

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

1	Problem Statement 1	3
1.1	Problem Statement	3
1.2	Equipment Required	3
1.3	Algorithm and Implementation	4
1.4	Program Structure	5
1.5	Difficulties faced	6
1.6	Screenshots	6
2	Problem Statement 2	8
2.1	Problem Statement	8
2.2	Equipment Required	8
2.3	Algorithm and Implementation	8
2.4	Program Structure	9
2.5	Difficulties faced	10
2.6	Screenshots	10
3	Appendix	11
3.1	Appendix-A : Problem Statement-1	11
3.2	Appendix-B : Problem Statement-2	15
3.3	Appendix-C : Problem Statement-2a	16

List of Figures

1	Problem Statement-1 Console input	6
2	Problem Statement-1 Overlapping	7
3	Problem Statement-1 Non-overlapping	7
4	Problem Statement-2 Message sent from Rpi	10
5	Problem Statement-2 Message received on Arduino	10

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)

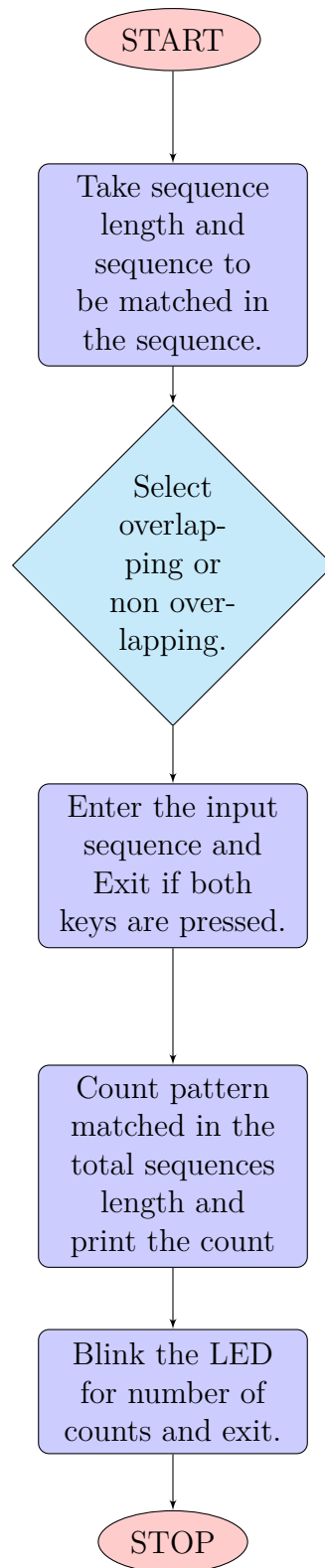
1.2 Equipment Required

- 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



1.5 Difficulties faced

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

1.6 Screenshots

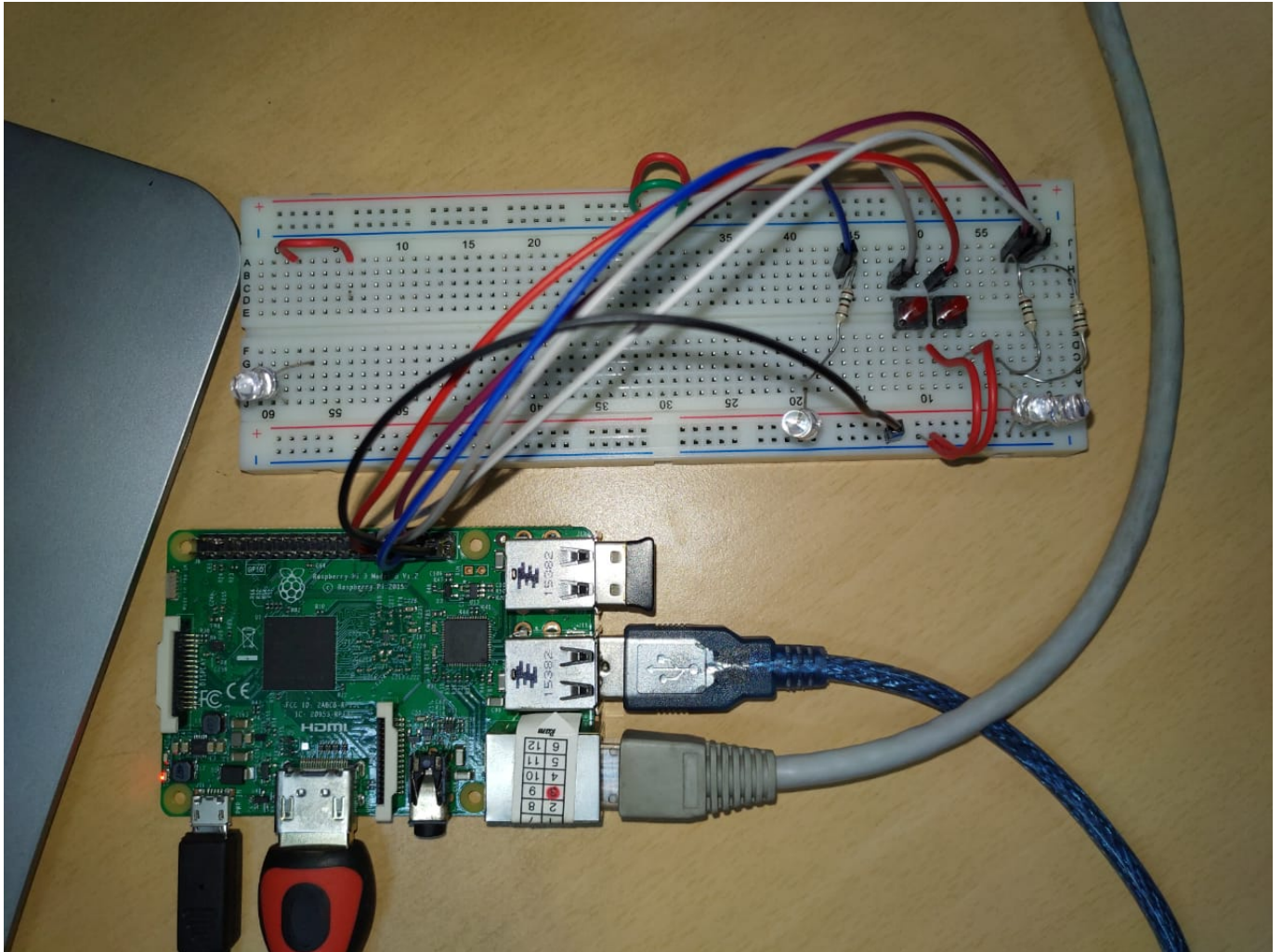


Figure 1: Problem Statement-1 Console input

```

===== RESTART =====
>>> %Run A8_psl.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', '1']
['0', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1', '0']
['0', '1', '0', '1', '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_psl.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

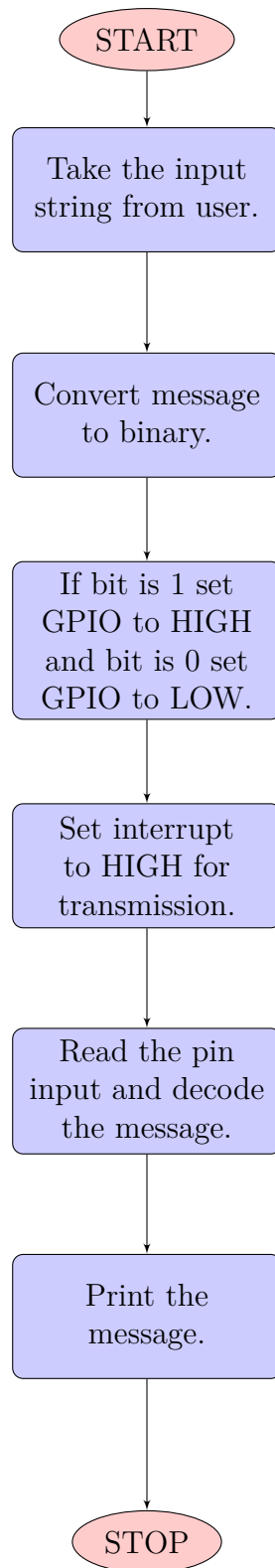
2.2 Equipment Required

- 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.

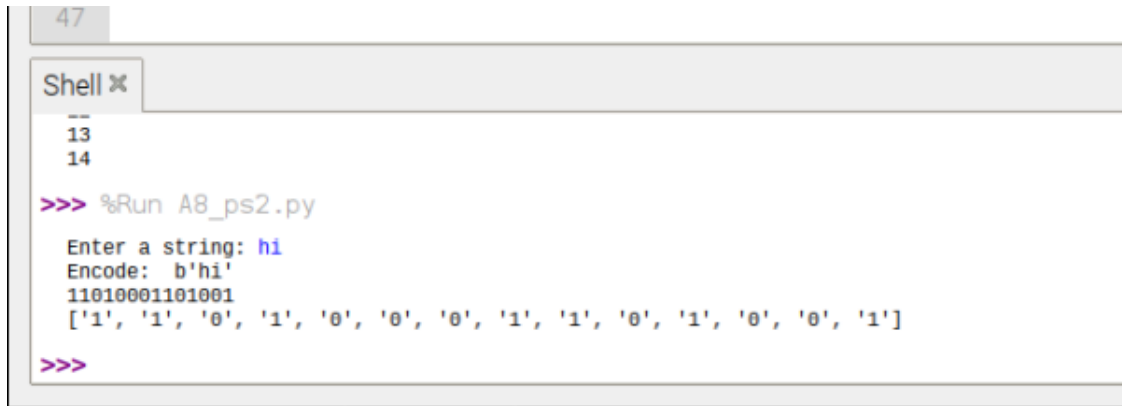
2.4 Program Structure



2.5 Difficulties faced

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

2.6 Screenshots



```
47
Shell x
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

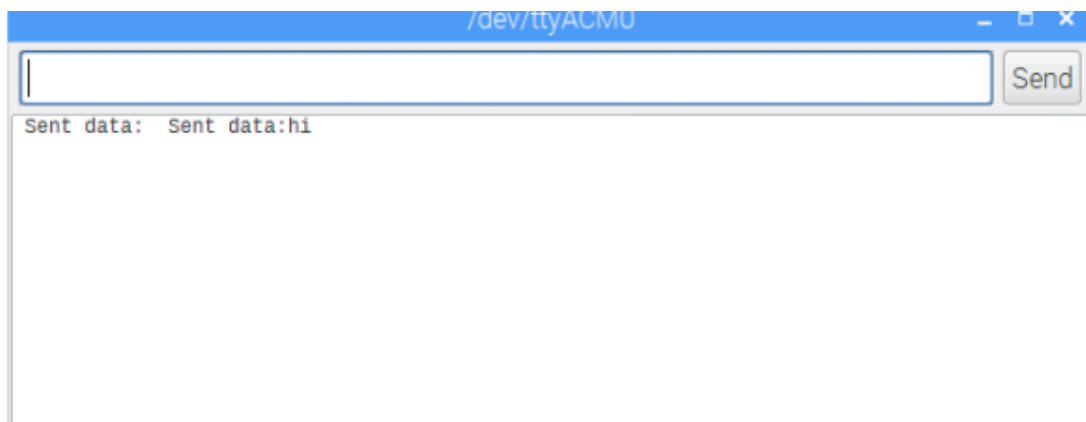


Figure 5: Problem Statement-2 Message received on Arduino

3 Appendix

3.1 Appendix-A : Problem Statement-1

```
1
2 # Problem Statement 1 – Sequence detector
3
4 # * Read button input
5 # * Output on LEDs (Blinking n times)
6
7 import RPi.GPIO as GPIO
8 import time
9 GPIO.setwarnings(False)
10
11 # Pin Definitions
12 button_Pin1 = 29
13 button_Pin2 = 31
14
15 led_Pin1 = 32
16 led_Pin2 = 36
17 led_Pin_op = 35
18
19 sequence_given=[]
20 sequence_total=[]
21
22
23 # Setup GPIO
24 GPIO.setmode(GPIO.BOARD)
25 GPIO.setup(button_Pin1,GPIO.IN, pull_up_down=GPIO.PUD_UP)
26 GPIO.setup(button_Pin2,GPIO.IN, pull_up_down=GPIO.PUD_UP)
27 GPIO.setup(led_Pin1,GPIO.OUT)
28 GPIO.setup(led_Pin2,GPIO.OUT)
29 GPIO.setup(led_Pin_op,GPIO.OUT)
30
31 def pattern_match(list1,list2):
32     len1 = len(list1)
33     len2 = len(list2)
34     l1=[]
35     l2=[]
36     p = 0
37     j=0
38     while j < len2:
39         l1.clear()
40         l2.clear()
41         for i in range(len1):
42             x=list1[i]
43             y=list2[i+j]
44             l1.append(x)
45             l2.append(y)
46         #print("l1",l1)
47         #print("l2",l2)
48         if l1==l2:
49             p=p+1
```

```

50         j=j+len1
51     print(p)
52     return
53
54 # Infinite Loop
55 try:
56     length_seq = int(input("\n Enter the length of the sequence to be detected
57 : "))
58     print(length_seq)
59     print("Select the type for sequence detection: ")
60     print("(1) Overlapping ")
61     print("(2) Non-Overlapping ")
62     case_type = input()
63     print("Select type : ",case_type)
64     #####
65     if case_type=='1':
66         # take input sequence from the buttons
67         while 1:
68             if GPIO.input(button_Pin1) == GPIO.HIGH :                # Not
69 Pressed =====
70                 GPIO.output(led_Pin1 , GPIO.LOW)
71             if GPIO.input(button_Pin2) == GPIO.HIGH :                # Not
72 Pressed =====
73                 GPIO.output(led_Pin2 , GPIO.LOW)
74                 if GPIO.input(button_Pin1) == GPIO.LOW :
75                     GPIO.output(led_Pin1 , GPIO.HIGH)
76                     sequence_given.append("1")
77                     print(sequence_given)
78                     time.sleep(0.5)
79                     if len(sequence_given)==length_seq:
80                         print("Given sequence: ", sequence_given )
81                         break
82
83                 if GPIO.input(button_Pin2) == GPIO.LOW :
84                     GPIO.output(led_Pin2 , GPIO.HIGH)
85                     sequence_given.append("0")
86                     print(sequence_given)
87                     time.sleep(0.5)
88                     if len(sequence_given)==length_seq:
89                         print("Given sequence: ", sequence_given )
90                         break
91
92     # Run Mode for listening
93     while 1:
94         if ((GPIO.input(button_Pin1) == GPIO.LOW) & (GPIO.input(
95 button_Pin2) == GPIO.LOW)): #both pressed simlteniously
96             x=len(sequence_total)
97             a=0
98             b=length_seq
99             count=0
100             check=[]
101             while True:
102                 check=sequence_total[a:b]

```

```

100         a=a+1
101         b=b+1
102         if(sequence_given==check):
103             count=count+1
104         if (b==x+1):
105             break
106     print("The count of pattern is: ",count)
107     time.sleep(2)
108     for i in range(0,a):
109         GPIO.output(led_Pin_op , GPIO.HIGH)
110         time.sleep(1)
111         GPIO.output(led_Pin_op , GPIO.LOW)
112         time.sleep(1)
113     print("Thank you !")
114     #exit()
115     if GPIO.input(button_Pin1) == GPIO.HIGH :           # Not
Pressed =====
116         GPIO.output(led_Pin1 , GPIO.LOW)
117     if GPIO.input(button_Pin2) == GPIO.HIGH :           # Not
Pressed =====
118         GPIO.output(led_Pin2 , GPIO.LOW)
119     if GPIO.input(button_Pin1) == GPIO.LOW :
120         GPIO.output(led_Pin1 , GPIO.HIGH)
121         sequence_total.append("1")
122         print(sequence_total)
123         time.sleep(0.5)
124     if GPIO.input(button_Pin2) == GPIO.LOW :
125         GPIO.output(led_Pin2 , GPIO.HIGH)
126         sequence_total.append("0")
127         print(sequence_total)
128         time.sleep(0.5)
129
130
131
132     elif case_type=='2':
133         # Calibration mode take input sequence from the buttons
134         while 1:
135             if GPIO.input(button_Pin1) == GPIO.HIGH :           # Not
Pressed =====
136                 GPIO.output(led_Pin1 , GPIO.LOW)
137             if GPIO.input(button_Pin2) == GPIO.HIGH :           # Not
Pressed =====
138                 GPIO.output(led_Pin2 , GPIO.LOW)
139             if GPIO.input(button_Pin1) == GPIO.LOW :
140                 GPIO.output(led_Pin1 , GPIO.HIGH)
141                 sequence_given.append("1")
142                 print(sequence_given)
143                 time.sleep(0.5)
144             if len(sequence_given)==length_seq:
145                 print("Given sequence: ", sequence_given )
146                 break
147
148             if GPIO.input(button_Pin2) == GPIO.LOW :
149                 GPIO.output(led_Pin2 , GPIO.HIGH)

```

```

150         sequence_given.append("0")
151         print(sequence_given)
152         time.sleep(0.5)
153         if len(sequence_given)==length_seq:
154             print("Given sequence: ", sequence_given )
155             break
156
157     # Run Mode for listening
158     while 1:
159         if ((GPIO.input(button_Pin1) == GPIO.LOW) & (GPIO.input(
button_Pin2) == GPIO.LOW)):    #both pressed simultaniously
160             print("The sequence you entered is: ",sequence_total)
161             #pattern_match(sequence_given , sequence_total )
162             x=len(sequence_total)
163             a=0
164             b=length_seq
165             count=0
166             check=[]
167             while True:
168                 check=sequence_total[a:b]
169                 a=a+1
170                 b=b+1
171                 if(sequence_given==check):
172                     count=count+1
173                 if b==x+1:
174                     break
175             print("The count of pattern is: ",count)
176             print("Thank you !")
177             time.sleep(2)
178             for i in range(0,a):
179                 GPIO.output(led_Pin_op , GPIO.HIGH)
180                 time.sleep(1)
181                 GPIO.output(led_Pin_op , GPIO.LOW)
182                 time.sleep(1)
183             #exit()
184             if GPIO.input(button_Pin1) == GPIO.HIGH :                # Not
Pressed =====
185                 GPIO.output(led_Pin1 , GPIO.LOW)
186             if GPIO.input(button_Pin2) == GPIO.HIGH :                # Not
Pressed =====
187                 GPIO.output(led_Pin2 , GPIO.LOW)
188             if GPIO.input(button_Pin1) == GPIO.LOW :
189                 GPIO.output(led_Pin1 , GPIO.HIGH)
190                 sequence_total.append("1")
191                 print(sequence_total)
192                 time.sleep(0.5)
193             if GPIO.input(button_Pin2) == GPIO.LOW :
194                 GPIO.output(led_Pin2 , GPIO.HIGH)
195                 sequence_total.append("0")
196                 print(sequence_total)
197                 time.sleep(0.5)
198
199     else :
200         print("Invalid choice select again! ")

```

```

201
202
203
204 except KeyboardInterrupt:
205     pwm.stop()
206     GPIO.cleanup()
207
208
209

```

3.2 Appendix-B : Problem Statement-2

```

1 #importing the required libraries
2
3 import serial                                #for serial communication
4 import RPi.GPIO as GPIO                    # To use GPIO pins
5 import time                                 # to introduce delay
6 GPIO.setwarnings(False)                   #to avoid any false warnings
7 GPIO.setmode(GPIO.BOARD)                  #using board channel
8
9
10 ser = serial.Serial('/dev/ttyACM0', 9600)    #as found from ls /dev/tty/ACM*
11
12 string_input=input("Enter a string: ")      #taking input from user
13 string_input_encode = string_input.encode()  #encoding to
14     send to serial monitor
15 print("Encode: ",string_input_encode)
16 ser.write(string_input_encode)              #to write to serial
17     monitor
18
19 # Converting String to binary
20 encode_list = ''.join(format(i, 'b') for i in bytearray(string_input, encoding
21     ='utf-8'))
22 print(encode_list)
23 const=str(encode_list)
24
25 #taking into list
26 list_temp = []
27 for c in const:
28     list_temp.append(c)
29 print(list_temp)
30
31 GPIO.setup(10, GPIO.OUT)
32 GPIO.setup(11, GPIO.OUT)
33
34 #Start sending the bits
35 GPIO.output(10,GPIO.HIGH)
36 for k in range (len(list_temp)+1):
37     if (list_temp[k-1]==1):
38         GPIO.output(11,GPIO.HIGH)
39     else:
40         GPIO.output(11,GPIO.LOW)
41
42 GPIO.output(10,GPIO.LOW)

```



```
40 #stop sending the bits
41
42
43
44
45
46
47
```

3.3 Appendix-C : Problem Statement-2a

```
1
2 const byte buttonPin = 8;
3 int buttonState= 0;
4
5
6 void setup() {
7   pinMode(2, INPUT_PULLUP);
8   attachInterrupt(digitalPinToInterrupt(2), blink, CHANGE);
9   Serial.begin(9600);
10 }
11
12 void loop() {
13 }
14
15 void blink() {
16
17   buttonState = digitalRead(buttonPin);
18
19   if (buttonState == HIGH) {
20     Serial.println ("1");
21     delay(2000);
22   }
23   else {
24     Serial.println ("0");
25
26   }
27 }
28
29
30 char r;
31 //volatile byte state = LOW;
32 void setup()
33 {
34   // put your setup code here, to run once:
35   Serial.begin(9600);
36   pinMode(7,INPUT);
37   pinMode(3,INPUT);
38   Serial.print(" Sent data:");
39   //attachInterrupt(digitalPinToInterrupt(3), blink, HIGH);
40 }
41
42 void loop()
43 {
```

```
44 if (Serial.available()) {           //From RPi to Arduino
45     r = Serial.read() ; //conveting the value of chars to integer
46
47     Serial.print(r);
48
49 }
50 }
51
52
53
```

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-python>
- [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/>.