(180, -1, -10):
        set_angle(angle)
        sleep(0.05)

    sleep(1)
```

Result

The simulation of hardware connections were made and the program was executed to observe the rotation of motor .

Experiment 4

Develop a program to simulate the working of LCD and print the given text on LCD.

Introduction:

A Liquid Crystal Display (LCD) is a flat-panel display that uses liquid crystals to modulate light. The 16x2 LCD module is widely used in embedded systems because it can display 16 characters per line on 2 rows, making it ideal for projects that require text-based information output.

Unlike LEDs or 7-segment displays, the LCD allows the display of letters, numbers, and custom characters. By interfacing the LCD with a microcontroller such as the Arduino Uno or Raspberry Pi Pico, we can print messages and simulate the working of the LCD. The microcontroller sends data and control signals to the LCD either in parallel mode (4-bit/8-bit) or via I2C communication, depending on the hardware.

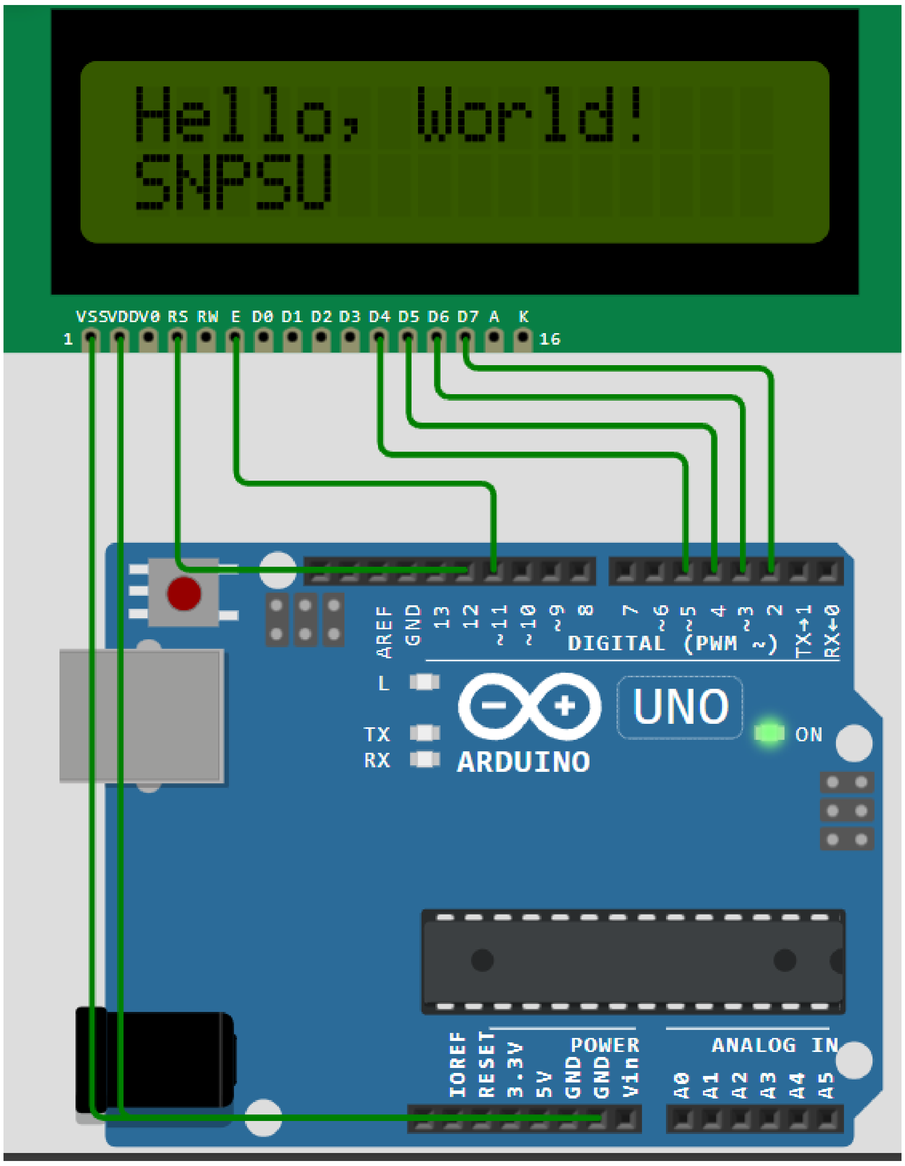
- **4-bit mode** is used (only D4–D7 are connected).
- **RS pin** selects whether you're sending **command** or **data**.
- **E pin** is the "enable strobe" that tells LCD when to read data.
- **VCC** and **GND** power the LCD module (5V operation).

This experiment demonstrates how to simulate the working of an LCD and display user-defined text.

Hardware Required:

Component Name	Quantity
Aurdino Uno	1
LCD (16x2)	1
Jumper wires	Several

Connection Diagram:


Hello, World!
SNPSU
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
1
16
AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX→1 RX←0
DIGITAL (PWM ~)
L
TX
RX
UNO
ARDUINO
ON
IOREF RESET 3.3V 5V GND GND Vin
POWER
ANALOG IN
A0 A1 A2 A3 A4 A5

Steps of working

1. Connect the LCD module to the microcontroller (Arduino Uno) as per the pin configuration.

LCD Pin	Arduino UNO Pin	Function
RS	D12	Register Select (data/command select)
E	D11	Enable signal
D4	D5	Data bit 4
D5	D4	Data bit 5

D6	D3	Data bit 6
D7	D2	Data bit 7
VDD	5V	Power supply (+5V)
VSS	GND	Ground (0V)

2. Upload the code as given below. Save with .ino extension.
3. Run the code and observe the LCD display "Hello, World!" on the first line and "SNPSU" on the second line..

Code

```
#include <LiquidCrystal.h>

// LCD pin mapping: RS=12, E=11, D4=5, D5=4, D6=3, D7=2
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);            // Initialize LCD as 16x2
  lcd.print("Hello, World!");   // Print text on first line
}

void loop() {
  lcd.setCursor(0, 1);          // Move to second line
  lcd.print("SNPSU"); // Print text
}
```

Result

The simulation of hardware connections were made and the program was executed to observe the output in lcd display.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Experiment 5

A program to calculate the distance of an object using ultrasonic sensor.

Introduction:

The HC-SR04 is a widely used ultrasonic distance sensor designed to measure the distance between the sensor and an object without physical contact. It works on the principle of ultrasonic sound waves and the time-of-flight method. The sensor transmits high-frequency sound waves (40 kHz), which travel through the air. When these waves hit an object, they are reflected back toward the sensor. By measuring the time taken for the echo to return, the distance to the object can be calculated. The working principle is as follows:

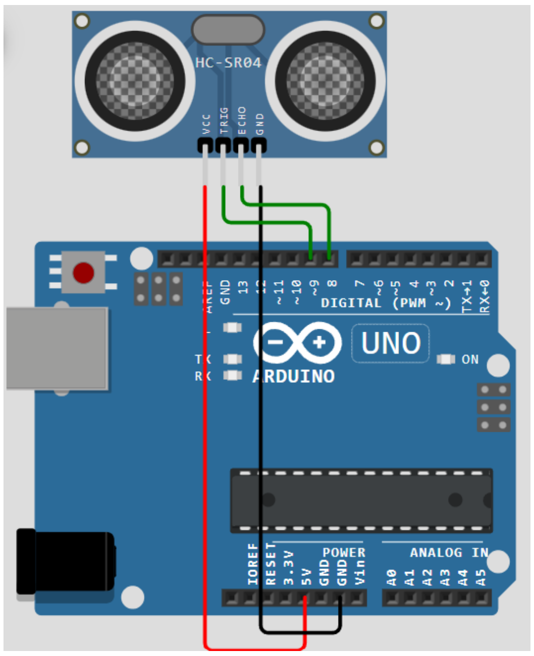
1. Arduino sends 10µs trigger pulse to TRIG.
2. Sensor transmits 8 ultrasonic waves (40 kHz).
3. Waves hit an obstacle and reflect back.
4. Sensor sets ECHO pin HIGH for the duration of travel.
5. Arduino measures the HIGH pulse width.
6. Distance is calculated using the formula:

Distance (cm)=$\frac{\boldsymbol{Time}\ (\mu s)\times \mathbf{0.03432}}{\mathbf{2}}$

Hardware Required:

Component Name	Quantity
Aurdino Uno	1
HC-SR04	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Connect HC-SR04 to Aurdino board as follows:

Ultrasonic Sensor Pin	**Arduino UNO Pin**	**Purpose**
VCC	5V	Powers the ultrasonic sensor
GND	GND	Common ground
TRIG	D9	Arduino sends a 10µs pulse to start measurement
ECHO	D8	Sensor outputs HIGH pulse proportional to distance

2. Upload the code as given below.
3. Run the code and observe the distance in serial monitor.
4. To change the distance in Wokwi, click on the **Ultrasonic Sensor** to see a **distance slider** (e.g., 10 cm → 400 cm). Move the slider . Then the value in Serial Monitor changes in real time

Code

```
#define TRIG_PIN 9
#define ECHO_PIN 8

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  // Send a 10µs pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Read the echo pulse duration
  long duration = pulseIn(ECHO_PIN, HIGH);

  // Convert time to distance (cm)
  float distance = duration * 0.0343 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(500);
}
```

Result

The simulation of hardware connections were made and the program was executed to observe the real time sensor output in serial monitor.

Experiment 6

A program to interface temperature sensor to read the room temperature, humidity and heat index and print the readings on the serial monitor.

Introduction:

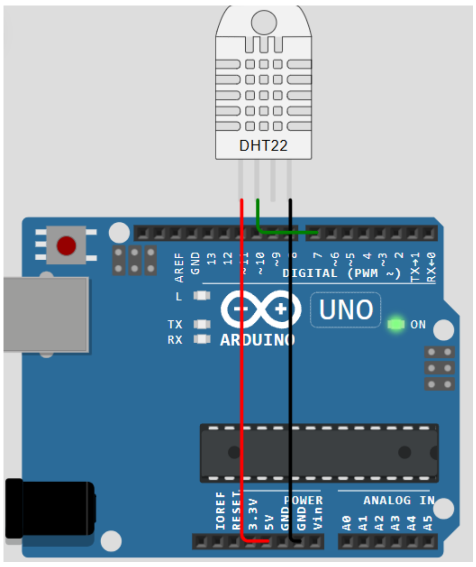
The DHT22 (also known as AM2302) is a low-cost, digital-output sensor used to measure temperature and relative humidity. It combines a capacitive humidity sensor and a thermistor in a single package and provides calibrated digital output via a single data line, making it simple to interface with microcontrollers like Arduino, Raspberry Pi, and ESP32. The working principle is as follows:

- Arduino sends a start signal to the DHT22 through the DATA pin.
- The DHT22 responds with a digital signal containing temperature and humidity values.
- Arduino decodes this signal using the DHT library.
- The code also computes the heat index using temperature + humidity formula.
- Finally, all values are printed on the Serial Monitor.

Hardware Required:

Component Name	Quantity
Aurdino UNO	1
DHT22	1
Jumper wires	Several

Connection Diagram:



Steps of working

1. Connect DHT22 sensor to Aurdino Board as follows:

DHT22 Pin	Arduino UNO Pin	Purpose
VCC	5V	Provides operating voltage to the sensor
GND	GND	Common ground
SDA	D7	Arduino reads temperature & humidity data

2. Upload the code as given below.
3. Run the Simulation to see the readings in serial monitor
4. Click on the DHT22 sensor in the workspace. Adjust the temperature and humidity sliders provided in Wokwi. Observe the updated readings (Temperature, Humidity, Heat Index) on the Serial Monitor.

Code

```
#include "DHT.h"

#define DHTPIN 7       // Pin connected to the DHT sensor
#define DHTTYPE DHT22  // Can be DHT11 or DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT22 Sensor Reading...");
  dht.begin();
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();        // Celsius
  float heatIndex = dht.computeHeatIndex(temperature, humidity, false);

  // Check if any reading failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  Serial.print("Heat Index: ");
  Serial.print(heatIndex);
  Serial.println(" °C");

  Serial.println("------------------------");

  delay(2000); // Wait 2 seconds before next reading
}
```

Result

The simulation of hardware connections were made and the program was executed to observe the real time sensor output in serial monitor.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Experiment 7

A program to illustrate the working of LED with a push button using ESP32/Aurdino UNO.

Introduction

This program demonstrates the basic concept of digital input and output using Arduino Uno or ESP32. A push button is used as a digital input device and an LED as an output device. When the button is pressed, the LED turns ON, and when released, the LED turns OFF. This simple experiment helps in understanding how microcontrollers read input signals and control output devices in real time.

Hardware Required

Component Name	Quantity
ESP32 Development Board (or Arduino)	1
Push Button	1
LED	1
220Ω Resistor (for LED)	1
Patch Chords	Several

Circuit Connection

- Connect LED anode (+) → GPIO pin 2 (ESP32) or pin 13 (Arduino Uno) via 220Ω resistor.
- Connect LED cathode (–) → GND.
- Connect push button:
  - One side → 3.3V/5V.
  - Other side → GPIO pin 4 (ESP32) or pin 7 (Arduino).
  - Add 10kΩ resistor from input pin to GND (pull-down)

Steps of working

- Plug the ESP32/Arduino into your computer using a USB cable.
- In Arduino IDE select **File → New**
- Type the code given below.
- Select the correct **Board** and **Port** in Arduino IDE.
- Click **Upload**.The IDE will compile and transfer the code to the board.
- Observe the output.

Code

```
int buttonPin = 4;    // Push button pin
int ledPin = 2;       // LED pin
int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);  // LED ON
  } else {
    digitalWrite(ledPin, LOW);   // LED OFF
  }
}
```

Result

The hardware connections were made and the program was executed to observe that LED turns ON when push button is pressed and turns OFF when push button is released.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Experiment 8

A program to illustrate LED Blinking with User-Defined Functions using ESP32/Aurdino UNO.

Introduction

This program demonstrates how to blink an LED using user-defined functions in Arduino Uno or ESP32. An LED is connected to a digital output pin of the microcontroller. A user-defined function is created to turn the LED ON and OFF with a specified delay. The function is called repeatedly in the main loop to make the LED blink at a regular interval.

Hardware Required

Component Name	Quantity
ESP32 Development Board (or Arduino)	1
LED	1
220Ω Resistor (for LED)	1
Patch Chords	Several

Circuit Connection

- Connect LED anode (+) to GPIO pin 2 (ESP32) or pin 13 (Arduino Uno) via 220Ω resistor.
- Connect LED cathode (–) to GND.

Steps of working

- Plug the ESP32/Arduino into your computer using a USB cable.
- In Arduino IDE select File → New
- Type the code given below.
- Select the correct **Board** and **Port** in Arduino IDE.
- Click **Upload**. The IDE will compile and transfer the code to the board.
- Observe the output.

Code

```
int ledPin = 2;   // LED pin (ESP32), use 13 for Arduino Uno
// User-defined function for blinking
void blinkLED(int delayTime) {
  digitalWrite(ledPin, HIGH);
  delay(delayTime);
  digitalWrite(ledPin, LOW);
  delay(delayTime);
}
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  blinkLED(200);   // Fast blink
  blinkLED(500);   // Medium blink
  blinkLED(1000);  // Slow blink
}
```

Result

The hardware connections were made and the program was executed to observe that LED blinks at 200ms , 500ms and 1000ms intervals repeatedly.

Experiment 9

Develop a program to simulate the working of LCD and print the given text on LCD using ESP32.

Introduction

This experiment demonstrates how to interface a TFT LCD display with ESP32 to display text or sensor data. The TFT display acts as a visual output unit, enabling real-time monitoring of system information. By using appropriate communication protocols such as SPI, the ESP32 sends data to the display, which can then be formatted to show messages, numerical values, or live sensor readings.

Hardware Required

Component Name	Quantity
ESP32 Development Board	1
TFT LCD Display (SPI/I2C interface)	1
Patch Chords	Several

Circuit Connection

TFT LCD Pin	ESP 32 Pin
VCC(Displayed as LED in Kit)	3.3 V or 5 V (module supports both)
GND	Ground
SCK (Clock)	GPIO 18
SDA / MOSI	GPIO 23
CS	GPIO 5
DC(Data/Command) (Displayed as A0 in Kit)	GPIO 2
RST	GPIO 4

Steps of working

1. Install **Adafruit GFX** and **Adafruit ST7735** libraries in Arduino IDE:
   - Go to **Sketch → Include Library → Manage Libraries**
   - Search for **"Adafruit GFX"** → Install
   - Search for **"Adafruit ST7735 and ST7789 Library"** → Install
2. Wire the TFT as per the pins defined (CS=5, DC=2, RST=4, MOSI=23, SCK=18, VCC=3.3/5V, GND=GND).
3. Upload the code.

SAPTHAGIRI NPS UNIVERSITY
UNMATCHED EXCELLENCE, UNLIMITED POTENTIAL

Code

```
#include <Adafruit_GFX.h>      // Core graphics library
#include <Adafruit_ST7735.h>   // Hardware-specific library for ST7735
// Define connections (adjust if wired differently)
#define TFT_CS     5    // Chip Select
#define TFT_DC     2    // Data/Command
#define TFT_RST    4    // Reset
// Create display object
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
void setup() {
  // Initialize the display
  tft.initR(INITR_BLACKTAB);   // Use INITR_GREENTAB / REDTAB / BLACKTAB
depending on your LCD
  tft.fillScreen(ST77XX_BLACK);

  // Set text properties
  tft.setRotation(1);          // 0–3: rotate display (try 1 for landscape)
  tft.setTextColor(ST77XX_WHITE);
  tft.setTextSize(2);

  // Print message
  tft.setCursor(20, 40);
  tft.println("Hello World!");
}
void loop() {
  // Nothing here – message is static
}
```

Result

The hardware connections were made and the program was executed to observe the text in the display.

Experiment 10

A program to interface temperature sensor to read the room temperature, humidity and heat index and print the readings on the serial monitor using ESP32.

Introduction

The DHT11 is a low-cost, digital-output sensor used to measure temperature and relative humidity. It combines a capacitive humidity sensor and a thermistor in a single package and provides calibrated digital output via a single data line, making it simple to interface with microcontrollers like Arduino, Raspberry Pi, and ESP32. The working principle is as follows:

- Arduino sends a start signal to the DHT11 through the DATA pin.
- The DHT11 responds with a digital signal containing temperature and humidity values.
- Arduino decodes this signal using the DHT library.
- The code also computes the heat index using temperature + humidity formula.
- Finally, all values are printed on the Serial Monitor.

Hardware Required

Component Name	Quantity
ESP32 Development Board	1
DHT11 temperature & humidity sensor module	1
Patch Chords	Several

Circuit Connection

- Connect VCC of DHT11 to 3.3V (ESP32) / 5V (Arduino)
- Connect GND of DHT11 to GND
- Connect Data of DHT11 to GPIO 4 (ESP32) / Digital Pin 7 (Arduino)

Steps of working

- Plug the ESP32/Arduino into your computer using a USB cable.
- In Arduino IDE select File → New
- Type the code given below.
- Select the correct **Board** and **Port** in Arduino IDE.
- Click **Upload**. The IDE will compile and transfer the code to the board.
- Open **Tools → Serial Monitor** in Arduino IDE.
- Set baud rate to **9600**.
- Observe real-time outputs.

Code

```
#include "DHT.h"
#define DHTPIN 4        // Pin connected to DHT11 Data
#define DHTTYPE DHT11   // DHT11 Sensor
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // Celsius
  float f = dht.readTemperature(true); // Fahrenheit
  float hi = dht.computeHeatIndex(f, h);
 if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT11 sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" °C\t");
  Serial.print("Heat Index: ");
  Serial.print(dht.computeHeatIndex(t, h, false));
  Serial.println(" °C");
  delay(2000); }
```

Result

The hardware connections were made and the program was executed to observe the real time sensor output in serial monitor.