Visvesvaraya National Institute of Technology (VNIT), Nagpur

Embedded Systems (ECL403)

Lab Report

Submitted by :
Pranav Kathar (BT19ECE058)
Semester 5

Submitted to :
Dr. Ankit A. Bhurane and Dr. Manisha Parlewar
(Course Instructors)
Department of Electronics and Communication Engineering,
VNIT Nagpur

Contents

Experiment-1: Interface LEDs with 8051.

Aim:: To turn on and off all and alternate leds with some delay .

Concept:: LEDs are semiconductor devices used in a wide range of electrical devices for signal transmission and power indication. It's affordable and comes in a broad variety of forms, colours, and sizes. LEDs are also used in message display boards and traffic control signal lights, among other applications. It has two terminals, one positive and the other negative.

Code with explanation:

```
1  //ALP code
2  MAIN: MOV A, #0FFH
3  MOV P1, A
4  ACALL DELAY
5  MOV A, #00H
6  MOV P1, A
7  ACALL DELAY
8  MOV A, #55H
9  MOV P1, A
10 ACALL DELAY
11 MOV A, #00H
12 MOV P1, A
13 ACALL DELAY
14 SJMP MAIN
15
16 DELAY:
17 MOV R4, #100D
18 REP MOV TMOD, #01H
19 MOV TH1, #0DBH
20 MOV TL0, #0FFH
21 SETB TCON.4
22
23 H1: JNB TCON.5, H1
24 CLR TCON.4
25 CLR TCON.5
26 DJNZ R4, REP
27 RET
28 END
29
30 //C code
31 //FOR ALL LEDS ON AND OFF
32 #include<reg51.h>
33 void delay(unsigned int del)
```

```
{
  int delay gen;
  for (delay gen = 0; i < del; delay gen++);
}
void main()
{
  while (1)
  {
    leds = -leds;
    delay(1000);
  }
}
//FOR ALTERNATE LED ON AND OFF
#include<reg51.h>
#define led P2
void delay(unsigned int del);
void main(void)
{
  unsigned int delay = 0;
  led = 0x55;
  while (1)
  {
    led = -led;
    delay(1);
  }
}
void delay(unsigned int del)
{
  unsigned int loop = 0;
  unsigned int delay gen = 0;
  for (loop = 0; loop < del; loop++)
    for (delay gen = 0; delay gen < 1000; delay gen++);
}
```

Output:

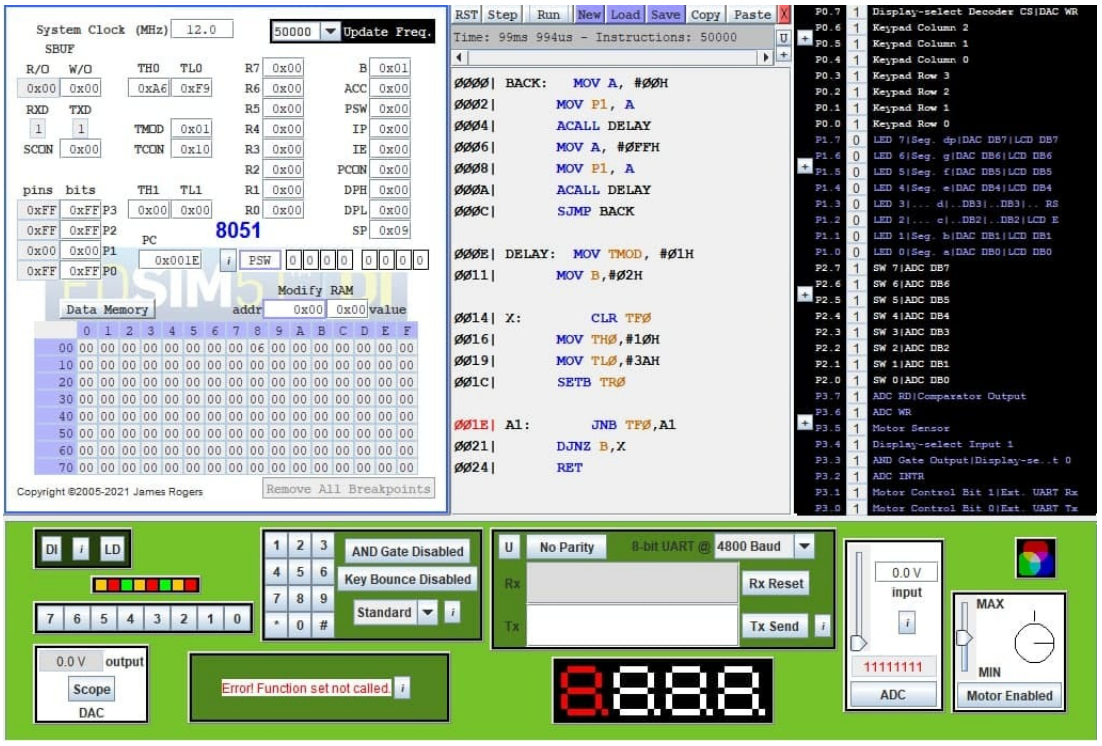

Figure 1: Part A: Blinking all LEDs using 8051 microcontroller

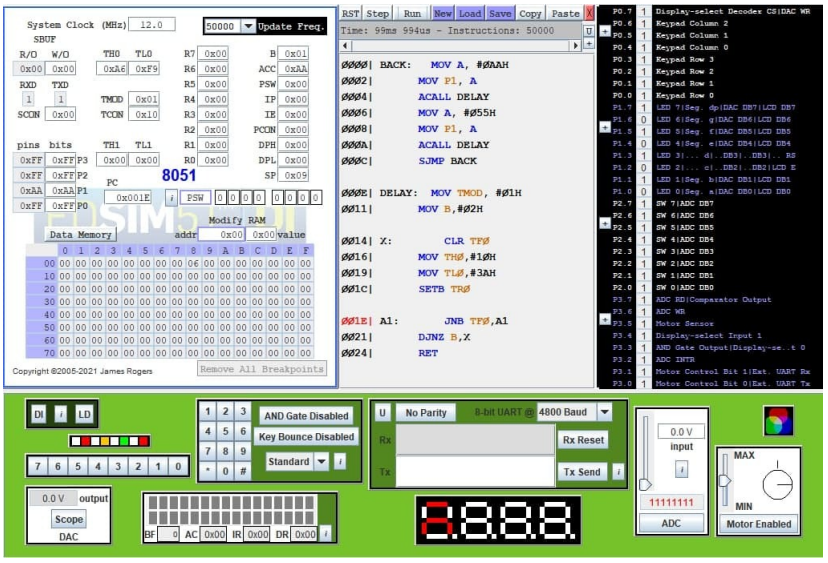

Figure 2: Part B: Blinking Alternate LEDs using 8051 microcontroller

Conclusion: LEDs were interfaced successfully with the 8051,

Experiment-2: Interface 7 segment display with 8051.

Aim:: To write the ALP and c program to display 0-9 on 7 seg display

Concept:: A seven-segment display is the most basic electrical display. When the appropriate LED combinations are switched on, it displays digits from 0 to 9. It is made up of eight LEDs that are connected in a sequence to display digits from 0 to 9. On a 7-segment display, seven LEDs are used to portray digits from 0 to 9, with the eighth LED serving as a dot. Numeric data is displayed on 7-segment screens in a variety of applications. They can only show one digit at a time on the screen.

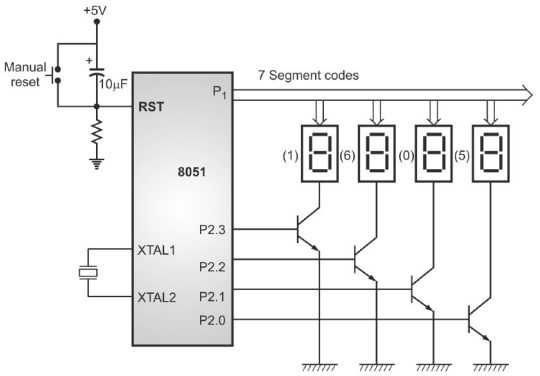

Figure 3: Interfacing with seven segment display

Code with explanation:

```
68 //ALP code
69 //interfacing 7 segment display
70 ORG 0000H
71 MAIN: MOV R1,#10H
72 MOV DPTR,#4000H
73 BACK: CLR A
74 MOVC A,@A+DPTR
75 MOV P2,A
76 ACALL DELAY
77 INC DPTR
78 DJNZ R1,BACK
79 SJMP MAIN
80 ORG 4000H
81 DB 3FH,O6H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
82 DELAY: MOV R2,#08H
83 H2: MOV R3,#0FFH
```

```
84  H1: MOV R4,#0FFH
85  REP: DJNZ R4,REP
86  DJNZ R3,H1
87  DJNZ R2,H2
88  RET
89  END
90
91  //C code
92  #include<reg51.h>
93  void main()
94  {
95    unsigned char ...
96    seg[10] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, ...
          0x80, 0x90}
97              unsigned char num;
98    unsigned int del;
99    P1 = 0x00; //output config
100   while (1)
101   {
102     for (num = 0; num < 10; num++)
103     {
104       P1 = seg[num];
105       for (del = 0; del < 1000; del++)
106       }
107   }
108 }
```

Output:

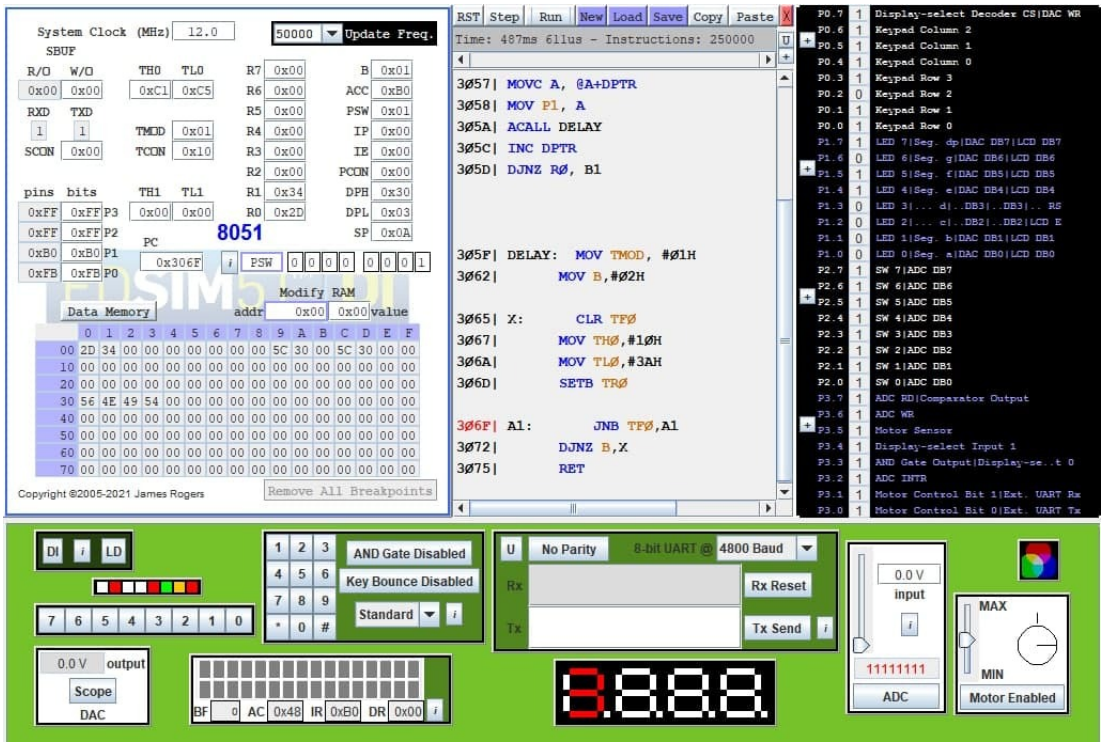

Figure 4: 7 segment display is interfaced and we displayed number 3

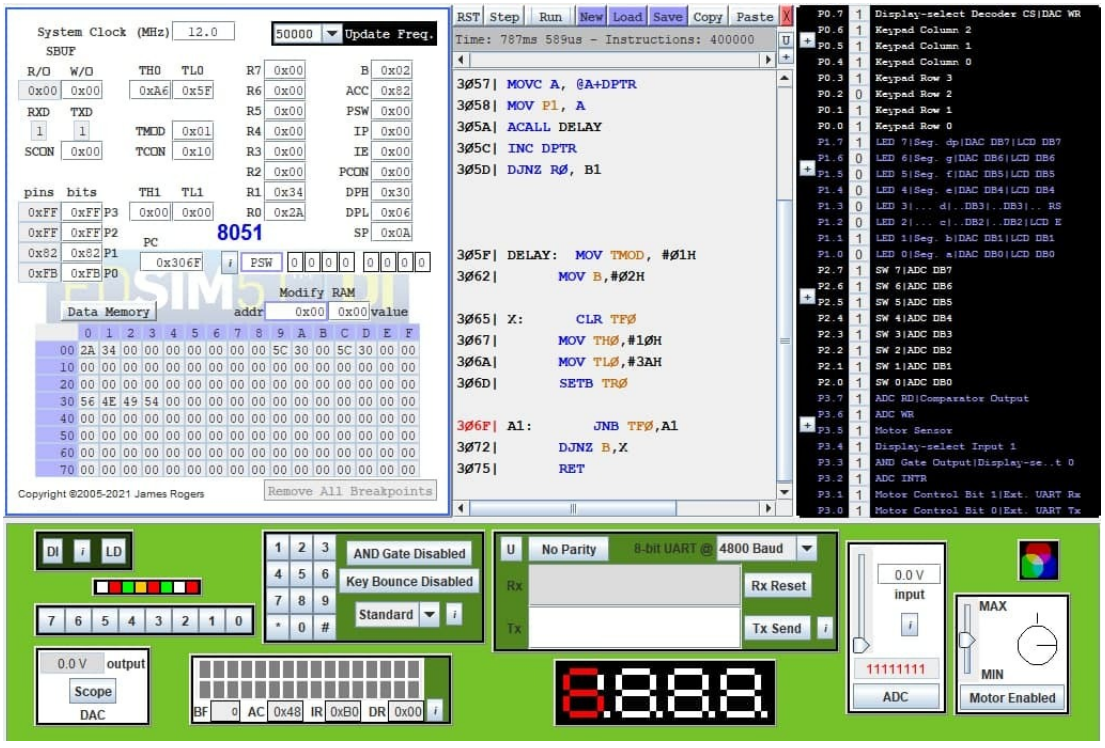

Figure 5: 7 segment display is interfaced and we displayed number 6

Conclusion: 7 segment display was interfaced successfully with the 8051,

Experiment-3: Interface keypad with 8051.

Aim:: Write an assembly and C language program to check which key is pressed in the keypad and display the corresponding number on the 7 segment display

Concept::

Working of a Key

The simplest basic input device is a keyboard. The port pin must be configured as an input pin when a Key is connected to it. We check the value of the pin to see if a key has been pressed.

For a key, most circuits employ active low logic.

A pull up resister is used to link the Keys to Vcc. The keys have a default value of one. In the event that a

When a key is pressed, it is linked to GND, resulting in a value of 0.

Concept of a matrix Keyboard

We simply need 2-3 keys for a small keypad such as a car remote key. We could simply wire them up on each port line.

However, a huge keyboard, such as a TV remote or a computer keyboard, has a vast number of keys. We can't afford to give each key its own port pin.

As a result, we use a matrix to connect the keys. This allows us to have more keys while utilising fewer lines. We can figure out which key is pressed by identifying the row and column.

4x4 Matrix Keyboard

As rows, we use four lines from a port, such as P1.0... P1.3. We use columns for lines of a different port, such as P2.0... P2.3. Columns will be input, while rows will be output. Due to the Vcc with a pull up resister, the columns will have a default value of 1. This results in 16 intersecting spots, each of which we place a key. The relevant row and column will come into contact when the key is "Pressed."

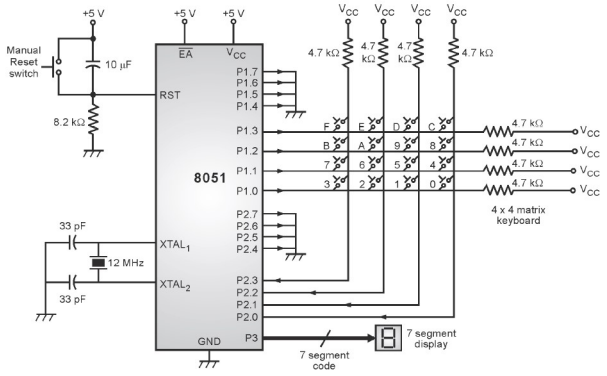

Figure 6: Interfacing with seven segment display

Code with explanation:

```
110 //ALP code for keypad interfacing
111 MOV DPTR, #0400H ; Initialising DPTR
112 MOV P2, #0FFH ;
113 Check : MOV P1, #00H ;
114 MOV A, P2 ;
115 CJNE A, #0FH, Pressed ;
116 SJMP Check ;
117 Pressed: ACALL Delay ;
118 MOV R0, #00H ;
119 MOV P1, #0EH ;
120 MOV A, P2 ;
121 CJNE A, #0FH, Colcheck ; I,
122 MOV R0, #04H ;
123 MOV P1, #0DH ;
124 MOV A, P2 ;
125 CJNE A, #0FH, Colcheck ;
126
127 MOV R0, #08H ;
128 MOV P1, #0BH ;
129 MOV A, P2 ;
130 CJNE A, #0FH, Colcheck ; If A != 0FH ,row is identified,
131 MOV R0, #0CH ;
132 MOV P1, #07H ;
133 MOV A, P2 ;
134 Colcheck : RR A ; rotate right
135 JNC Display ;
136 INC R2 ;
137 SJMP Colcheck ;
138 Display : MOV A, R2 ;
```

```
139  MOVC A, @A + DPTR ;
140  MOV P3, A ;
141  ACALL Delay ;
142  SJMP Check ;
143
144
145  //C code
```

Output:

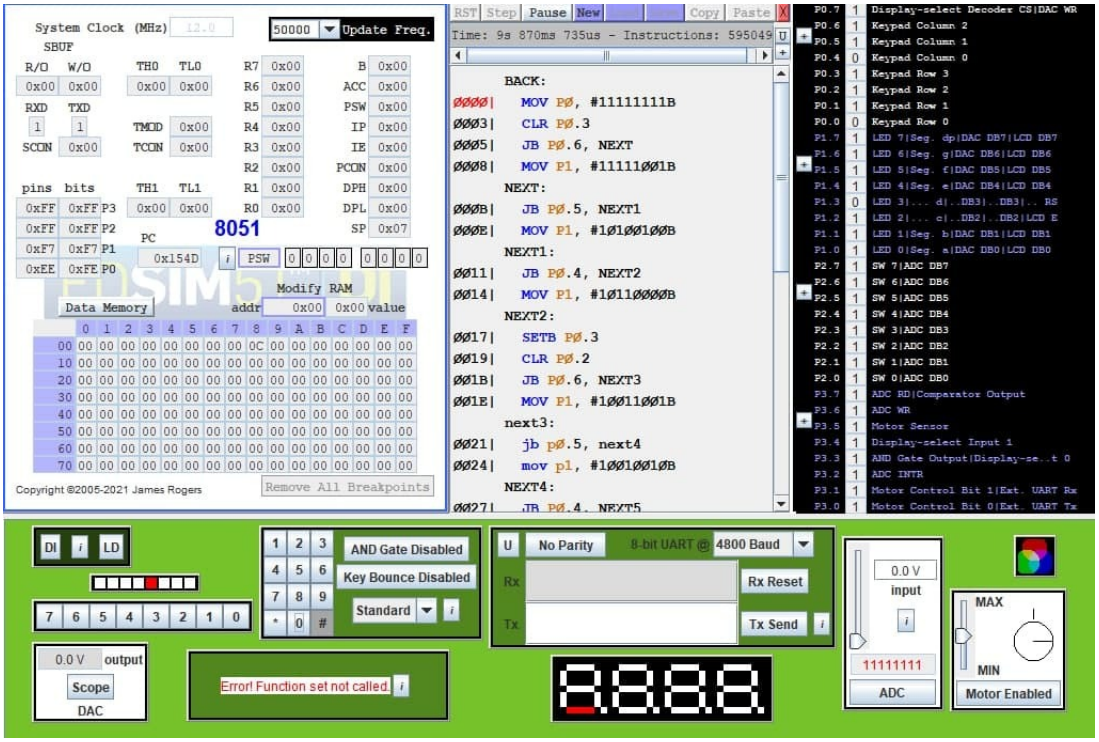

Figure 7: Output display for 'hash'

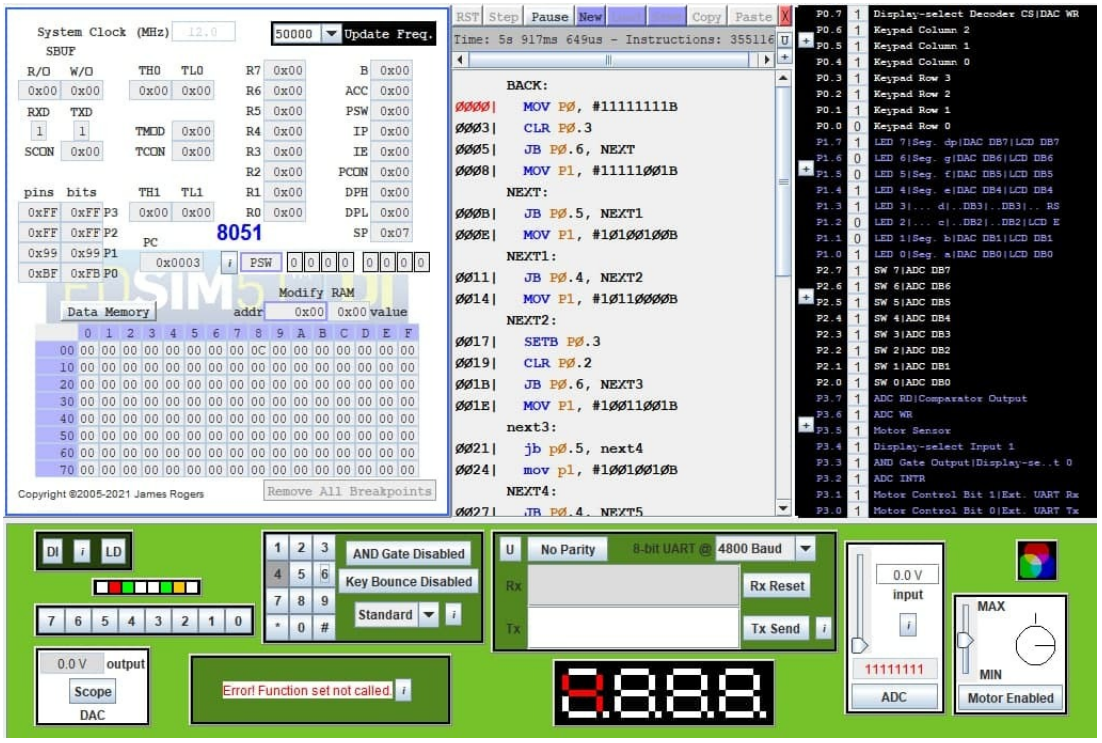

Figure 8: Output display for '4'

Conclusion: keypad was interfaced successfully with the 8051,

Experiment-4: Serial communication through 8051.

Aim:: Write an assembly and C language program to transfer the data serially

Concept:: The number of bits that can be exchanged at a time in serial communication is governed by the number of data lines available for transmission, both asynchronous and synchronous.

Code with explanation:

```
148 //ALP code for serial transmission.
149 ORG 00H
150 MOV SCON, #50H
151 MOV TMOD, #20H
152 MOV TH1, #0FDH
153 SETB TR1
154
155 DIS:
156 MOV A, #"V"
157 ACALL DEL
158 MOV A, #"N"
159 ACALL DEL
160 MOV A, #"I"
161 ACALL DEL
162 MOV A, #"T"
163 ACALL DEL
164 ACALL DELAY
165 SJMP DIS
166 DEL: MOV SBUF, A
167 SECOND: JNB TI, SECOND
168 CLR TI
169 RET
170
171 DELAY:
172 LOOPB: MOV R2, #200
173 LOOPA: MOV R3, #250
174 NOP
175 NOP
176 MOV TH0, #00H
177 DJNZ R2, LOOPB
178 DJNZ R2, LOOPB
179 RET
180 END
181
182 //C code
183 #include<reg51.h>
```

```
184 void main()
185 {
186   SCON = 0X50;
187   TMOD = 0X20;
188   TH1 = -3; //setting BAUD RATE at 9600
189   TR1 = 1;
190   SBUF = "VNIT
191          while(TI==0)
192          TI=0;
193 }
```

Output:

Figure 9: "VNIT" getting displayed serially

Conclusion: Data was successfully transmitted serially with 8051,

Experiment-5: Interfacing of stepper motor to 8051.

Aim:: Write an assembly and C language program to rotate stepper motors using 8051

Concept::

Introduction to Stepper Motor

Stepper motors are high-precision devices that rotate in tiny, precise increments while being controlled by a computer. Stepper motors are preferred over DC motors in applications where stepwise rotation is required, such as robotic arms and printers. They are simple to programme and interface with. They can also transition between clockwise and anticlockwise motions with ease. Figure 5.3.1 Four stator magnets and one rotor magnet make up a stepper motor.

Code with explanation:

```
195 //ALP code for interfacing stepper motor
196 START:
197 MOV TMOD, #50H ; sets timer1 as counter
198 SETB TR1 ;Starts the counter
199 SETB P3.0 ;sets A of bridge driver to 1
200 CLR P3.1 ; sets B of bridge driver to 0
201     ; at TL1, number of rotations can be viewed
202 JMP START ;
203
204
205 //C code for interfacing stepper motor
206 #include<reg51.h>
207 void delay()                              // delay function defintion
208 {int i,j;
209     for(i=0;i<1000;i++)
210     {       for(j=0;j<1000;j++)
211             {
212             }
213     }
214 }
215 void main()
216 {
217   while(1)
218   {
219         P2= 0x01; //Rotates Motor Clock-wise
220     delay();
```

```
221         }
222     }
```

Output:

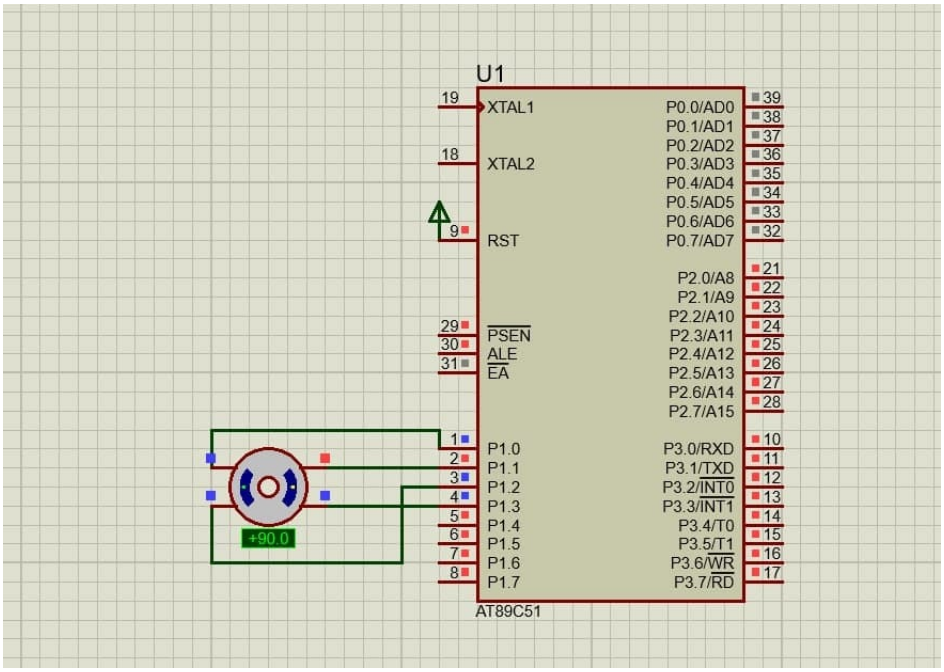

Figure 10: stepper motor interfaced with 8051 in proteus software

Conclusion: stepper motor was interfaced successfully with the 8051,

Experiment-6: 16x2 LCD interfacing with 8051.

Aim:: Write an assembly and C language program to transfer string data from 8051 to 16x2 LCD and display on it.

Concept::

Introduction to 16x2 Generic LCD Display

Because LCD screens can correctly display text, they are significantly more helpful than 7 segment LED displays. A matrix of pixels can readily show complex characters such as W, M, K, Q, and so on. A 5x7 matrix is typically used to display one character, five columns, and seven rows.

Code with explanation:

```
225 //ALP code for displaying 'VNIT' on LCD display
226 CLR P0.3
227 MOV P1,#38H
228 SETB P0.2
229 CLR P0.2
230 CALL DELAY
231 MOV P1,#0EH
232 SETB P0.2
233 CLR P0.2
234 CALL DELAY
235 MOV P1,#06H
236 SETB P0.2
237 CLR P0.2
238 CALL DELAY
239 ; VINAY
240 SETB P0.3
241 REPEAT:
242 MOV P1,#'V'
243 SETB P0.2
244 CALL DELAY
245 CLR P0.2
246
247 MOV P1,#'N'
248 SETB P0.2
249 CALL DELAY
250 CLR P0.2
251
252 MOV P1,#'I'
253 SETB P0.2
```

```
254 CALL DELAY
255 CLR P0.2
256
257 MOV P1,#'T'
258 SETB P0.2
259 CALL DELAY
260 CLR P0.2
261 jmp ENDING
262 DELAY:
263     MOV R0, #030H
264     DJNZ R0, $
265     RET
266 ENDING:
267 JMP $
268
269
270
271 //C code
272 #include<reg51.h>
273 #include<stdio.h>
274
275 sbit rs = P2^0;
276 sbit rw = P2^1;
277 sbit en = P2^2;
278
279 void lcd(unsigned char A)
280 {
281   P1 = A;
282 }
283 void delay()
284 {int i;
285   en =1;
286   for(i=0;i<10000;i++)
287   {
288   }
289   en =0;
290 }
291 void main()
292 { int j;
293   P1=0x00;
294   rw=0;
295   rs=0;
296     lcd(0x38);
297   delay();
298   lcd(0x01);
299   delay();
300   lcd(0x10);
301   delay();
302   lcd(0x0c);
```

```
303     delay();
304
305     while(j<1)
306     {
307     rs = 0x01;
308     delay();
309     lcd('V');
310     delay();
311     lcd('N');
312     delay();
313     lcd('I');
314     delay();
315     lcd('T');
316     delay();
317     j++;
318  }
319     }
```

Output:

Figure 11: 'VNIT' getting displayed on LCD Display

Conclusion: 16x2 LCD display was interfaced successfully with the 8051,

Task-0: ESP32- Overview.

Task:: ESP32- Overview

Concept::

What is ESP32?

ESP32 is a dual core integrated Micro controller unit with on-board Wi-Fi and Bluetooth facilities and can be used for wide range of applications in the field of automation because of its wide set of peripheral interfaces.

Figure 12: ESP32

Features

- Robust design - can work in industry environments
- Low power consumption - Can be dropped even further using power modes like deep sleep.
- Integrated chip - Comes with on-board Wi-Fi and Bluetooth LE facilities. It has inbuilt antennas, filters and amplifiers. It has also got CP2102 driver.
- Act like a fully self-contained system or as a slave device to a host Micro controller unit.

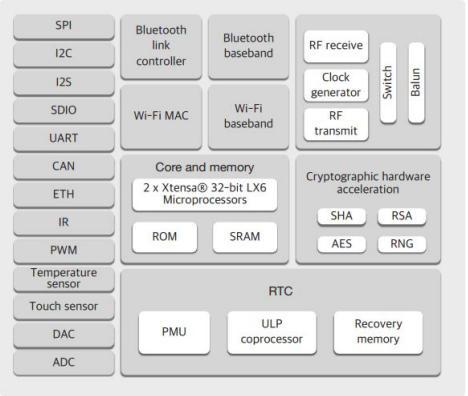

Figure 13: ESP32 features

Specifications

Specifications - ESP32 DEVKIT V1 DOIT	
Number of cores	2 (Dual core)
Wi-Fi	2.4 GHz up to 150 Mbit/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Clock frequency	Up to 240 MHz
RAM	512 KB
Pins	30
Peripherals	Capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

Figure 14: ESP32 specifications

ESP32 DEVKIT V1 - DOIT pin-out (30 GPIOs)

Programming ESP32

There are various programming environments available for programming ESP32. In this course we have programmed ESP using Arduino IDE.Before we program, we need to make sure that we have selected the correct board and COM port and installed required libraries.Also there are lot of example codes given in this IDE.

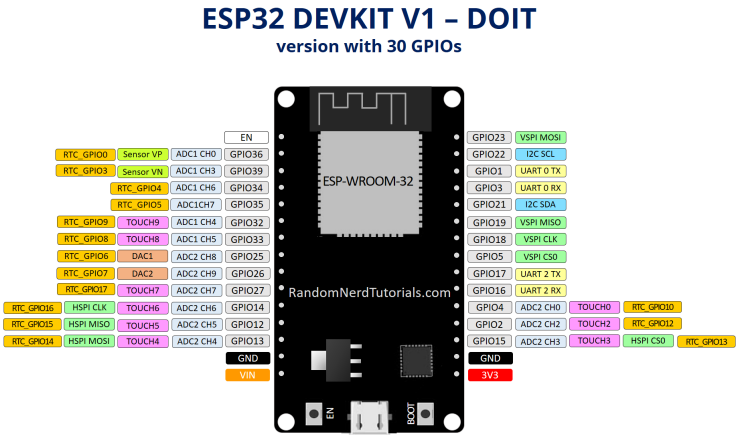

Figure 15: ESP32 pin-out (30 GPIOs)

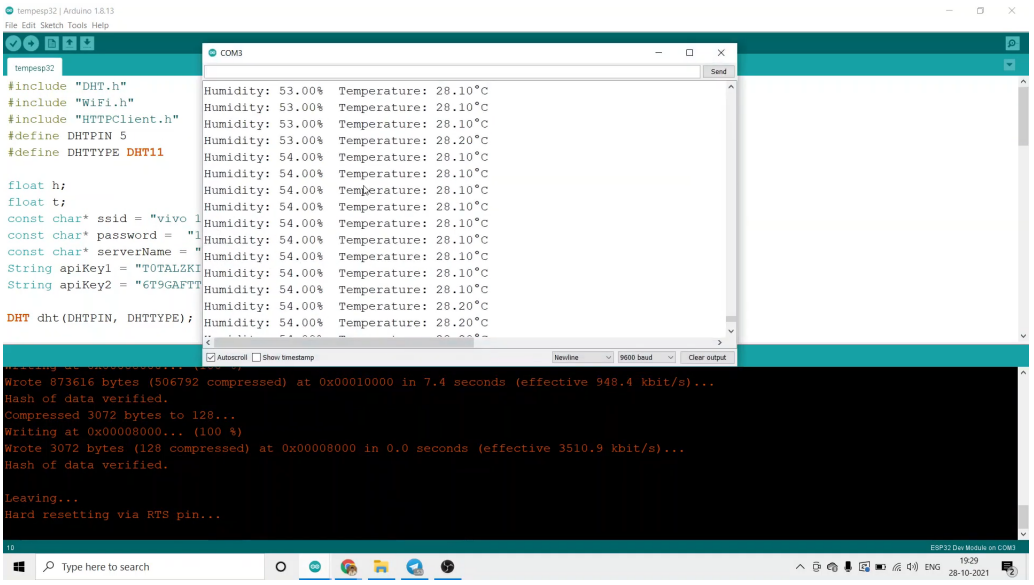

Figure 16: Arduino IDE

Problems Faced (if any):

- Many a times there will be an error like 'Failed to connect to ESP32: Timed out' - In this case while uploading the code we need to keep the boot button pressed
- COM Port not found - this can be due to two reasons
    1. USB cable which we are using is not a data cable.
    2. CP2102 drivers not installed

References/ Citations:

1. Random nerd tutorials
2. https://www.espressif.com/en/products/socs/esp32
3. www.exploreembedded.com

Youtube link of my channel:

- $https://www.youtube.com/channel/UC5b_j7Z_FgdH1KtplY12lcA$

Task-1: To explore touch functionalities of esp32.

Task:: To control on-board led of esp using touch pins available.

Concept:: There are ten capacitive touch GPIOs on the ESP32.
These GPIOs can detect changes in anything that has an electrical charge, such as the skin. As a result, they may detect fluctuations caused by a finger touching the GPIOs. These pins may easily be incorporated into capacitive pads and can be used to replace mechanical buttons.

Locate the 10 distinct touch sensors on your board by looking at the pinout.

Pins that are sensitive are highlighted in a pink tint.

Code with explanation:

```
321
322 const int touchPin = 4;
323 const int ledPin = 16;
324 const int threshold = 20;
325 int val;//for storing touch sensor reading
326
327 void setup() {
328   Serial.begin(115200);
329   delay(1000);
330   // pinmode
331   pinMode (ledPin, OUTPUT);
332 }
333 void loop() {
334   val = touchRead(touchPin);
335   Serial.print(val);
336    //here we are using touch pins as switch to turn the led on ...
         and off
337   if (val < threshold) {
338     digitalWrite(ledPin, HIGH);
339     Serial.println(" - LED on");
340   }
341   else {
342
343     digitalWrite(ledPin, LOW);
344     Serial.println(" - LED off");
345   }
346   delay(500);
347 }
```

References/ Citations:

1. ESP32 Blogs of Random nerd tutorials

Further scope: You can set a threshold value to make something happen when it detects touch.This can be useful for controlling appliances too.

Task-2: To detect temperature and humidity of a room using DHT11 Sensor and Publish the Readings to ThingSpeak.

Task:: To detect temperature and humidity of a room using DHT11 Sensor and Publish the Readings to ThingSpeak

Concept:: In this experiment, We are going to detect Temperature and Humidity of a room using DHT11 Sensor. DHT11 is a easy to use, low cost capacitive humidity sensor which measures temperature and relative humidity.
It has a chip that converts readings from analog to digital and outputs a digital signal for the same.

- Detecting humidity - Capacitive sensing element is used.
- Detecting Temperature - Thermistor

There is one advancement to to DHT11 i.e. DHT22. It's bit expensive but gives better resolution, requires lesser delay and provides wider range for measurement.

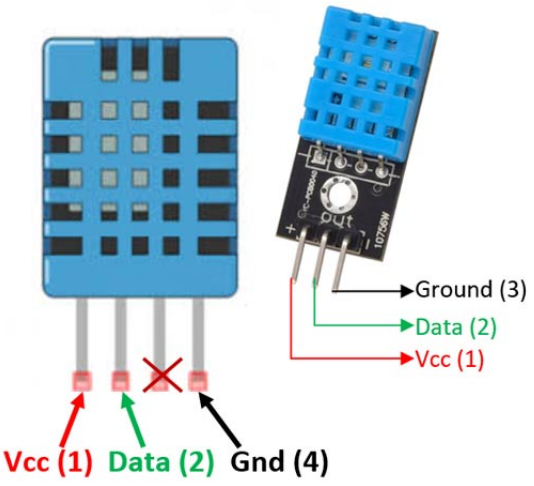

Figure 17: DHT11 Pinout configuration

Now let's make cicuit for the same

So now we will need to upload the readings to thingspeak platform which is one of the super easy to use IoT analytics platforms. We can send live data streams of our DHT11 sensor readings and publish them on thingspeak. It will also take care of the data visualizatiion part and we will able to see clear plots. We can access these readings from anywhere in the world.

DHT pin	Connect to
1	3.3V
2	Any digital GPIO; also connect a 10k Ohm pull-up resistor
3	Don't connect
4	GND

Figure 18: DHT11 Pin configuration table

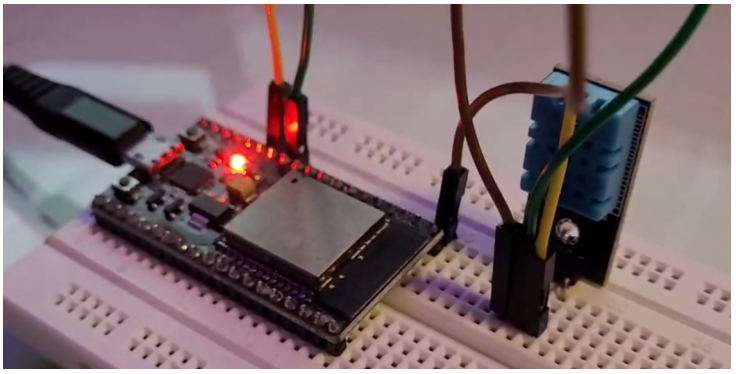

Figure 19: Circuit for this task

So we will create two seperate channels on thingspeak for Temperature and humidity readings. We can also customize the type of plot we want. Final step is to copy API keys and we will adopt them in the code. They will help us for sending readings of DHT from the ESP32 to ThingSpeak.

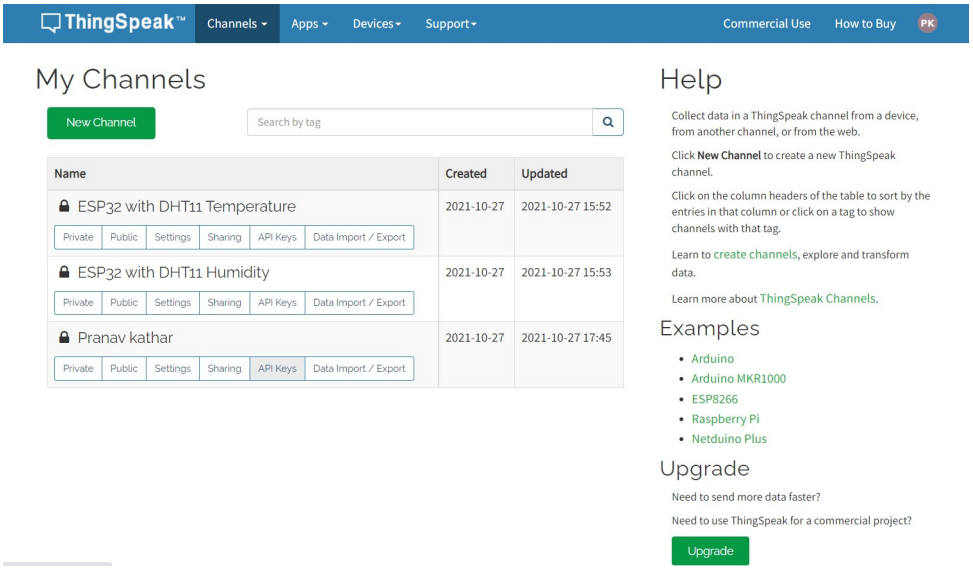

Figure 20: Thingspeak channels

Libraries required :

- DHT Sensor library from Adafruit
- Adafruit Unified Sensor
- thingspeak-arduino library.

Code with expalnation:

```
348 //included libraries reuired for dht sensor
349 #include "DHT.h"
350 #include <WiFi.h>
351 #include <HTTPClient.h>
352 #define DHTPIN 4//this gpio we will use for data
353 #define DHTTYPE DHT11
354 float h;
355 float t;
356 //wifi credentials
357 const char* ssid = "Pranav_pk";
358 const char* password =  "Aarushi!23";
359 //we will need these api keys for constantly updting live sensor ...
       readings
360 const char* serverName = "https://api.thingspeak.com/update";
361 String apiKey1 = "T0TALZKIBV46WH67";
362 String apiKey2 = "6T9GAFTTQW229AAD";
363
364 DHT dht(DHTPIN, DHTTYPE);
365
366 void setup() {
367   Serial.begin(9600);//baud rate set to 9600
368   WiFi.begin(ssid, password);//wifi initialized
369   Serial.println("Connecting");
370   while(WiFi.status() != WL_CONNECTED)//constantly checking
371   {
372     delay(500);
373     Serial.print(".");
374     }
375   Serial.println("");
376   Serial.print("Connected!");
377   Serial.println(F("DHT11 test!"));
378
379   dht.begin();
380 }
381
382 void loop() {
383   if(WiFi.status() == WL_CONNECTED)
384   {
385     HTTPClient http;
386     http.begin(serverName);
387   // Wait a few seconds between measurements.
388     delay(2000);
389
390     h = dht.readHumidity();
391     // Read temperature as Celsius (the default)
392     t = dht.readTemperature();
```

```
393
394       // Check if any reads failed and exit early (to try again).
395       if (isnan(h) || isnan(t)) {
396         Serial.println(F("Failed to read from DHT sensor!"));
397         return;
398       }
399       //publishing on thingspeak
400       String HumidityDataSent = "api_key=" + apiKey1 + "&field1=" ...
              + String(h);
401       String TempDataSent = "api_key=" + apiKey2 + "&field1=" + ...
              String(t);
402
403       http.POST(HumidityDataSent);
404       http.POST(TempDataSent);
405
406
407     }
408     //printing sensor readings on serial monitor
409     Serial.print(F("Humidity: "));
410     Serial.print(h);
411     Serial.print(F("%  Temperature: "));
412     Serial.print(t);
413     Serial.print(F[U+FFFD]"C"));
414     Serial.println(" ");
415
416   }
```

Output/Demonstration:

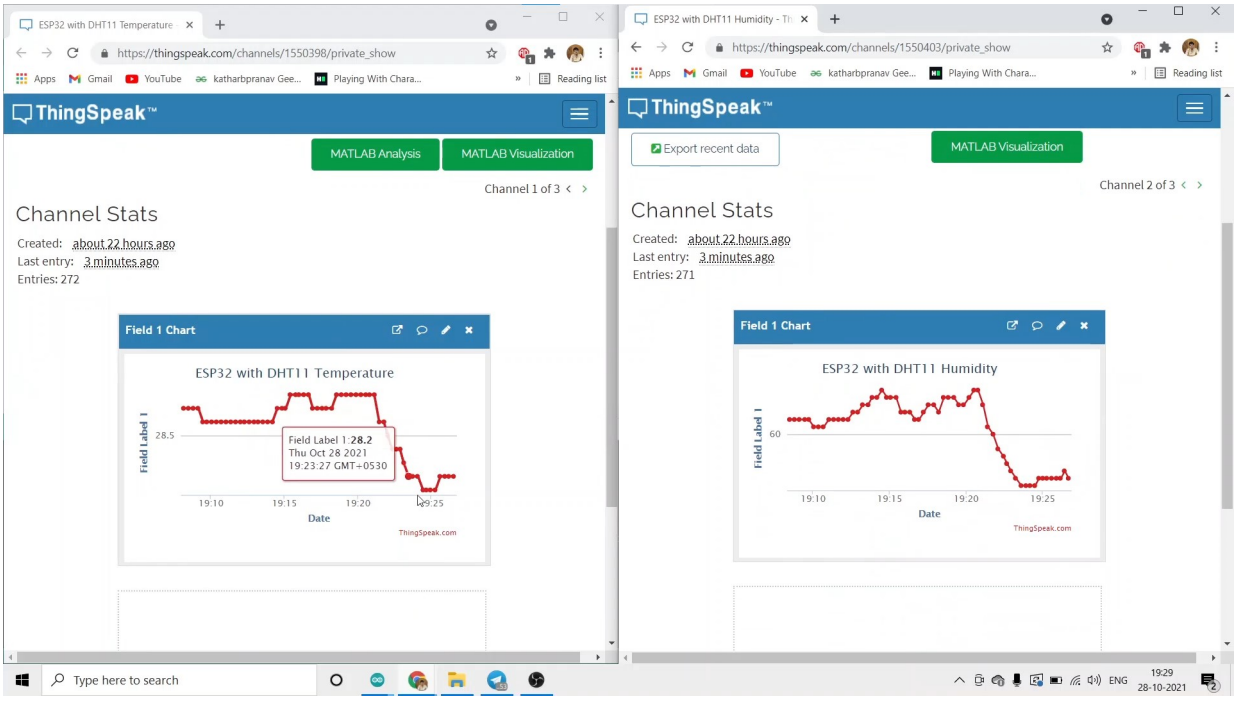

Figure 21: Data getting published on Thingspeak

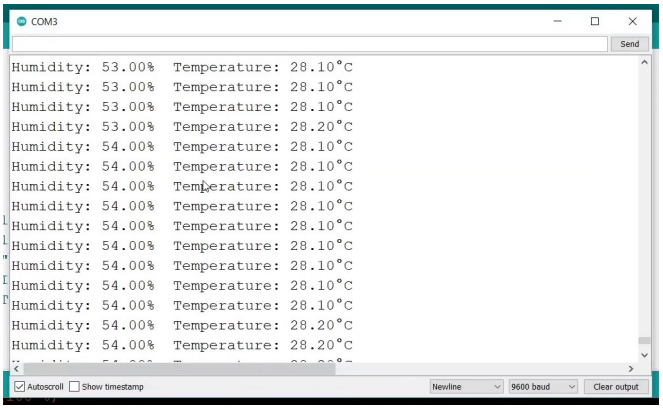

Figure 22: Output of Serial Monitor

Youtube Video Link: https://www.youtube.com/watch?v=TdiiW2BAtVA

Problems Faced (if any):

- Fatal error: Adafruit Sensor.h: No such file or directory - Actually this Problem was caused because I forgot to install the above mentioned libraries. Got resolved after installing.
- I connected DHT to GPIO4 first but wasn't getting any output. After sometime tried connecting to GPIO5 and it worked.

References/ Citations:

1. ESP32 Blogs of Random nerd tutorials https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/ //https://randomnerdtutorials.com/es thingspeak-publish-arduino/

Further scope: Here we have Thingspeak platform for data visualization. We can go one step further and create a web server which will do our job of displaying readings in our smarthphone.

Task-3: To control GPIOs of ESP32 using telegram.

Task:: To control GPIOs of ESP32 using telegram

Concept:: In this task, we are going to control on board LED of ESP32 with the help of a telegram bot. Telegram chat bots are third party applications which are easy to create and handle. Also we can control them from anywhere in the world. So that's an additional advantage. So the GPIO pin we have used here is GPIO2. Commands which we will give here are

- /start
- /led on
- /led off
- /state

We will create bot for our ESP32 MCU using Botfather. (t.me/botfather). At the end , we will get one bot token which we will adopt in our code for our esp to interact with newly created bot.
After creating a bot , the issue we face is , Anyone can interact with that bot just by knowing bot username. For ignoring messages which are not from our account , we need our Telegram User ID. We will adopt this ID in our code and in this way ESP32 can ignore the messages from other sender checking whether the ID is appropriate or not. so for this purpose we will need "IDBot". (t.me/myidbot)

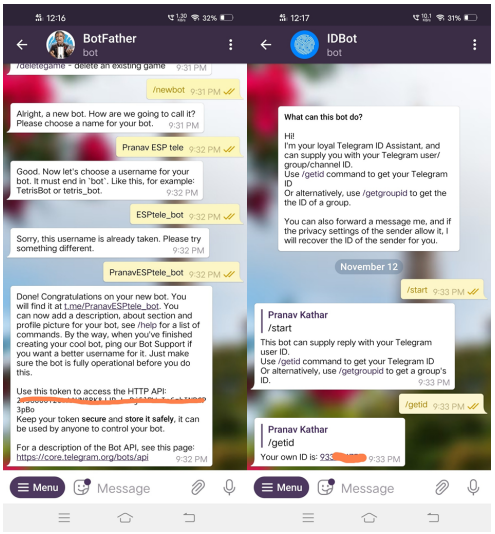

Figure 23: Botfather and IDbot.

Libraries required :

- Universal Telegram Bot Library
- ArduinoJson Library

Code with expalnation:

```
418 #include <WiFi.h>//libraries for setting up wifi
419 #include <WiFiClientSecure.h>
420 #include <UniversalTelegramBot.h>//for telegram
421 #include <ArduinoJson.h>
422 //credentials
423 const char* ssid = "vivo 1951";
424 const char* password = "12345678";
425 //we need these tokens so that we can interact with the bot
426 #define BOTtoken "2136800120:AAHN8RK8JJP-L-Pj5lBWoIo6zbINPGP3pBo"
427 #define CHAT_ID "933504755"
428
429 WiFiClientSecure client;
430 UniversalTelegramBot bot(BOTtoken, client);
431
432 int bot_delay = 1000;
433 unsigned long lastTimeBotRan;
434 const int ledPin = 2;
435 bool ledState = LOW;
436
437 //Handling new messages
438 void handleNewMessages(int numNewMessages) {
439   Serial.println("Handling New Message");
440   Serial.println(String(numNewMessages));
441
442   for (int i=0; i<numNewMessages; i++) {
443
444     String chat_id = String(bot.messages[i].chat_id);
445     //verifying the chat id
446     if (chat_id != CHAT_ID){
447       bot.sendMessage(chat_id, "Unauthorized user", "");
448       continue;
449     }
450
451     String user_text = bot.messages[i].text;
452     Serial.println(user_text);
453
454     String your_name = bot.messages[i].from_name;
455     //start command
456     if (user_text == "/start") {
```

```
457       String welcome = "Welcome, " + your_name + ".\n";
458       welcome += "Use the following commands to control the ...
              ESP32.\n\n";
459       welcome += "Send /led_on to turn GPIO2 ON \n";
460       welcome += "Send /led_off to turn GPIO2 OFF \n";
461       welcome += "Send /get_status to request current GPIO2 ...
              stateus \n";
462       bot.sendMessage(chat_id, welcome, "");
463     }
464     //ledon command
465     if (user_text == "/led_on") {
466       bot.sendMessage(chat_id, "LED is now ON", "");
467       ledState = HIGH;
468       digitalWrite(ledPin, ledState);
469     }
470     //led off command
471     if (user_text == "/led_off") {
472       bot.sendMessage(chat_id, "LED is now OFF", "");
473       ledState = LOW;
474       digitalWrite(ledPin, ledState);
475     }
476     //and last one get status command
477     if (user_text == "/get_status") {
478       if (digitalRead(ledPin)){
479         bot.sendMessage(chat_id, "LED is ON.", "");
480       }
481       else{
482         bot.sendMessage(chat_id, "LED is OFF.", "");
483       }
484     }
485   }
486 }
487
488 void setup() {
489   Serial.begin(115200);
490   pinMode(ledPin, OUTPUT);
491   digitalWrite(ledPin, ledState);
492
493   WiFi.mode(WIFI_STA);
494   WiFi.begin(ssid, password);
495
496   client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root ...
          certificate for api.telegram.org
497
498   while (WiFi.status() != WL_CONNECTED) {
499     delay(1000);
500     Serial.println("Connecting to WiFi..");
501   }
502   Serial.println(WiFi.localIP());
```

```
503 }
504
505 void loop() {
506   if (millis() > lastTimeBotRan + bot_delay)  {
507     int numNewMessages = ...
            bot.getUpdates(bot.last_message_received + 1);
508
509     while(numNewMessages) {
510       Serial.println("Got Response!");
511       handleNewMessages(numNewMessages);
512       numNewMessages = bot.getUpdates(bot.last_message_received ...
              + 1);
513     }
514     lastTimeBotRan = millis();
515   }
516 }
```

Output/Demonstration:

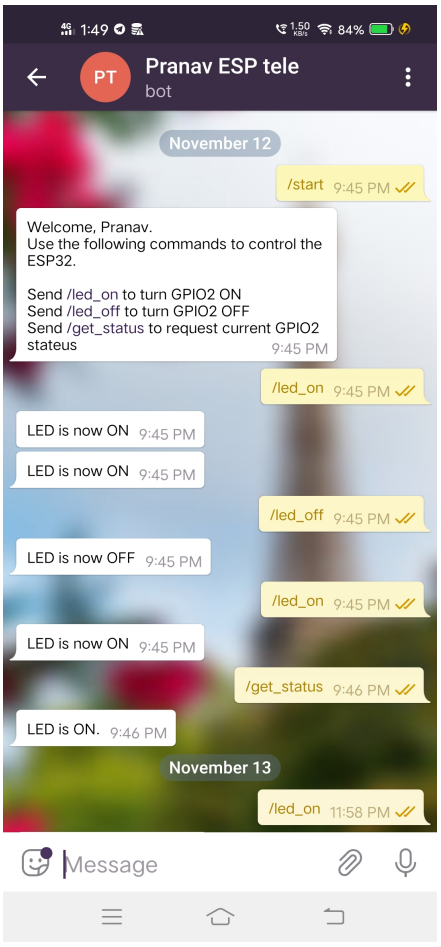

Figure 24: Pranav ESP32 Telebot.

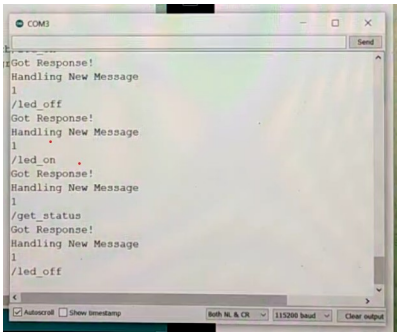

Figure 25: Output on serial monitor.

Youtube Video Link: https://www.youtube.com/watch?v=VrFf1-fs30g

Problems Faced (if any):

- Unauthorized user - Problem was solved by adopting our Telegram user ID.
- Poor internet connectivity may create a lag between commands and output.

References/ Citations:

1. ESP32 Blogs of Random nerd tutorials
https://randomnerdtutorials.com/telegram-control-esp32-esp8266-nodemcu-outputs/

Further scope: Here we have controlled LED using telebot but we can modify a bit and control other home appliances or sensors too and try for a good home automation project.

Task-4: To control on board LED of ESP32 using voice commands over Google Assistant.

Task:: To control on board LED of ESP32 using voice commands over Google Assistant.

Concept:: In this task, we will be using adafruit IO and IFTTT, some of the popular cloud platforms to build IoT and Automation based projects easily and on a large scale too.

MQTT is a lightweight messaging protocol that makes it easy to distribute telemetry data to network clients with limited resources and Adafruit Io works on MQTT.
Adafruit IO – https://io.adafruit.com/
We will create an Adafruit account first then create a ledcontrol feed and ledswitch dashboard.
Now into the same dashbaord we will create a block consisting of UI components like Buttons. And in the end, we will copy the AIO Active key generated so that we can adopt it in the code. Its a confedential key through which we will control our LED.

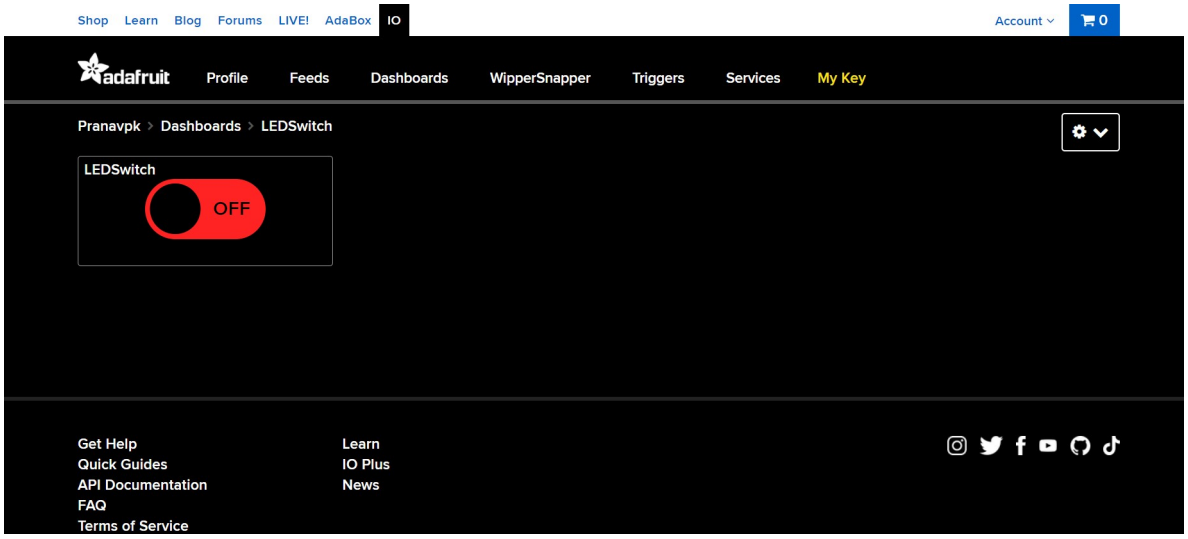

Figure 26: Adafruit IO dashboard.

Now let's connect our Google Assistant to the Adafruit IO MQTT Broker to allow us to control the lights with voice commands. for this we will need IFTTT platform.
IFTTT (If This Then That)– https://ifttt.com/
After creating account on IFTTT, We will create two applets using according to the

LED on/off condition and the voice commands we would like to give in the 'If this' part. And in the 'Then that' part, we will select option for sending data to adafruit choosing the feed we created in adafruit IO and giving appropriate command on that particular 'if this' condition.

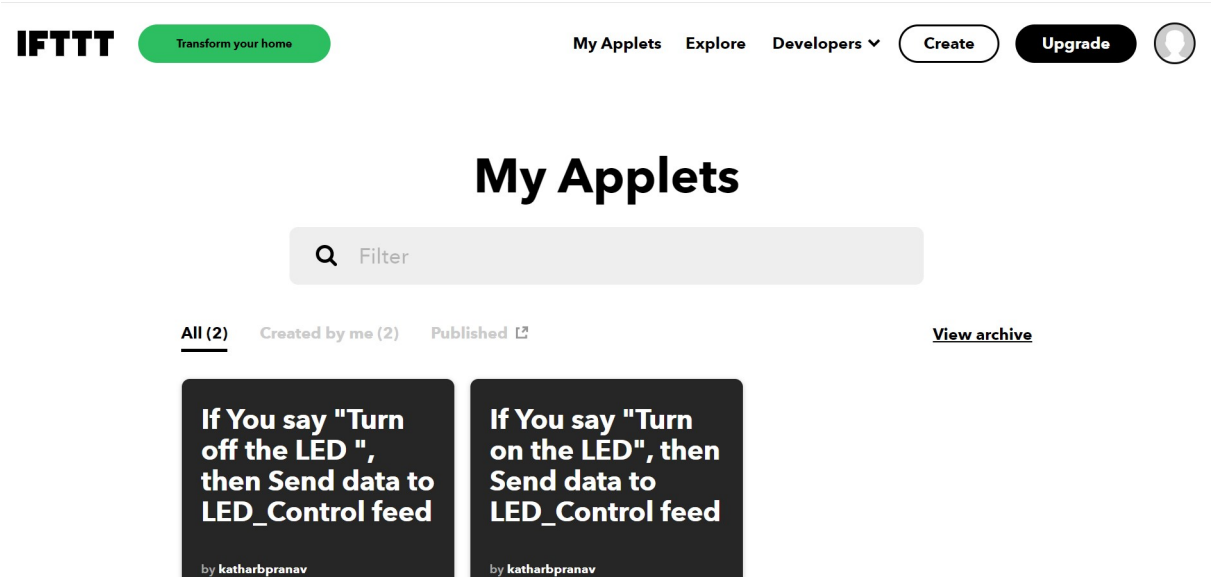

Figure 27: IFTTT dashboard.

Libraries required :

- Adafruit MQTT Client Library

Code with expalnation:

```
518 #include <WiFi.h>
519 #include "Adafruit_MQTT.h"
520 #include "Adafruit_MQTT_Client.h"
521 //wifi
522 #define SSID        "Ashish"
523 #define PASS        "12345678"
524
525 #define AIO_SERVER      "io.adafruit.com"
526 #define AIO_SERVERPORT  1883
527 //adafruit keys to send data on adafruit
528 #define AIO_USERNAME    "Pranavpk"
529 #define AIO_KEY         "aio_YbOB44s8tfzENRnGihOVvluHl3dq"
530
```

```
531 int output = 2;
532
533 WiFiClient client;
534 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, ...
       AIO_USERNAME, AIO_KEY);
535 Adafruit_MQTT_Subscribe LED_Control = ...
       Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME ...
       "/feeds/LED_Control");
536
537 void MQTT_connect();
538
539 void setup() {
540    //setting up serial monitor
541   Serial.begin(115200);
542   delay(10);
543   pinMode(2, OUTPUT);
544   Serial.println(); Serial.println();
545   Serial.print("Connecting to ");
546   Serial.println(SSID);
547   WiFi.begin(SSID, PASS);
548   while (WiFi.status() != WL_CONNECTED) {
549     delay(500);
550     Serial.print(".");
551   }
552   Serial.println();
553   Serial.println("WiFi connected");
554   Serial.println("IP address: "); Serial.println(WiFi.localIP());
555   mqtt.subscribe(&LED_Control);\
556 }
557 uint32_t x = 0;
558 void loop() {
559   MQTT_connect();
560   Adafruit_MQTT_Subscribe *subscription;
561   while ((subscription = mqtt.readSubscription(5000))) {
562     if (subscription == &LED_Control) {
563       Serial.print(F("Got: "));
564       Serial.println((char *)LED_Control.lastread);
565       if (!strcmp((char*) LED_Control.lastread, "ON"))
566       {
567         digitalWrite(2, HIGH);
568       }
569       else
570       {
571         digitalWrite(2, LOW);
572       }
573     }
574   }
575 }
576 //connecting to adafruit mqtt
```

```
577 void MQTT_connect() {
578
579   int8_t ret;
580   if (mqtt.connected()) {
581     return;
582   }
583   Serial.print("Connecting to MQTT... ");
584   uint8_t retries = 3;
585   while ((ret = mqtt.connect()) != 0) { // connect will return 0 ...
          for connected
586     Serial.println(mqtt.connectErrorString(ret));
587     Serial.println("Retrying MQTT connection in 5 seconds...");
588     mqtt.disconnect();
589     delay(5000);
590     retries--;
591     if (retries == 0) {
592       while (1);
593     }
594   }
595   Serial.println("MQTT Connected");
596 }
```

Output/Demonstration:

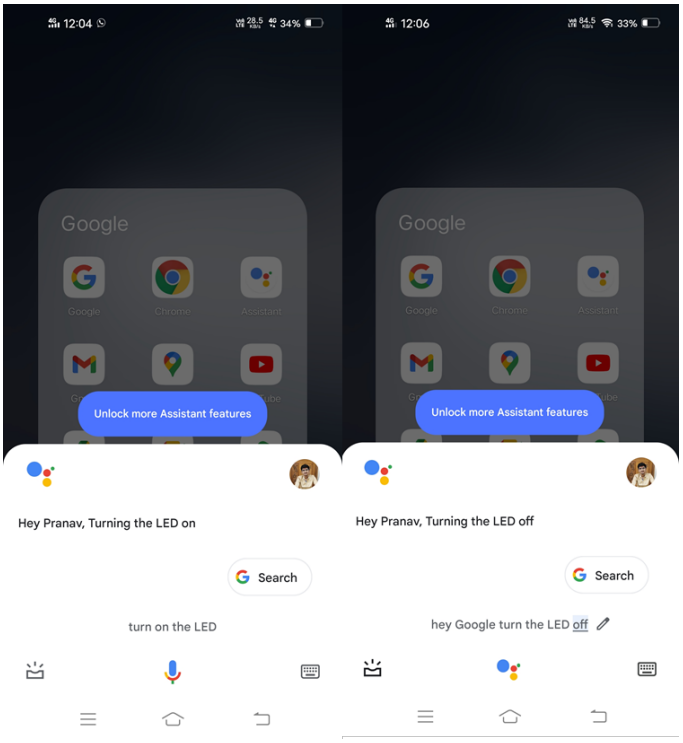

Figure 28: Commands over Google assistant.

Youtube Video Link: https://www.youtube.com/watch?v=aplKMZn2gQs

Problems Faced (if any):

- Poor internet connectivity may create a lag between our voice commands and results.Because MQTT will get disconnected in this case.

References/ Citations:

1. ESP32 Blogs of iotdesignpro
https://iotdesignpro.com/projects/google-assistant-controlled-led-using-ESP32-and-adafruit-io

Further scope: Here we have controlled LED using google assistant but we can modify our code a bit and remotely control AC home appliances or sensors anywhere using internet.

Task-5.1: RTOS Basics.

Task:: RTOS basics

Concept:: We use RTOS for tasks to run in concurrency on multiple cores. We want them to work in parallel. We can use delay function but it will block usage of our MCU for other purposes.As we have got 2 cores on ESP32 , Lets use it to the fullest

Code with expalnation:

```
598 //exploring basic functions of RTOS
599 int c1 = 0;
600 void t1(void *par){
601   for(;;){
602     Serial.print("Task1 running on core ");
603     // to get to know core on which task is running
604     Serial.println(xPortGetCoreID());
605     Serial.print("Task 1 count:");
606     Serial.print(c1++);
607     Serial.println();
608     vTaskDelay(1000);
609   }
610 }
611 void setup() {
612   // put your setup code here, to run once:
613   Serial.begin(9600);
614   //func for pinning task t1 to a particular core
615   xTaskCreatePinnedToCore(
616     t1,       // Task func
617     "Task 1", // Name
618     1000,     // Stack
619     NULL,     // Parameter
620     1,        // Priority, .
621     NULL,     // Task handle
622     0);       // Core
623 }
624 void loop() {
625   // put your main code here, to run repeatedly:
626 }
```

Output/Demonstration:

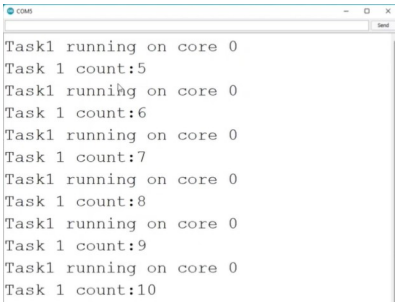

Figure 29: Output on serial monitor

References/ Citations:

1. Entire list of RTOS functions: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html
2. https://www.freertos.org/a00106.html

Further scope: We use it in home-automation or any other IoT application where multitasking is needed

Task-5.2: RTOS Task.

Task:: RTOS task

Concept:: Here we are trying to make 2 tasks which will run of different cores and increment the counters simultaneouly. And after certain period of time we will delete/suspend/resume task 1.

Code with expalnation:

```
627 // RTOS Task
628 // Two parallel (concurrent) tasks
629 // Two counters on two different cores
630 int c1 = 0;
631 int c2 = 0;
632 //task handles are required if we want to do some operation
633 //lets say delete a task outside the task function or in
634 //some other function
635 TaskHandle_t t1h = NULL;
636 TaskHandle_t t2h = NULL;
637 void t1(void *par){
638   for(;;){
639     Serial.print("Task1 running on core ");
640     Serial.println(xPortGetCoreID());
641     Serial.print("Task 1 count:");
642     Serial.print(c1++);
643     Serial.println();
644     vTaskDelay(1000);
645     if(c1>10){
646       //deleting the task after 10 seconds
647       //similarly we can try for suspend,resume functions in rtos
648       vTaskDelete(NULL);
649     }
650   }
651 }
652 void t2(void *par){
653   for(;;){
654     Serial.print("Task2 running on core ");
655     Serial.println(xPortGetCoreID());
656     Serial.print("Task 2 count:");
657     Serial.print(c2++);
658     Serial.println();
659     vTaskDelay(1000);
660   }
661 }
662 void setup() {
```

```
663   // put your setup code here, to run once:
664   Serial.begin(9600);
665   xTaskCreatePinnedToCore(
666     t1,        // Task function
667     "Task 1", // Task Name
668     1000,      // Stack size
669     NULL,      // Parameter for the task
670     1,         // Priority
671     NULL,      // Task handle
672     0);        // Core
673   xTaskCreatePinnedToCore(
674     t2,
675     "Task 2",
676     1000,      // Stack size
677     NULL,      // Parameter for the task
678     1,         // Priority, high for high no.
679     NULL,      // Task handle
680     1);        // Core
681 }
682 void loop() {
683   // put your main code here, to run repeatedly:
684 }
```

Output/Demonstration:

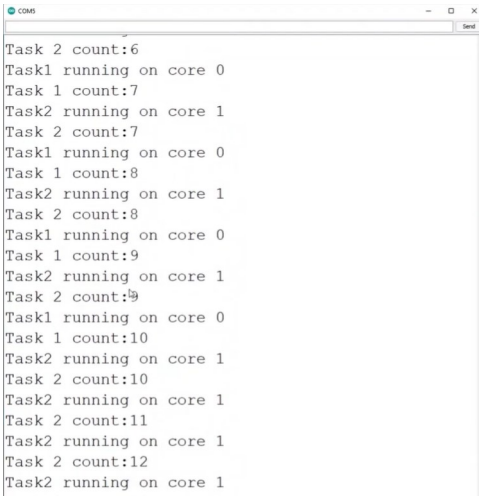

Figure 30: Output on serial monitor

References/ Citations:

1. https://savjee.be/videos/programming-esp32-with-arduino/what-is-freertos

Further scope: We use it in home-automation or any other IoT application where multitasking is needed

Task-6: Wi-Fi controlled DIY car.

Task:: Wi-Fi controlled DIY car

Concept::

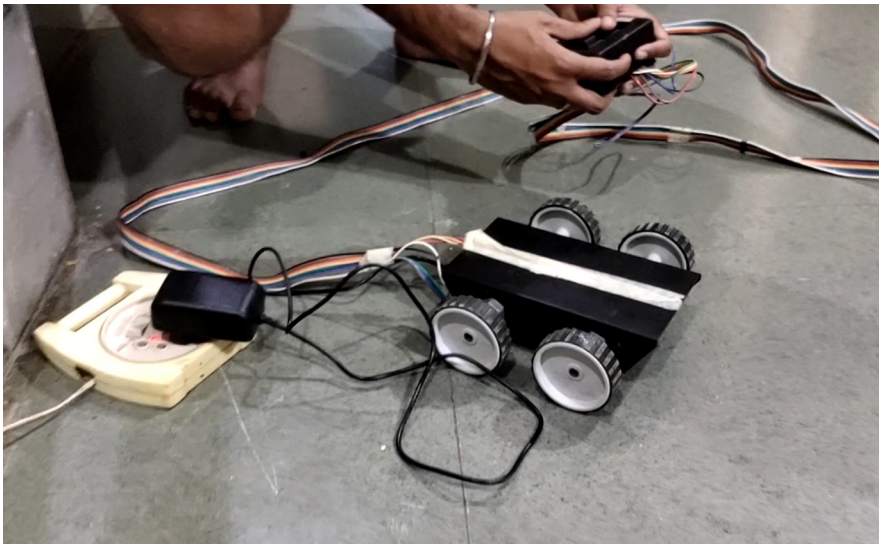

Figure 31: You can see the wires required for controlling this DIY using remote.

So lets try to make it wifi controlled using web server

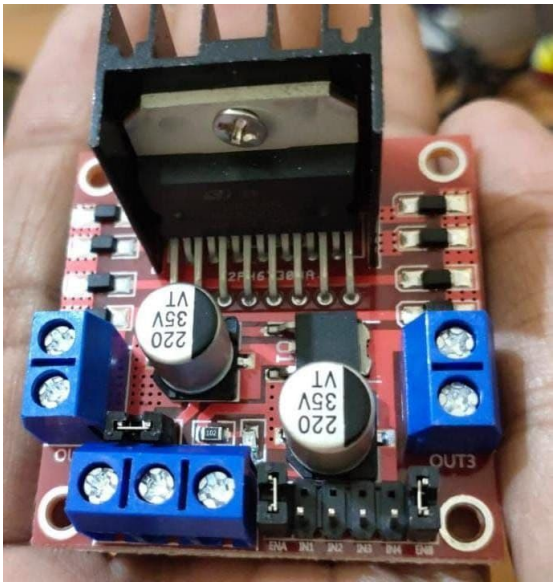

Figure 32: Motor drivers for controlling motor speed and direction

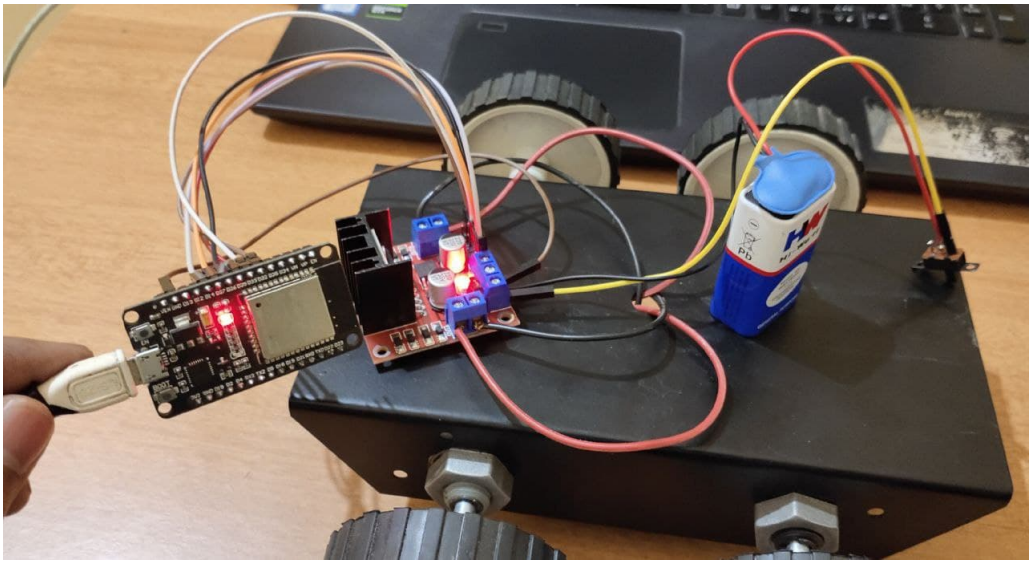

Figure 33: testing the connections

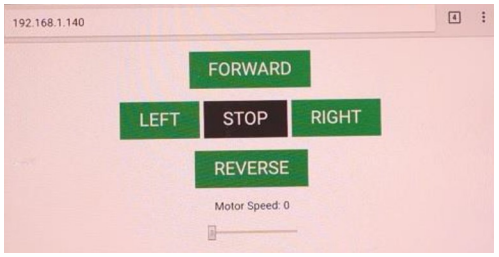

Figure 34: This is how web server for controlling this DIY looks like

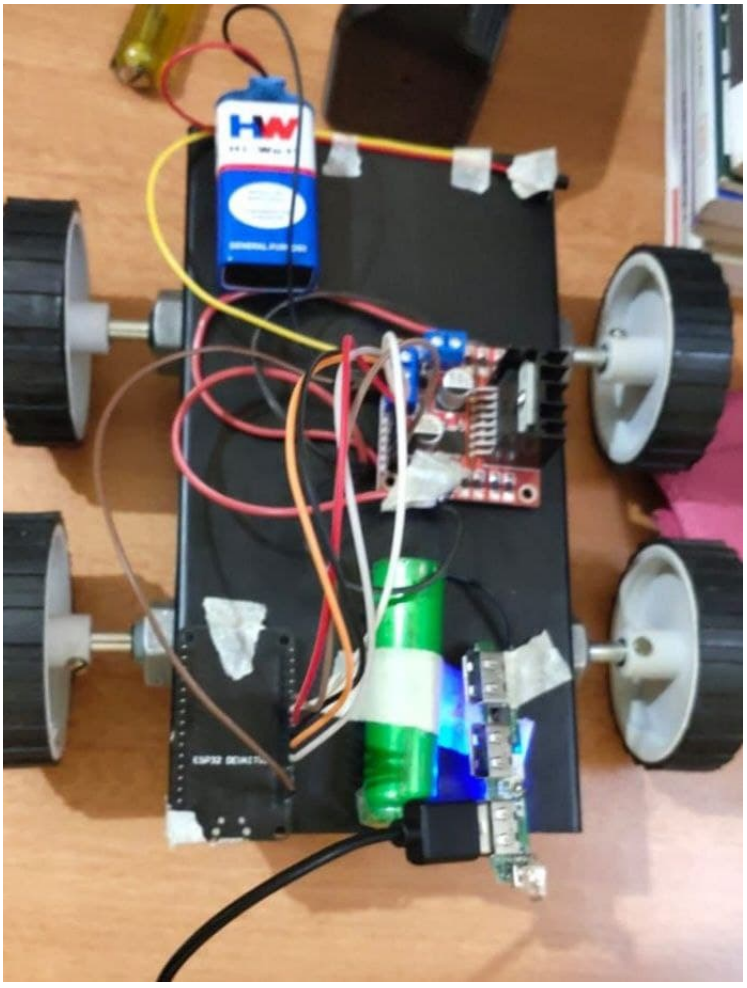

Figure 35: Final circuit, fixed on the chassis

Code with expalnation:

```
686
687 #include <Arduino.h>
688 #include "WiFi.h"
689
690 #define WIFI_NETWORK "vivo 1951"
691 #define WIFI_PASSWORD "12345678"
692 #define WIFI_TIMEOUT_MS 20000
693 #define WIFI_RECOVER_TIME_MS 30000
694
695 //trying to keep wifi alive by creating rtos func
696 void keepWiFiAlive(void * parameter){
697     for(;;){
698         if(WiFi.status() == WL_CONNECTED){
699             vTaskDelay(10000 / portTICK_PERIOD_MS);
700             continue;
701         }
702         Serial.println("[WIFI] Connecting");
703         WiFi.mode(WIFI_STA);
704         WiFi.begin(WIFI_NETWORK, WIFI_PASSWORD);
705         unsigned long startAttemptTime = millis();
706         while (WiFi.status() != WL_CONNECTED &&
707                 millis() - startAttemptTime < WIFI_TIMEOUT_MS){}
708
709         if(WiFi.status() != WL_CONNECTED){
710             Serial.println("[WIFI] FAILED");
711             vTaskDelay(WIFI_RECOVER_TIME_MS / portTICK_PERIOD_MS);
712         continue;
713         }
714
715         Serial.println("[WIFI] Connected: " + WiFi.localIP());
716     }
717 }
718
719 WiFiServer server(80);//port 80
720 String header;
721
722 // Motor1
723 int motor1Pin1 = 27;
724 int motor1Pin2 = 26;
725 int enable1Pin = 14;
726 // Motor2
727 int motor2Pin1 = 33;
728 int motor2Pin2 = 25;
729 int enable2Pin = 32;
730
731 // Setting PWM
```

```
732 const int freq = 30000;
733 const int pwmChannel = 0;
734 const int resolution = 8;
735 int dutyCycle = 0;
736
737 // Decode HTTP GET value
738 String valueString = String(5);
739 int pos1 = 0;
740 int pos2 = 0;
741
742 // Current time
743 unsigned long currentTime = millis();
744 // Previous time
745 unsigned long previousTime = 0;
746 // Define timeout time in milliseconds (example: 2000ms = 2s)
747 const long timeoutTime = 2000;
748
749 void setup() {
750   Serial.begin(115200);
751   xTaskCreatePinnedToCore(
752     keepWiFiAlive,
753     "keepWiFiAlive",  // Task name
754     5000,             // Stack size (bytes)
755     NULL,             // Parameter
756     1,                // Task priority
757     NULL,             // Task handle
758     CONFIG_ARDUINO_RUNNING_CORE
759   );
760
761   // Set the Motor pins as outputs
762   pinMode(motor1Pin1, OUTPUT);
763   pinMode(motor1Pin2, OUTPUT);
764   pinMode(motor2Pin1, OUTPUT);
765   pinMode(motor2Pin2, OUTPUT);
766
767   // Configure PWM channel functionalitites
768   ledcSetup(pwmChannel, freq, resolution);
769
770   // Attach the PWM channel 0 to the enable pins which are the ...
        GPIOs to be controlled
771   ledcAttachPin(enable1Pin, pwmChannel);
772   ledcAttachPin(enable2Pin, pwmChannel);
773
774   // Produce a PWM signal to both enable pins with a duty cycle 0
775   ledcWrite(pwmChannel, dutyCycle);
776
777 /*  // Connect to Wi-Fi network with SSID and password
778   Serial.print("Connecting to ");
779   Serial.println(ssid);
```

```
780   WiFi.begin(ssid, password);
781   while (WiFi.status() != WL_CONNECTED) {
782     delay(500);
783     Serial.print(".");
784   }
785   // Print local IP address and start web server
786   Serial.println("");
787   Serial.println("WiFi connected.");
788   Serial.println("IP address: ");
789   Serial.println(WiFi.localIP());
790   server.begin();
791 */
792 }
793
794
795 void loop(){
796   WiFiClient client = server.available();   // Listen for ...
          incoming clients
797
798   if (client) {                             // If a new client ...
          connects,
799     currentTime = millis();
800     previousTime = currentTime;
801     Serial.println("New Client.");          // print a message ...
          out in the serial port
802     String currentLine = "";                // make a String to ...
          hold incoming data from the client
803     while (client.connected() && currentTime - previousTime ≤ ...
          timeoutTime) {  // loop while the client's connected
804       currentTime = millis();
805       if (client.available()) {             // if there's bytes ...
            to read from the client,
806         char c = client.read();             // read a byte, then
807         Serial.write(c);                    // print it out the ...
              serial monitor
808         header += c;
809         if (c == '\n') {                    // if the byte is a ...
              newline character
810           // if the current line is blank, you got two newline ...
                characters in a row.
811           // that's the end of the client HTTP request, so send ...
                a response:
812           if (currentLine.length() == 0) {
813             // HTTP headers always start with a response code ...
                  (e.g. HTTP/1.1 200 OK)
814             // and a content-type so the client knows what's ...
                  coming, then a blank line:
815             client.println("HTTP/1.1 200 OK");
816             client.println("Content-type:text/html");
```

```
817             client.println("Connection: close");
818             client.println();
819
820             // Controls the motor pins according to the button ...
                    pressed
821             if (header.indexOf("GET /forward") ≥ 0) {
822               Serial.println("Forward");
823               digitalWrite(motor1Pin1, LOW);
824               digitalWrite(motor1Pin2, HIGH);
825               digitalWrite(motor2Pin1, LOW);
826               digitalWrite(motor2Pin2, HIGH);
827             }  else if (header.indexOf("GET /left") ≥ 0) {
828               Serial.println("Left");
829               digitalWrite(motor1Pin1, LOW);
830               digitalWrite(motor1Pin2, LOW);
831               digitalWrite(motor2Pin1, LOW);
832               digitalWrite(motor2Pin2, HIGH);
833             }  else if (header.indexOf("GET /stop") ≥ 0) {
834               Serial.println("Stop");
835               digitalWrite(motor1Pin1, LOW);
836               digitalWrite(motor1Pin2, LOW);
837               digitalWrite(motor2Pin1, LOW);
838               digitalWrite(motor2Pin2, LOW);
839             } else if (header.indexOf("GET /right") ≥ 0) {
840               Serial.println("Right");
841               digitalWrite(motor1Pin1, LOW);
842               digitalWrite(motor1Pin2, HIGH);
843               digitalWrite(motor2Pin1, LOW);
844               digitalWrite(motor2Pin2, LOW);
845             } else if (header.indexOf("GET /reverse") ≥ 0) {
846               Serial.println("Reverse");
847               digitalWrite(motor1Pin1, HIGH);
848               digitalWrite(motor1Pin2, LOW);
849               digitalWrite(motor2Pin1, HIGH);
850               digitalWrite(motor2Pin2, LOW);
851             }
852             // Display the HTML web page
853             client.println("<!DOCTYPE HTML><html>");
854             client.println("<head><meta name=\"viewport\" ...
                    content=\"width=device-width, initial-scale=1\">");
855             client.println("<link rel=\"icon\" href=\"data:,\">");
856             // CSS to style the buttons
857             // Feel free to change the background-color and ...
                    font-size attributes to fit your preferences
858             client.println("<style>html { font-family: ...
                    Helvetica; display: inline-block; margin: 0px ...
                    auto; text-align: center;}");
859             client.println(".button { -webkit-user-select: none; ...
                    -moz-user-select: none; -ms-user-select: none; ...
```

```
                user-select: none; background-color: #4CAF50;");
860             client.println("border: none; color: white; padding: ...
                12px 28px; text-decoration: none; font-size: ...
                26px; margin: 1px; cursor: pointer;}");
861             client.println(".button2 {background-color: ...
                #555555;}</style>");
862             client.println("<script ...
                src=\"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min
863
864             // Web Page
865             client.println("<p><button class=\"button\" ...
                onclick=\"moveForward()\">FORWARD</button></p>");
866             client.println("<div style=\"clear: ...
                both;\"><p><button class=\"button\" ...
                onclick=\"moveLeft()\">LEFT </button>");
867             client.println("<button class=\"button button2\" ...
                onclick=\"stopRobot()\">STOP</button>");
868             client.println("<button class=\"button\" ...
                onclick=\"moveRight()\">RIGHT</button></p></div>");
869             client.println("<p><button class=\"button\" ...
                onclick=\"moveReverse()\">REVERSE</button></p>");
870             client.println("<p>Motor Speed: <span ...
                id=\"motorSpeed\"></span></p>");
871             client.println("<input type=\"range\" min=\"0\" ...
                max=\"100\" step=\"25\" id=\"motorSlider\" ...
                onchange=\"motorSpeed(this.value)\" value=\"" + ...
                valueString + "\"/>");
872
873             client.println("<script>$.ajaxSetup({timeout:1000});");
874             client.println("function moveForward() { ...
                $.get(\"/forward\"); {Connection: close};}");
875             client.println("function moveLeft() { ...
                $.get(\"/left\"); {Connection: close};}");
876             client.println("function stopRobot() ...
                {$.get(\"/stop\"); {Connection: close};}");
877             client.println("function moveRight() { ...
                $.get(\"/right\"); {Connection: close};}");
878             client.println("function moveReverse() { ...
                $.get(\"/reverse\"); {Connection: close};}");
879             client.println("var slider = ...
                document.getElementById(\"motorSlider\");");
880             client.println("var motorP = ...
                document.getElementById(\"motorSpeed\"); ...
                motorP.innerHTML = slider.value;");
881             client.println("slider.oninput = function() { ...
                slider.value = this.value; motorP.innerHTML = ...
                this.value; }");
882             client.println("function motorSpeed(pos) { ...
                $.get(\"/?value=\" + pos + \"&\"); {Connection: ...
```

```
                close};}</script>");
883
884             client.println("</html>");
885
886             //Request example: GET /?value=100& HTTP/1.1 - sets ...
                   PWM duty cycle to 100% = 255
887             if(header.indexOf("GET /?value=")>=0) {
888               pos1 = header.indexOf('=');
889               pos2 = header.indexOf('&');
890               valueString = header.substring(pos1+1, pos2);
891               //Set motor speed value
892               if (valueString == "0") {
893                 ledcWrite(pwmChannel, 0);
894                 digitalWrite(motor1Pin1, LOW);
895                 digitalWrite(motor1Pin2, LOW);
896                 digitalWrite(motor2Pin1, LOW);
897                 digitalWrite(motor2Pin2, LOW);
898               }
899               else {
900                 dutyCycle = map(valueString.toInt(), 25, 100, ...
                       200, 255);
901                 ledcWrite(pwmChannel, dutyCycle);
902                 Serial.println(valueString);
903               }
904             }
905             // The HTTP response ends with another blank line
906             client.println();
907             // Break out of the while loop
908             break;
909           } else { // if you got a newline, then clear currentLine
910             currentLine = "";
911           }
912         } else if (c != '\r') {  // if you got anything else but ...
              a carriage return character,
913           currentLine += c;      // add it to the end of the ...
                 currentLine
914         }
915       }
916     }
917     // Clear the header variable
918     header = "";
919     // Close the connection
920     client.stop();
921     Serial.println("Client disconnected.");
922     Serial.println("");
923   }
924 }
```

Output/Demonstration:

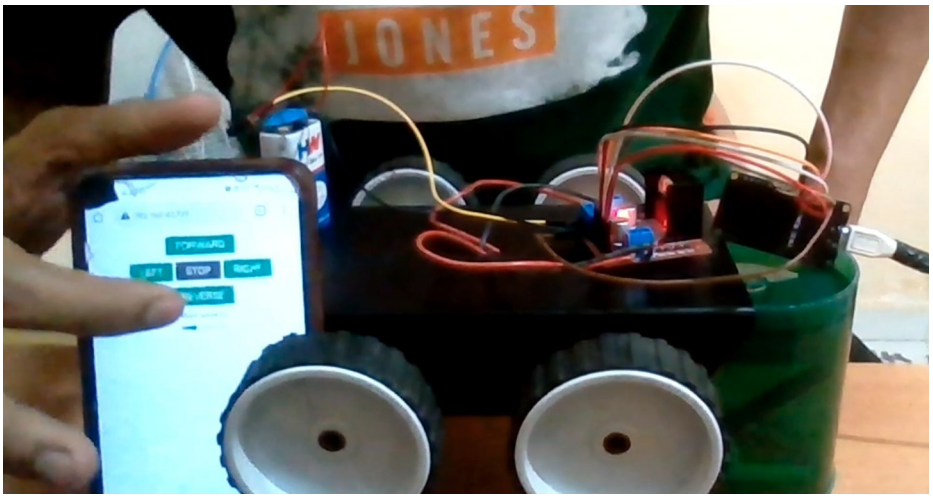

Figure 36: Performing demo

Youtube Video Link: https://youtu.be/m0jm8an1I6o

Problems Faced (if any):

- Wi-Fi wasn't alive after power cut - problem solved using rtos

References/ Citations:

1. ESP32 Blogs of Random nerd tutorials and the book by ruis santos

Further scope: Well, this Wi-Fi controlled DIY has lot of scope. We can modify it as per our requirements. We can add PIR sensors, ultrasonic sensors, proximity sensors and do more exciting stuff. Also, we can interface cam module and with enough knowledge of image processing we can make some nice projects. We can even make it long range by adopting Lo-Ra transreciever