Q1 Write a program in GO language using a function to check whether the accepted number is palindrome or not.

Ans:-

```go
package main


import (
    "fmt"
)

// isPalindrome function checks whether the number is a palindrome or not
func isPalindrome(num int) bool {
    var reversed int = 0
    var remainder int

    // reverse the number
    temp := num
    for {
        remainder = temp % 10
        reversed = reversed*10 + remainder
        temp /= 10

        if temp == 0 {
            break
        }
    }

    // compare the original and reversed numbers
    if num == reversed {
        return true
    } else {
        return false
    }
}

func main() {
    var num int
    fmt.Println("Enter a number: ")
    fmt.Scanf("%d", &num)

    if isPalindrome(num) {
        fmt.Printf("%d is a palindrome number.\n", num)
    } else {
        fmt.Printf("%d is not a palindrome number.\n", num)
    }
}
```

Q1 Write a program in GO language to create an interface shape that includes area and volume. Implements these methods in square and rectangle type.

Ans

```go
package main

import (
    "fmt"
)

// shape interface defines methods for calculating area and volume
type shape interface {
    area() float64
    volume() float64
}

// square type implements the shape interface
type square struct {
    side float64
}

func (s square) area() float64 {
    return s.side * s.side
}

func (s square) volume() float64 {
    return 0
}

// rectangle type implements the shape interface
type rectangle struct {
    length float64
    width  float64
    height float64
}

func (r rectangle) area() float64 {
    return r.length * r.width
}

func (r rectangle) volume() float64 {
    return r.length * r.width * r.height
}
```

```go
func main() {
    var s shape
    var length, width, height, side float64
    var shapeType string

    fmt.Println("Enter shape type (square/rectangle):")
    fmt.Scanln(&shapeType)

    switch shapeType {
    case "square":
        fmt.Println("Enter side length:")
        fmt.Scanln(&side)
        s = square{side: side}
    case "rectangle":
        fmt.Println("Enter length:")
        fmt.Scanln(&length)
        fmt.Println("Enter width:")
        fmt.Scanln(&width)
        fmt.Println("Enter height:")
        fmt.Scanln(&height)
        s = rectangle{length: length, width: width, height: height}
    default:
        fmt.Println("Invalid shape type!")
        return
    }

    fmt.Printf("Area of %s: %.2f\n", shapeType, s.area())
    fmt.Printf("Volume of %s: %.2f\n", shapeType, s.volume())
}
```

a. 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

Ans
b -

```python
import RPi.GPIO as GPIO
import time
```

```
# set up GPIO mode and pin number
GPIO.setmode(GPIO.BCM)
LED_PIN = 4
GPIO.setup(LED_PIN, GPIO.OUT)

# blink LED 5 times
for i in range(5):
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(1)

# clean up GPIO
GPIO.cleanup()
```

c - Input refers to the data or signals that are received by a system or device, while output refers to the data or signals that are generated or produced by the system or device. In the case of the above program, the input is the user code, which instructs the program to blink the LED. The output is the visual blinking of the LED, which is controlled by the program based on the input provided by the user.

d - The result of the program is the blinking of the LED 5 times, with each blink lasting for 1 second. The conclusion is that the program successfully controlled the GPIO pin of the Raspberry Pi to turn the LED on and off, thereby achieving the desired output.

Slip 2

Q1 Write a program in GO language to create an interface and display its values with the help of type assertion.

Ans

```
package main

import "fmt"

// define a simple interface
type myInterface interface {
    getValue() string
}

// define a struct that implements the interface
```

```go
type myStruct struct {
    value string
}

func (s myStruct) getValue() string {
    return s.value
}

func main() {
    // create an object of type myStruct and assign it to the interface variable
    var iface myInterface = myStruct{value: "Hello, world!"}

    // use type assertion to extract the value from the interface
    if s, ok := iface.(myStruct); ok {
        fmt.Println("Value:", s.getValue())
    } else {
        fmt.Println("Error: unexpected type")
    }
}
```

Q1 Write a program in GO language to read and write Fibonacciseries to the using channel.
Ans

```go
package main

import "fmt"

func fibonacci(n int, c chan int) {
    a, b := 0, 1
    for i := 0; i < n; i++ {
        c <- a
        a, b = b, a+b
    }
    close(c)
}

func main() {
    // create a channel to hold the Fibonacci series
    c := make(chan int)

    // generate the Fibonacci series and write it to the channel
    go fibonacci(10, c)

    // read the Fibonacci series from the channel and print it
    for i := range c {
        fmt.Println(i)
    }
}
```

Q2 a. 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

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

buzzer_pin = 18  # Pin number where buzzer is connected

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

while True:
    GPIO.output(buzzer_pin, GPIO.HIGH)  # Turn on the buzzer
    time.sleep(1)  # Wait for 1 second
    GPIO.output(buzzer_pin, GPIO.LOW)  # Turn off the buzzer
    time.sleep(1)  # Wait for 1 second
```

c - The observations on input and output would be as follows:

Input: The input to the program is the pin number where the buzzer is connected and the delay time.
Output: The output of the program is the continuous ON/OFF cycle of the buzzer.

d - Result and Conclusion:

The program successfully turns on and off the buzzer in a continuous cycle, which can be used in various applications such as alarm systems, notifications, etc.

Slip 3

Q1 Write a program in GO language to check whether the accepted number is two digit or not.

Ans

```go
package main

import "fmt"

func main() {
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scan(&num)

    if num >= 10 && num <= 99 {
        fmt.Printf("%d is a two-digit number.\n", num)
    } else {
        fmt.Printf("%d is not a two-digit number.\n", num)
    }
}
```

Q1 Write a program in GO language to create a buffered channel, store few values in it and find channel capacity and length. Readvalues from channel and find modified length of a channel

Ans

```go
package main

import "fmt"

func main() {
    // create a buffered channel with capacity 3
    ch := make(chan int, 3)

    // add some values to the channel
    ch <- 1
    ch <- 2
    ch <- 3

    // find the capacity of the channel
    fmt.Printf("Channel capacity: %d\n", cap(ch))

    // find the length of the channel
    fmt.Printf("Channel length: %d\n", len(ch))

    // read values from the channel
    fmt.Printf("Values from the channel: ")
    for i := 0; i < cap(ch); i++ {
        fmt.Printf("%d ", <-ch)
    }

    // find the modified length of the channel
    fmt.Printf("\nChannel length after reading values: %d\n", len(ch))
}
```

Q2 a. 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

Ans

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

# turn ON buzzer
GPIO.output(18, GPIO.HIGH)
time.sleep(1)

# turn OFF buzzer
GPIO.output(18, GPIO.LOW)

GPIO.cleanup()
```

c Observations on Input and Output:

The input to the program is the GPIO pin number and the duration to turn ON the buzzer. The output is the buzzer being turned ON/OFF for the specified duration.

D Result and Conclusion:
In this tutorial, we have seen how to interface IR Sensor/Temperature Sensor/Camera with the Raspberry Pi board and how to write a program in Python to turn ON/OFF the buzzer using Raspberry Pi GPIO pins. By connecting sensors and actuators to the Raspberry Pi board and controlling them using software, we can build complex systems for various applications like home automation, robotics, and IoT.

Slip 4

Q1 Write a program in GO language to swap two numbers using call by reference concept

Ans

```
package main

import "fmt"

func swap(a *int, b *int) {
    temp := *a
    *a = *b
    *b = temp
}

func main() {
    var x, y int

    fmt.Print("Enter two numbers: ")
    fmt.Scan(&x, &y)

    fmt.Printf("Before swapping, x = %d and y = %d\n", x, y)

    swap(&x, &y)

    fmt.Printf("After swapping, x = %d and y = %d\n", x, y)
}
```

Q1 Write a program in GO language that creates a slice of integers, checks numbers from the slice are even or odd and further sent torespective go routines through channel and display values received by goroutines.

Ans

```go
package main

import "fmt"

func even(c chan int, nums []int) {
    for _, num := range nums {
        if num%2 == 0 {
            c <- num
        }
    }
    close(c)
}

func odd(c chan int, nums []int) {
    for _, num := range nums {
        if num%2 != 0 {
            c <- num
        }
    }
    close(c)
}

func main() {
    nums := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
    evenC := make(chan int)
    oddC := make(chan int)

    go even(evenC, nums)
    go odd(oddC, nums)

    fmt.Println("Even Numbers:")
    for num := range evenC {
        fmt.Println(num)
    }

    fmt.Println("\nOdd Numbers:")
    for num := range oddC {
        fmt.Println(num)
    }
}
```

Q2 a. 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

Ans

```
import RPi.GPIO as GPIO
import time

# Set up GPIO pins for LED control
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)  # LED1
GPIO.setup(18, GPIO.OUT)  # LED2

# Toggle the LEDs 10 times
for i in range(10):
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(18, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.HIGH)
    time.sleep(0.5)

# Cleanup GPIO pins
GPIO.cleanup()
```

c Observations:

- When the program is executed, the LEDs connected to the GPIO pins or digital pins start toggling.
- The program will continue to run until it is stopped manually.

d Result and Conclusion:

- The program successfully toggles the two LEDs using the GPIO or digital pins.
- We can conclude that interfacing LEDs with GPIO or digital pins on a Raspberry Pi, Beagle board, or Arduino Uno is a simple task that can be accomplished using the appropriate libraries and functions.

Slip 5

Q1 Write a program in GO language to print sum of all even and odd numbers separately between 1 to 100
Ans

```go
package main

import "fmt"

func main() {
    evenSum := 0
    oddSum := 0

    for i := 1; i <= 100; i++ {
        if i%2 == 0 {
            evenSum += i
        } else {
            oddSum += i
        }
    }

    fmt.Println("Sum of even numbers between 1 to 100:", evenSum)
    fmt.Println("Sum of odd numbers between 1 to 100:", oddSum)
}
```

Q1 Write a function in GO language to find the square of a number and write a benchmark for it.

Ans

```go
package main

import "fmt"

func Square(n int) int {
    return n * n
}

func main() {
    result := Square(5)
    fmt.Println("The square of 5 is:", result)
}
```

Here's how you can write a benchmark for this function using Go's built-in benchmarking package `testing`:

```go
package main

import "testing"

func BenchmarkSquare(b *testing.B) {
    for i := 0; i < b.N; i++ {
        Square(5)
    }
}
```

To run this benchmark, save the above code in a file named `squares_test.go` and run the following command:

```
go test -bench=.
```

Q2 a. 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

Ans

```python
import serial
import time

# Establish serial communication with Arduino
ser = serial.Serial('COM3', 9600)

# Delay to allow the serial communication to be established
time.sleep(2)

# Turn on LED1
ser.write(b'1')

# Delay for 1 second
time.sleep(1)

# Turn off LED1 and turn on LED2
ser.write(b'2')

# Delay for 1 second
time.sleep(1)
```

```
# Turn off LED2 and turn on LED1
ser.write(b'1')

# Delay for 1 second
time.sleep(1)

# Turn off LED1 and close serial communication
ser.write(b'0')
ser.close()
```

c Observations:

> When we run this code, we can observe that the LED1 turns on for 1 second and then turns off. After that, LED2 turns on for 1 second and then turns off. Finally, LED1 turns on for 1 second and then turns off. The program then closes the serial communication.

d Results and Conclusion:

> This code shows that we can use Python to control and toggle the state of two LEDs connected to an Arduino board. We can modify the code to control more LEDs or other components connected to the board

Slip 6

Q1 Write a program in GO language to demonstrate working of slices (like append, remove, copy etc.)

Ans

```
package main

import "fmt"

func main() {
    // Initializing a slice with some elements
    slice1 := []int{1, 2, 3, 4, 5}
    fmt.Println("Original Slice:", slice1)

    // Appending an element to the slice
    slice1 = append(slice1, 6)
    fmt.Println("Slice After Appending an Element:", slice1)
```

```go
    // Removing an element from the slice
    slice1 = remove(slice1, 3)
    fmt.Println("Slice After Removing an Element:", slice1)

    // Copying a slice to another slice
    slice2 := make([]int, len(slice1))
    copy(slice2, slice1)
    fmt.Println("Copied Slice:", slice2)
}

// Function to remove an element from the slice
func remove(slice []int, s int) []int {
    return append(slice[:s], slice[s+1:]...)
}
```

Q2 Write a program in GO language using go routine and channel that will print the sum of the squares and cubes of the individual digits of a number. Example if number is 123 then squares = (1 * 1) + (2 * 2) + (3 * 3) cubes = (1 * 1 * 1) + (2 * 2 * 2) + (3 * 3 * 3).

Ans

```go
package main

import (
    "fmt"
    "strconv"
)

func digitSquareCubeSum(num int, sqChan, cubeChan chan int) {
    squaresSum := 0
    cubesSum := 0
    digits := strconv.Itoa(num)

    for _, digit := range digits {
        digitInt, _ := strconv.Atoi(string(digit))
        squaresSum += digitInt * digitInt
        cubesSum += digitInt * digitInt * digitInt
    }

    sqChan <- squaresSum
    cubeChan <- cubesSum
}
```

```
func main() {
    num := 123

    sqChan := make(chan int)
    cubeChan := make(chan int)

    go digitSquareCubeSum(num, sqChan, cubeChan)

    squaresSum := <-sqChan
    cubesSum := <-cubeChan

    fmt.Printf("Number: %d\n", num)
    fmt.Printf("Sum of squares: %d\n", squaresSum)
    fmt.Printf("Sum of cubes: %d\n", cubesSum)
}
```

Q2 a. 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

Ans

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

# Turn ON buzzer
GPIO.output(11, GPIO.HIGH)
time.sleep(1)

# Turn OFF buzzer
GPIO.output(11, GPIO.LOW)
```

C Observations:

- When the buzzer is turned ON, it produces a sound.
- When the buzzer is turned OFF, it stops producing sound.
-

D Result and Conclusion:

The Python program can effectively control the state of the buzzer and turn it ON or OFF as desired. However, it is important to ensure that the correct GPIO pin number is used and that the buzzer is connected properly to the board. Additionally, appropriate safety measures should be taken when working with hardware to avoid any damage or injury.

Slip 7

Q1 Write a program in GO language to demonstrate function returnmultiple values.

Ans

```
package main

import "fmt"

// function that returns multiple values
func divideAndRemainder(dividend, divisor int) (int, int) {
    quotient := dividend / divisor
    remainder := dividend % divisor
    return quotient, remainder
}

func main() {
    // call the function and receive the multiple return values
    quotient, remainder := divideAndRemainder(10, 3)

    fmt.Printf("Quotient: %d, Remainder: %d", quotient, remainder)
}
```

Q1 Write a program in GO language to read XML file into structure and display structure

Ans:

```

The notes.xml file is read with the ioutil.ReadFile() function and a byte slice
is returned, which is then decoded into a struct instance with the
xml.Unmarshal() function. The struct instance member values are used to print the
decoded data.
-------------------------------------------------------------------
*/
```

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
package main

import (
    "encoding/xml"
    "fmt"
    "io/ioutil"
)

type Notes struct {
    To      string `xml:"to"`
    From    string `xml:"from"`
    Heading string `xml:"heading"`
    Body    string `xml:"body"`
}

func main() {
    data, _ := ioutil.ReadFile("notes.xml")

    note := &Notes{}

    _ = xml.Unmarshal([]byte(data), &note)

    fmt.Println(note.To)
    fmt.Println(note.From)
    fmt.Println(note.Heading)
    fmt.Println(note.Body)
}
```

Q2 a. 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

Ans

```
import RPi.GPIO as GPIO
import time

# Set the GPIO pin numbers
led_pin1 = 18
led_pin2 = 23

# Set up the GPIO pins as outputs
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin1, GPIO.OUT)
GPIO.setup(led_pin2, GPIO.OUT)

# Toggle the LEDs on and off
while True:
    GPIO.output(led_pin1, GPIO.HIGH)
    GPIO.output(led_pin2, GPIO.LOW)
    time.sleep(1)
    GPIO.output(led_pin1, GPIO.LOW)
    GPIO.output(led_pin2, GPIO.HIGH)
    time.sleep(1)
```

c Inputs are signals or data that are provided to a system, while outputs are the resulting signals or data that are produced by the system. In the case of the LED toggling program above, the inputs are the signals sent to the GPIO pins, and the outputs are the resulting light emitted from the LEDs.

D The result of the LED toggling program is that the two LEDs connected to the GPIO pins will alternate between being turned on and off every second. The conclusion is that it is possible to use a programming language like Python to control the behavior of electronic devices like LEDs by interfacing with the GPIO pins of a Raspberry Pi.

Slip 8

Q1 Write a program in GO language to create a user defined packageto find out the area of a rectangle

Ans

```
package main

import (
    "fmt"

    "rectangle" // Import the user-defined rectangle package
)

func main() {
    r := rectangle.Rectangle{Length: 4, Width: 6}

    area := rectangle.Area(r)

    fmt.Printf("The area of the rectangle is %.2f square units.\n", area)
}
```

Q2 Write a program in GO language that prints out the numbers from 0to 10, waiting between 0 and 250 ms after each one using the delay function.

Ans

```
package main

import (
    "fmt"
    "math/rand"
    "time"
)

func main() {
    rand.Seed(time.Now().UnixNano()) // seed the random number generator with the
current time

    for i := 0; i <= 10; i++ {
        fmt.Println(i)
        time.Sleep(time.Duration(rand.Intn(250)) * time.Millisecond) // wait
between 0 and 250 ms
```

```
    }
}
```

Q2 a. 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

Ans

```python
import RPi.GPIO as GPIO
import time

# Set the GPIO pin number
led_pin = 18

# Set up the GPIO pin as an output
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

# Blink the LED on and off
while True:
    GPIO.output(led_pin, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(led_pin, GPIO.LOW)
    time.sleep(1)
```

c. Inputs are signals or data that are provided to a system, while outputs are the resulting signals or data that are produced by the system. In the case of the LED blinking program above, the input is the signal sent to the GPIO pin, and the output is the resulting light emitted from the LED.

d. The result of the LED blinking program is that the LED connected to the GPIO pin will turn on and off repeatedly, with a delay of one second between each change. The conclusion is that it is possible to use a programming language like Python to control the behavior of electronic devices like LEDs by interfacing with the GPIO pins of a Raspberry Pi.

Slip 9

Q1 Write a program in GO language to print a multiplication table ofnumber using function.

Ans

```
package main

import "fmt"

func multiplicationTable(num int) {
    for i := 1; i <= 10; i++ {
        fmt.Printf("%d x %d = %d\n", num, i, num*i)
    }
}

func main() {
    var num int
    fmt.Print("Enter a number to print its multiplication table: ")
    fmt.Scanln(&num)
    multiplicationTable(num)
}
```

Q1 Write a program in GO language using a user defined package calculator that performs one calculator operation as per the user'schoice.

Ans

```
package calculator
// I'm creating a simple calculator that
// performs one calculator operation as per the
// user's choice. For readability of code,
// I named the package as "calculator"
// And remember, the first executable line
// must always be as mentioned above:
// the keyword package followed by a name
// that you wish to give to your package*
//* indicates very very important

import "fmt"
// importing fmt package for basic
// printing & scan operations

func Calc() {
```

```go
// a simple Calc function that contains
// all code within and has no return
// type mentioned
// Println prints the input string in new line
fmt.Println("Welcome to calculator")
fmt.Println("********************MAIN MENU*************************")
fmt.Println("1. Add")
fmt.Println("2. Subtract")
fmt.Println("3. Multiply")
fmt.Println("4. Divide")
fmt.Println("******************************************************")
var choice int

// choice will store the user's
// input as per the menu shown above
fmt.Scan(&choice)
var a, b int

// After the choice of operation, user
// will be asked to enter 2 int
// values one by one to perform
// the operation on
fmt.Println("Enter value of a: ")
fmt.Scan(&a)
fmt.Println("Enter value of b: ")
fmt.Scan(&b)
if( choice == 1 ){
    // choice 1 activates this part --> addition
    ans := a + b
    fmt.Println("Answer = ", ans)
} else if( choice == 2 ){
    // choice 2 activates this part --> subtraction
    ans := a - b
    fmt.Println("Answer = ", ans)
} else if( choice == 3 ){
    // choice 3 activates this part --> multiplication
    ans := a * b
    fmt.Println("Answer = ", ans)
} else {
    // choice 4 activates this part --> division
    // remember not to enter second value as 0
    // as that would raise a DivideByZero error
    // or may display infinity
    ans := a / b
```

```
        fmt.Println("Answer = ", ans)
    }
    fmt.Println("*******************************************************")
    fmt.Println("Thank you for using calculator! Have a nice day ahead. ^-^")
    fmt.Println("*******************************************************")
}
```

Calculator.go

```
package main

import "calculator"

func main() {
    calculator.Calc()
    }
```

Q2 a. 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

Ans

b

```
import os
import time
import glob

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'
```

```python
def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c

while True:
    print(read_temp())
    time.sleep(1)
```

c. Observations on Input and Output:

In the temperature sensor program, the input is the temperature readings from the sensor, and the output is the temperature in Celsius. The input and output are both printed to the console for observation.

In the buzzer program, the input is the Raspberry Pi's GPIO pin state (HIGH or LOW), and the output is the buzzer turning on and off. The input and output are both printed to the console for observation.

d. Result and Conclusion:

In this exercise, we learned how to interface a Raspberry Pi with a temperature sensor and a buzzer using the GPIO pins and wrote programs in Python to read the temperature and turn the buzzer on and off. By observing the input and output, we can verify that the programs are working as expected.

Slip 10

Q1 Write a program in GO language to illustrate the function returning multiple values(add, subtract).

Ans

```go
package main

import "fmt"

func addSubtract(a, b int) (int, int) {
    return a+b, a-b
}

func main() {
    sum, diff := addSubtract(10, 5)
    fmt.Println("Sum:", sum)
    fmt.Println("Difference:", diff)
}
```

Q1 Write a program in the GO language program to open a file inREAD only mode.

Ans

```go
package main

import (
    "fmt"
    "os"
)

func main() {
    filename := "example.txt"
    file, err := os.Open(filename)
    defer file.Close()

    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    fmt.Println("Successfully opened file", filename)

    // Read file content here
}
```

Q2 a. 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.

Ans

```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
    GPIO.output(18, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    time.sleep(1)
```

c. Observations on Input and Output:

In both programs, the input is the pin number to which the buzzer is connected, and the output is the sound produced by the buzzer when it is turned ON.

d. Results and Conclusion:

By running the above programs, we can successfully turn the buzzer ON/OFF using the Raspberry Pi board and the specified pin. The buzzer produces a sound when it is turned ON, and stops producing sound when it is turned OFF. We can use this functionality in various applications, such as sound alerts, alarms, and notifications.

Slip 11

Q1 Write a program in Go language to add or append content at theend of a text file.

Ans

```go
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    // Open the file for appending
    file, err := os.OpenFile("filename.txt", os.O_APPEND|os.O_WRONLY, 0644)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }
    defer file.Close()

    // Ask the user for input to append
    reader := bufio.NewReader(os.Stdin)
    fmt.Print("Enter text to append: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    // Append the text to the file
    _, err = file.WriteString(text)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    fmt.Println("Text appended successfully!")
}
```

Q1 Write a program in Go language how to create a channel and illustrate how to close a channel using for range loop and close function.

Ans

```go
package main

import (
    "fmt"
)

func main() {
    // Create a channel of type int
    ch := make(chan int)

    // Start a goroutine to send values to the channel
    go func() {
        for i := 0; i < 5; i++ {
            ch <- i
        }
        close(ch)
    }()

    // Use the for range loop to read values from the channel
    for val := range ch {
        fmt.Println("Received:", val)
    }

    fmt.Println("Channel is closed")
}
```

Q2 a. 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.

Ans

```python
import RPi.GPIO as GPIO
import time

# Set up GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)

# Toggle the two LEDs
while True:
    GPIO.output(18, GPIO.HIGH)
    GPIO.output(23, GPIO.LOW)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(23, GPIO.HIGH)
    time.sleep(1)
```

c. The input to this program is the code itself, which sets up the GPIO pins and toggles the LEDs. The output is the visual effect of the LEDs turning on and off in a repeating pattern.

d. The result of this program should be two LEDs turning on and off in a repeating pattern. The conclusion is that the program successfully toggles the two LEDs using the GPIO pins of the Raspberry Pi, demonstrating how to control external devices with Python code.

Slip 19