



# VISVESVARAYA NATIONAL INSTITUTE OF TECHNOLOGY (VNIT), NAGPUR

---

## Embedded Systems (ECL403 & ECP403)

### Lab Report

---

*Submitted by :*

Maddipatla Vignesh Srinivas (BT19ECE065)

Semester 5

*Submitted to :*

Dr. Ankit A. Bhurane

(Course Instructor)

Department of Electronics and Communication Engineering,  
VNIT Nagpur

# Contents

1	Experiment-1: Alternate LED Blink . . . . .	2
2	Experiment-2: Seven segment display Interface . . . . .	5
3	Experiment-3: keypad Interfacing . . . . .	8
4	Experiment-4: LCD Interfacing . . . . .	12
5	Experiment-5: Serial Communication . . . . .	16
6	Experiment-6: Interfacing Stepper Motor . . . . .	18
7	Experiment-7: Touch Sensor in ESP-32 . . . . .	21
8	Experiment-8: Bluetooth Module in ESP-32 . . . . .	23
9	Experiment-9: Control ESP-32 using Wifi . . . . .	25
10	Experiment-10: Connect ESP-32 with cloud . . . . .	27
11	Experiment-11: Control ESP-32 with Telegram Bot . . . . .	30
12	Experiment-12: Control ESP-32 with Google Assistant . . . . .	35
13	Experiment-13: FreeRTOS in ESP-32 . . . . .	39
14	Experiment-14: Task Scheduling in ESP-32 . . . . .	42
15	Experiment-15: Connect ESP-32 to Firebase Database . . . . .	44

## Experiment-1: Alternate LED Blink

**Aim:** To interface LEDs with 8051 micro controller and make them blink alternatively with a delay of 1 second

**Requirements:** EdSim51, Keil 8051

**Description:** In this experiment, we'll use the Assembly and C programming languages to create a logic that will display alternate LED blinks with a 1 second delay using 8051 micro controller.

**Code simulation and Comments:** The ALP and C code

```

1  BACK:MOV A,#0AAH ...
      ;Alternate 1s ( 0AAH = ...
      10101010)
2  MOV P1,A
3  ACALL DELAY
4  MOV A,#055H ; Next ...
      alternate blink (055H = ...
      01010101)
5  MOV P1,A
6  ACALL DELAY
7  SJMP BACK ;Continuously ...
      blink LED
8  DELAY:
9  MOV R7,#064H ;Using this ...
      to create a delay of ...
      1second
10 LOOP:MOV TMOD,#01H
11
12      ;10ms delay
13      MOV TH0, #0DBH
14      MOV TLO, #0FFH
15
16      SETB TR0
17 L1:  JNB TFO,L1
18      CLR TR0
19      CLR TFO
20      DJNZ R7, LOOP
21      RET

```

```

1  #include<reg51.h>
2  int i=-1;
3  P2 = 0x00;
4  void time_delay(void) {
5  for(i=-1;i<499;i++){
6  //timer(16-bit)
7  TMOD=0x01;
8
9  //store least byte
10 TLO=0xC3;
11
12 //store highest byte
13 TH0=0xEF;
14 TR0=1;
15 while(TFO!=0){}
16 TR0 = 0;
17 // here, we clear the bit ...
   in the flag for ...
   further calc
18 TFO = 0;
19 }
20 void main(){
21 while(1){
22     P2 = 0xAA;
23     time_delay();
24     P2 = 0x55;
25     time_delay();
26 }
}

```

**Calculations:** To create a 10 milliseconds (10ms) delay, we need to store 0DBH and 0FFH values in TH0 and TL0 respectively

$$TH0 = 0DBH$$

$$TL0 = 0FFH$$

Now, to create 1 second(1s) delay,

$$1s = 100*(10ms)$$

So, we will run the delay programme 100 times

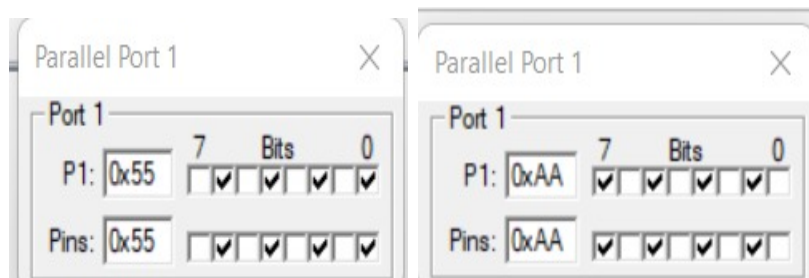
So, store 100D(064H) in R7 and decrease it till 0. This will create a delay of 1 second

**Output:** The outputs of the Assembly programme



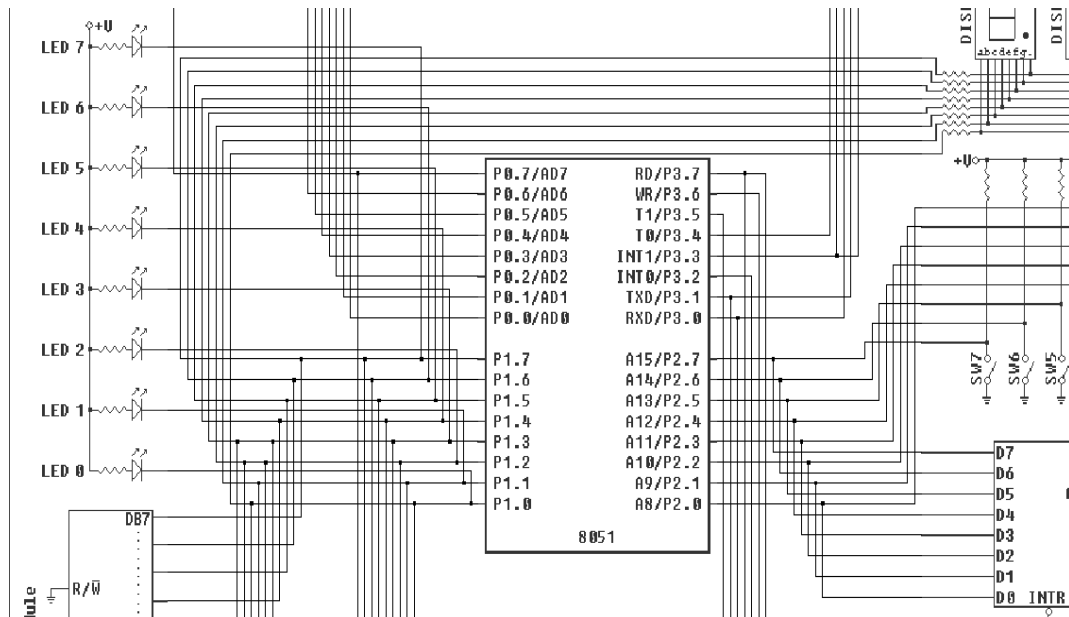
Figure(1.1): Alternate LED blinking

The outputs of the Embedded C programme (Keil)



Figure(1.1): Alternate LED blinking

**Observations & Discussions:** LEDs are basically active low. They are connected in a common anode setup. So, we'll be sending matching inputs to it to make LEDs glow in an other manner (alternate manner).The logic is basically to pass relevant input to blink alternate LEDs and create a delay of 1 second and after the delay, LEDs that are off initially will glow now. We can observe the outputs in the above figures



Figure(1.2): The interfacing of LEDs with 8051 microcontroller

**Conclusion:** We wrote Assembly and C language programmes to do alternate blinking of LEDs with a 1 second time delay in this experiment. We also looked at the logic diagram, which shows how the LED bank is connected with the 8051 microcontroller circuit.

## Experiment-2: Seven segment display Interface

**Aim:** To interface 7-segment display with 8051 micro controller and display numbers on it one after another

**Requirements:** EdSim51, Keil& Proteus 8

**Description:** In this experiment, we need to write a logic in ALP and C languages to display numbers(0-9) on 7 segment display that is interfaced with 8051 micro-controller

**Code simulation and Comments:** -

```

1  START: MOV RO, #0AH ...
        ;counter (from 10 to 0)
2      MOV DPTR, #val ...
        ;DPTR to ...
        string(to 1st ...
        element)
3  BACK: MOV A, #00H
4  LABEL: MOVC A, @A+DPTR
5      MOV P1, A ;send value ...
        to p1
6      ACALL DELAY
7      INC DPTR ; DPTR gets ...
        pointed to next ...
        string value
8      DJNZ RO, BACK; Decrease ...
        counter
9  SJMP START
10 DELAY:
11      MOV RO, #08H
12  LP2:  MOV R1, #0FFH
13  LP1:  MOV R2, #0FFH
14  LP3:  DJNZ R2, LP3
15      DJNZ R1, LP1
16      DJNZ R0, LP2
17      RET
18  val:
19  DB ...
        COH, F9H, A4H, B0H, 99H, 92H, 82H,
20  F8H, 80H, 90H

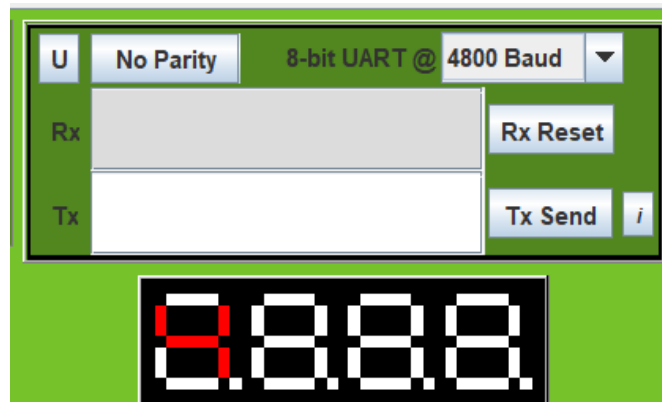
```

```

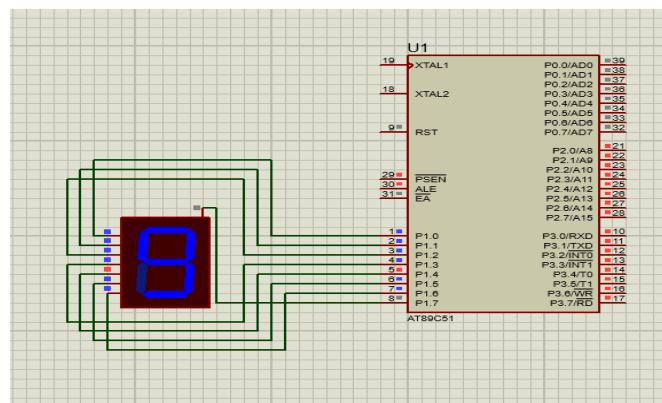
1  #include "reg51.h"
2
3  //this function is used to ...
   create delay
4  void create_delay(unsigned ...
   int run_ms)
5  {   unsigned int i;
6      while(run_ms!=0){
7          i = 1120;
8          while(i!=0){
9              i--;
10             }
11             run_ms--;
12         }
13     }
14     int main(){
15         char c;
16         char val_seg[]={
17             0x3f,0x06,0x5b,0x4f,0x66,
18             0x6d,0x7d,0x07,0x7f,0x6f,
19             0x77,0x7c,0x39,0x5e,0x79,0x71
20         };
21         c = 0x00;
22         while(1){
23             while(c<=0x0f){
24                 P2 = seg_code[c]; //num ...
                to be displayed
25                 create_delay(1200); //creating ...
                delay
26                 c++;
27             }
28         }

```

**Output:** Numbers will be displayed from 0 to 9. Below image is taken while it is displaying number 4

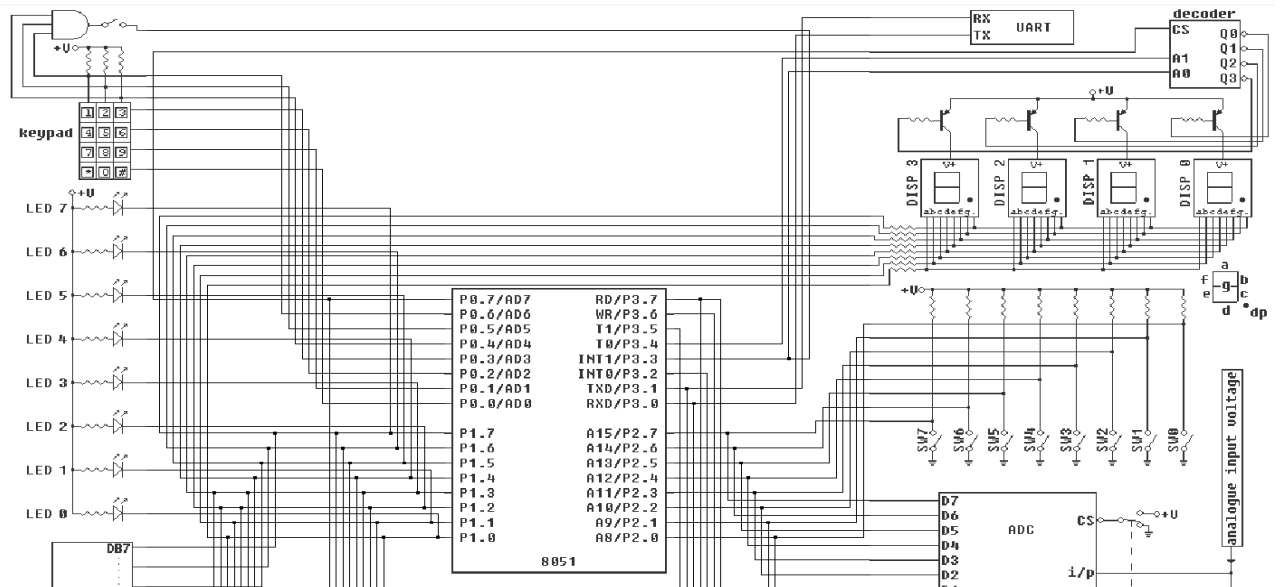


Figure(2.1): 7 segment Display displaying number 4



Figure(2.2): 7 segment Display displaying number 9 (Proteus 8)

**Observations & Discussions:** A Seven segment display is basically madeup of 8 LEDs which are connected in a definite way such that each LED represents a segment. Figure(2.3) shows how 8051 micro controller is interfaced with the seven segment display. We basically initialise a counter R0 and with the help of this counter, we take data sequentially from 0 to 9 (counter reduces for every successive number). The necessary operations are performed and the value of the number in the string is displayed. It starts displaying numbers from 0 to 9 with some delay. Figure(2.1) shows number 4 which is a part of the sequential display



Figure(2.3): Circuit Diagram connecting 7 segment display and 8051 micro-controller

**Conclusion:** Thus, we have interfaced seven segment display with the 8051 micro controller and displayed data (numbers from 0 to 9) sequentially with some delay.



## Experiment-3: keypad Interfacing

**Aim:** Interfacing keypad with 8051 micro controller

**Requirements:** EdSim51 simulator, Keil 8051, Proteus 8

**Description:** Develop a logic in Assembly and C languages to display the selected key (on keypad) on LED array with default connections

**Code simulation and Comments:** -

```

1  LOOP:
2  MOV PO, #11111110B
3
4  ;As we can't show star
5  ;Assume 'C' as star
6  JNB PO.6, STAR
7  JNB PO.5, ZERO
8
9  ;As we can't show Hash
10 ;Assume '3 lines' ...
    as Hash
11 JNB PO.4, HASH
12
13 MOV PO, #11111101B
14 JNB PO.6, SEVEN
15 JNB PO.5, EIGHT
16 JNB PO.4, NINE
17
18 MOV PO, #11111011B
19 JNB PO.6, FOUR
20 JNB PO.5, FIVE
21 JNB PO.4, SIX
22
23 MOV PO, #11110111B
24 JNB PO.6, ONE
25 JNB PO.5, TWO
26 JNB PO.4, THREE
27
28 MOV A, #0FFH ;IF ...
    NONE PRESSED-ALL OFF
29 JMP HERE
30 ;code continues in ...
    next page

```

```

1  #include<reg51.h>
2  //Array of values 0-9
3  unsigned char Seg[] = ...
    {0xC0,0xF9,0xA4,0xB0,0x99,
4    0x92,0x82,0xF8,0x80,0x90};
5  void display(unsigned char num)
6  {
7    //LED binary Equivalent
8    P0 = ~(num);
9    //7-seg display
10    P2 = Seg[num];
11    return;
12 }
13
14 void main()
15 {
16   while(1)
17   {
18     //Initially LEDs are OFF.
19     P0 = 0xFF;
20     P2 = 0xFF;
21     P1 = 0xFF;
22
23     P1_0 = 0; //Row 4 check
24     //Column check
25     if( P1_5 == 0)
26     {display(0);
27      break;}
28     P1_0=P1_1=P1_2=P1_2=1;
29     P1_3=0; //Row 1 check
30
31    // ...Code continues in next page...

```

```

1 ;code continues
2 ZERO: MOV A, #0C0H
3 JMP HERE
4 ONE: MOV A, #0F9H
5 JMP HERE
6 TWO: MOV A, #0A4H
7 JMP HERE
8 THREE: MOV A, #0B0H
9 JMP HERE
10 FOUR: MOV A, #099H
11 JMP HERE
12 FIVE: MOV A, #092H
13 JMP HERE
14 SIX: MOV A, #082H
15 JMP HERE
16 SEVEN: MOV A, #0F8H
17 JMP HERE
18 EIGHT: MOV A, #080H
19 JMP HERE
20 NINE: MOV A, #090H
21 JMP HERE
22 STAR: MOV A, #046H
23 JMP HERE
24 HASH: MOV A, #036H
25 JMP HERE
26
27 HERE: MOV P1, A
28 JMP LOOP

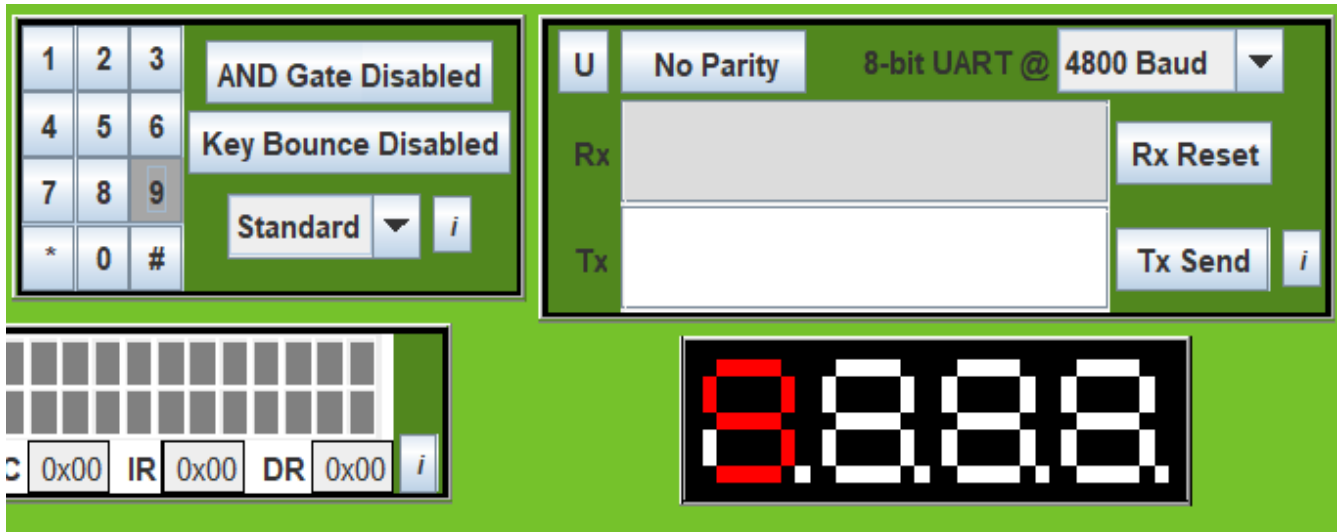
```

```

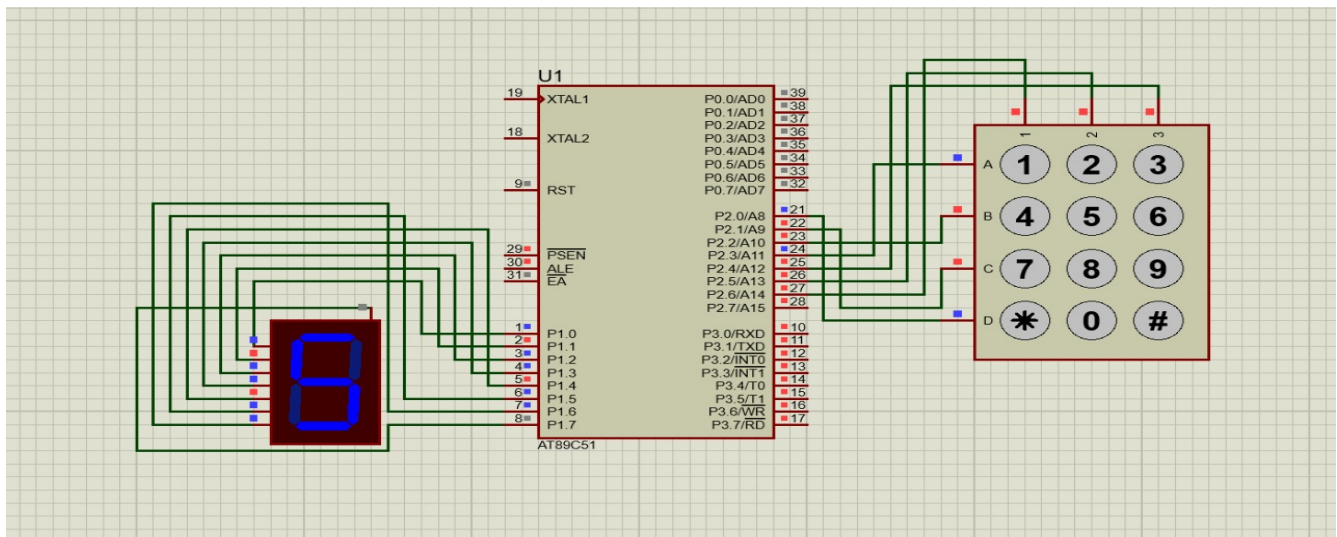
1 //code continues
2
3 //Column check
4 if (P1_6 == 0) //C1 check
5     {display(1);
6     break;}
7 else if(P1_5 == 0) //C2 check
8     {display(2);
9     break;}
10 else if(P1_4 == 0) //C3 check
11     {display(3);
12     break;}
13
14 P1_0=P1_1=P1_2=P1_2=1;
15 P1_2 =0; //Row 2 check
16 //Column check
17 if (P1_6 == 0)
18     {display(4);
19     break;}
20 else if(P1_5 == 0)
21     {display(5);
22     break;}
23 else if(P1_4 == 0)
24     {display(6);
25     break;}
26
27 P1_0=P1_1=P1_2=P1_2=1;
28 P1_1=0; //Row 3 check
29 //Column check
30 if (P1_6 == 0)
31     {display(7);
32     break;}
33 else if(P1_5 == 0)
34     {display(8);
35     break;}
36 else if(P1_4 == 0)
37     {display(9);
38     break;}
39 }
40 }

```

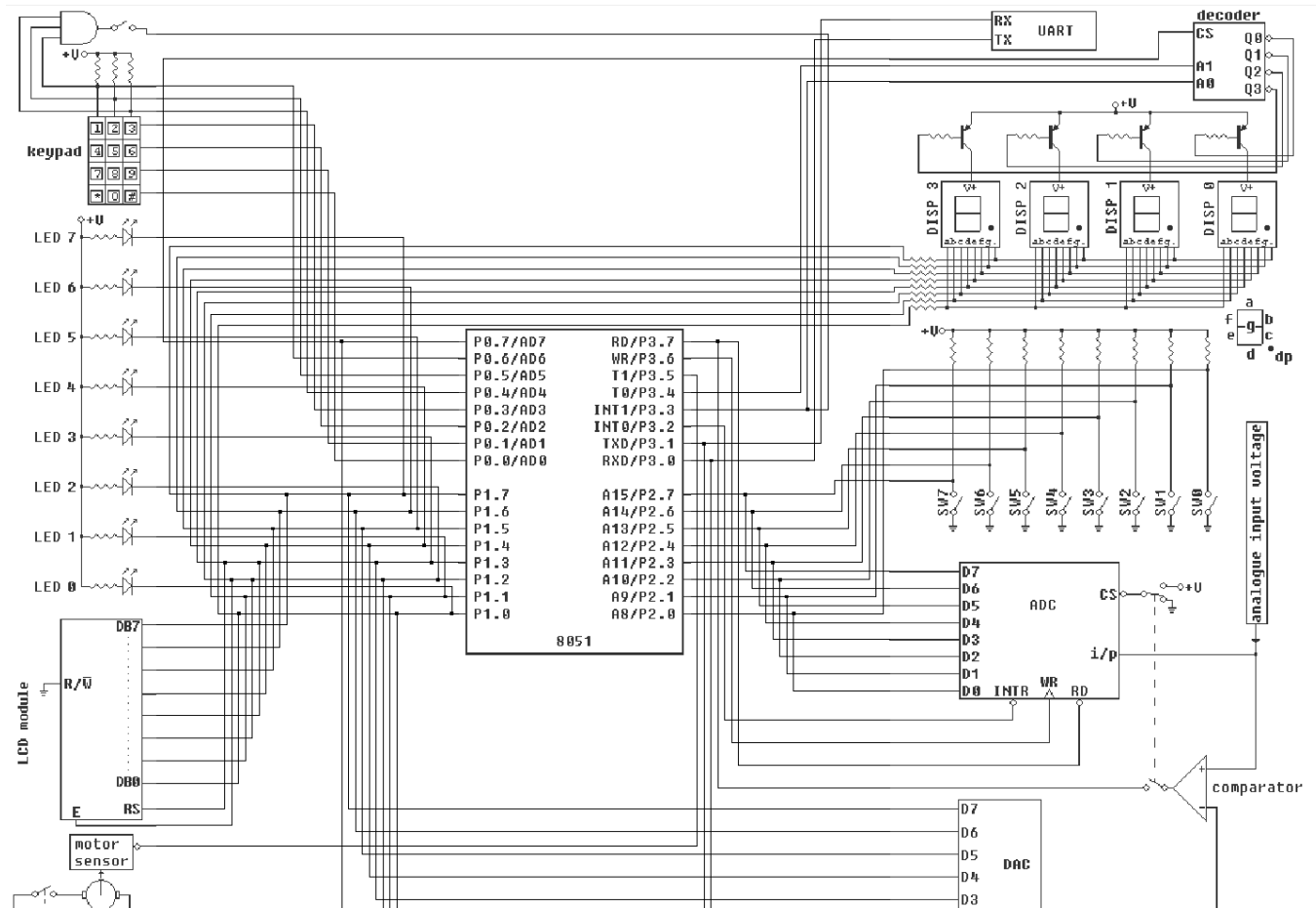
**Output:** Key 9 is pressed on the keypad and it is displayed on LED



Figure(3.1): Button 9 is pressed



Figure(3.2): Button 5 is pressed



**Observations & Discussions:** Here, the logic is developed in such a way that, first, we check whether any key on the keypad is pressed or not (If nothing is pressed, the LED will be off). We can see in the logic diagram that how port 0 pins are connected to the keypad rows and columns. The corresponding row checks whether a button is clicked using the respective port. If clicked, a particular row comes to know and the command is sent to respective function which takes in, value of that particular number and displays it on the LED.

**Conclusion:** Therefore, we have studied how the keypad is interfaced with the micro controller and learnt to display the pressed key on the LED.

## Experiment-4: LCD Interfacing

**Aim:** Interfacing LCD with 8051 micro controller and displaying text on it

**Requirements:** EdSim51 simulator, Keil 8051, Proteus 8

**Description:** Develop a logic in Assembly and C programming languages to interface LCD and display some text on it

**Code simulation and Comments:** -

```

1 CLR P0.3 ; Clearing RS
2
3 MOV P1, #38H;
4
5 SETB P0.2; Set EN high
6 CALL DELAY
7 CLR P0.2; Clearing EN
8
9 ; cursor and display on
10 MOV P1, #0EH
11
12 SETB P0.2
13 CALL DELAY
14 CLR P0.2
15
16 ; move cursor
17 MOV P1, #06H
18
19 SETB P0.2
20 CALL DELAY
21 CLR P0.2
22
23 SETB P0.3
24
25 ; the lookup table has the ...
   string that should be ...
   displayed
26 MOV A, #00H
27 MOV DPTR, #LOOKUP
28 MOV B, #00H
29 LOOP:
30 MOVC A,@A+DPTR
31
32 ;...code is continued in ...
   next page...

```

```

1 #include<reg51.h>
2 #include<string.h>
3
4 sbit rs=P3^1;
5 sbit rw=P3^2;
6 sbit en=P3^3;
7
8 void create_delay(unsigned ...
   int ms)
9 {   int i;
10     while (ms!=0){
11         i = 10;
12         while(i!=0){
13             i--;
14         }
15         ms--;
16     }
17     return;
18 }
19 void set_LCD(unsigned char ...
   set){
20     // write mode select
21     rw = 0;
22
23     // command mode
24     rs = 0;
25
26     P2 = set;
27     en = 1;
28     create_delay(1000);
29     en = 0;
30     return;
31 }
32 //...code is continued in ...
   next page

```

```

1  MOV P1,A
2  SETB P0.2
3  CALL DELAY
4  CLR P0.2
5  INC B
6  MOV A, B
7  CJNE A, #16, LOOP
8  RET
9  DELAY:
10 MOV RO, #0FOH
11 DJNZ RO, $
12 RET
13
14 LOOKUP:
15 DB "Vignesh"

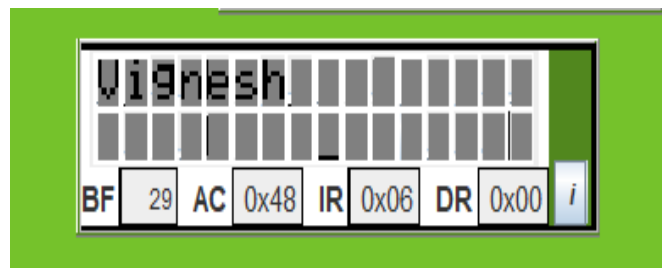
```

```

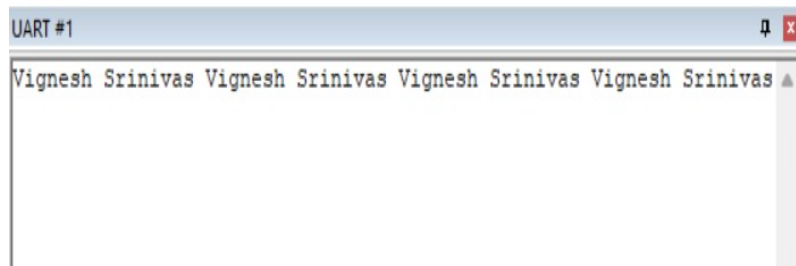
1  void LCD_input(unsigned ...
    char *input){
2      int i,len;
3      len = strlen(input);
4      for (i=0; i<len; i++){
5          // Data mode
6          rs = 1;
7
8          // write mode
9          rw = 0;
10
11         //passing ...
            character 1 by 1
12         P2 = input[i];
13
14         en = 1;
15         create_delay(1000);
16         en = 0;
17     }
18     return;
19 }
20 void main(){
21     set_LCD(0x38); // ...
                Setup 2 lines LCD
22     set_LCD(0x0E); // ...
                cursor setup
23     set_LCD(0x06); // ...
                increment cursor
24
25     LCD_input("Vignesh ...
                Srinivas");
26     set_LCD(0xC0);
27     create_delay(1000);
28 }

```

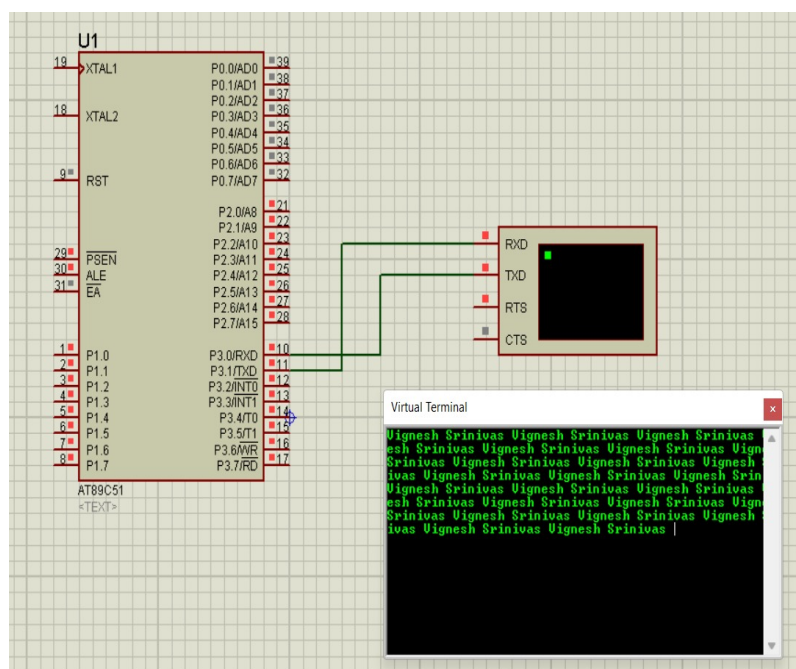
### Output:



Figure(4.1): Displaying the text provided in the look up

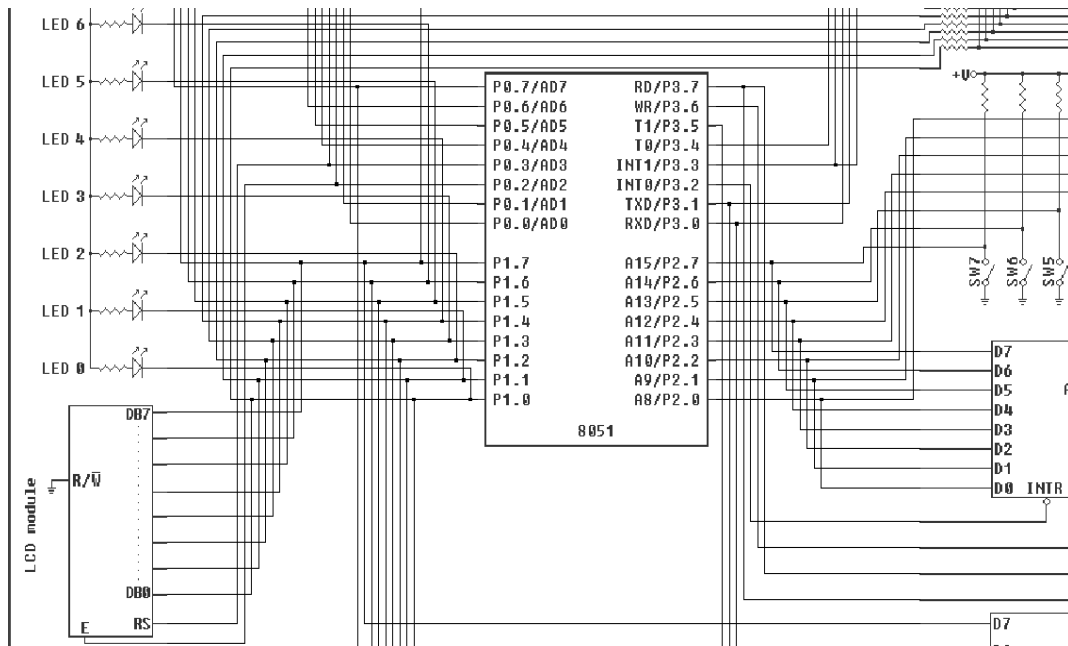


Figure(4.2): Displaying the text in UART in Keil



Figure(4.3): Proteus 8 output

**Observations & Discussions:** Figure(4.2) is the circuit diagram of interfacing of LCD with 8051 micro controller. The change in the circuit diagram (when compared with previous experiments) is that the EN and RS pins of LCD is connected to Port 0 instead of Port 1 (By Default, its actually connected to port 1). It's because port 1 is connected to LCD in such a way that Data,RS and E are connected to it. This gives an error. So, to avoid this, we do it. And all the necessary steps are followed in the code which uses the look up that contains the string that should be displayed.



Figure(4.2): Circuit Diagram of interface of 8051 with LCD

We can observe the text which is printed on the LCD in Figure(4.1).

**Conclusion:** Thus, In this experiment, we have interfaced the LCD with the 8051 micro controller, changed the port connections and displayed required text on it.



## Experiment-5: Serial Communication

**Aim:** To display name using Serial Communication/Transmission

**Requirements:** EdSim51 simulator, Keil 8051

**Description:** Develop a logic to display name using Universal Asynchronous Receiver-Transmitter serial communication. In this experiment, we use some important registers to serially transmit data. They are SCON(control center of serial communication), PCON(Improves Baudrate), TMOD(controls Transmission speed) and SBUF(keeps data for transmission and receiving).

We perform Mode 0 serial communication here. Initially, SCON is set to 00H. After that, the data is serially delivered one by one to SBUF, and it will wait until each character has been transmitted.

**Code simulation and Comments:** -

```

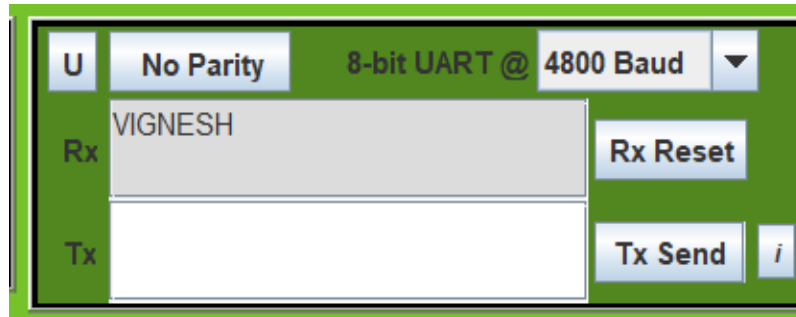
1  ;Storing Strings
2  MOV 49H, #'V'
3  MOV 4AH, #'I'
4  MOV 4BH, #'G'
5  MOV 4CH, #'N'
6  MOV 4DH, #'E'
7  MOV 4EH, #'S'
8  MOV 4FH, #'H'
9  MOV R1, #049H
10 MOV SCON, #40H
11 MOV PCON, #80H
12 MOV TMOD, #20H
13 MOV TH1, #0F3H
14 MOV TL1, #0F3H
15 SETB TR1
16 BACK: MOV A, @R1
17         JZ EXIT
18         INC R1
19         MOV SBUF, A
20 L1: JNB TI, L1
21 CLR TI
22 JMP BACK
23 EXIT: JMP EXIT

```

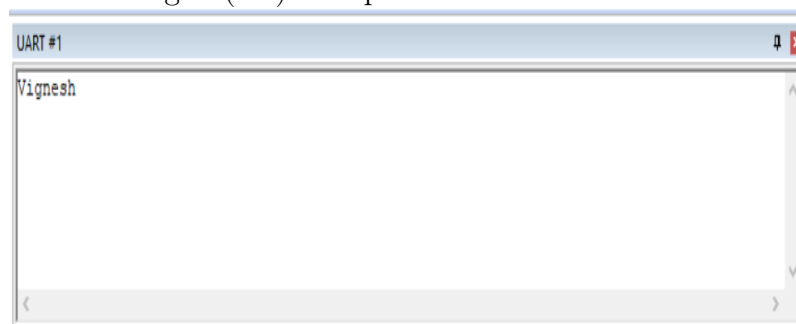
```

1  #include <reg51.h>
2  void main()
3  {
4      unsigned int i=0;
5      char string[] = "Vignesh";
6      SCON = 0x00;    // mode 0
7
8      while(i<sizeof(string)){
9          // character is sent
10         SBUF = string[i];
11
12         //Wait till the ...
13             transmission completes
14         while(TI==0);
15
16         // Reset the flag when ...
17             transmission is done
18         TI = 0;
19         i++;
20     }
21 }

```

**Output:**

Figure(5.1): Output of Edsim simulator



Figure(5.2): Output of Serially transmitted Data through mode 0 (Keil C)

**Observations & Discussions:** Serial communication is the method of transferring data bit by bit at a time in a serial fashion. Generally, wire is commonly used to transport data directly, however it does not operate well over long distances. For large distances, serial transmission is ideal. Universal Asynchronous Receiver-Transmitter (UART) is asynchronous serial communication protocol. This UART frame consists of the data to be transmitted, the start and the stop bits.

We performed the process mentioned above and successfully transmitted and displayed "Vignesh" (our data) on the monitor.

**Conclusion:** Thus we have successfully transferred data (through Mode 0) using serial transmission

## Experiment-6: Interfacing Stepper Motor

**Aim:** To interface stepper motor with 8051 microcontroller

**Requirements:** EdSim51 simulator , Keil 8051, Proteus 8

**Code simulation and Comments:** -

```

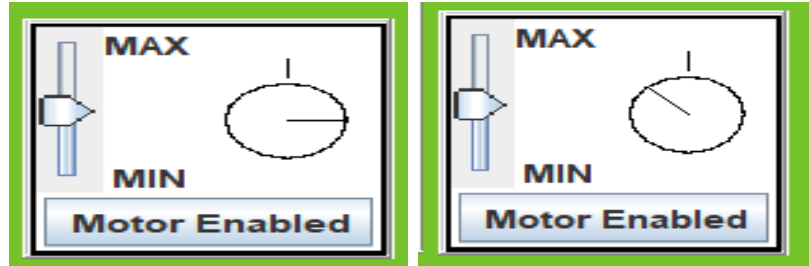
1  ;ALP to rotate motor with ...
    certain angle
2  ;From Figure(6.3), P3.0 ...
    has motor control bit 0 ...
    and P3.1 has motor ...
    control bit 1
3  LOOP:
4  SETB P3.1;sets the bit ...
    operand to 1
5
6  ACALL ROT ;ROT is our ...
    rotate fun
7
8  ROT:MOV R1,#02H
9
10 ; Time delay for rotation
11 DELAY: DJNZ R1,DELAY
12
13 ;if A=0, control is sent ...
    to given address
14 JZ FUNC
15
16 FUNC: CLR P3.1
17 JMP LOOP

```

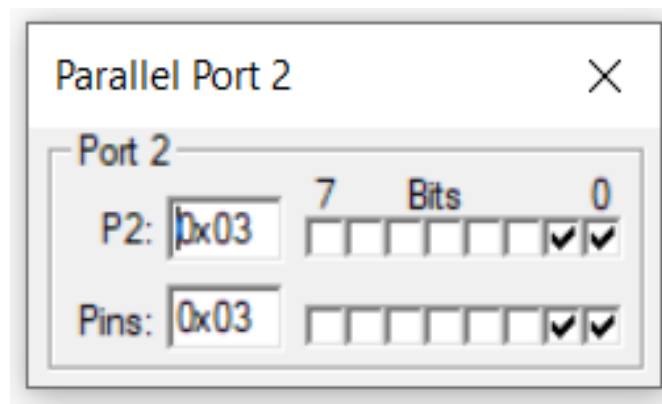
```

1  #include<reg51.h>
2  //delay function
3  void create_delay(unsigned ...
    int del){
4      unsigned int i,j;
5      for(i=0;i<del;i++){
6          for(j=0;j<120;j++){
7              continue;}}
8  }
9  //forward rotate
10 void clock(){
11     create_delay(120);
12     P2 = 0x0C;
13     create_delay(120);
14     P2 = P2-0x06;
15     create_delay(120);
16     P2 = P2-0x03;
17     create_delay(120);
18     P2 = P2+0x06;
19 //backward rotate
20 void anticlock(){
21     create_delay(120);
22     P2 = 0x09;
23     create_delay(120);
24     P2 = P2-0x06;
25     create_delay(1200);
26     P2 = P2+0x03;
27     create_delay(120);
28     P2 = P2+0x06;
29 void main(){
30     while(1){
31         clock();
32         create_delay(400);
33         anticlock();
34         create_delay(400);}
35 }

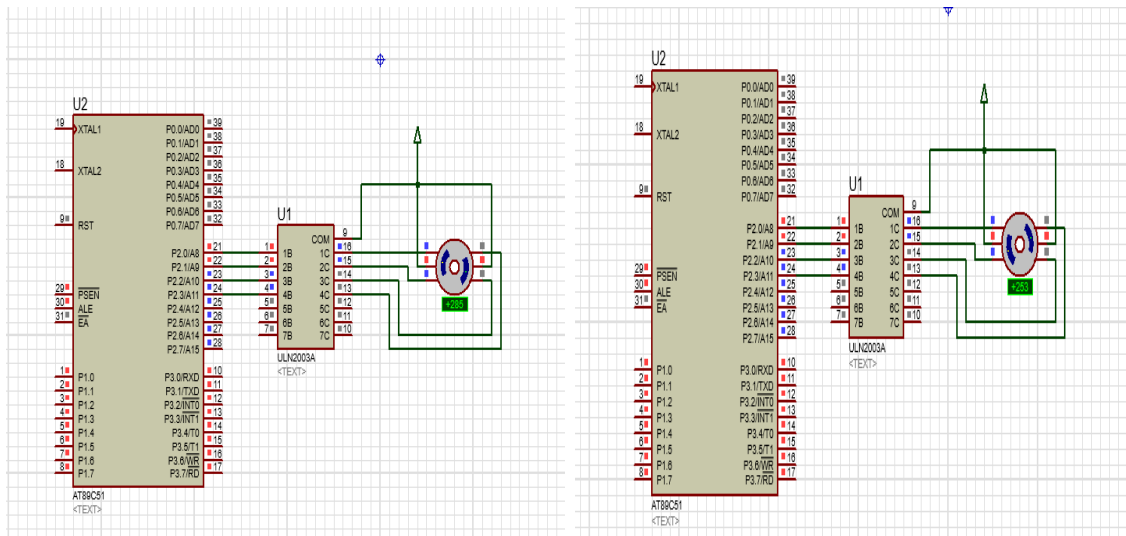
```

Output:

Figure(6.1): Stepper motor rotation

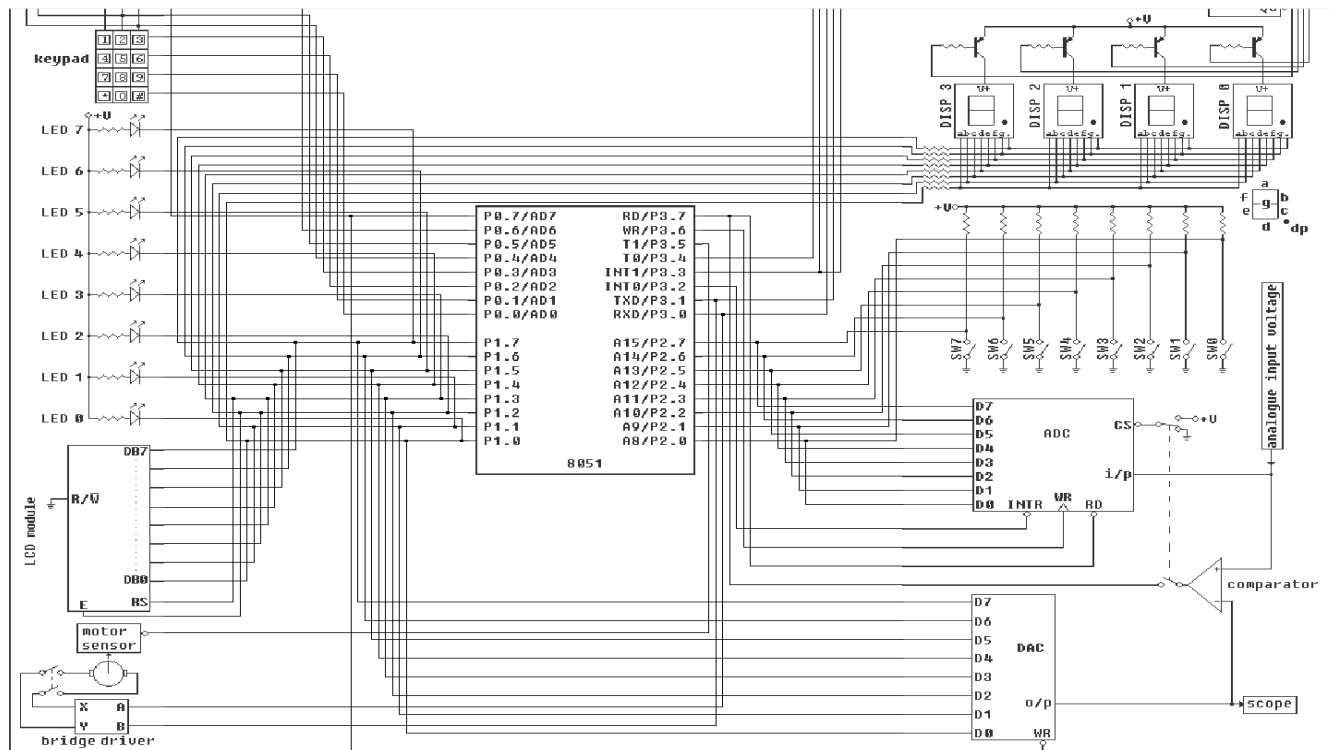


Figure(6.2): Output of Port-2 in Keil



Figure(6.3): Proteus 8 output

### Observations & Discussions:



Figure(6.3): Interfacing of Motor with 8051

In the above diagram, we can see how the stepper motor is connected to the microcontroller. Now, we need to continuously rotate this motor with some desired angle. So, by running the above code, we can obtain this. In keil, we have written an embedded C code that makes the motor rotate in forward and backward direction with some delay between both the operations. We have built the code and generated the hex file. We created a simulation in proteus 8. We interfaced motor with 8051 micro controller and used the hex file to perform the simulation. After doing this, we started the simulation and we got our desired outputs.

**Conclusion:** Thus, we have successfully interfaced stepper motor with 8051 microcontroller and rotated it.

## Experiment-7: Touch Sensor in ESP-32

**Aim:** To control the built-in LED of ESP-32 micro controller using touch sensor present in it. Also display the readings of sensor on Serial Monitor

**Requirements:** ESP-32 micro controller,

**Description:** This is the most basic ESP-32 application. We will learn how to use the touch sensor on the ESP32 development board and how to use the ESP32 touch sensor as a push button. We will blink ESP32's in-built LED using a touch

**Code simulation and Comments:** -

```
1 void setup() {
2   pinMode(2,OUTPUT); //output displayed by in-built LED
3   pinMode(4,INPUT); // GPIO pin 4 as input
4   Serial.begin(115200); //serial communication with BR 115200
5 }
6
7 void loop() {
8   int touch_read;
9   touch_read = touchRead(4); //Read values from GPIO pin 4
10  Serial.println(touch_read);
11  if(touch_read>81){
12    digitalWrite(2,LOW);
13  }
14  else{
15    digitalWrite(2,HIGH);
16  }
17 }
```

**Output:** The above process can be verified on the ESP-32 Hardware. As we have selected pin 4 to give touch value, if we touch that pin, LED glows else it won't. (The Blue LED is the one which should glow. Red one is just an indication that ESP-32 is connected properly to the PC).

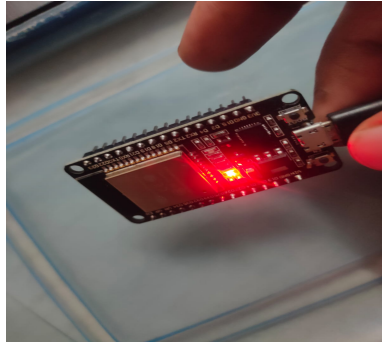


Figure 1: No touch

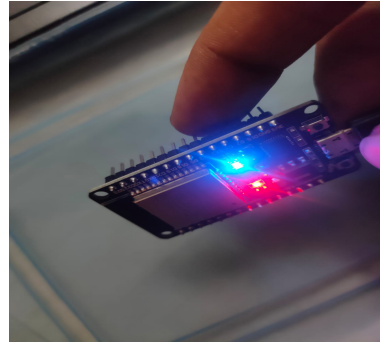
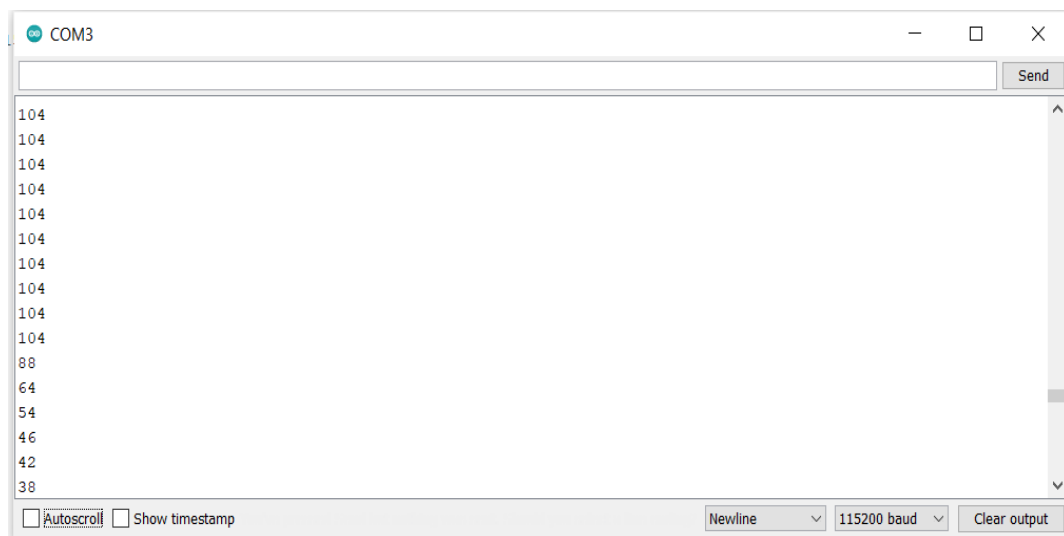


Figure 2: When touched



**Observations & Discussions:** Here, in this experiment, we have programmed an embedded C code to blink the in-built LED in ESP-32 using touch sensor. We should use any GPIO pin to take the touch as an input response. Here, we used GPIO pin 4 to do this. After giving the touch, it will store the readings of the touch in a variable. We see observe the serial monitor, if touch is not given, the value is default 104 and when the touch is given, it is less than 80. So if the value is less than 80 i.e, when the touch is given, the LED glows.

**Conclusion:** In this experiment, we have studied how to glow the in-built LED of ESP-32 micro-controller. This is one of the simplest codes and a building block in learning ESP-32

## Experiment-8: Bluetooth Module in ESP-32

**Aim:** To use and explore Bluetooth module of ESP-32 microcontroller

**Requirements:** ESP-32 microcontroller, serial bluetooth Terminal app (Android application)

**Description:** Install the serial bluetooth Terminal Application in your Android/iOS device. Upload the below code into your ESP-32 microcontroller (Install the libraries if you don't have). After uploading, follow the procedure mentioned below the code

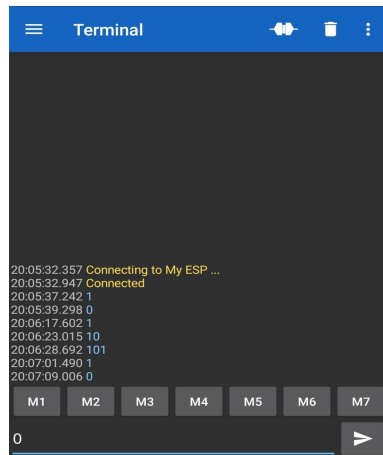
**Code simulation and Comments:** -

```
1 #include"BluetoothSerial.h"
2 BluetoothSerial SerialBT; //same as serial but uses bluetooth
3 char led_status;
4 void setup() {
5   SerialBT.begin("My ESP"); //Name of my ESP-32 Bluetooth(you can ...
   give any name)
6   pinMode(2,OUTPUT);
7 }
8 void loop() {
9   led_status = SerialBT.read(); //read from the app
10
11   // when 1 is clicked in app, LED ON
12   if(led_status == '1'){
13     digitalWrite(2,HIGH);
14   }
15
16   // when 0 is clicked in app, LED OFF
17   if(led_status == '0'){
18     digitalWrite(2,LOW);
19   }
20 }
```

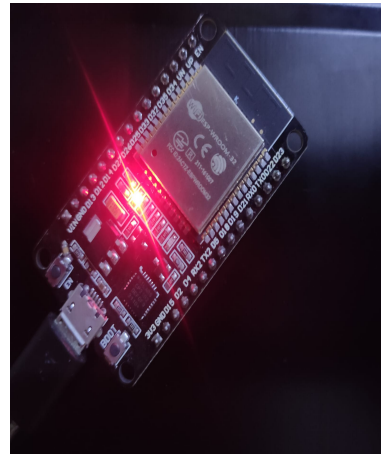
If you successfully upload this code then it implies your bluetooth module is activated. Now, your mobile phone recognises your ESP-32 bluetooth (To verify this, go to bluetooth and check devices name. You can find your bluetooth name there (In my case, it's "My ESP"). After you find this, go to the application and go to the "Devices" option and select your device. Now, you have connected the Bluetooth Application with your ESP-32 microcontroller.

**Output:**

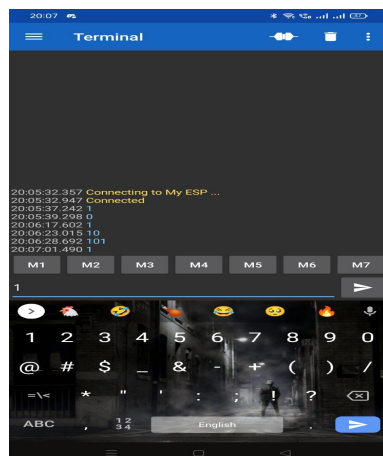




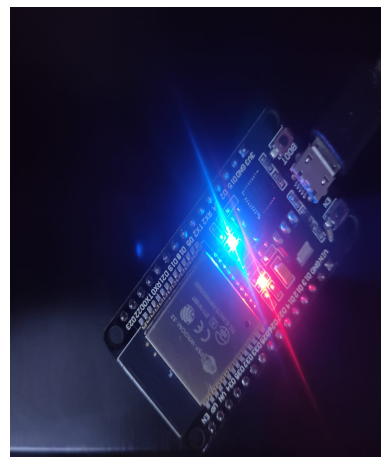
When 0 is given



Hardware output



When 1 is given



Hardware output

**Observations & Discussions:** In ESP-32 Micro controller, we have Bluetooth and Wifi modules which is why this micro controller is extensively used. In this experiment, we have used this bluetooth module to connect this with the Android Serial bluetooth Terminal application. We have already discussed this process in the Description section. Now, from the above outputs, we can observe that when 1 is given, LED glows and it doesn't if 0 is given.

**Conclusion:** Therefore, in this experiment, we have explored the bluetooth module in ESP-32 and learnt how to connect and use it to bluetooth accessible devices. We have connected ESP-32 with a bluetooth application and controlled the in-built LED in it using that Application.

## Experiment-9: Control ESP-32 using Wifi

**Aim:** To control ESP-32 micro controller using the WiFi module

**Requirements:** ESP-32 micro controller

**Description:** In this programme, we are going to explore the WiFi module in ESP-32. We have seen in the previous experiment that when we activate bluetooth module of ESP-32, we can see the connecting option in nearby bluetooth accessible device(for example, android phone). Now, if we activate the WiFi module, we can see it in "available WiFi network " option in our android device. Now, to do this, Upload the below code into your ESP-32 micro controller. We will activate the WiFi module and connect it with Android device. After that, we will create a webpage with two buttons and we will control the ESP-32 micro controller using that webpage

**Code simulation and Comments:** -

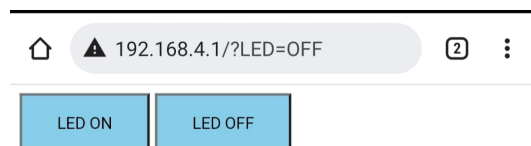
```

1  #include <WiFi.h>
2  //HTML code with stylings
3  String html = "<!DOCTYPE html>\n
4  <html>\n
5  <body>\n
6  <form>\n
7  <button name=\"LED\" button value=\"ON\" type=\"submit\" ...
      class=\"click\" style=\"background-color: skyblue;color: ...
      black;padding: 14px 28px; text-align:center\">LED ...
      ON</button> \n
8  <button name=\"LED\" button value=\"OFF\" type=\"submit\" ...
      class=\"click\" style=\"background-color: skyblue;color: ...
      black;padding: 14px 28px; text-align:center\">LED ...
      OFF</button> \n
9  </form> \n
10 </body \n
11 </html>";
12
13 WiFiServer server(80); //begin WiFi server
14 void setup() {
15     WiFi.softAP("V-ESP", "tony1234"); //Set your ESP-32 WiFi
16     Serial.begin(115200); //set baudrate(change in serial monitor ...
        too)
17     pinMode(2, OUTPUT); //Configure pin 2 as output
18     IPAddress IP = WiFi.softAPIP();
19     Serial.print("IP: " + String(IP));
20     server.begin();

```

```
21   digitalWrite(2, LOW); //Keeping LED OFF initially
22 }
23 void loop() {
24   WiFiClient client = server.available();
25   if(client){
26     String req = client.readStringUntil('\r');
27     if (req.indexOf("LED=ON")≥0){
28       digitalWrite(2, HIGH);
29     }
30     else if (req.indexOf("LED=OFF")≥0){
31       digitalWrite(2, LOW);
32     }
33     client.print(html);
34     req=" ";
35   }
36 }
```

Open the webpage by typing the IP address 192.168.1.184 in your browser. You'll see a HTML page



Figure(9.1): Webpage you created

**Observations & Discussions:** In ESP-32 Micro controller, we have Wifi module and using this, we need to connect to our android phone/PC. I.e, just as how wifi option appears in our mobile/pc when someone put on Wifi, the same happens here too (In my case Wifi name is "Wifiname" and password is "wifipass". Now, after connecting to this wifi, we should search the address 192.168.1.184 in your browser. We'll get the webpage you created like in Figure(9.1) (We can even add stylings). Now, if we click on LED ON button, the LED turns on and when LED OFF is clicked, it puts off.

**Conclusion:** Therefore, we have explored about the Wifi module in ESP-32 and learnt how to connect and use it with Wifi accessible devices. We have controlled the in-built LED of ESP-32 using the webpage.

## Experiment-10: Connect ESP-32 with cloud

**Aim:** Connect ESP-32 micro controller with thingspeak cloud, send data to it and visualise the Data

**Requirements:** ESP-32 microcontroller, Account in Thingspeak

**Description:** Create an account in thingspeak and use the "write API" key (which you'll find in your Thingspeak account) in the below code.

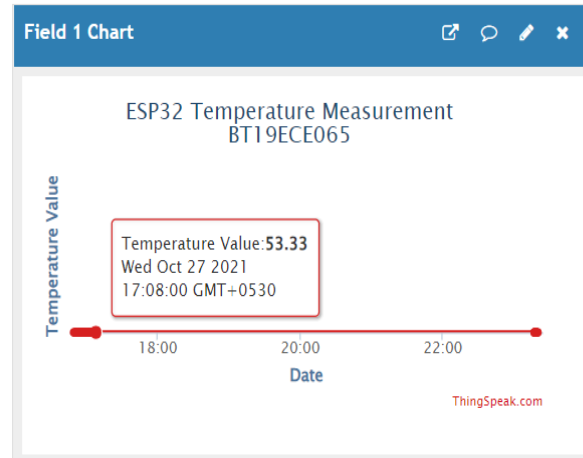
**Code simulation and Comments:** -

```
1 //for measuring temperature
2 #ifndef __cplusplus
3 extern "C" {
4 #endif
5 uint8_t temprature_sens_read();
6 #ifdef __cplusplus
7 }
8 #endif
9 uint8_t temprature_sens_read();
10 #include <WiFi.h>
11 #include <HTTPClient.h>
12 String apiKey = " "; // Enter your Write API ...
    key from ThingSpeak
13 const char *ssid = " "; // ...
    replace with your home wifi/ mobile hotspot
14 const char *pass = " "; // ...
    password for the above
15 const char* server = "http://api.thingspeak.com/update";
16 WiFiClient client;
17 void setup()
18 {
19     Serial.begin(115200);
20     delay(100);
21     Serial.println("Connecting to ");
22     Serial.println(ssid);
23     WiFi.begin(ssid, pass);
24     while (WiFi.status() != WL_CONNECTED)
25     {
26         delay(400);
27         Serial.print(".");
28     }
29     Serial.println("");
30     Serial.println("WiFi connected");
```

```
31 }
32 }
33 void loop()
34 {
35
36 float t =0;
37
38 t = ((temperature_sens_read()-32)/1.8);    //changing ...
      temperature parameter to celsius
39
40 if (WiFi.status() == WL_CONNECTED)
41 {
42     HTTPClient http;
43     http.begin(server);
44     String DataSent = "api_key=" + apiKey + ...
      "&field1=" + String(t);
45     int Response = http.POST(DataSent);
46     Serial.print("Room Temperature:");
47     Serial.print(t);
48     Serial.print(" degree Celsius\n");
49     Serial.println("% Sending to Thingspeak %");
50     http.end();
51 }
52     client.stop();
53     Serial.println("Waiting");
54     delay(12500);
55 }
```

In the above code, you should enter your home wifi/mobile hotspot ssid and password to connect the ESP-32 with Wifi. And you should also enter thingspeak "write API" as mentioned in the comments.

**Observations & Discussions:** In this experiment, we have connected ESP-32 to the cloud. We can easily transfer data to Cloud applications from ESP-32 and can visualise them easily. We can see, in the above figure, the plot of values. The internal temperature sensor always measures an approximate value (won't be as accurate as a original temperature sensor) and the value is 53.3°C. It fluctuates between 52-55 but most of the time it is 53.33°C . The flow is simply to connect ESP-32 with home wifi/ personal hotspot and if it is connected, send the values to the Cloud. This is one of the important features of ESP-32



Figure(10.1): Thingspeak graphical visualisation

I have recorded a video on hardware demonstration which can be found [here](#)

**Conclusion:** In this experiment, we have studied how to transfer ESP-32 data values to the thingspeak cloud and have learnt how to visualise the data.

## Experiment-11: Control ESP-32 with Telegram Bot

**Aim:** Control the ESP-32 micro controller using the Telegram bot

**Requirements:** ESP-32 microcontroller, Telegram

**Description:** Telegram is used for a wide range of applications. Various telegram channels are used for various purposes. Here, we use the same to create a bot and control ESP-32 micro controller using it. Go to telegram and search "BotFather" Channel. After joining, send the command "/start" to it. After this, it replies the instructions. Follow the necessary instructions given by it to create a bot.

**Code simulation and Comments:** -

```

1 //for temperature measurement
2 #ifdef __cplusplus
3 extern "C" {
4 #endif
5 uint8_t temperature_sens_read();
6 #ifdef __cplusplus
7 }
8 #endif
9 uint8_t temprature_sens_read();
10
11 #include <WiFi.h>
12 #include <WiFiClientSecure.h>
13 #include <UniversalTelegramBot.h> //our telegram bot library
14
15 // Wifi network/ Mobile hotspot credentials
16 #define WIFI_SSID " " //Wifi Username
17 #define WIFI_PASSWORD " " Wifi password
18 #define BOT_TOKEN " " //your telegram bot token
19
20 const unsigned long BOT_MTBS = 1000; // mean time between scan ...
    messages
21
22 WiFiClientSecure secured_client;
23 UniversalTelegramBot bot(BOT_TOKEN, secured_client);
24 unsigned long bot_lasttime; // last time messages' scan has ...
    been done
25
26 const int ledPin = 2;
27 int ledStatus = 0;
28
29 void handleNewMessages(int numNewMessages)

```

```
30 {
31     float t =0;
32     char temp[10];
33     Serial.print("handleNewMessages ");
34     Serial.println(numNewMessages);
35
36     for (int i = 0; i < numNewMessages; i++)
37     {
38         String chat_id = bot.messages[i].chat_id;
39         String text = bot.messages[i].text;
40
41         String from_name = bot.messages[i].from_name;
42         if (from_name == "")
43             from_name = "Guest";
44
45         if (text == "/ledon")
46         {
47             digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is ...
48                 the voltage level)
49             ledStatus = 1;
50             bot.sendMessage(chat_id, "Led is ON", "");
51         }
52
53         if (text == "/ledoff")
54         {
55             ledStatus = 0;
56             digitalWrite(ledPin, LOW); // turn the LED off (LOW is ...
57                 the voltage level)
58             bot.sendMessage(chat_id, "Led is OFF", "");
59         }
60
61         if (text == "/status")
62         {
63             if (ledStatus)
64             {
65                 bot.sendMessage(chat_id, "Led is ON", "");
66             }
67             else
68             {
69                 bot.sendMessage(chat_id, "Led is OFF", "");
70             }
71         }
72
73         if(text=="/temp"){
74             t = ((temperature_sens_read()-32)/1.8); //changing ...
75                 temperature parameter to celsius
76             bot.sendMessage(chat_id, "Temperature is:");
77             gcvt(t,4,temp);
78             bot.sendMessage(chat_id,temp);
79             bot.sendMessage(chat_id,"Celsius");
80         }
```



```
76     }
77
78     if (text == "/start")
79     {
80         String welcome = "Welcome to Universal Arduino Telegram ...
81             Bot library, " + from_name + ".\n";
82         welcome += "This is Flash Led Bot example.\n\n";
83         welcome += "/ledon : to switch the Led ON\n";
84         welcome += "/ledoff : to switch the Led OFF\n";
85         welcome += "/status : Returns current status of LED\n";
86         bot.sendMessage(chat_id, welcome, "Markdown");
87     }
88 }
89
90
91 void setup()
92 {
93     Serial.begin(115200);
94     Serial.println();
95
96     pinMode(ledPin, OUTPUT); // initialize digital ledPin as an ...
97         output.
98     delay(10);
99     digitalWrite(ledPin, HIGH); // initialize pin as off (active LOW)
100
101     // attempt to connect to Wifi network:
102     Serial.print("Connecting to Wifi SSID ");
103     Serial.print(WIFI_SSID);
104     WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
105     secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add ...
106         root certificate for api.telegram.org
107     while (WiFi.status() != WL_CONNECTED)
108     {
109         Serial.print(".");
110         delay(500);
111     }
112     Serial.print("\nWiFi connected. IP address: ");
113     Serial.println(WiFi.localIP());
114
115     Serial.print("Retrieving time: ");
116     configTime(0, 0, "pool.ntp.org"); // get UTC time via NTP
117     time_t now = time(nullptr);
118     while (now < 24 * 3600)
119     {
120         Serial.print(".");
121         delay(100);
122         now = time(nullptr);
123     }
```

```
122     Serial.println(now);
123 }
124
125 void loop()
126 {
127     if (millis() - bot_lasttime > BOT_MTBS)
128     {
129         int numNewMessages = ...
            bot.getUpdates(bot.last_message_received + 1);
130
131         while (numNewMessages)
132         {
133             Serial.println("got response");
134             handleNewMessages(numNewMessages);
135             numNewMessages = bot.getUpdates(bot.last_message_received ...
                + 1);
136         }
137
138         bot_lasttime = millis();
139     }
140 }
```

Before verifying the above code, make sure you install all the libraries used in the code especially the "Universal telegram Bot" library.

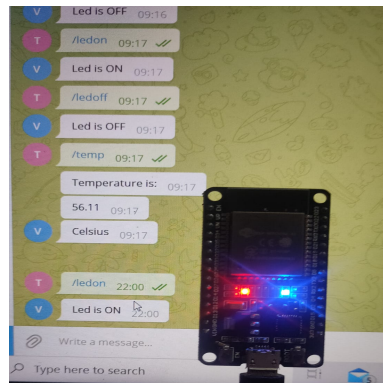
Once you have setup your telegram bot, insert your home wifi/mobile hotspot ssid and password in the code to connect ESP-32 with Wifi. We should also insert the API key provided by the bot in the code. This establishes connection between the bot and the micro controller.

After doing, the variable "text" in the code is used to read the messages in telegram. I.e, if you send a command to the bot, the ESP-32 stores it in the "text" variable and performs operation. If you want to change command name, simply change the if-else statements (which uses text variables) in the code.

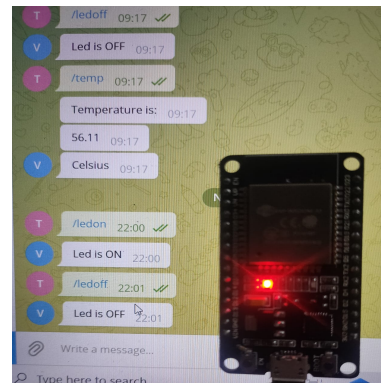
After successful connection, start sending the commands. The output will be as follows

**Observations & Discussions:** Below are the commands given to the ESP-32 using the bot and the respective outputs. We can control the LED on ESP-32 using the telegram bot and we also used it to measure temperature of the room.

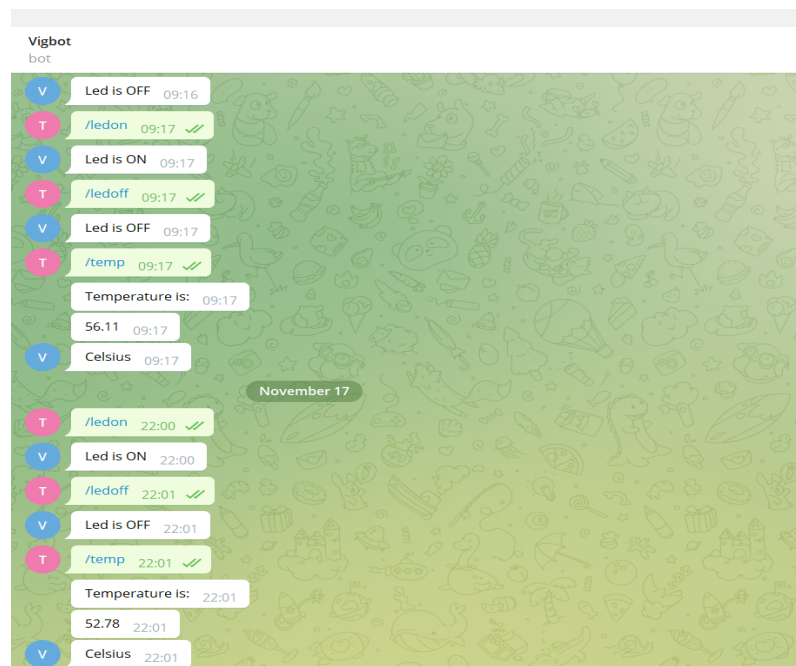
I have recorded a video on hardware demonstration which can be found [here](#)



When LED ON command is given



When LED OFF command is given



Figure(11.1): LED status and Temperature of the room

**Conclusion:** In this experiment, we have studied how to create a telegram bot and connect it to ESP-32. We controlled the LED and measured the temperature of the room using that bot.

## Experiment-12: Control ESP-32 with Google Assistant

**Aim:** Control the ESP-32 micro controller using Google Assistant

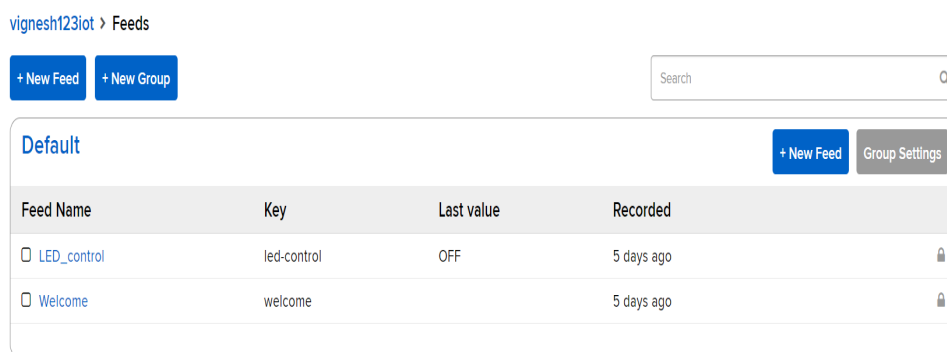
**Requirements:** ESP-32 microcontroller, an account in Adafruit and IFTTT

**Description:** This is one of the most interesting applications of ESP-32. The basic principle is, connect your mail with Adafruit and IFTTT, Give your own command to put on the LED, burn the code into the Micro controller and start giving commands from Google Assistant. The clear explanation is given below.

Create an account in Adafruit and start creating a Feed (You need to mention this Feed name in the code too). After creating a feed, create an account in IFTTT and start creating two applets. Connect these applets with the Feed in Adafruit. After doing this, For first applet, give a phrase (this is the instruction which we will give to the google Assistant). In our case, the phrase is "Turn on LED" which puts on LED and same process goes for second applet but the phrase is "Turn off LED".

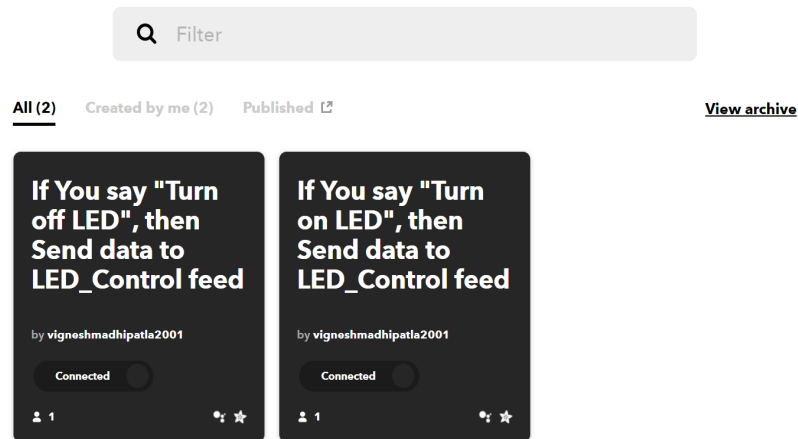
After doing all these things, mention the home wifi/mobile hotspot username and password to connect it with Wifi/hotspot and also, we need to mention required Adafruit IO key in the code along with Adafruit username.

After doing all the necessary changes, make sure all the libraries are installed as Adafruit is not a default library



Feed Name	Key	Last value	Recorded
<input type="checkbox"/> LED_control	led-control	OFF	5 days ago
<input type="checkbox"/> Welcome	welcome		5 days ago

Figure(12.1): Adafruit Feeds



Figure(10.2): IFTTT Applets

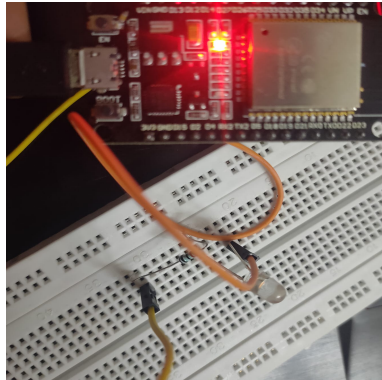
Code simulation and Comments: -

```

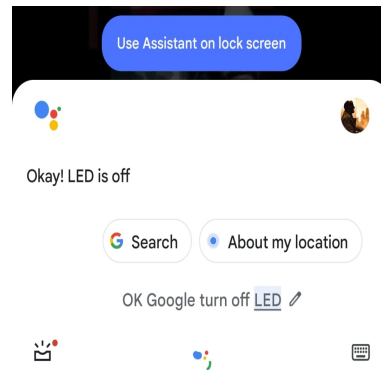
1  #include <WiFi.h>
2  #include <WiFiClient.h>
3  #include "Adafruit_MQTT.h"
4  #include "Adafruit_MQTT_Client.h"
5
6  #define WLAN_SSID      " " // Wifi name
7  #define WLAN_PASS      " " // Wifi password
8  #define AIO_SERVER     "io.adafruit.com"
9  #define AIO_SERVERPORT 1883
10 #define AIO_USERNAME   "vignesh123iot" // Replace with your ...
    Adafruit username
11 #define AIO_KEY        " " // Give your Adafruit IO key
12
13 WiFiClient client;
14
15 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, ...
    AIO_USERNAME, AIO_KEY);
16 Adafruit_MQTT_Subscribe LED_control = ...
    Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME ...
    "/feeds/LED_control");
17 // here feeds/LED_control is given which is my feed name
18
19 void MQTT_connect() {
20     int ret;
21     if (mqtt.connected()) {
22         return;

```

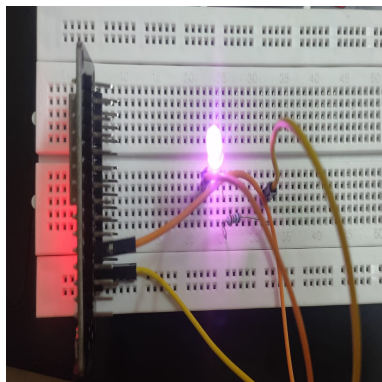
```
23     }
24     Serial.print("Connecting to MQTT... ");
25     while ((ret = mqtt.connect()) != 0) {
26         Serial.println(mqtt.connectErrorString(ret));
27         Serial.println("Retrying MQTT connection in 5 seconds...");
28         delay(5000);
29     }
30     Serial.println("MQTT Connected!");
31 }
32
33 void setup() {
34     // put your setup code here, to run once:
35     Serial.begin(115200);
36     delay(100);
37     pinMode(4, OUTPUT);
38     WiFi.mode(WIFI_STA);
39     WiFi.begin(WLAN_SSID, WLAN_PASS);
40     WiFi.setSleep(false);
41
42     Serial.println("Connecting");
43     while(WiFi.status() != WL_CONNECTED) {
44         delay(500);
45         Serial.print(".");
46     }
47
48     Serial.println("");
49     Serial.println("Connected");
50
51     mqtt.subscribe(&LED_control);
52 }
53
54 void loop() {
55     // put your main code here, to run repeatedly:
56     MQTT_connect();
57     Adafruit_MQTT_Subscribe *subscription;
58     while((subscription = mqtt.readSubscription(5000))) {
59         if(subscription == &LED_control) {
60             Serial.print(F("Got: "));
61             Serial.println((char *)LED_control.lastread);
62             if (!strcmp((char*) LED_control.lastread, "ON")) {
63                 digitalWrite(4, HIGH);
64
65             } else {
66                 digitalWrite(4, LOW);
67             }
68         }
69     }
70 }
```



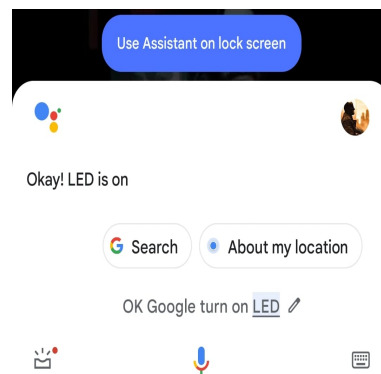
LED OFF



When LED OFF command is given



LED ON



When LED ON command is given

**Observations Discussions:** I have recorded a video on hardware demonstration which can be found [here](#)

In the above application, we learnt how to create our own commands for google assistant, pass them and get desired output. We used this application to control an External LED. we can even extend the same to Home appliances

**Conclusion:** Therefore, we have studied how to connect and control use ESP-32 with google Assitant using Adafruit.io and IFTTT

## Experiment-13: FreeRTOS in ESP-32

**Aim:** To create a multi-threaded programme on ESP-32 micro controller using freeRTOS

**Requirements:** ESP-32 microcontroller

**Description:** In general, micro controllers have one or more than one cpu cores. We can use these multiple cores for multiple purposes at same time. RTOS is generally a part of a big programme that determines the next task, assigns priority to the tasks, manages the task messages, & coordinates all the tasks. ESP-32 is a dual core microcontroller i.e, we can perform any two operations at the same time. This has a lot of advantages. We can use this amazing feature with the help of freeRTOS

In this experiment, we will design two tasks, one is to increase a counter and other is to blink LED. Both at the same time

**Code simulation and Comments:** -

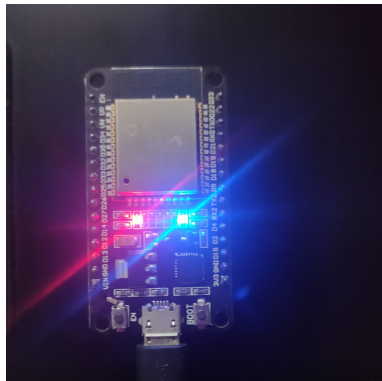
```

1  int counter=0;
2  void task1(void* parameters){ //task to increment counter
3      for(;;){
4          Serial.print("Task 1:");
5          Serial.println(counter++);
6          vTaskDelay(1000/portTICK_PERIOD_MS); //1sec time delay
7      }
8  }
9  // The above programme has a delay of 1sec. After highest prior ...
   programme is executed, then At this moment, the next ...
   less/equally programmes
10 void task2(void* parameters){ //task to blink LED
11     for(;;){
12         Serial.println("Task 2 LED");
13         pinMode(2,OUTPUT);
14         digitalWrite(2,HIGH);
15         vTaskDelay(500/portTICK_PERIOD_MS); // 0.5 sec time delay
16         digitalWrite(2,LOW);
17         vTaskDelay(500/portTICK_PERIOD_MS);
18     }
19 }
20 void setup() {
21     Serial.begin(9600);
22     xTaskCreate(
23         task1, // function
24         "Task 1", // Descriptiom

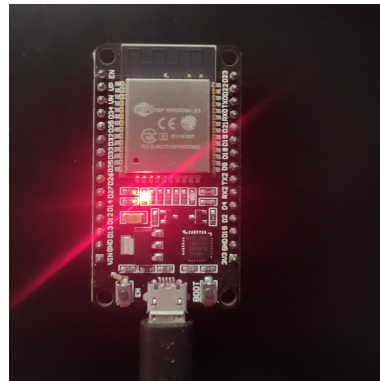
```



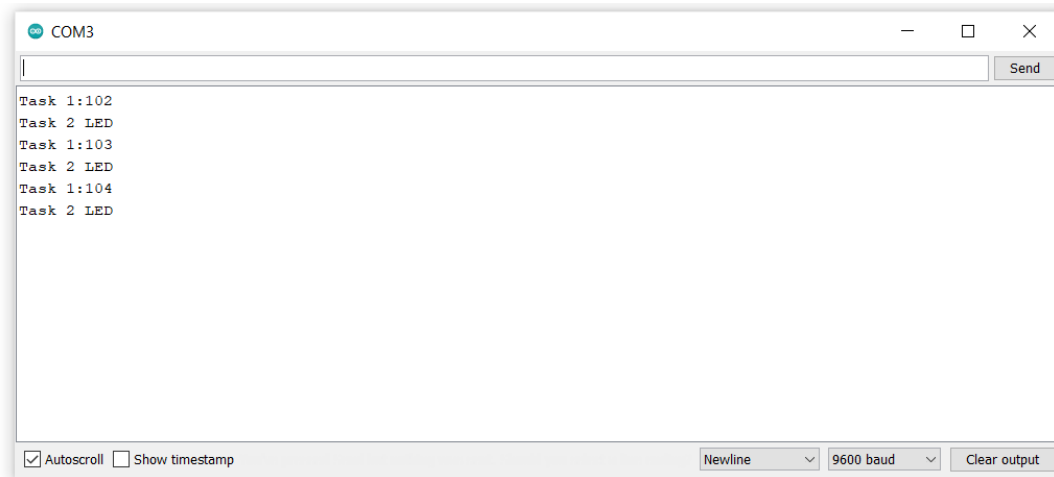
```
25     1000, // Stack size
26     NULL, // task parameters
27     1, //Priority level
28     NULL // handle
29 );
30 xTaskCreate(
31     task2,
32     "Task 2",
33     1000,
34     NULL,
35     1,
36     NULL
37 );
38 }
39 void loop() {
40     // put your main code here, to run repeatedly:
41
42 }
```



LED ON for 0.5 seconds



LED OFF for 0.5 seconds



Figure(13.3): Both tasks running in parallel (both with 1 second delay)

**Observations Discussions:** I have recorded a video on detailed hardware demonstration which can be found [here](#)

We can see, the serial monitor gets updated for every 1 second which was the time delay provided by us. When LED goes ON and OFF for 0.5 second each, it makes a total of 1 second which says the LED task is done. The counter gets updated for every 1 second. The clear explanation is given in the above video

FreeRTOS is widely used in many micro controller applications. We don't need to use any extra micro controller to perform the other tasks. Also, we can even setup priority. More the priority, more preference and time is given to the task and less for task with less priority.

**Conclusion:** Therefore, in this experiment, we have learnt how to perform multi-tasking on ESP-32 micro controller using freeRTOS.

## Experiment-14: Task Scheduling in ESP-32

**Aim:** To learn about task scheduling mechanism in ESP-32 microcontroller

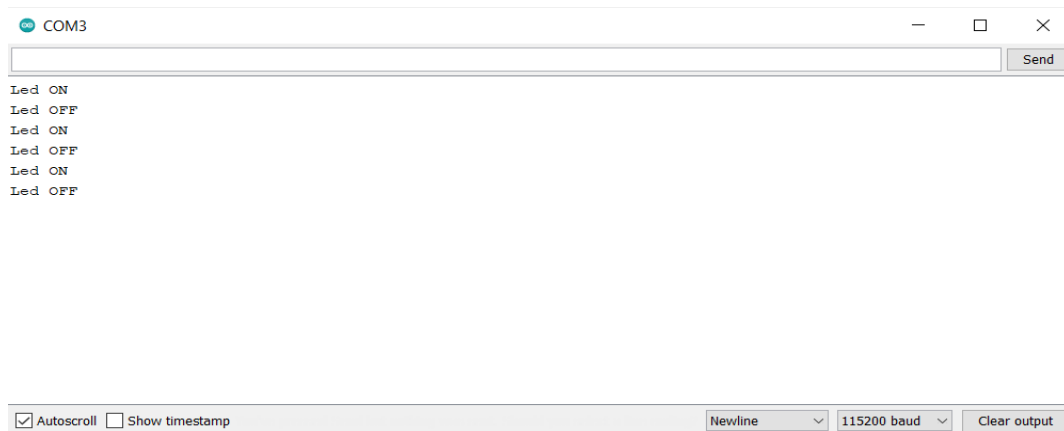
**Requirements:** ESP-32 microcontroller

**Description:** In this experiment, we will perform task scheduling in ESP-32 microcontroller. We control internal LED of ESP-32 microcontroller using task scheduler. In general we use delay to blink the LED. But here, we will follow a complete different procedure. We will build a scheduler at first which basically manages the task. And after this, we will create our required task which will be executed for every second. After that, our task is added to our Scheduler and from there, it performs necessary operations.

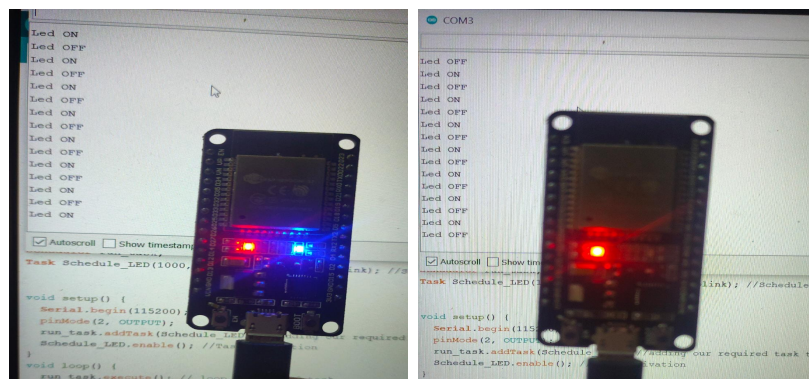
**Code simulation and Comments:** -

```
1  #include <TaskScheduler.h> // Task scheduler library
2  int LED = 1;
3  void led_blink() { //function to put on and put off LED
4      LED = !LED;
5      if(LED){
6          digitalWrite(2, HIGH);
7          Serial.println("Led ON");
8      }
9      if(!LED){
10         digitalWrite(2, LOW);
11         Serial.println("Led OFF");
12     }
13 }
14
15 Scheduler run_task;
16 Task Schedule_LED(1000, TASK_FOREVER, &led_blink); //Schedule ...
    task for every second
17
18 void setup() {
19     Serial.begin(115200);
20     pinMode(2, OUTPUT);
21     run_task.addTask(Schedule_LED); //adding our required task to ...
        the scheduler
22     Schedule_LED.enable(); //Task activation
23 }
24 void loop() {
25     run_task.execute(); // loop running of task
26 }
```

**Output:** I have recorded a video on detailed hardware demonstration and output obtained which can be found [here](#)



Figure(14.1): Output of serial monitor (updates for every 1 second)



Figure(14.2): LED blink with task scheduler

**Observations and Discussions:** We have used a Task scheduler to perform this operation. The procedure is discussed above. Now, if we see the output, we have scheduled the task for every 1 second, So, as a second passes, the LED blinks and after a second, it stops.

This feature has a lot of advantages. Because, Generally, if we give delay for certain time, the controller will still be in high power mode. But here, if there are no tasks to be performed, this method pushes the microcontroller into a less power consuming mode. So, if LED is OFF, it consumes less power and consumes more if it's ON.

**Conclusion:** Therefore, we have studied about Task scheduling in ESP-32 micro-controller and performed LED blink operation using it

## Experiment-15: Connect ESP-32 to Firebase Database

**Aim:** To connect ESP-32 with Firebase Database and pass data to it (additional task)

**Requirements:** ESP-32 microcontroller

**Description:** Firebase database is a Database management system developed by google. In this project, we will create a real time database using Firebase and we will connect our microcontroller with this database. After connecting, we will pass our data from the ESP-32 microcontroller to the Real time database.

**Code simulation and Comments:** -

```

1  #include <WiFi.h> // WiFi library
2  #include <Firebase_ESP_Client.h> // Firebase ESP library
3  #include "addons/TokenHelper.h" //Token helper lib
4  #include "addons/RTDBHelper.h"
5
6  FirebaseAuth auth;
7  unsigned long sendDataPrevMillis = 0;
8  int sign_up = 0;
9  FirebaseConfig config;
10
11 void setup(){
12     Serial.begin(115200);
13     WiFi.begin("Wifi_name", "Wifi_password"); //Wifi name and ...
        password
14     Serial.print("Connecting to WiFi:");
15     while (WiFi.status() != WL_CONNECTED){
16         Serial.print(".");
17         delay(300);
18     }
19     Serial.print("\nConnected to IP: ");
20     Serial.println(WiFi.localIP());
21
22     config.database_url = "Enter URL"; //Database URL. You'll get ...
        this when you create your project in Firebase
23     config.api_key = "Enter API"; //Database API key. You'll get ...
        this in your real time database section
24     int flag = Firebase.signUp(&config, &auth, "", ""); //Signing up
25     if (flag){
26         sign_up = 1;
27         Serial.print("Signup done\n");
28     }

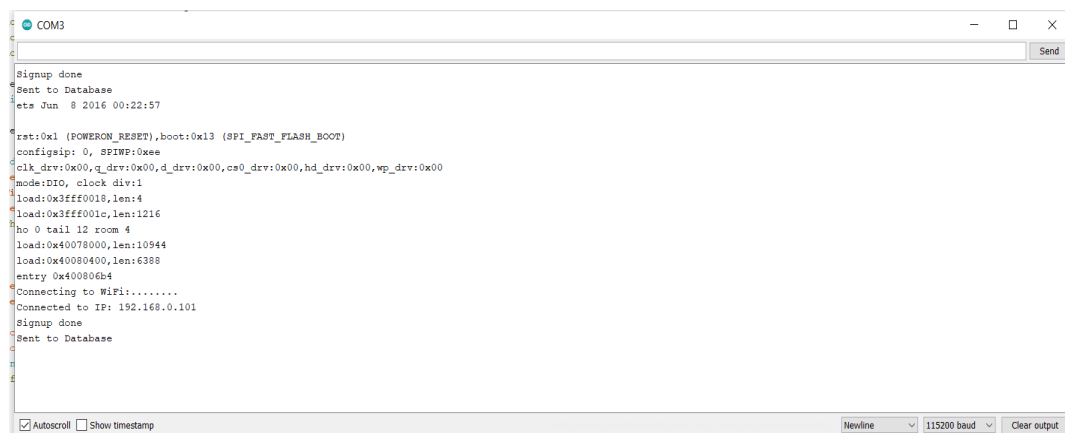
```

```

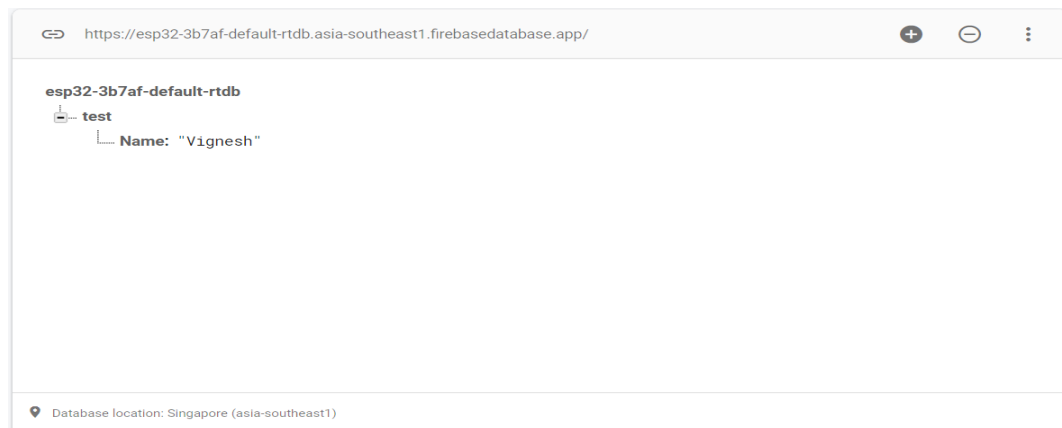
29 config.token_status_callback = tokenStatusCallback;
30
31 Firebase.begin(&config, &auth);
32 Firebase.reconnectWiFi(true);
33 }
34
35 FirebaseData fb_obj;
36 void loop(){
37     if(sign_up){
38         if(Firebase.ready()){
39             if((millis() - sendDataPrevMillis > 15000 || ...
                sendDataPrevMillis == 0)){
40                 sendDataPrevMillis = millis();
41                 if (Firebase.RTDB.setString(&fb_obj, "test/Name", ...
                    "Vignesh")){ //passing our string to the Database
42                     Serial.print("Sent to Database\n");
43                 }
44                 else {
45                     Serial.println("Failed to send to Database");
46                 }
47             }
48         }
49     }
50 }

```

### Output:



Figure(15.1): Output of serial monitor



Figure(15.2): Output of Real time Database

**Observations:** Just as how we did in telegram bot, Thingspeak, this method is also same almost. Take the required API, the project URL link and connect the micro controller with the Cloud. Then, we need to perform required operations using required libraries and required functions. We can see that our data "Vignesh" has been displayed on the Real time database which was sent using the function "setString".

**Conclusion:** Therefore, we have successfully connected the Fire base database with ESP-32 microcontroller and transferred data using it.