ARDUINO CODE

1. **Ultrasonic Sensor:**

Code:

```
// Define pins
#define trigPin 9
#define echoPin 10

// Define variables
long duration;
int distance;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set pin modes
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Send a 10 microsecond pulse to the trigger pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse
  duration = pulseIn(echoPin, HIGH);

  // Convert the duration to distance in centimeters
  distance = duration * 0.034 / 2;

  // Print the distance to the serial monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Wait for 500 milliseconds before measuring again
  delay(500);
}
```

- **Gas Sensor**

CODE:

```
// Define pins
#define gasPin A0

// Define variables
int gasValue;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set pin mode
  pinMode(gasPin, INPUT);
}

void loop() {
  // Read the analog value from the gas sensor
  gasValue = analogRead(gasPin);

  // Print the gas Relayto the serial monitor
  Serial.print("Gas Value: ");
  Serial.println(gasValue);

  // Check if gas level is above a threshold
  if (gasValue > 500) {
    Serial.println("Gas detected!");
  }

  // Wait for 500 milliseconds before reading again
  delay(500);
}
```

Relay

CODE:

```
// Define pins
#define relayPin 8

void setup() {
  // Set pin mode
  pinMode(relayPin, OUTPUT);

  // Turn off the relay at the start
  digitalWrite(relayPin, LOW);
}

void loop() {
  // Turn on the relay for 5 seconds
  digitalWrite(relayPin, HIGH);
  delay(5000);

  // Turn off the relay for 5 seconds
  digitalWrite(relayPin, LOW);
  delay(5000);
}
```

LDR

CODE:

```
// Define pins
#define ldrPin A0

// Define variables
int ldrValue;

void setup() {
  // Initialize serial communication
```

```
  Serial.begin(9600);

  // Set pin mode
  pinMode(ldrPin, INPUT);
}

void loop() {
  // Read the analog value from the LDR
  ldrValue = analogRead(ldrPin);

  // Print the LDR value to the serial monitor
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  // Map the LDR value to a range of 0-255
  int brightness = map(ldrValue, 0, 1023, 0, 255);

  // Print the brightness to the serial monitor
  Serial.print("Brightness: ");
  Serial.println(brightness);

  // Wait for 500 milliseconds before reading again
  delay(500);
}
```

Servomotor

CODE:

```
// Include the Servo library
#include <Servo.h>

// Define pins
#define servoPin 9
```

```
// Create a Servo object
Servo servo;

void setup() {
  // Attach the servo to the appropriate pin
  servo.attach(servoPin);
}

void loop() {
  // Move the servo to position 0 degrees
  servo.write(0);
  delay(1000);

  // Move the servo to position 90 degrees
  servo.write(90);
  delay(1000);

  // Move the servo to position 180 degrees
  servo.write(180);
  delay(1000);
}
```

- **Temperature Sensor**

Code

```
// Define pins
#define tempPin A0

// Define variables
float tempC;
float tempF;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);
}

void loop() {
  // Read the analog value from the temperature sensor
  int sensorValue = analogRead(tempPin);

  // Convert the sensor value to temperature in Celsius
  tempC = (5.0 * sensorValue * 100.0) / 1024.0;

  // Convert the temperature to Fahrenheit
  tempF = (tempC * 9.0 / 5.0) + 32.0;

  // Print the temperature to the serial monitor
  Serial.print("Temperature in Celsius: ");
  Serial.print(tempC);
  Serial.print(" °C | Temperature in Fahrenheit: ");
  Serial.print(tempF);
  Serial.println(" °F");

  // Wait for 1 second before reading again
  delay(1000);
}
```

- **Humidity Sensor**

Code

```
// Include the DHT library
#include <DHT.h>

// Define pins
#define dhtPin 2

// Create a DHT object
DHT dht(dhtPin, DHT11);

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize the DHT sensor
  dht.begin();
}

void loop() {
  // Read the humidity from the sensor
  float humidity = dht.readHumidity();

  // Print the humidity to the serial monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  // Wait for 1 second before reading again
  delay(1000);
}
```

RAS PI CODES:

- DHT11

CODE:

```
import Adafruit_DHT

# Define sensor type and pin number
sensor = Adafruit_DHT.DHT11
pin = 4

while True:
    # Read the temperature and humidity from the sensor
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

    # Check if the reading was successful
    if humidity is not None and temperature is not None:
        # Print the temperature and humidity to the console
        print('Temperature: {0:0.1f} °C | Humidity: {1:0.1f} %'.format(temperature,
humidity))
    else:
        # Print an error message if the reading failed
        print('Failed to retrieve data from the sensor!')

    # Wait for 1 second before taking another reading
    time.sleep(1)
```

- **Relay**

CODE:

```
import RPi.GPIO as GPIO
import time

# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)

# Define the GPIO pin number for the relay
relay_pin = 18

# Set the GPIO pin as an output
GPIO.setup(relay_pin, GPIO.OUT)

# Function to turn on the relay
def relay_on():
    GPIO.output(relay_pin, GPIO.HIGH)
    print('Relay is ON')

# Function to turn off the relay
def relay_off():
    GPIO.output(relay_pin, GPIO.LOW)
    print('Relay is OFF')

# Turn on the relay for 5 seconds, then turn it off
relay_on()
time.sleep(5)
relay_off()

# Clean up the GPIO pins
GPIO.cleanup()
```

- **LDR**

CODE:

```
import RPi.GPIO as GPIO
import time

# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)

# Define the GPIO pin number for the LDR
ldr_pin = 17

# Function to read the LDR sensor value
def read_ldr():
    # Set the GPIO pin as an output
    GPIO.setup(ldr_pin, GPIO.OUT)
    GPIO.output(ldr_pin, GPIO.LOW)
    time.sleep(0.1)

    # Set the GPIO pin as an input
    GPIO.setup(ldr_pin, GPIO.IN)

    # Measure the duration of the pulse
    start_time = time.time()
    end_time = time.time()
    while GPIO.input(ldr_pin) == GPIO.LOW:
        start_time = time.time()
    while GPIO.input(ldr_pin) == GPIO.HIGH:
        end_time = time.time()

    # Calculate the resistance of the LDR based on the duration of the pulse
    pulse_duration = end_time - start_time
    resistance = pulse_duration / 0.1

    # Calculate the LDR sensor value based on the resistance
    ldr_value = 1 / resistance
```

```
    return ldr_value

while True:
    # Read the LDR sensor value
    ldr_value = read_ldr()

    # Print the LDR sensor value to the console
    print('LDR sensor value: {0:.2f}'.format(ldr_value))

    # Wait for 1 second before taking another reading
    time.sleep(1)

# Clean up the GPIO pins
GPIO.cleanup()
```

- **PWM**

CODE:

```
import RPi.GPIO as GPIO
import time

# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)

# Define the GPIO pin number for the PWM signal
pwm_pin = 18

# Set the frequency of the PWM signal to 100 Hz
frequency = 100

# Set the duty cycle of the PWM signal to 50%
duty_cycle = 50

# Create a PWM object
pwm = GPIO.PWM(pwm_pin, frequency)

# Start the PWM signal with the specified duty cycle
pwm.start(duty_cycle)

# Wait for 5 seconds
time.sleep(5)

# Change the duty cycle of the PWM signal to 75%
pwm.ChangeDutyCycle(75)

# Wait for 5 seconds
time.sleep(5)

# Change the duty cycle of the PWM signal to 25%
pwm.ChangeDutyCycle(25)
```

```
# Wait for 5 seconds
time.sleep(5)

# Stop the PWM signal
pwm.stop()

# Clean up the GPIO pins
GPIO.cleanup()
```

- **LED Blinking**

CODE:

```
import RPi.GPIO as GPIO
import time

# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)

# Define the GPIO pin number for the LED
led_pin = 18

# Set the LED pin as an output
GPIO.setup(led_pin, GPIO.OUT)

# Set the blink interval to 1 second
blink_interval = 1

# Loop infinitely
while True:
    # Turn on the LED
    GPIO.output(led_pin, GPIO.HIGH)

    # Wait for the blink interval
    time.sleep(blink_interval)

    # Turn off the LED
```

```
    GPIO.output(led_pin, GPIO.LOW)

    # Wait for the blink interval
    time.sleep(blink_interval)

# Clean up the GPIO pins
GPIO.cleanup()
```

- **Push-button**

CODE:

```
import RPi.GPIO as GPIO


# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)


# Define the GPIO pin numbers for the push button and LED
button_pin = 18
led_pin = 23


# Set the push button pin as an input with a pull-up resistor
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)


# Set the LED pin as an output
GPIO.setup(led_pin, GPIO.OUT)


# Loop infinitely
while True:
```

```
    # Check if the push button is pressed
    if GPIO.input(button_pin) == GPIO.LOW:
        # Turn on the LED
        GPIO.output(led_pin, GPIO.HIGH)
    else:
        # Turn off the LED
        GPIO.output(led_pin, GPIO.LOW)

# Clean up the GPIO pins
GPIO.cleanup()
```