Assignment-8

ELP - 720 Telecommunication Networks Laboratory

Sudhanshu Chaudhary

2019JTM2207
2020-21

A report presented for the assignment on

Raspberry Pi

Bharti School Of

Telecommunication Technology and Management

IIT Delhi

India

March 3, 2020

Contents

List of Figures

1 Problem Statement 1

1.1 Problem Statement

You have to design a sequence detector using Rpi.

- Use two push buttons (P1 & P0) for ‘1’ and ‘0.’
- Perform the following tasks
  - **Calibration mode**
    1. Ask the user to enter the length of the detecting sequence(through the keyboard) ex-4
    2. Now enter the sequence(through push button) ex-1010
    3. Ask for overlapping or non-overlapping case
  - **Run mode**
    1. Now go to the listening mode(i.e., sequence detecting mode)
    2. Use P1 & P0 to enter the input sequence
    3. Count the no of times sequence is detected
  - **Exit mode**
    1. If P1 & P2 are pressed at the same time then the system should enter the exit mode
    2. Blink an LED ‘n’ number of times(n is the number of times sequence is detected)

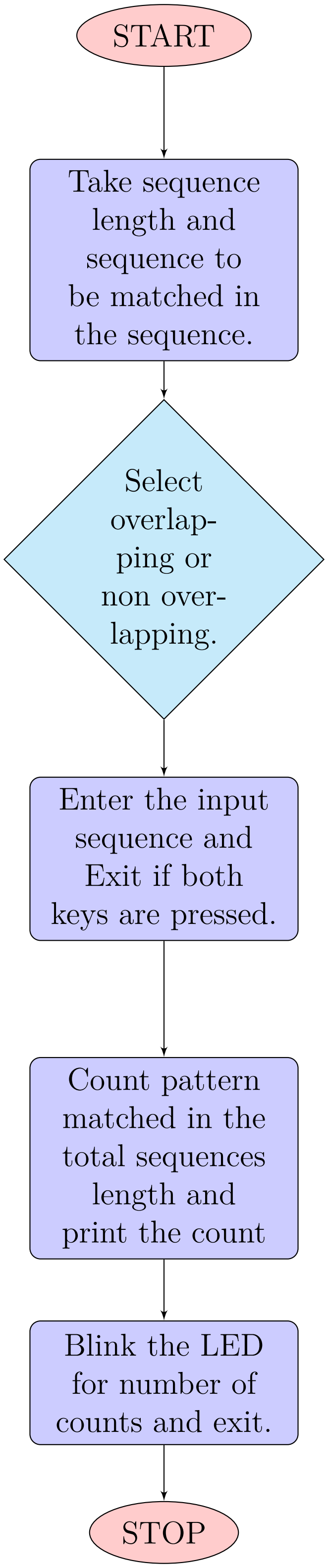
1.2 Equipment Required

- Raspberry Pi
- 1 USB Cable
- 1 HDMI cable
- Breadboard, LEDs and Buttons

1.3 Algorithm and Implementation

- Use two push buttons (P1 and P0) for ‘1’ and ‘0.’ and LED’s for the buttons as well as for output blinking.
- Take sequence length from user.
- Take the sequence using push button to be matched in the sequence.
- Ask for overlapping or non-overlapping.
- Select the choice for the same.
- Enter the input sequence.
- Exit if both keys are pressed.
- Count number of times pattern matched in the total sequences length.
- Print the count.
- Blink the LED for number of counts and exit.

1.4 Program Structure


START
Take sequence length and sequence to be matched in the sequence.
Select overlapping or non overlapping.
Enter the input sequence and Exit if both keys are pressed.
Count pattern matched in the total sequences length and print the count
Blink the LED for number of counts and exit.
STOP

1.5 Difficulties faced

- Setting up the hardware.
- Button values and delay.

1.6 Screenshots

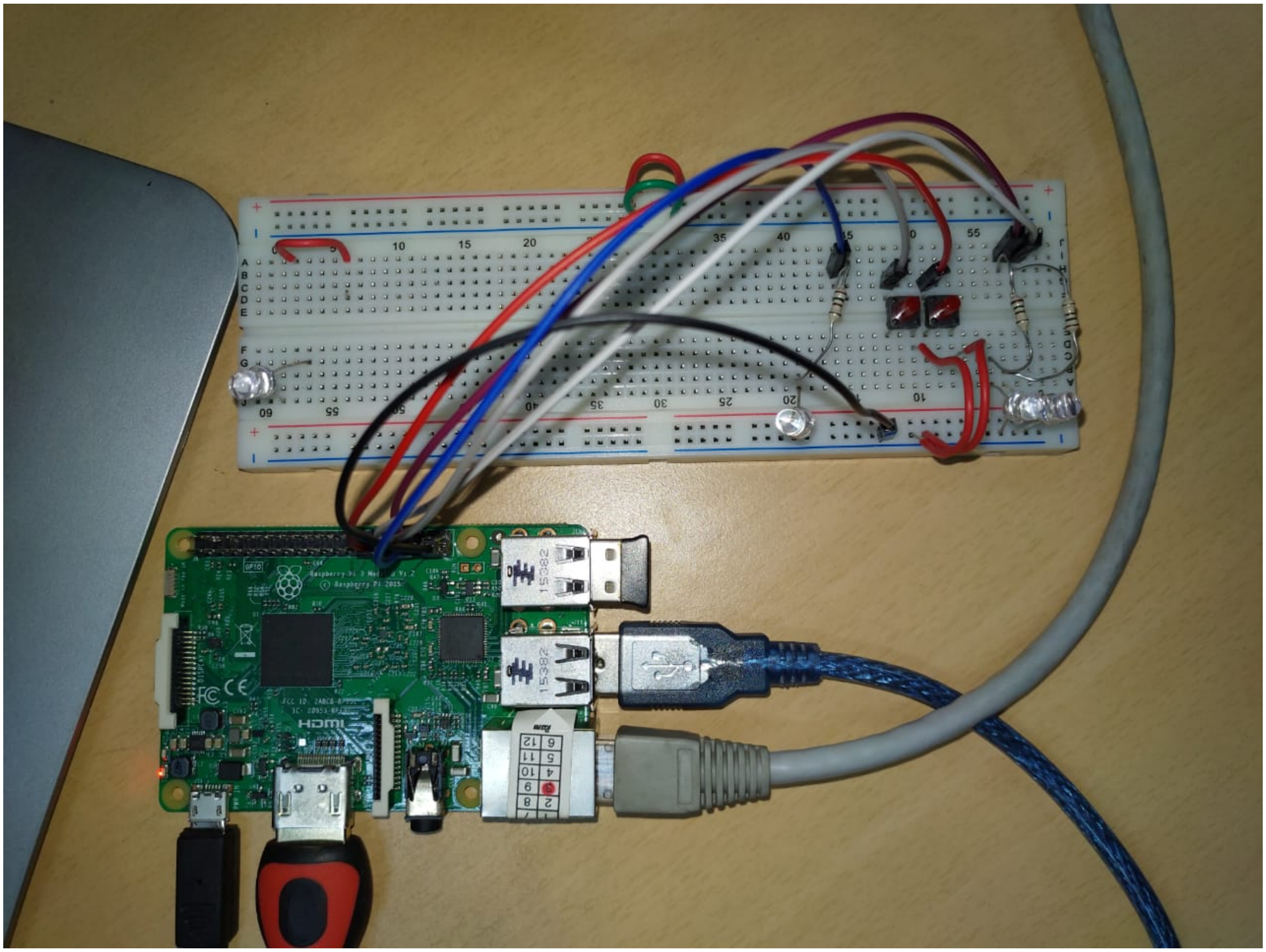
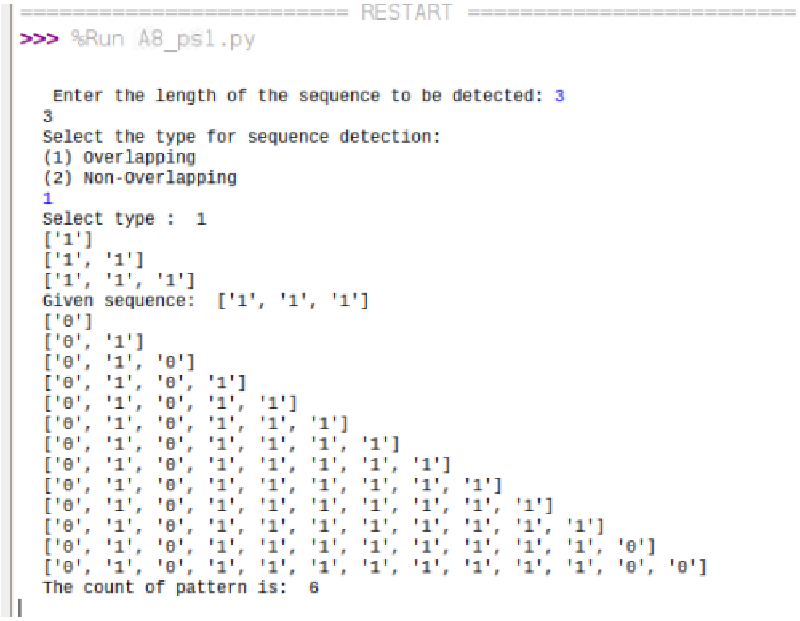


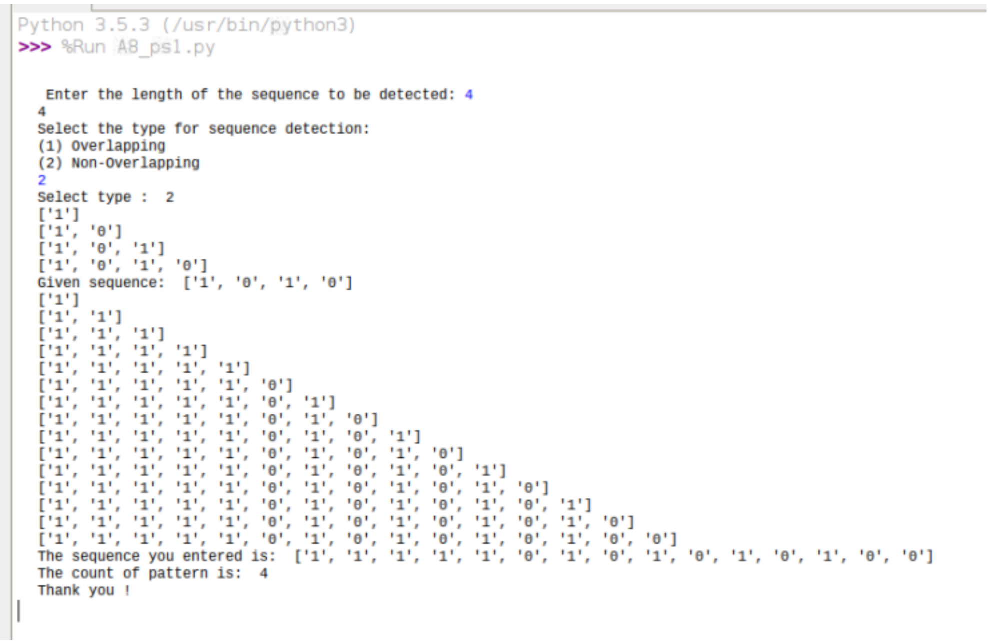
Figure 1: Problem Statement-1 Console input



```
========================= RESTART =========================
>>> %Run A8_ps1.py

 Enter the length of the sequence to be detected: 3
3
Select the type for sequence detection:
(1) Overlapping
(2) Non-Overlapping
1
Select type :  1
['1']
['1', '1']
['1', '1', '1']
Given sequence:  ['1', '1', '1']
['0']
['0', '1']
['0', '1', '0']
['0', '1', '0', '1']
['0', '1', '0', '1', '1']
['0', '1', '0', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1']
['0', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '0']
['0', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '0', '0']
The count of pattern is:  6
```

Figure 2: Problem Statement-1 Overlapping



```
Python 3.5.3 (/usr/bin/python3)
>>> %Run A8_ps1.py

 Enter the length of the sequence to be detected: 4
4
Select the type for sequence detection:
(1) Overlapping
(2) Non-Overlapping
2
Select type :  2
['1']
['1', '0']
['1', '0', '1']
['1', '0', '1', '0']
Given sequence:  ['1', '0', '1', '0']
['1']
['1', '1']
['1', '1', '1']
['1', '1', '1', '1']
['1', '1', '1', '1', '1']
['1', '1', '1', '1', '1', '0']
['1', '1', '1', '1', '1', '0', '1']
['1', '1', '1', '1', '1', '0', '1', '0']
['1', '1', '1', '1', '1', '0', '1', '0', '1']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1', '0']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1', '0', '1']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1', '0', '1', '0']
['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1', '0', '1', '0', '0']
The sequence you entered is:  ['1', '1', '1', '1', '1', '0', '1', '0', '1', '0', '1', '0', '1', '0', '0']
The count of pattern is:  4
Thank you !
```

Figure 3: Problem Statement-1 Non-overlapping

2 Problem Statement 2

2.1 Problem Statement

Take any input string from your Rpi and send it to your Arduino and display the string on the serial monitor.
Note: Use only GPIO pins.
You can use ESP32 if Arduino is not available

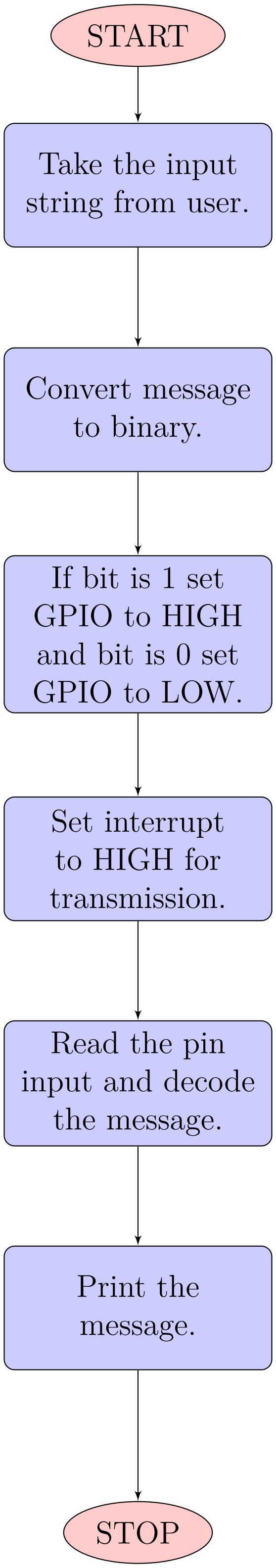
2.2 Equipment Required

- Raspberry pi
- Arduino

2.3 Algorithm and Implementation

- Take the input string from user.
- Convert message to binary.
- If bit is 1 set GPIO to HIGH.
- If bit is 0 set GPIO to LOW.
- Set interrupt to HIGH for transmission.
- Read the pin input.
- Decode the message.
- Print the message.

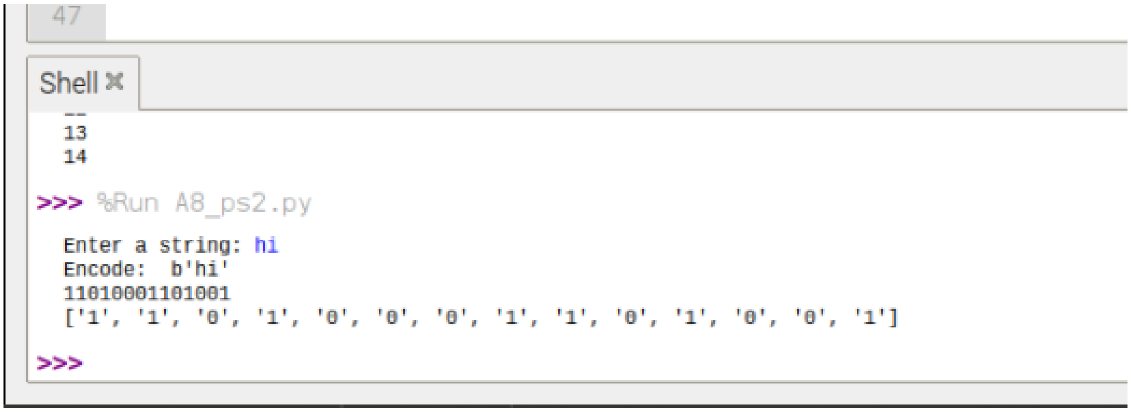
2.4 Program Structure


START
Take the input string from user.
Convert message to binary.
If bit is 1 set GPIO to HIGH and bit is 0 set GPIO to LOW.
Set interrupt to HIGH for transmission.
Read the pin input and decode the message.
Print the message.
STOP

2.5 Difficulties faced

- Receiving the bit to arduino in same sequence sent from Rpi.

2.6 Screenshots

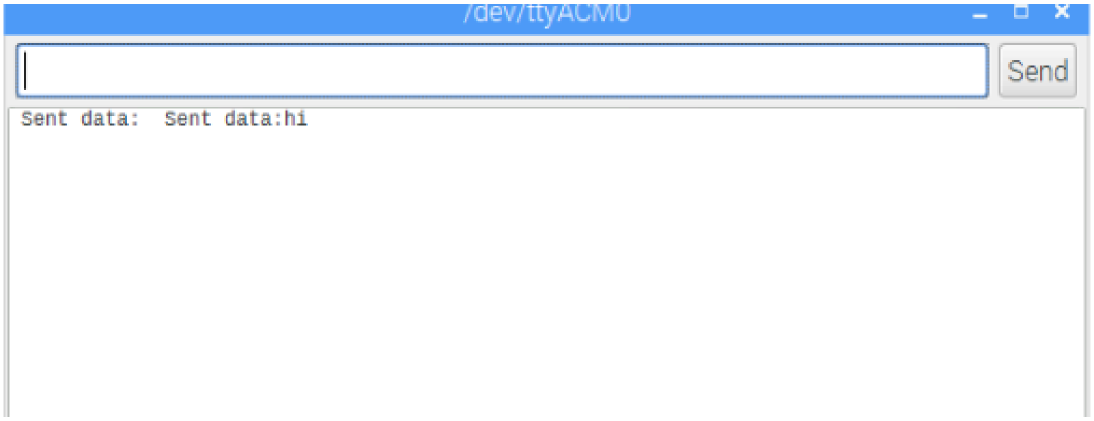


```
  13
  14
>>> %Run A8_ps2.py

  Enter a string: hi
  Encode:  b'hi'
  11010001101001
  ['1', '1', '0', '1', '0', '0', '0', '1', '1', '0', '1', '0', '0', '1']

>>>
```

Figure 4: Problem Statement-2 Message sent from Rpi



```
Sent data:  Sent data:hi
```

Figure 5: Problem Statement-2 Message received on Arduino

3 Appendix

3.1 Appendix-A : Problem Statement-1

```

# Problem Statement 1 -  Sequence detector

#   * Read button input
#   * Output on LEDs (Blinking n times)

import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

# Pin Definitions
button_Pin1 = 29
button_Pin2 = 31

led_Pin1 = 32
led_Pin2 = 36
led_Pin_op = 35

sequence_given=[]
sequence_total=[]


# Setup GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(button_Pin1,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(button_Pin2,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led_Pin1,GPIO.OUT)
GPIO.setup(led_Pin2,GPIO.OUT)
GPIO.setup(led_Pin_op,GPIO.OUT)

def pattern_match(list1,list2):
    len1 = len(list1)
    len2 = len(list2)
    l1=[]
    l2=[]
    p = 0
    j=0
    while j < len2:
        l1.clear()
        l2.clear()
        for i in range(len1):
            x=list1[i]
            y=list2[i+j]
            l1.append(x)
            l2.append(y)
        #print("l1",l1)
        #print("l2",l2)
        if l1==l2:
            p=p+1
        j=j+len1
    print(p)
    return

# Infinite Loop
try:
    length_seq = int(input("\n Enter the length of the sequence to be detected : "))
    print(length_seq)
    print("Select the type for sequence detection :  ")
    print("(1) Overlapping ")
    print("(2) Non-Overlapping ")
    case_type = input()
    print("Select type : ",case_type)
    #############################

    if case_type=='1':
        # take input sequence from the buttons
        while 1:
            if  GPIO.input(button_Pin1) == GPIO.HIGH :              # Not Pressed ========
                GPIO.output(led_Pin1, GPIO.LOW)
            if  GPIO.input(button_Pin2) == GPIO.HIGH :              # Not Pressed ========
                GPIO.output(led_Pin2, GPIO.LOW)
            if  GPIO.input(button_Pin1) == GPIO.LOW :
                GPIO.output(led_Pin1, GPIO.HIGH)
                sequence_given.append("1")
                print(sequence_given)
                time.sleep(0.5)
                if len(sequence_given)==length_seq:
                    print("Given sequence: ", sequence_given )
                    break

            if  GPIO.input(button_Pin2) == GPIO.LOW :
                GPIO.output(led_Pin2, GPIO.HIGH)
                sequence_given.append("0")
                print(sequence_given)
                time.sleep(0.5)
                if len(sequence_given)==length_seq:
                    print("Given sequence: ", sequence_given )
                    break

        # Run Mode for listening
        while 1:
                if  ((GPIO.input(button_Pin1) == GPIO.LOW) &  (GPIO.input(button_Pin2) == GPIO.LOW)):    #both pressed simulteniously
                    x=len(sequence_total)
                    a=0
                    b=length_seq
                    count=0
                    check=[]
                    while True:
                        check=sequence_total[a:b]
                        a=a+1
                        b=b+1
                        if(sequence_given==check):
                            count=count+1
                        if (b==x+1):
                            break
                    print("The count of pattern is: ",count)
                    time.sleep(2)
                    for i in range(0,a):
                        GPIO.output(led_Pin_op, GPIO.HIGH)
                        time.sleep(1)
                        GPIO.output(led_Pin_op, GPIO.LOW)
                        time.sleep(1)
                    print("Thank you !")
                    #exit()
                if GPIO.input(button_Pin1) == GPIO.HIGH :              # Not Pressed ========
                    GPIO.output(led_Pin1, GPIO.LOW)
                if GPIO.input(button_Pin2) == GPIO.HIGH :              # Not Pressed ========
                    GPIO.output(led_Pin2, GPIO.LOW)
                if GPIO.input(button_Pin1) == GPIO.LOW :
                    GPIO.output(led_Pin1, GPIO.HIGH)
                    sequence_total.append("1")
                    print(sequence_total)
                    time.sleep(0.5)
                if GPIO.input(button_Pin2) == GPIO.LOW :
                    GPIO.output(led_Pin2, GPIO.HIGH)
                    sequence_total.append("0")
                    print(sequence_total)
                    time.sleep(0.5)



    elif case_type=='2':
        # Calibration mode take input sequence from the buttons
        while 1:
            if GPIO.input(button_Pin1) == GPIO.HIGH :              # Not Pressed ========
                GPIO.output(led_Pin1, GPIO.LOW)
            if GPIO.input(button_Pin2) == GPIO.HIGH :              # Not Pressed ========
                GPIO.output(led_Pin2, GPIO.LOW)
            if GPIO.input(button_Pin1) == GPIO.LOW :
                GPIO.output(led_Pin1, GPIO.HIGH)
                sequence_given.append("1")
                print(sequence_given)
                time.sleep(0.5)
                if len(sequence_given)==length_seq:
                    print("Given sequence: ", sequence_given )
                    break

            if GPIO.input(button_Pin2) == GPIO.LOW :
                GPIO.output(led_Pin2, GPIO.HIGH)
                sequence_given.append("0")
                print(sequence_given)
                time.sleep(0.5)
                if len(sequence_given)==length_seq:
                    print("Given sequence: ", sequence_given )
                    break

        # Run Mode for listening
        while 1:
                if  ((GPIO.input(button_Pin1) == GPIO.LOW) &  (GPIO.input(button_Pin2) == GPIO.LOW)):    #both pressed simulteniously
                    print("The sequence you entered is: ",sequence_total)
                    #pattern_match(sequence_given,sequence_total )
                    x=len(sequence_total)
                    a=0
                    b=length_seq
                    count=0
                    check=[]
                    while True:
                        check=sequence_total[a:b]
                        a=a+1
                        b=b+1
                        if(sequence_given==check):
                            count=count+1
                        if b==x+1:
                            break
                    print("The count of pattern is: ",count)
                    print("Thank you !")
                    time.sleep(2)
                    for i in range(0,a):
                        GPIO.output(led_Pin_op, GPIO.HIGH)
                        time.sleep(1)
                        GPIO.output(led_Pin_op, GPIO.LOW)
                        time.sleep(1)
                    #exit()
                if  GPIO.input(button_Pin1) == GPIO.HIGH :                  # Not Pressed ========
                    GPIO.output(led_Pin1, GPIO.LOW)
                if  GPIO.input(button_Pin2) == GPIO.HIGH :                  # Not Pressed ========
                    GPIO.output(led_Pin2, GPIO.LOW)
                if  GPIO.input(button_Pin1) == GPIO.LOW :
                    GPIO.output(led_Pin1, GPIO.HIGH)
                    sequence_total.append("1")
                    print(sequence_total)
                    time.sleep(0.5)
                if  GPIO.input(button_Pin2) == GPIO.LOW :
                    GPIO.output(led_Pin2, GPIO.HIGH)
                    sequence_total.append("0")
                    print(sequence_total)
                    time.sleep(0.5)

    else:
        print("Invalid choice select again! ")
```

```



except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()


```

3.2 Appendix-B : Problem Statement-2

```
#importing the required libraries

import serial                   #for serial communication
import RPi.GPIO as GPIO         # To use GPIO pins
import time                     # to introduce delay
GPIO.setwarnings(False)         #to avoid any false warnings
GPIO.setmode(GPIO.BOARD)        #using board channel


ser = serial.Serial('/dev/ttyACM0', 9600)    #as found from ls /dev/tty/ACM*

string_input=input("Enter a string: ")          #taking input from user
string_input_encod = string_input.encode()                      #encoding to send to serial monitor
print("Encode: ",string_input_encod)
ser.write(string_input_encod)                               #to write to serial monitor

# Converting String to binary
encode_list = ''.join(format(i, 'b') for i in bytearray(string_input, encoding='utf-8'))
print(encode_list)
const=str(encode_list)

#taking into list
list_temp = []
for c in const:
    list_temp.append(c)
print(list_temp)

GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)

#Start sending the bits
GPIO.output(10,GPIO.HIGH)
for k in range (len(list_temp)+1):
    if (list_temp[k-1]==1):
        GPIO.output(11,GPIO.HIGH)
    else:
        GPIO.output(11,GPIO.LOW)

GPIO.output(10,GPIO.LOW)
```

```
#stop sending the bits






________________________________________________________________________________
```

3.3 Appendix-C : Problem Statement-2a

```

const byte buttonPin = 8;
int buttonState= 0;


void setup() {
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), blink, CHANGE);
  Serial.begin(9600);
}

void loop() {
}

void blink() {

buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
Serial.println (”1”);
delay(2000);
  }
  else {
    Serial.println (”0”);

  }
}

________________________________________________________________________________
char r;
//volatile byte state = LOW;
void setup()
{
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(7,INPUT);
pinMode(3,INPUT);
Serial.print(”  Sent data:”);
//attachInterrupt(digitalPinToInterrupt(3), blink, HIGH);
}

void loop()
{
```

```
if(Serial.available()){           //From RPi to Arduino
    r =Serial.read() ;  //conveting the value of chars to integer

    Serial.print(r);

}
}


--------------------------------------------------------------------------------
```

References

[1] *How to Connect and Interface a Raspberry Pi With an Arduino*. https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino.

[2] *led-blink*. https://raspberrypihq.com/making-a-led-blink-using-the-raspberry-pi-and-pytho

[3] *pins/model*. https://pi4j.com/1.2/pins/model-3b-rev1.html.

[4] *Raspberry Pi Arduino Serial*. https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/.

[5] *VNC*. https://www.raspberrypi.org/documentation/remote-access/vnc/.