Exp-1

Step 1: Install the Arduino IDE

·Download the Arduino IDE software from the Arduino website.

·Run the installer and allow it to install the necessary drivers. It's recommended to install the drivers before connecting the board.

Step 2: Connect your Arduino board

·Plug the USB cable into your Arduino board and your computer. An LED on the board should illuminate.

Step 3: Select the correct board and port in the IDE

·Open the Arduino IDE.

·Go to the Tools menu and navigate to Board.

·From the dropdown list, select the model of your Arduino board (e.g., "Arduino Uno").

·Next, go back to the Tools menu and select Port.

·Choose the serial port that corresponds to your Arduino. If you're unsure, unplug the board, check the port list, and then plug it back in to see which new port appears.

Step 4: Upload your first sketch

·To test the configuration, load a simple example by going to File > Examples > 01.Basics > Blink.

·Click the Upload button (the right-pointing arrow icon) in the top-left corner of the IDE.

·The IDE will compile and upload the code, and the onboard LED on your Arduino should start blinking.

Prepare the OS

·Download and open the Raspberry Pi Imager on your computer.

·Insert the microSD card into your computer.

·In the imager, click "CHOOSE OS" and select a version of Raspberry Pi OS (32-bit is often more stable, while 64-bit is also an option).

·Click "CHOOSE STORAGE" and select your microSD card.

·Click "Next" and then "EDIT SETTINGS" to configure options like hostname, username, password, and Wi-Fi settings before writing.

·Click "SAVE" and then "YES" to confirm the OS customization.

·Click "WRITE" to begin flashing the OS to the card.

2. Connect hardware

·Insert the flashed microSD card into the slot on the Raspberry Pi 4.

·Connect your keyboard and mouse to the USB ports.

·Connect a monitor using a micro-HDMI cable.

·Plug the USB-C power adapter into the Pi and the wall. The Pi will boot automatically.

3. Complete initial setup

·Follow the on-screen instructions for the first boot, which will guide you through setting your country, language, timezone, and password.

·If you didn't set up Wi-Fi in the imager, you will be prompted to connect to a network.

·After the setup is complete, the Raspberry Pi will restart.

·Manually update the system by opening the Terminal and entering the following commands: sudo apt update and sudo apt upgrade.

4. Use raspi-config for advanced configuration
·After the initial setup, open the Terminal and run sudo raspi-config to access a text-based configuration tool.
·This tool allows you to:
o Change the hostname, username, and password.
o Enable SSH for remote access.
o Configure network settings.
o Change the boot order to boot from USB or network if no SD card is detected.
o Update the bootloader software.
o Adjust display and audio settings.

Configuring Python IDLE on a Raspberry Pi :primarily involves ensuring it's installed and then customizing its appearance or behavior.
1. Installing Python 3 and IDLE3:
Most Raspberry Pi OS installations include Python 3, but IDLE might not be pre-installed. To install or ensure it's up-to-date: Open a terminal window on your Raspberry Pi and Update your package list
```
sudo apt update
```
Install Python 3 and IDLE3.
```
sudo apt install python3 idle3
```
2. Launching IDLE:
Once installed, you can launch IDLE:
·Click the Raspberry Pi logo in the top-left corner of your desktop.
·Navigate to Programming > Python 3 (IDLE).
3. Customizing IDLE: You can adjust various settings within IDLE to suit your preferences
4. Writing and Running Scripts:
·New File:
In IDLE, go to File > New File to open a new editor window for writing Python scripts.
·Save File:
Save your script with a .py extension (e.g., my_script.py) using File > Save As....
·Run Module:
To execute your script, go to Run > Run Module or press F5. The output will appear in the IDLE shell.

Exp-2: Raspberry Pi Interface with LED Control

AIM: To interface an LED with a Raspberry Pi and control it (turn ON and OFF) using a Python program and GPIO pins.

Procedure

1. Hardware Setup

o Insert the SD card with Raspberry Pi OS and boot the Pi.
o Assemble the circuit.

2. Software Preparation

o Open a terminal and update packages:
o sudo apt update && sudo apt upgrade -y
o Ensure the GPIO library is installed (pre-installed on Raspberry Pi OS):
o sudo apt install python3-rpi.gpio

3. Programming

o Create a Python file: nano led_on_off.py.
o Enter the code (see below), save and exit (Ctrl+O, Enter, Ctrl+X).

4. Run the Program: python3 led_on_off.py

Source Code:

```
import RPi.GPIO as GPIO
import time
# Disable warnings (e.g., "GPIO already in use")
GPIO.setwarnings(False)
# Use BCM pin numbering
GPIO.setmode(GPIO.BCM)
# Set GPIO 17 as output
LED_PIN = 17
GPIO.setup(LED_PIN, GPIO.OUT)
try:
while True:
GPIO.output(LED_PIN, True) # LED ON
time.sleep(1) # 1 second delay
GPIO.output(LED_PIN, False) # LED OFF
time.sleep(1) # 1 second delay
except KeyboardInterrupt:
# Gracefully clean up on Ctrl+C
GPIO.cleanup()
```

Result and Conclusion:

· **Result**: The LED turns ON for 2 seconds and then turns OFF, demonstrating successful control through the Raspberry Pi GPIO pin.

· **Conclusion**: This experiment verifies that the Raspberry Pi can directly interface with basic output devices and be programmed using Python to control hardware, forming the foundation
for more complex IoT and embedded applications.

Exp-3: Raspberry Pi Interface with IR (Obstacle) Sensor

AIM

To detect the presence of an object using an Infrared (IR) obstacle sensor and indicate detection by printing a message on the Raspberry Pi terminal

Procedure

1. Hardware: Connect VCC to 5 V, GND to Pi ground, OUT to GPIO 18.
2. Software:

o Update OS packages:

o sudo apt update && sudo apt upgrade -y

o Ensure GPIO library is present (usually pre-installed):

o sudo apt install python3-rpi.gpio

3. Programming: Create a file: nano ir_sensor.py and insert the code below.
4. Run: python3 ir_sensor.py

Source Code

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN)
print("IR Sensor Test - Press Ctrl+C to stop")
try:
while True:
if GPIO.input(18) == GPIO.LOW: # LOW when object is close
print("Object Detected")
else:
print("No Object")
time.sleep(0.5)
except KeyboardInterrupt:
GPIO.cleanup()
```

Result & Conclusion

· **Result**: The terminal displays “Object Detected” whenever an object is within the IR sensor’s
range.

· **Conclusion**: Raspberry Pi successfully receives a digital signal from an IR obstacle sensor,
demonstrating digital input interfacing

Exp-4: Raspberry Pi Interface with Ultrasonic Sensor (HC-SR)

AIM: To measure the distance to an object using the HC-SR ultrasonic sensor and display it on the terminal.

Procedure

1. Assemble circuit with voltage divider on Echo (1 kΩ + 2 kΩ).
2. Update system if needed.
3. Install GPIO library (usually present).
4. Create program: nano ultrasonic.py.
5. Run: python3 ultrasonic.py

Source Code

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
TRIG = 20
ECHO = 21
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
try:
print("Press Ctrl+C to stop\n")
while True:
# Send trigger pulse
GPIO.output(TRIG, False)
time.sleep(0.0002)
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
# Capture echo times
while GPIO.input(ECHO) == 0:
start = time.time()
while GPIO.input(ECHO) == 1:
end = time.time()
# Calculate distance in cm
distance = (end - start) * 34300 / 2
# Use .format() for printing
print("Distance = {:.1f} cm".format(distance))
time.sleep(0.5)
except KeyboardInterrupt:
print("\nMeasurement stopped by user.")
GPIO.cleanup()
```

Result & Conclusion

· Result: Terminal displays distance to object in centimeters.
· Conclusion: Raspberry Pi can measure real-world distances using ultrasonic time-of-flight Sensing

Exp-5: Raspberry Pi Interface with DHT11 Temperature & Humidity Sensor

AIM: To read temperature and humidity data from a DHT11 sensor and display it on the Raspberry Pi terminal.

Procedure:
1. Connect the sensor as above (use a pull-up resistor if not a module).
2. Install required Python library:
3. sudo apt update
4. sudo apt upgrade
5. pip3 install --break-system-packages dht11
6. Create program: nano dht11_read.py.
7. Run: python3 dht11_read.py

Source Code:
```
import RPi.GPIO as GPIO
import dht11
import time
# GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
# Setup sensor
instance = dht11.DHT11(pin=21) # GPIO21
while True:
result = instance.read()
if result.is_valid():
print("Temperature: {} C Humidity: {} %".format(result.temperature,
result.humidity))
else:
print("Waiting for valid data...")
time.sleep(2)
```

Result & Conclusion
· Result: Terminal displays ambient temperature (°C) and relative humidity (%).
· Conclusion: Confirms Raspberry Pi can interface with single-wire digital sensors for environmental monitoring, a building block for IoT weather-station projects.

Exp-6: Ultrasonic Sensor and Relay Interface with Raspberry Pi

AIM: To measure the distance of an object using an HC-SR04 ultrasonic sensor and automatically energize or de-energize a relay based on the measured distance.

Procedure

1. Connect the sensor and relay as per the table.
2. Export GPIO pins using RPi.GPIO and set mode to BCM.
3. Send a 10 μs pulse to TRIG; measure time until ECHO goes LOW.
4. Calculate distance: distance = (time * 34300) / 2.
5. If distance < threshold (e.g., 20 cm) turn the relay ON, else OFF.

Source Code

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
TRIG = 23
ECHO = 24
RELAY = 18
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(RELAY, GPIO.OUT)
try:
while True:
GPIO.output(TRIG, False)
time.sleep(0.05)
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
while GPIO.input(ECHO) == 0:
start = time.time()
while GPIO.input(ECHO) == 1:
end = time.time()
distance = (end - start) * 17150
print(f"Distance: {distance:.1f} cm")
GPIO.output(RELAY, GPIO.HIGH if distance < 20 else GPIO.LOW)
except KeyboardInterrupt:
GPIO.cleanup()
```

Result & Conclusion

The relay switched ON whenever an object came within the preset distance (20 cm) and switched
OFF when the object moved away. This demonstrates successful distance-based actuation using an ultrasonic sensor.

Exp-7: IR Sensor and Relay Interface with Raspberry Pi

AIM: To detect the presence of an object using an infrared proximity sensor and activate a buzzer when the object is detected.

Procedure

1. Connect IR sensor output to GPIO 17 and buzzer to GPIO 27.
2. Configure GPIO; set buzzer as output and IR pin as input with pull-down.
3. Continuously read sensor output.
4. If logic LOW (object detected), drive buzzer HIGH.

Source Code

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
IR_PIN = 17
BUZZER = 27
GPIO.setup(IR_PIN, GPIO.IN)
GPIO.setup(BUZZER, GPIO.OUT)
try:
while True:
if GPIO.input(IR_PIN) == GPIO.LOW: # Object detected
GPIO.output(BUZZER, GPIO.HIGH)
else:
GPIO.output(BUZZER, GPIO.LOW)
time.sleep(0.1)
except KeyboardInterrupt:
GPIO.cleanup()
```

Result & Conclusion

The buzzer activated whenever an obstacle was detected by the IR sensor, proving effective object detection and alert generation.

Exp-8: DHT11 Sensor and Relay Interface with Raspberry Pi

AIM: To sense ambient temperature and humidity using a DHT11 sensor and control a relay when the temperature exceeds a threshold

Procedure

1. Wire the DHT11 and relay as listed.
2. Use the Adafruit_DHT library to read temperature and humidity.
3. If temperature > threshold (e.g., 30 °C), turn relay ON; else OFF.

Source Code

```
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
RELAY = 20
GPIO.setup(RELAY, GPIO.OUT)
sensor = Adafruit_DHT.DHT11
pin = 21
try:
while True:
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
if humidity is not None and temperature is not None:
print(f"Temp={temperature:.1f}°C Humidity={humidity:.1f}%")
GPIO.output(RELAY, GPIO.HIGH if temperature > 30 else GPIO.LOW)
else:
print("Sensor failure. Check wiring.")
time.sleep(2)
except KeyboardInterrupt:
GPIO.cleanup()
```

Result & Conclusion

The relay turned ON whenever the temperature crossed the preset 30 °C threshold and OFF otherwise. This verifies correct sensor interfacing and environmental control

Exp-9 IoT Data Processing with ESP32 + DHT22 + ThingSpeak

```
#include &lt;WiFi.h&gt;
#include &lt;HTTPClient.h&gt; // For HTTP POST
#include &lt;WiFiClientSecure.h&gt; // For HTTPS (secure)
#include &lt;DHTesp.h&gt; // DHT library for ESP32

// WiFi (Wokwi default)
const char* ssid = “Wokwi-GUEST’;
const char* password = “ ”;


const char* server = “api.thingspeak.com”;
const int channelID = 1234567;
const char* writeAPIKey = “XXXXXXXXXXXXXXXX”;

// Sensor &amp; LED pins
#define DHT_PIN 15
#define LED_ALERT_PIN 2

// Thresholds
#define TEMP_HIGH_THRESHOLD 30.0
#define TEMP_LOW_THRESHOLD 18.0
#define HUM_HIGH_THRESHOLD 80.0

DHTesp dht;
WiFiClientSecure client;

void setup() {
Serial.begin(115200);
pinMode(LED_ALERT_PIN, OUTPUT);
digitalWrite(LED_ALERT_PIN, LOW);

// Initialize DHT22
dht.setup(DHT_PIN, DHTesp::DHT22);

// Connect to WiFi
WiFi.begin(ssid, password);
Serial.print(&quot;Connecting to WiFi...&quot;);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(&quot;.&quot;);
}
Serial.println(&quot;\nWiFi connected&quot;);
Serial.print(&quot;IP address: &quot;);
Serial.println(WiFi.localIP());


client.setInsecure();

}
void loop() {
static unsigned long lastSend = 0;
if (millis() - lastSend &lt; 20000) {
delay(100);
return;
}
lastSend = millis();

TempAndHumidity data = dht.getTempAndHumidity();
if (dht.getStatus() != DHTesp::ERROR_NONE) {
Serial.println(“DHT read error!”);
return;
}

float temp = data.temperature;
float hum = data.humidity;

Serial.printf(“Temp: %.1f °C | Humidity: %.1f %%\n”, temp, hum);

int alert = 0;
String alertMsg = “Normal”;
digitalWrite(LED_ALERT_PIN, LOW);

if (temp > TEMP_HIGH_THRESHOLD) {
alert = 1;
alertMsg = "HIGH TEMP”;
digitalWrite(LED_ALERT_PIN, HIGH);
} else if (temp < TEMP_LOW_THRESHOLD) {
alert = 1;
alertMsg = “LOW TEMP”;
digitalWrite(LED_ALERT_PIN, HIGH);
} else if (hum > HUM_HIGH_THRESHOLD) {
alert = 1;

alertMsg = “HIGH HUMIDITY”;
digitalWrite(LED_ALERT_PIN, HIGH);
}

Serial.println(“Alert: “ + alertMsg);

// Send to ThingSpeak via HTTPS POST
if (WiFi.status() == WL_CONNECTED) {
HTTPClient http;
String url = “https://” + String(server) +”/update?api_key=” +
writeAPIKey +
“&field1=” + String(temp, 1) +
“&field2=” + String(hum, 1) +&field3=” + String(alert);

http.begin(client, url); // Secure connection

int httpCode = http.GET();

if (httpCode &gt; 0) {
Serial.printf(“ThingSpeak Response: %d\n”, httpCode);
if (httpCode == 200) {
Serial.println(“Data sent successfully!”);
}
} else {
Serial.println(“HTTP request failed”);
}
http.end();
} else {
Serial.println(“WiFi lost”);
}

delay(100);
```

```
import network
import urequests
import time
import machine

# WiFi (Wokwi default)
ssid = &quot;Wokwi-GUEST&quot;
password = &quot;&quot;

# ThingSpeak settings (replace with your own)
server = &quot;api.thingspeak.com&quot;
channel_id = 1234567 # YOUR CHANNEL ID
write_api_key = &quot;XXXXXXXXXXXXXXXX”

# Sensor &amp; LED pins
trigger_pin = machine.Pin(15, machine.Pin.OUT)
echo_pin = machine.Pin(14, machine.Pin.IN)
led_alert_pin = machine.Pin(13,
machine.Pin.OUT)

# Threshold
DISTANCE_CLOSE_THRESHOLD = 10.0 # cm

def connect_wifi():
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
print(&quot;Connecting to WiFi...&quot;)
while not wlan.isconnected():
time.sleep(0.5)
print(“.”)
print(“\nWiFi connected”)
print(“IP address:”, wlan.ifconfig()[0])

def measure_distance():
trigger_pin.off()
time.sleep_us(2)
trigger_pin.on()
time.sleep_us(10)

trigger_pin.off()

while echo_pin.value() == 0:
signal_off = time.ticks_us()
while echo_pin.value() == 1:
signal_on = time.ticks_us()

time_passed = signal_on - signal_off
distance = (time_passed * 0.0343)
divide by 2 for round trip
return distance

connect_wifi()

last_send = 0
while True:
if time.time() - last_send < 20:
interval
time.sleep(1)
continue
last_send = time.time()

# Read sensor
distance = measure_distance()
print(f&quot;Distance: {distance:.1f} cm”)

alert = 0
alert_msg = &quot;Normal&quot;
led_alert_pin.off()

if distance &lt; DISTANCE_CLOSE_THRESHOLD:
alert = 1
alert_msg = &quot;OBJECT TOO CLOSE&quot;
led_alert_pin.on()

print(&quot;Alert:&quot;, alert_msg)

# Send to ThingSpeak via HTTPS GET
url =
f”https://{server}/update?api_key={write_api_key}&field1{distance:.1f}
&field2={alert}”
try:
response = urequests.get(url)
print(&quot;ThingSpeak Response:&quot;, response.status_code)
if response.status_code == 200:
print(&quot;Data sent successfully!&quot;)
response.close()
except Exception as e:
print(&quot;HTTP request failed:&quot;, e)

time.sleep(1) # Small delay for stability
```