Universal Development Board(UDB)

One Development Board for Arduino Uno / Nano, ESP32, STM32, 8051 & PIC MCU's

ESP32 DEV KIT

Reference Manual

AMOTECH LABS

Embedded systems | IndustrialAutomation | Robotics

Scan to download So Copy of Manual & Online Purchase

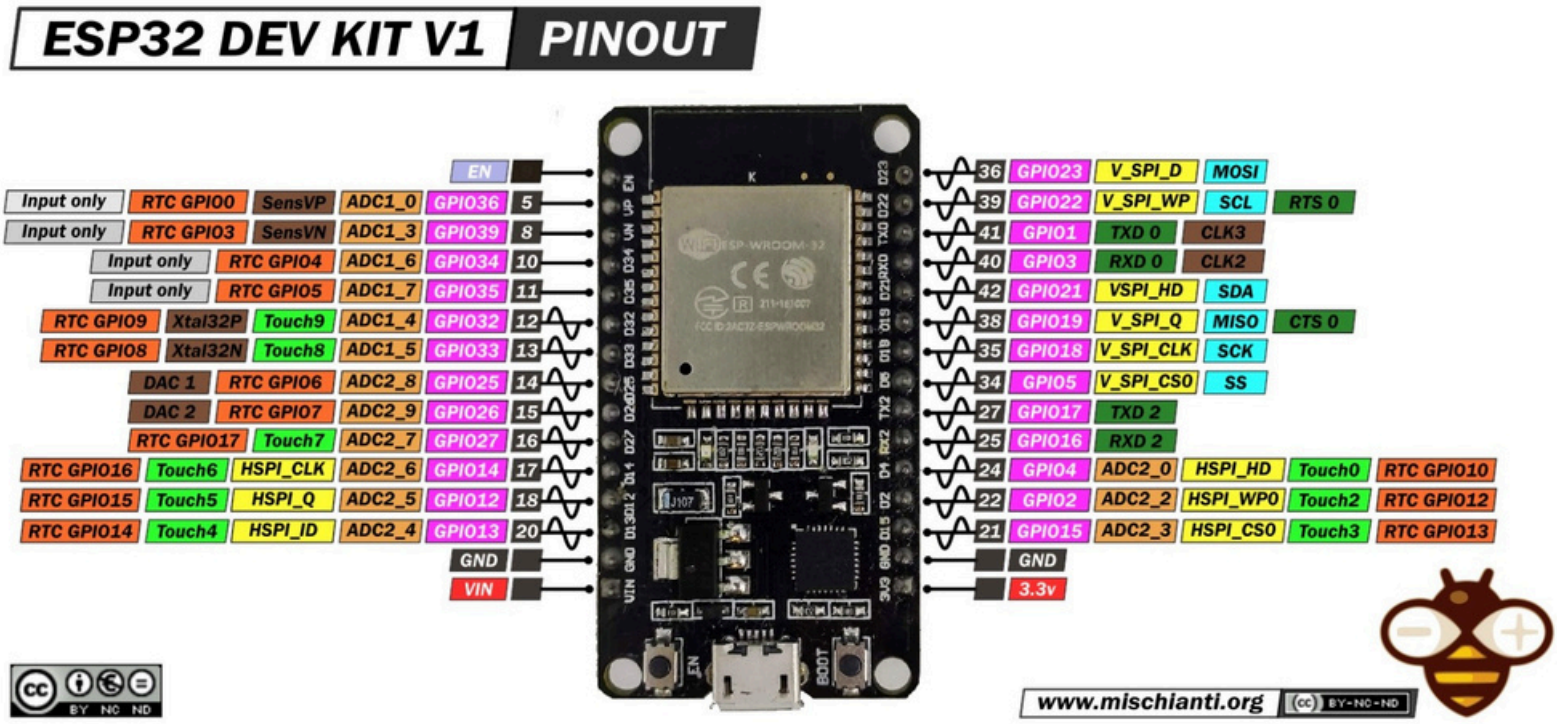
Scan for Video Tutorials

Table of Contents

Introduction

Embedded System ESP32 DEV KIT V1 Microcontroller:

Overview:



The ESP32 is a low-cost, high-performance microcontroller developed by Espressif Systems. It is widely used in IoT, automation, and embedded systems due to its integrated Wi-Fi, Bluetooth, and high processing capability.

Pin Name	Type	Function / Description
EN	Control	Chip enable pin; active HIGH to enable ESP32
VIN	Power	5 V input from USB or external power supply
3V3	Power	3.3 V regulated output from on-board regulator
GND	Power	Ground
GPIO0	I/O	General-purpose I/O; used for boot mode selection
GPIO1 (TX0)	UART	UART0 transmit pin (serial output)
GPIO3 (RX0)	UART	UART0 receive pin (serial input)
GPIO2, GPIO4	I/O	General-purpose I/O; supports PWM and digital I/O
GPIO12 – GPIO15	I/O	ADC, touch sensor, PWM, and general-purpose I/O
GPIO16 – GPIO17	I/O	UART2 TX/RX or general-purpose digital I/O
GPIO18 – GPIO19	I/O	SPI clock and MISO or general-purpose I/O
GPIO21 – GPIO23	I/O	I²C (SDA/SCL) or general-purpose I/O
GPIO25 – GPIO27	I/O	DAC, ADC, PWM, and digital I/O supported
GPIO32 – GPIO33	I/O	ADC, touch sensor, PWM supported
GPIO34 – GPIO39	Input Only	Input-only pins; ADC and touch sensor supported

Universal Development Board(UDB)

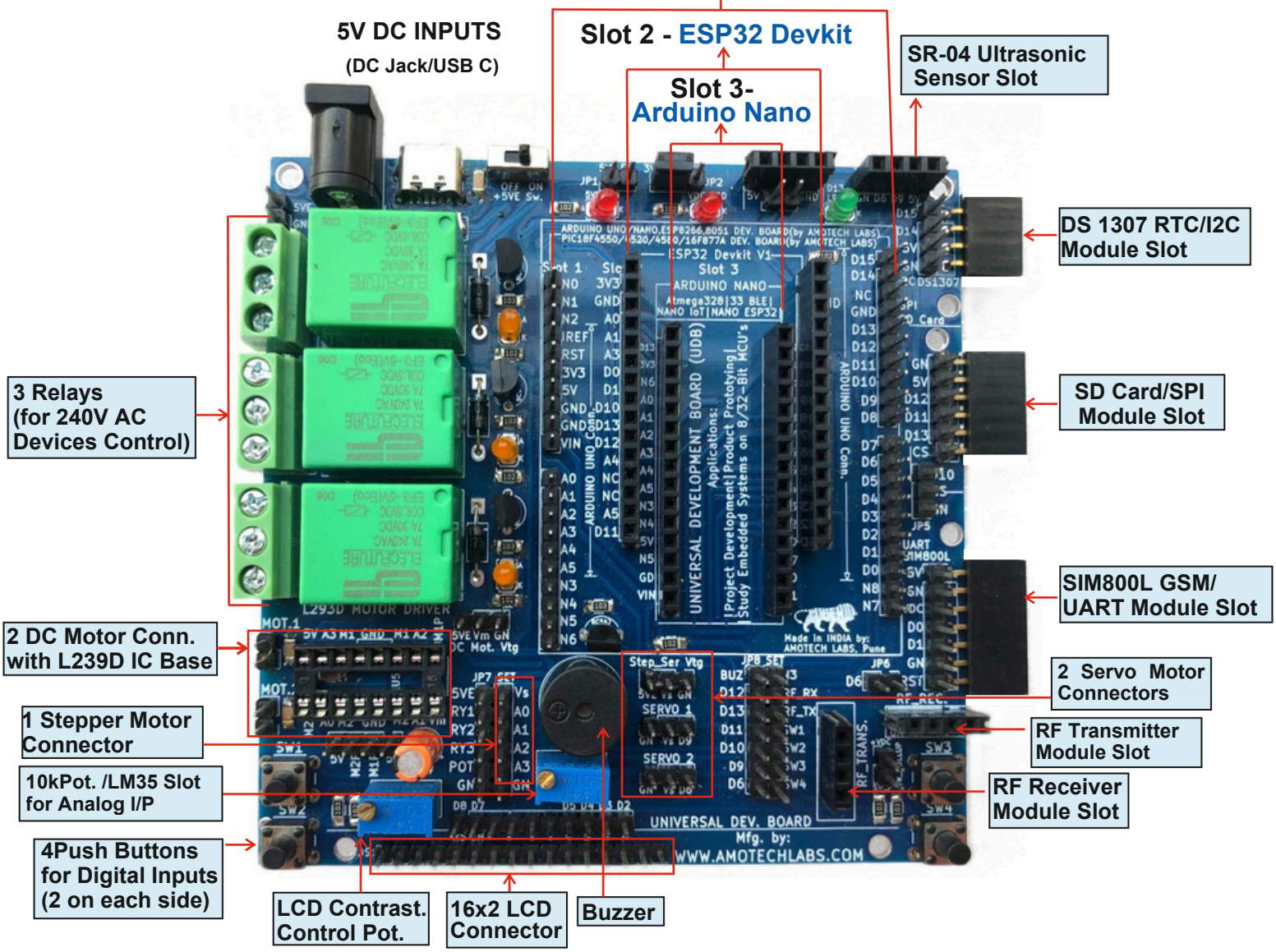
One Development for

Studying Embedded Systems | Project Development | Product Prototyping

on below multiple Microcontrollers

8051 Shield & PIC18F/16F Dev. Boards | Arduino Uno | ESP32 Devkit V1 | Arduino/STM32 Nano Boards

Slot 1- Arduino Uno Boards / 8051 & PIC Dev. Boards by Amotech Labs



Below Microcontroller Shields/Boards can be inserted into UDB and interfaced with all above Peripherals on it

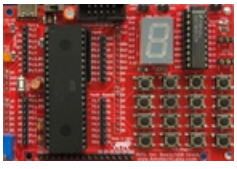
Slot 1

Arduino Uno

STM32F0/F4 NucleoBoards

8051 & PIC18F/16 Mini Dev. Boards by Amotech Labs

8051 & PIC18F/16 Dev. Boards by Amotech Labs



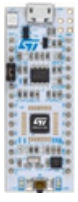
Slot 2

ESP32 Devkit V1

Slot 3

Arduino Nano

STM32 Nucleo Nano



Arduino NanoRp2040

& All Development Boards/Shields in Arduino Nano Footprint

Universal Development Board(UDB) kit

For Students

Universal Development Board(UDB)

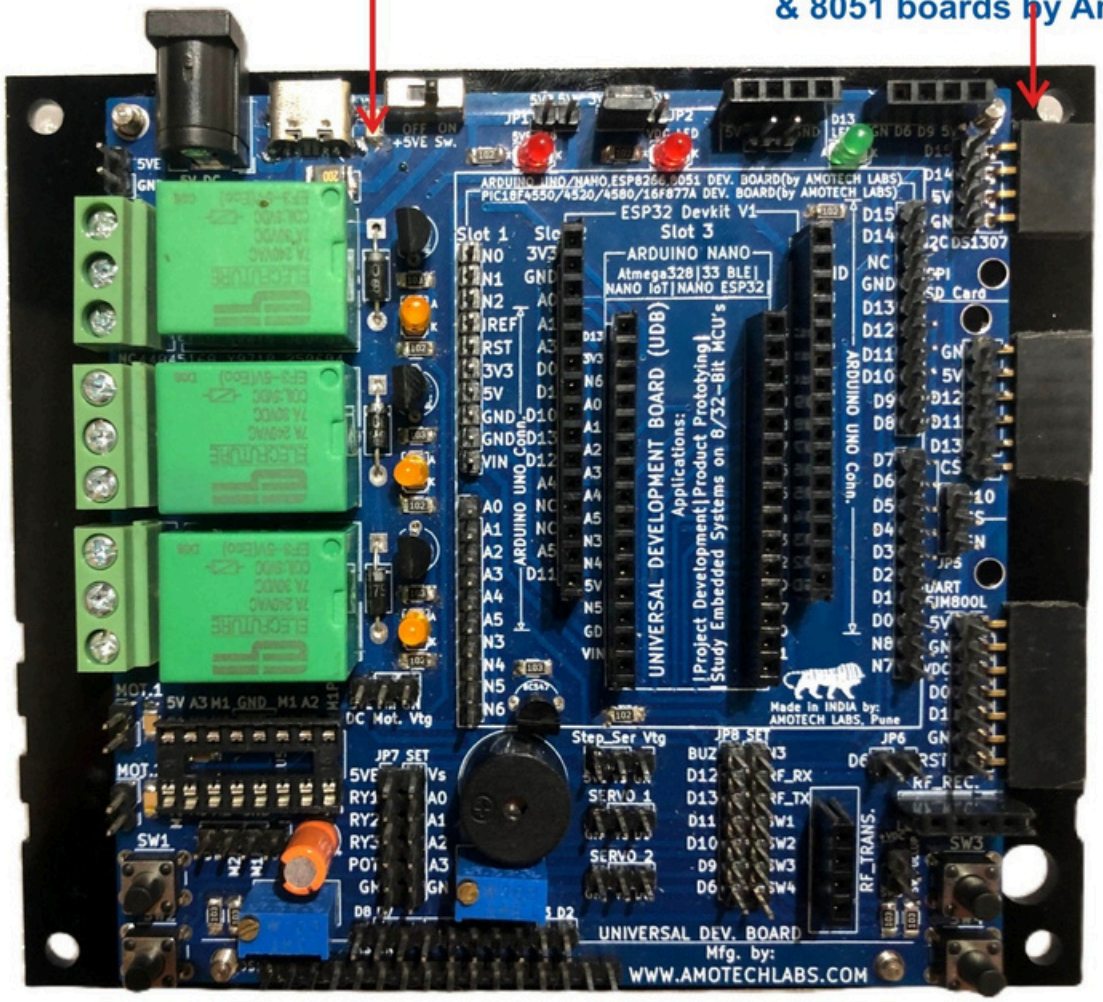
+

UDB 2 Wheel Robot Acrylic Sheet

This includes below two items as shown in the below

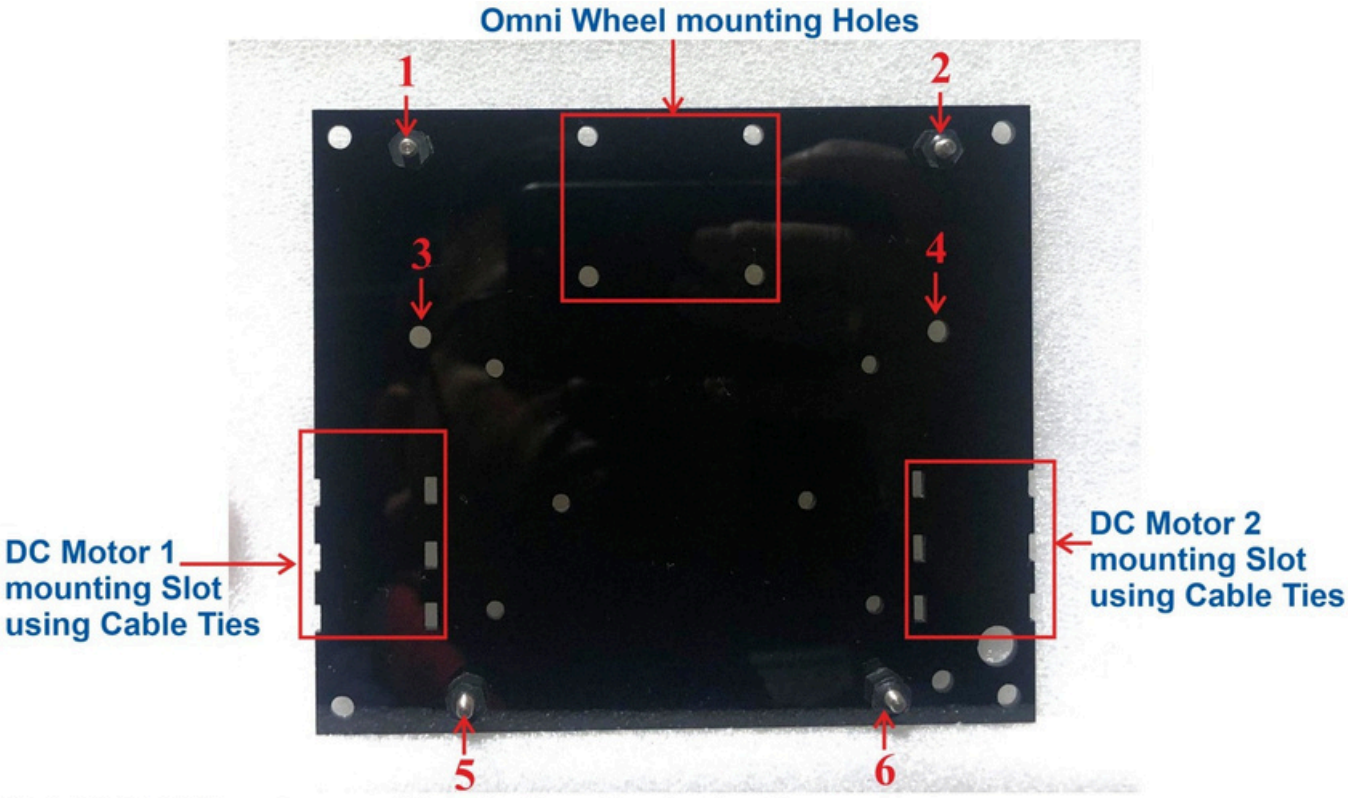
1- Universal Development Board(UDB)

2- UDB 2 Wheel Robot Acrylic Sheet (used for mounting UDB and also PIC & 8051 boards by Amotech Labs)



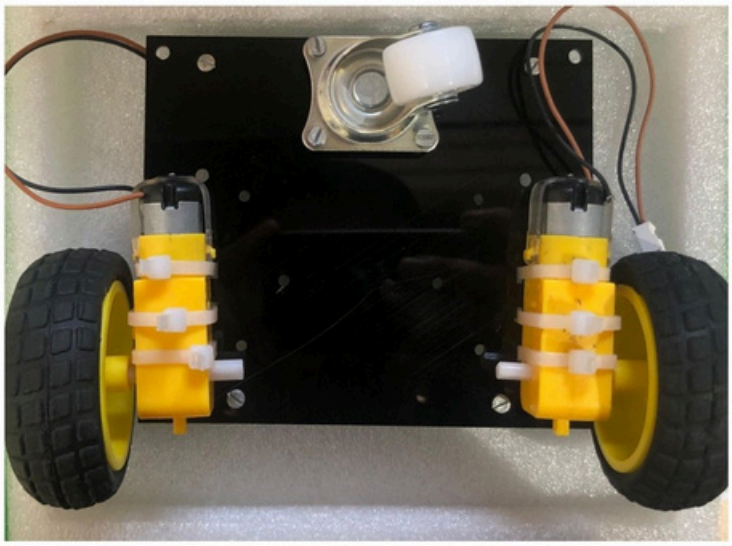
Note: 'UDB 2 Wheel Robot Acrylic Sheet' can be used to hold UDB and mount it on any other platform to use it for multiple Applications. It can be also converted into Robot by attaching 2 DC motors with wheels at it's back and 1 omni wheel at front. Refer UDB Manual or our Reference YouTube videos to see how it can be used as bot.

UDB 2 Wheel Robot Acrylic Sheet

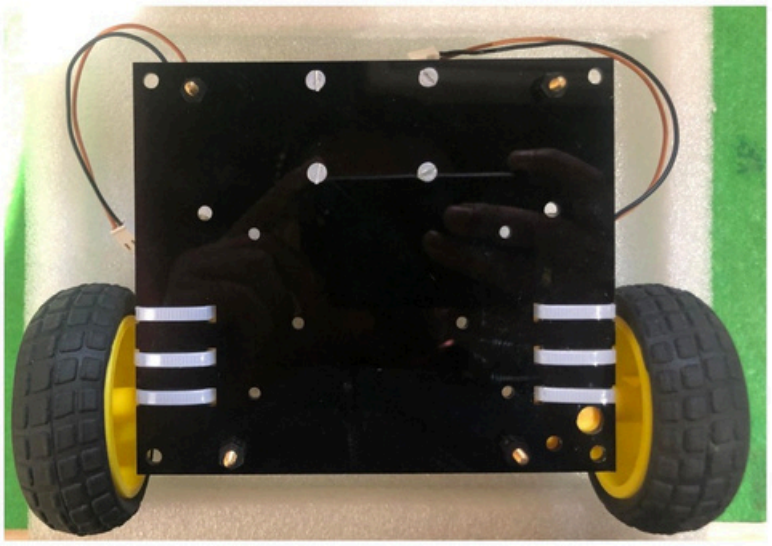


UDB & PIC/8051 Development Boards Mounting Instructions :
Universal Development Board : Use holes 1,2,5 & 6 for mounting UDB on above Acrylic Sheet.
PIC & 8051 Dev. Boards: Use holes 3,4,5 & 6 for mounting PIC18F/16F & 8051 Development Boards.

After Mounting Omni Wheel & 2 DC Motors along with their wheels



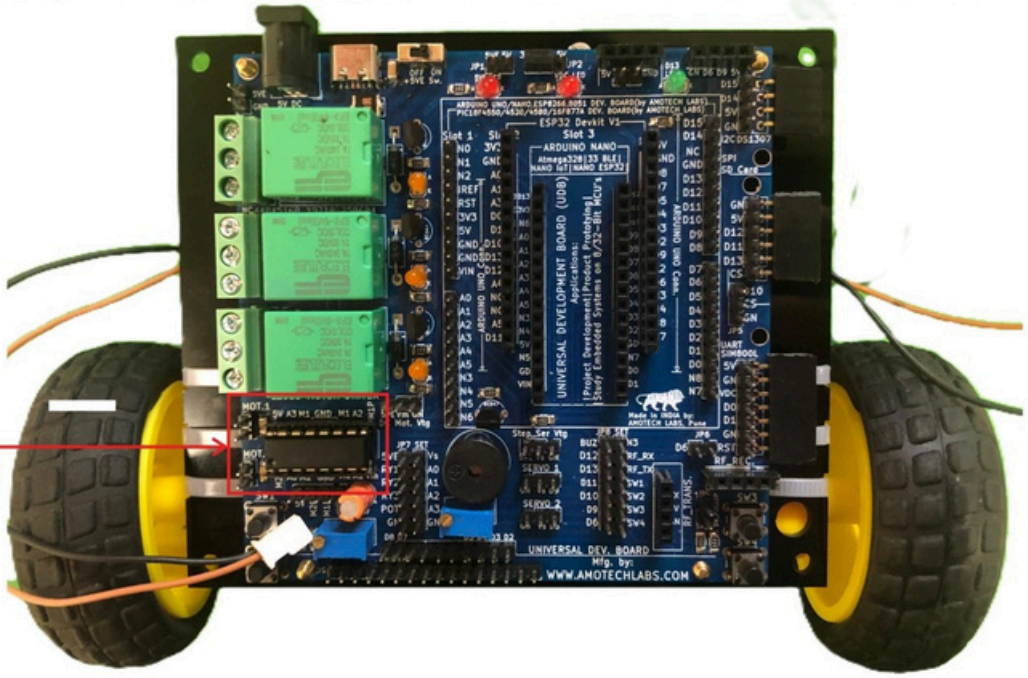
Backside View of Acrylic Sheet after mounting Omni Wheel and 2 DC Motors with their Wheels. Omni wheel is mounted using 10mm spacers with M3 nuts and DC motors can be mounted using Cable ties.



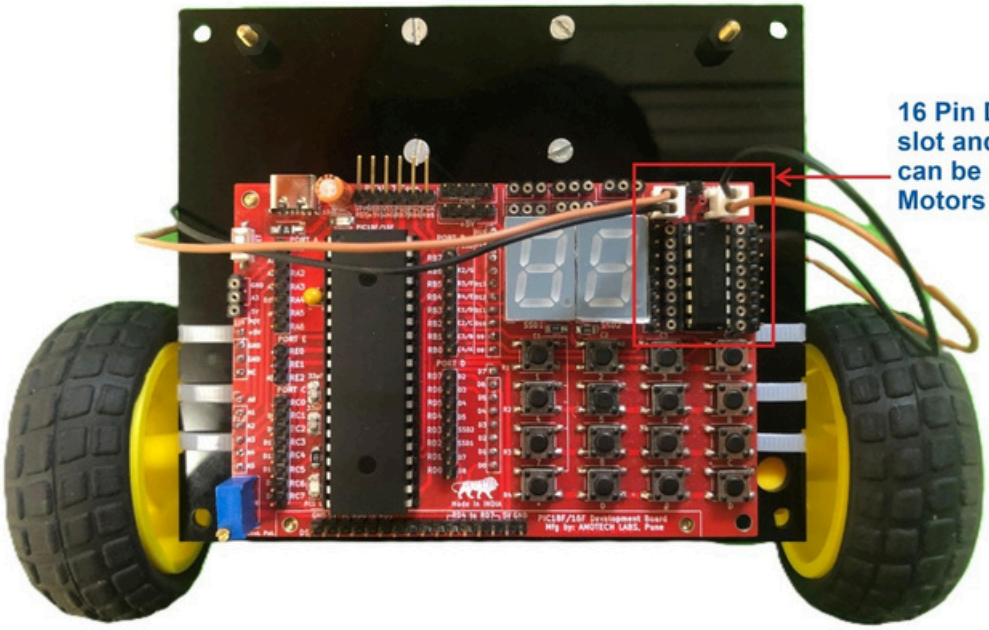
Front View of Acrylic Sheet after mounting Omni Wheel and 2 DC Motors with their Wheels. Now, on top 'UDB' and PIC/8051 Development Boards can be mounted.

Assembled 'UDB 2 Wheel Acrylic Robot Sheet' with UDB & 8051/PIC Boards

'Universal Development Board(UDB) mounted on assembled Robot Sheet



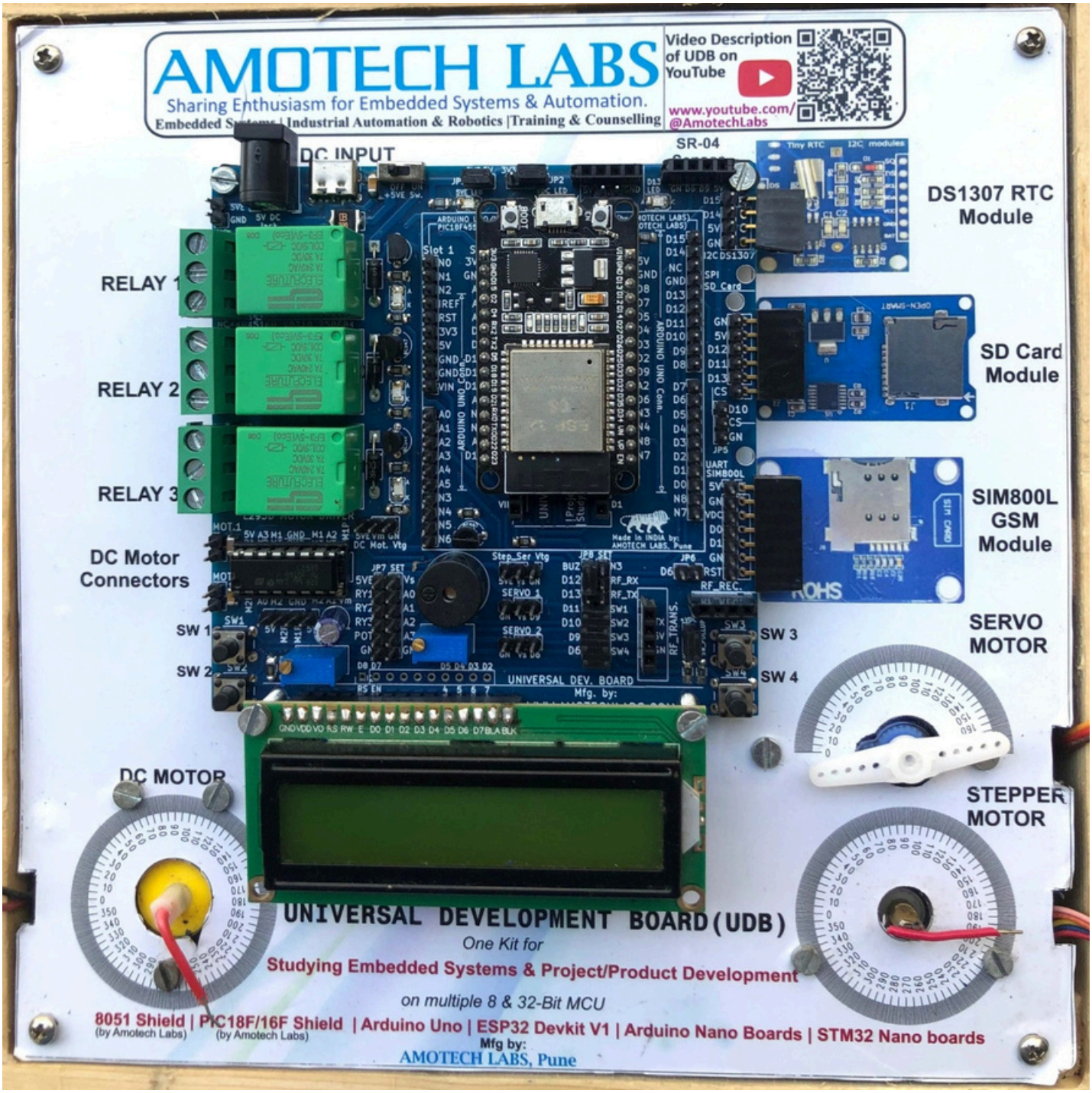
'PIC18F/16F Development Board' mounted on assembled Robot Sheet



Universal Development Board(UDB) kit

UDB Mounted in Wooden Box Enclosure with on board DC, Servo & Stepper Motors

Universal Development Board(UDB) + ESP32 Devkit V1



In above Kit, UDB is mounted on a sheet in a Wooden Enclosure Box along with DC, Servo and Stepper Motors. The ESP32 Devkit V1 must be attached to UDB to it's Slot 2 as seen in the Picture above.

To study & understand the 'Universal Development Board (UDB)' schematic visit our website for its Reference material, and also refer to our Youtube channel for video tutorials for more understanding

Website Link:

https://www.amotechlabs.com/

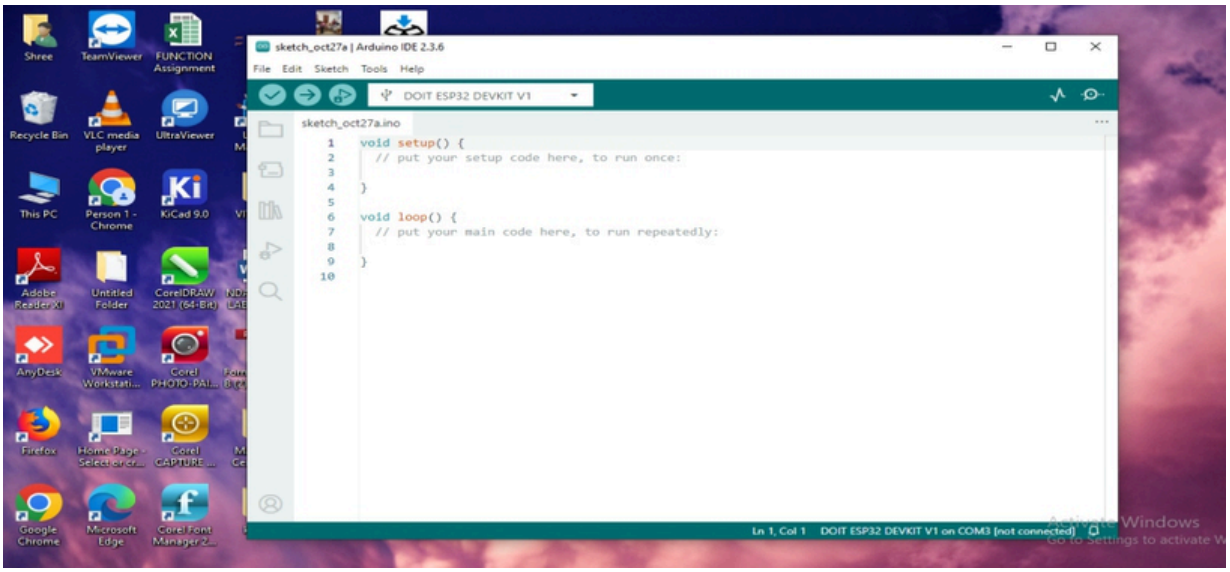
For Youtube Tutorial:

Scan for Video Tutorials

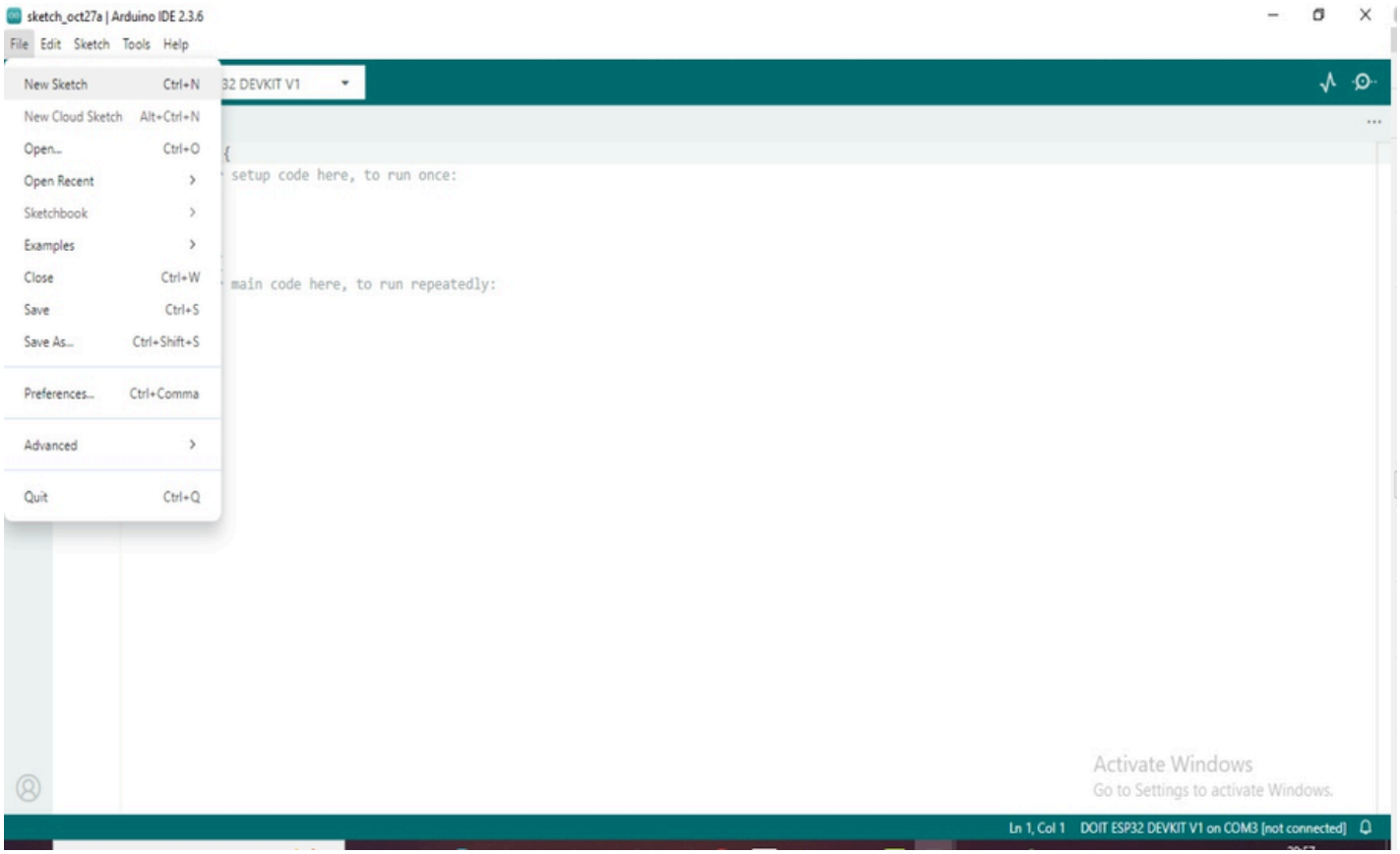
https://www.youtube.com/@AmotechLabs

User Guide for UDB So ware
Steps for use of Arduino IDE

1. Double click on Arduino IDE icon on the desktop

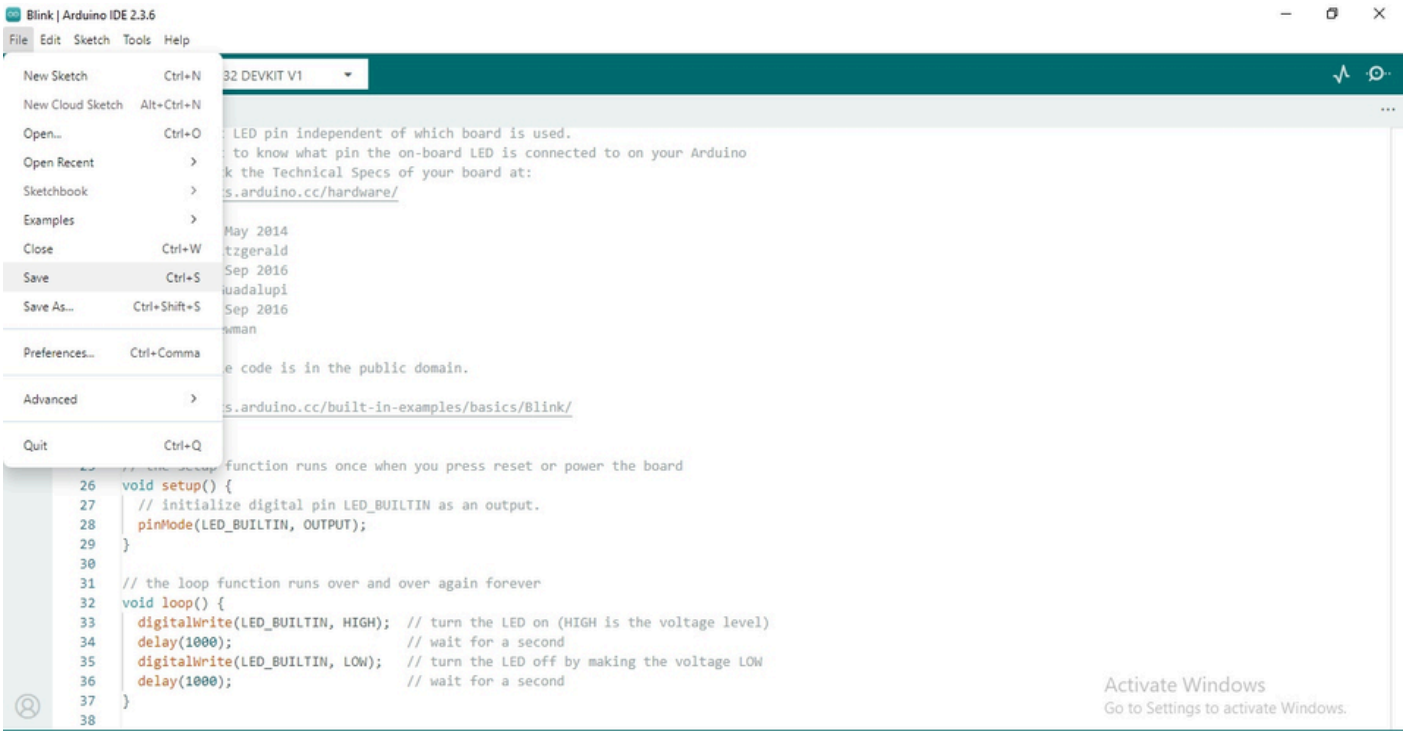


2. To create new project. Select File -> New Sketch



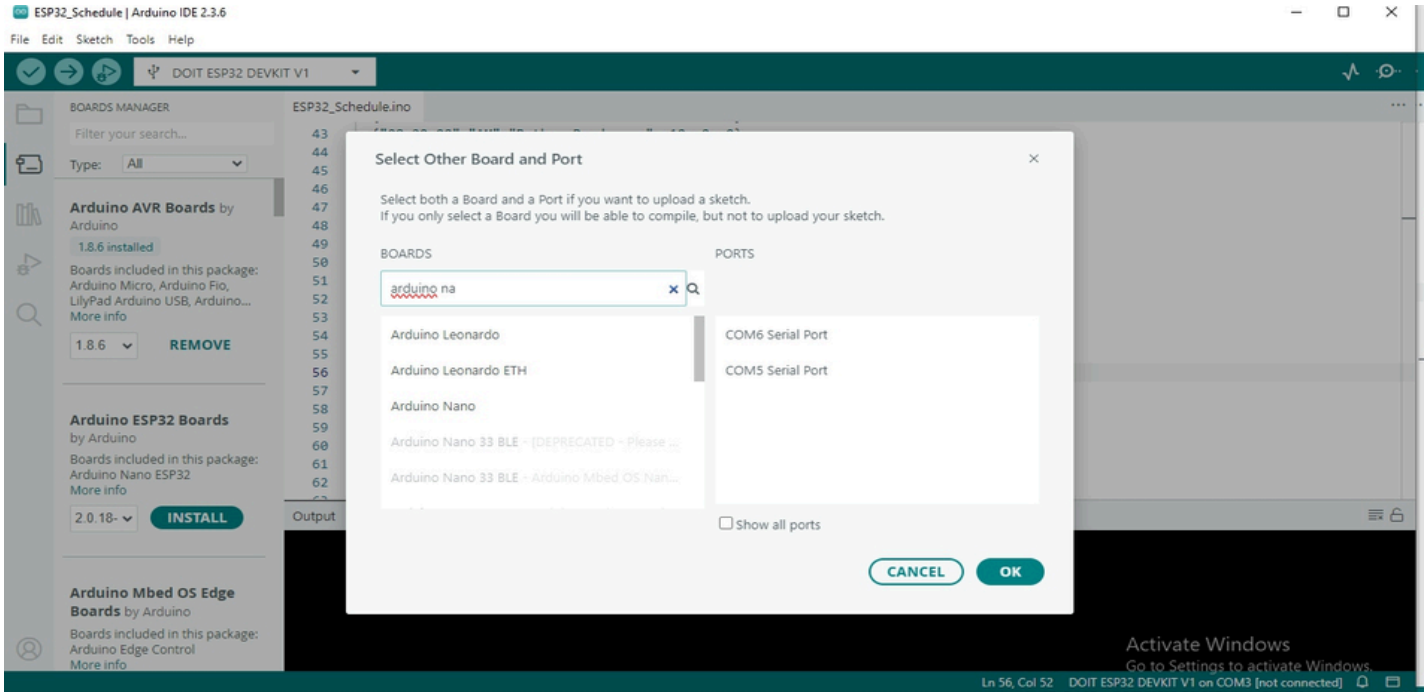
3. Save the sketch

Menu → File → Save

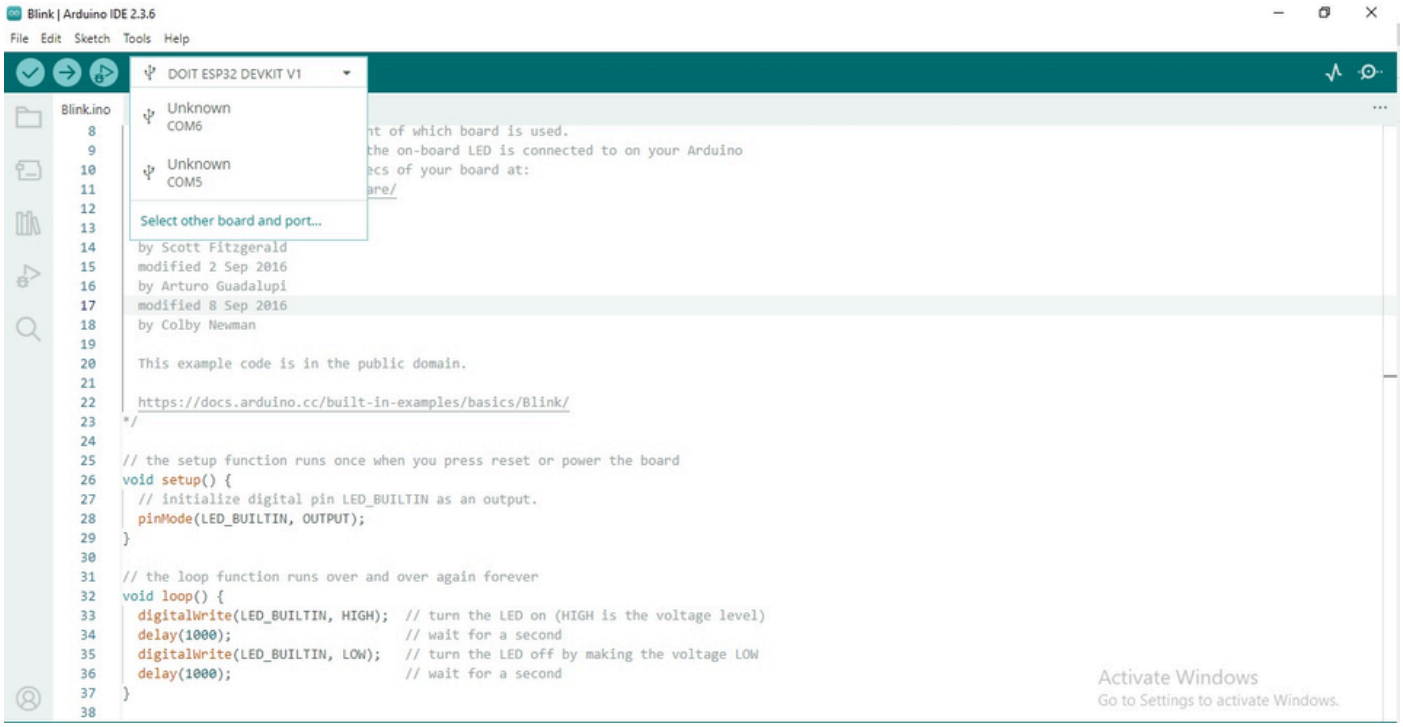


4. Select the board

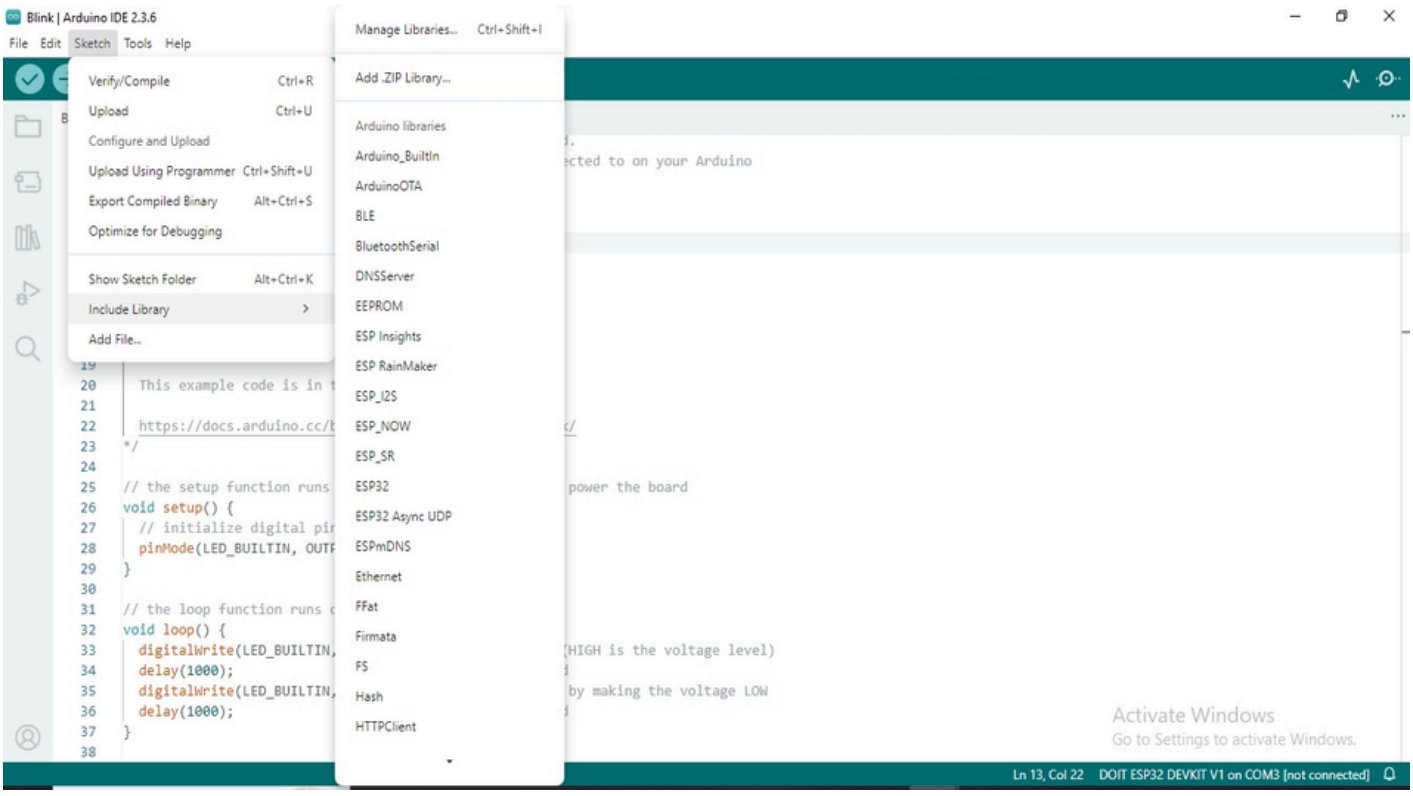
Tools → Board → {Your board type, e.g. Arduino Uno}.



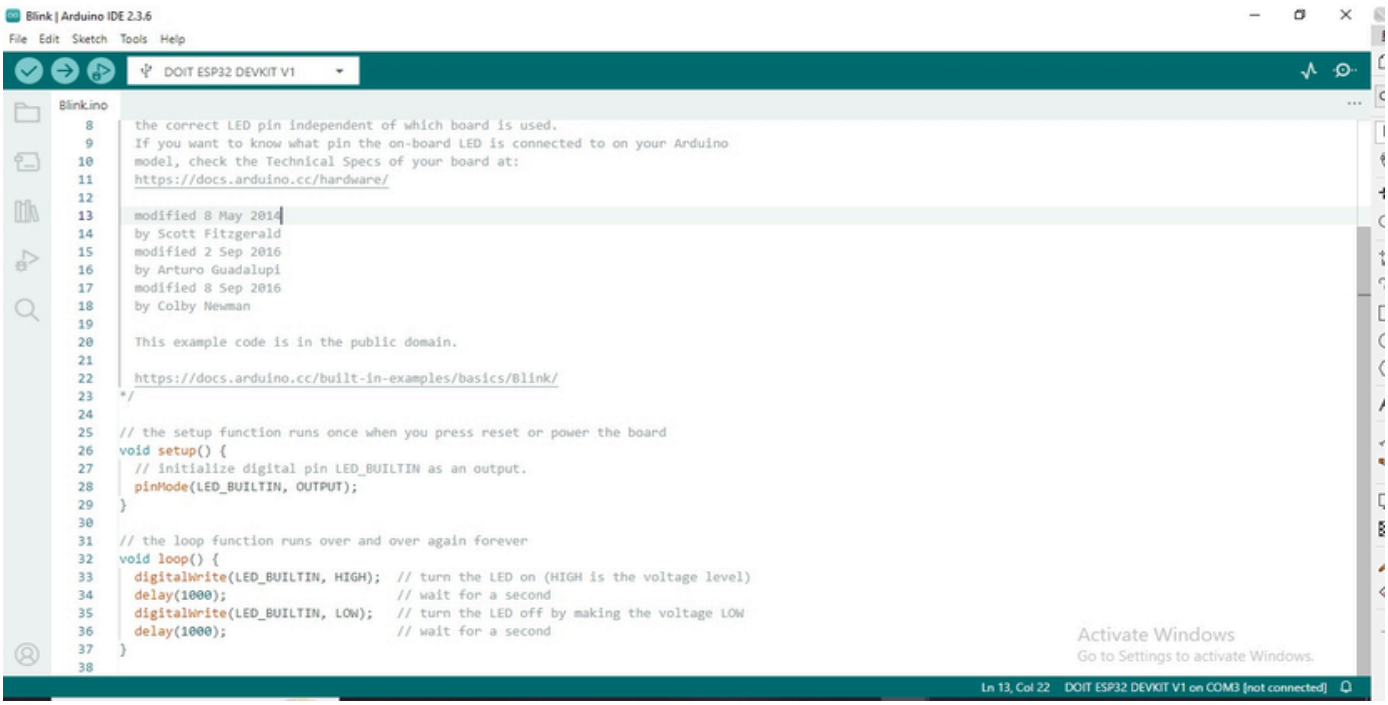
5.Select the port
Tools → Port → COMx



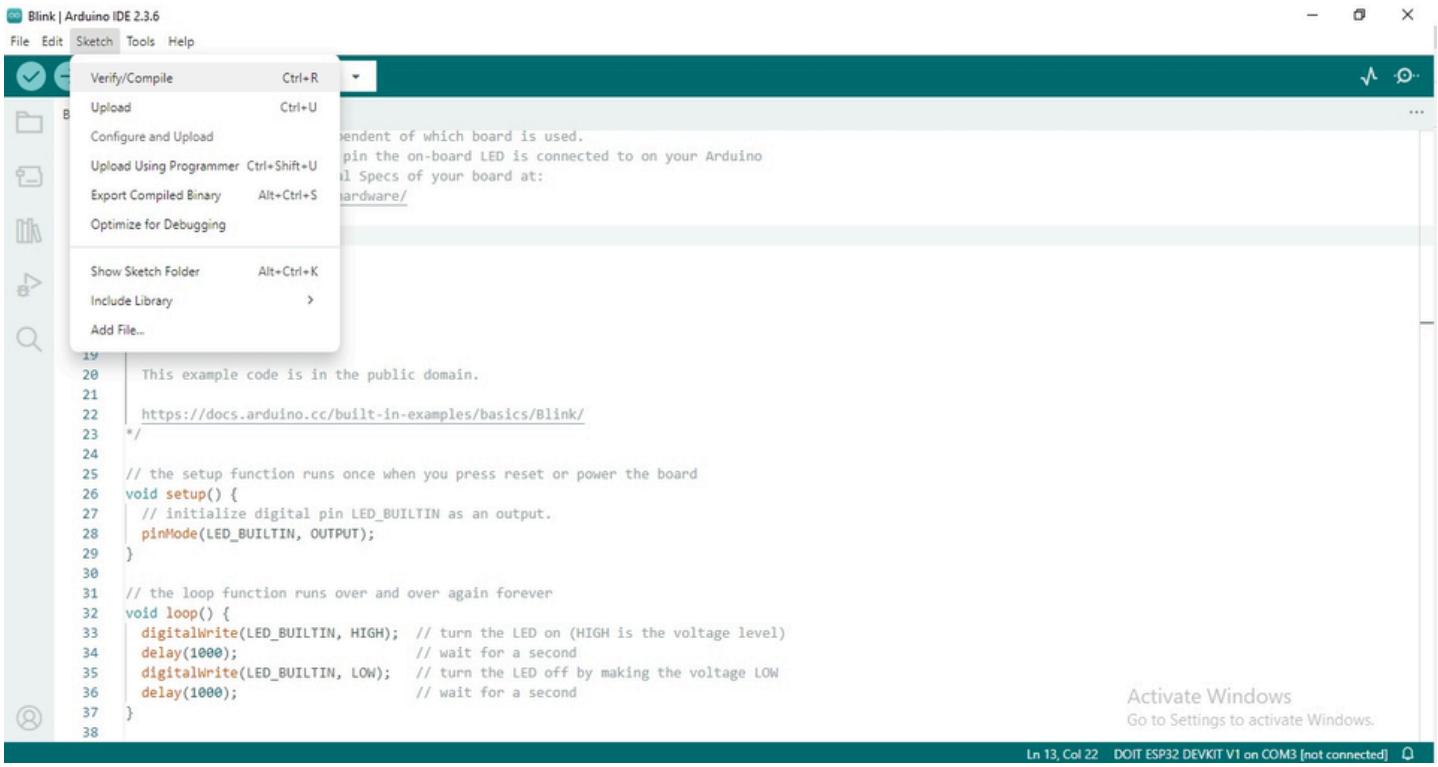
6.Install/Include libraries (if needed) Sketch → Include Library → Manage Libraries



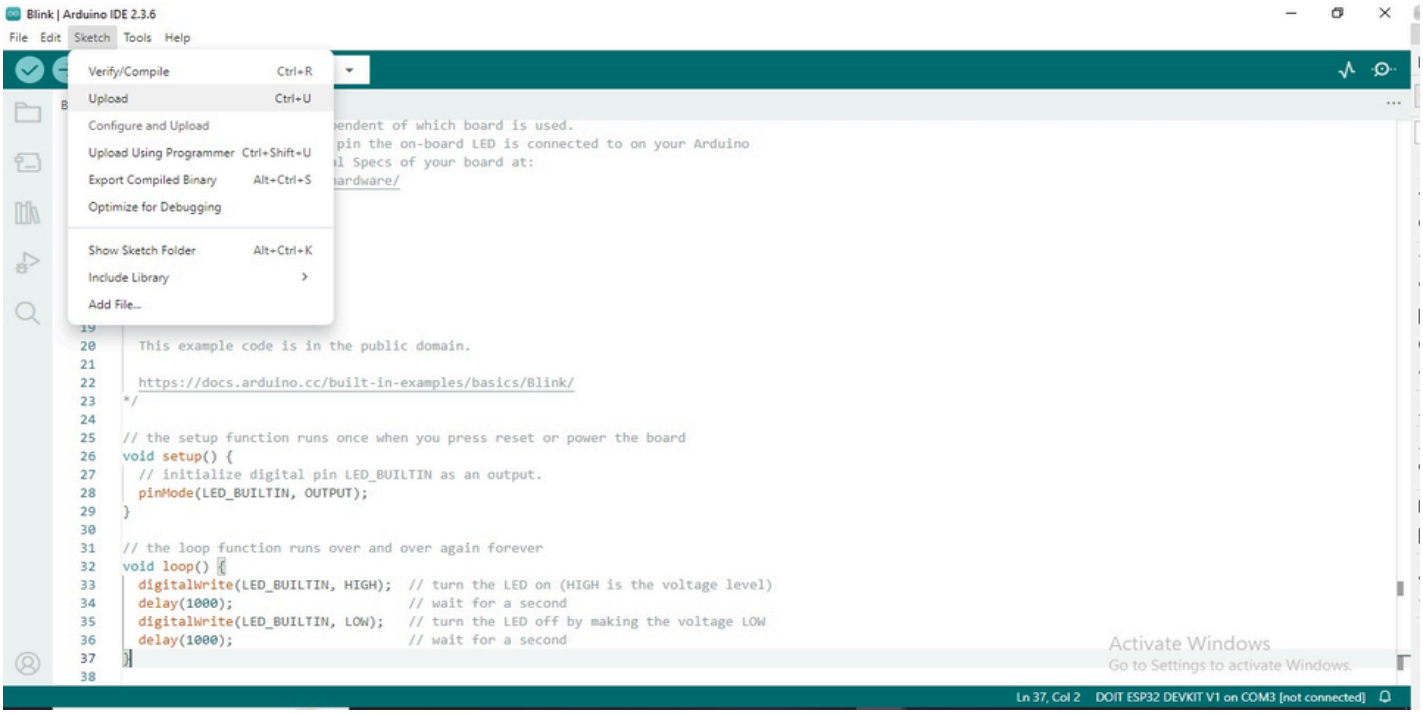
7. Write or paste code



8.Verify (compile)

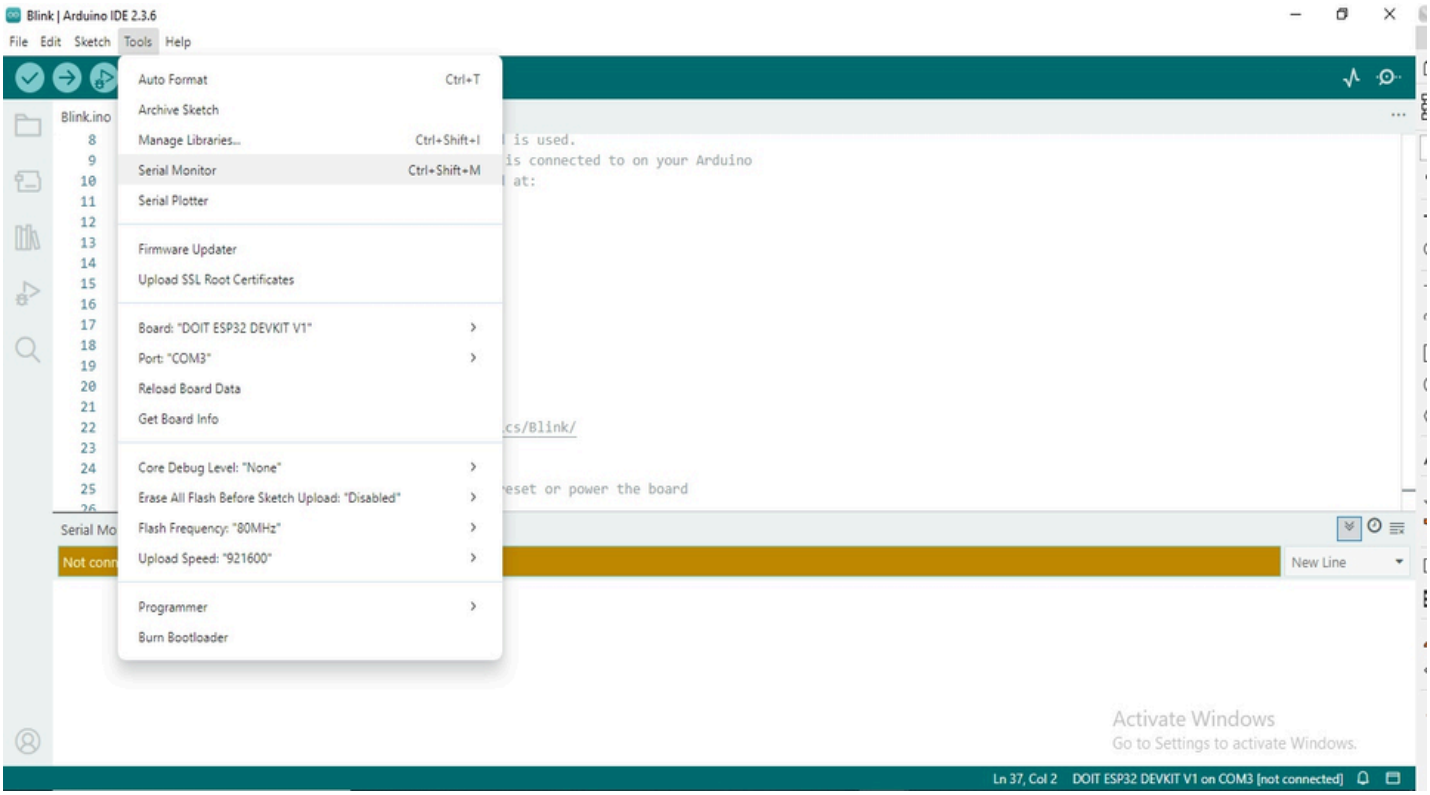


9.Upload to board

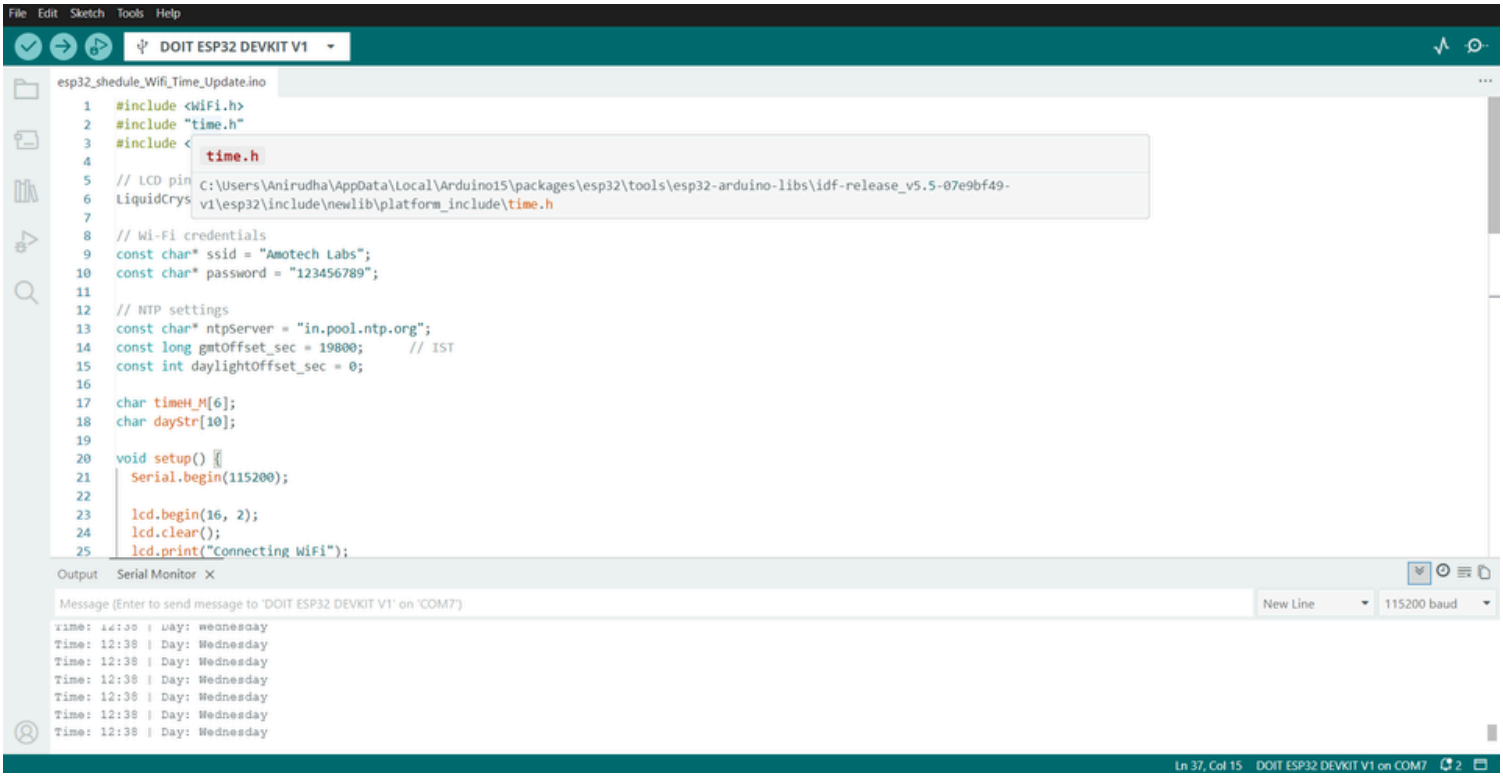


10.Open Serial Monitor

Tools → Serial Monitor or press Ctrl+Shi +M.



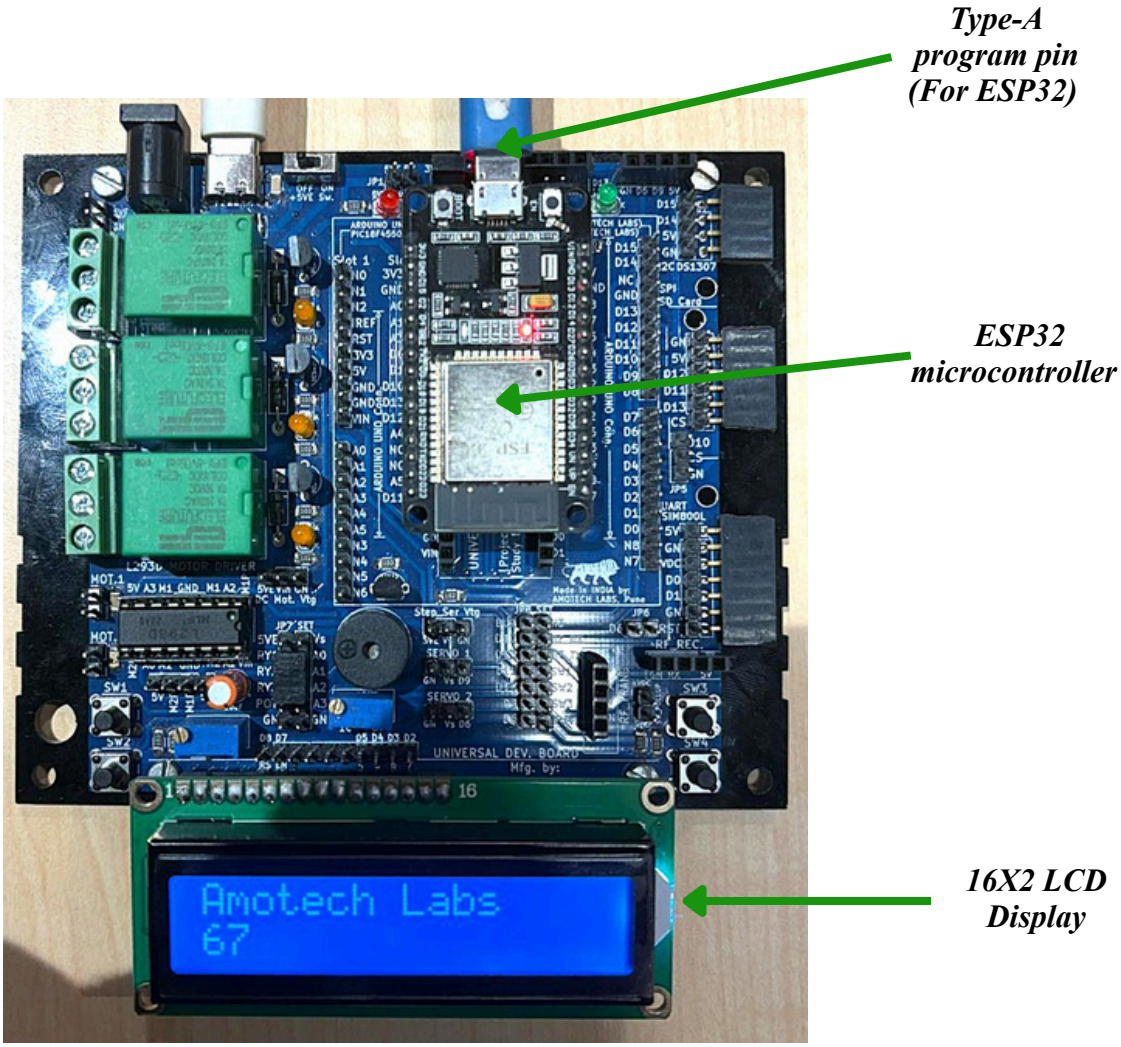
11. Below is the Final setup Example picture



Experiment 1
LCD Interfacing

Aim: LCD Interfacing with ESP32 using UDB.

Circuit Dig:



Procedure:

1. Open the Arduino IDE and follow the procedure mention in above section “Steps for use Arduino XIDE Software”.
2. Connect the power supply to board i.e. 3.3v.
3. Make all necessary connections as per above diagram.
4. Write and Upload the program into the ESP32 using Type-A cable.
5. Observe the output

Program:

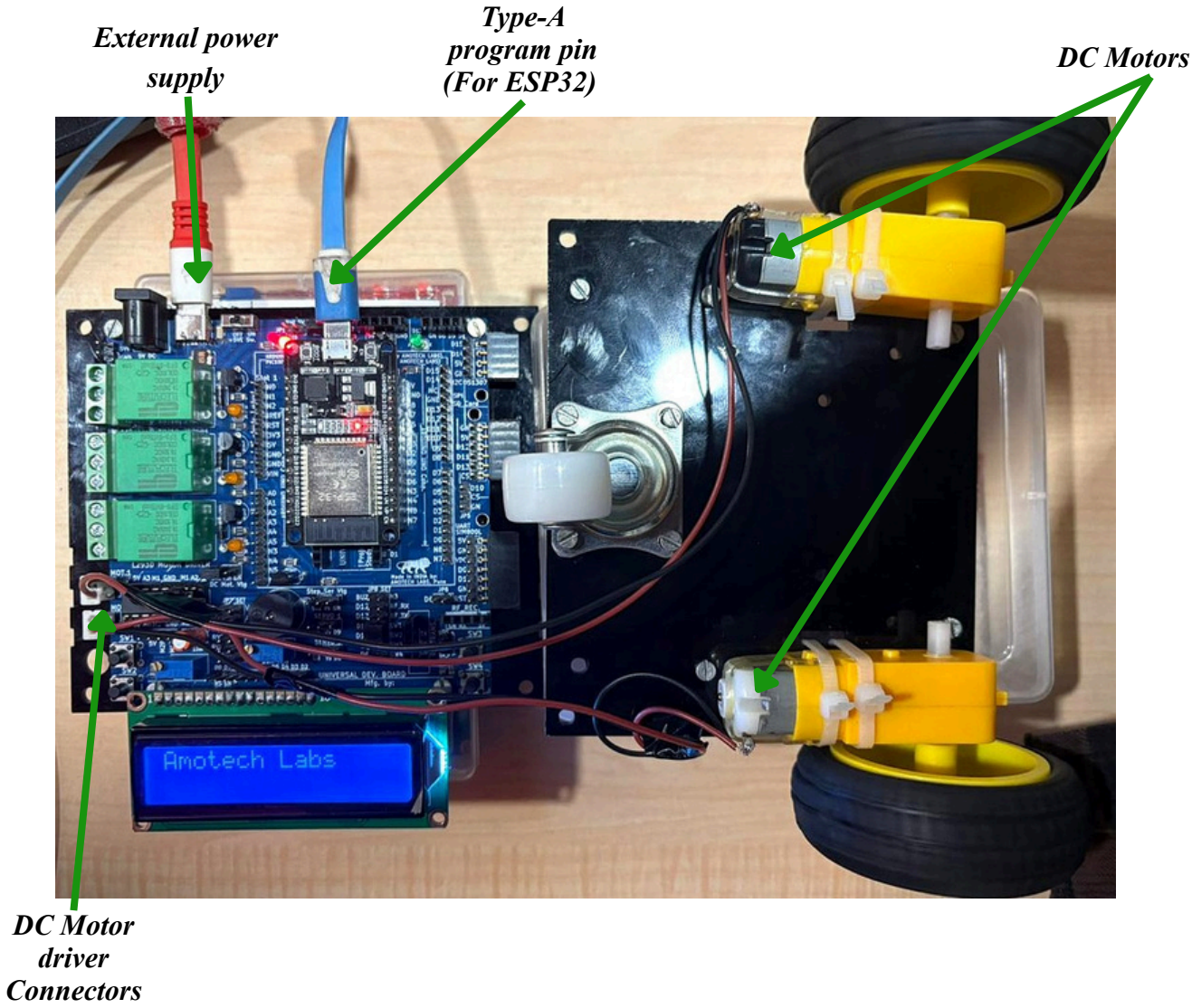
```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Amotech Labs");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Experiment 2

DC Motor Interfacing

Aim: To study the interfacing DC Motor Control using ESP32 with UDB and observe the output.

Circuit Dig:



Procedure:

1. Connect the DC motor to the motor driver module as shown in the circuit diagram.
2. Connect the motor driver control pins to the appropriate GPIO pins of the ESP32 Dev Kit.
3. Provide external power supply to the motor driver as required by the DC motor rating.
4. Connect the ESP32 Dev Kit to the computer using a USB cable and open the Arduino IDE.
5. Select ESP32 Dev Module and the correct COM port from the Tools menu.
6. Open the DC motor control program.
7. Compile and upload the program to the ESP32 Dev Kit.
8. Observe the DC motor operation after successful upload.
9. Verify the direction of rotation and speed control.

Program:

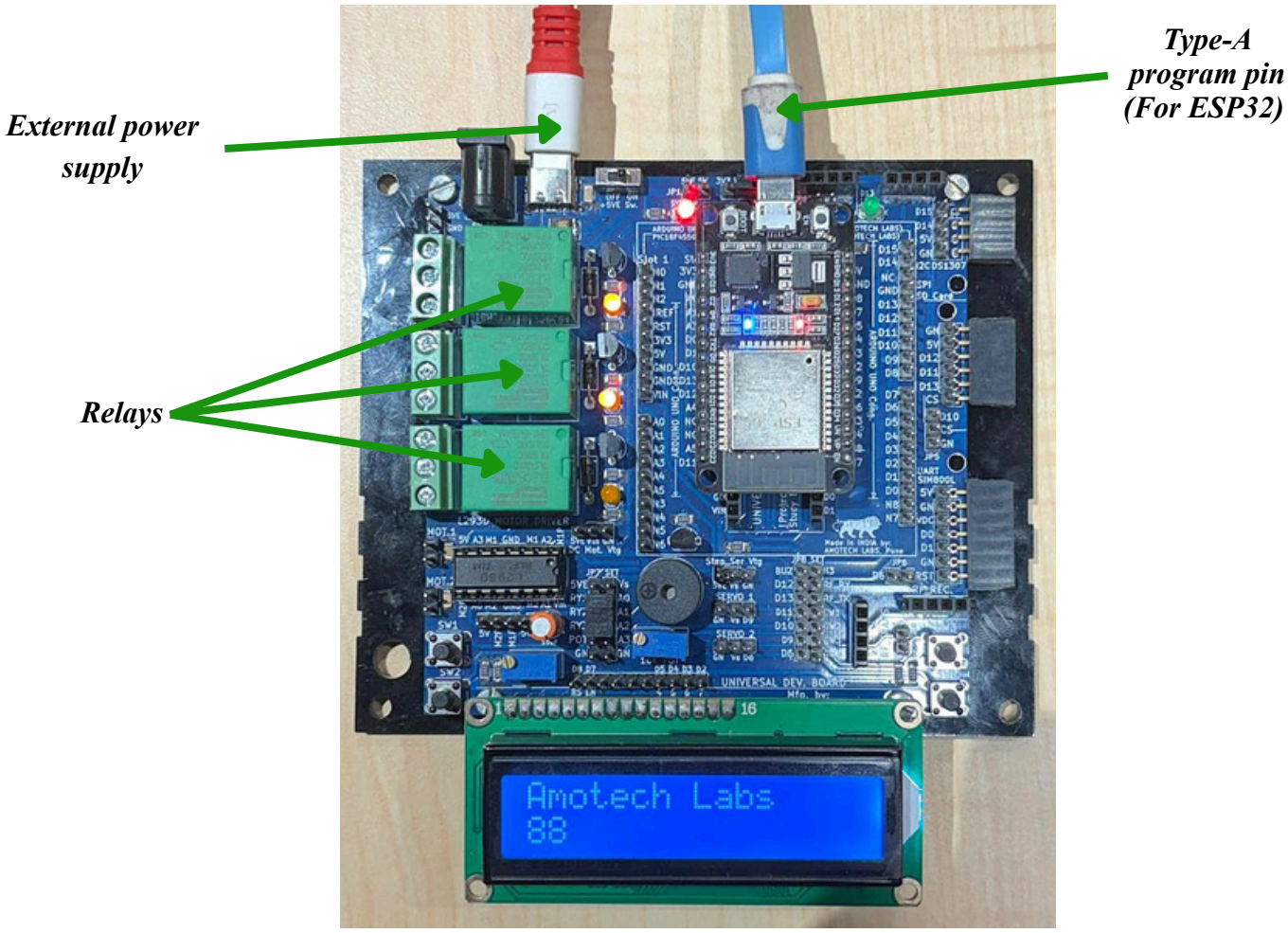
```
const int POT_input = A5;   /* assign ADC Channel */
bool d1 = HIGH;
bool d2 = LOW;
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

// set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Amotech Labs");

  pinMode(15, OUTPUT);  /* Motor 2 control pin 1 -  Marked as A0 on UDB*/
  pinMode(2, OUTPUT);  /* Motor 2 control pin 2 - Marked as A1 on UDB*/
  pinMode(32, OUTPUT);  /* Motor 1 control pin 1 - Marked as A2 on UDB */
  pinMode(4, OUTPUT);  /* Motor 1 control pin 1  - Marked as A3 on UDB*/
 // attachInterrupt(digitalPinToInterrupt(11), motor, RISING);  /* Interrupt on falling edge on pin 2 */
}

void loop() {
  int pwm_adc;
 // pwm_adc = analogRead(POT_input); /* Input from Potentiometer for speed control */
  digitalWrite(15,HIGH);   /* Rotate Motor 2 in Anti Clockwise */
  digitalWrite(2,LOW);

  digitalWrite(32,HIGH);   /* Rotate Motor 1 in Anti Clockwise */
  digitalWrite(4,LOW);

  delay(5000);

  digitalWrite(15,LOW);    /* Rotate Motor 2 in Clockwise */
  digitalWrite(2,HIGH);

  digitalWrite(32,LOW);   /* Rotate Motor 1 in Clockwise */
  digitalWrite(4,HIGH);
  delay(5000);
 // analogWrite(3, pwm_adc / 4);
}
/*
void motor(){
  d1 = !d1;
  d2 = !d2;
  _delay_ms(200);
}*/
```

Experiment 3

Relay control Interfacing

Aim: To study the Relay Control using ESP32 with UDB and observe the output.

Circuit Dig:



Procedure:

1. Connect the relay module to the Universal Development Board (UDB) as shown in the circuit diagram.
2. Connect the relay control input pins to the corresponding GPIO pins of the ESP32 Dev Kit using jumpers.
3. Provide the required external power supply to the relay module.
4. Connect the ESP32 Dev Kit to the computer using a USB cable (Type-A) and open the Arduino IDE.
5. Select ESP32 Dev Module and the correct COM port from the Tools menu.
6. Open the relay control program.
7. Compile and upload the program to the ESP32 Dev Kit.
8. Observe the relay ON/OFF operation.

Program:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
//int Relay1 = 15;    //D0
//int Relay2 = 2;    //D0
//int Relay3 = 32;     //D2

int Relay1 = 15;    //D0
int Relay2 = 2;    //D0
int Relay3 = 32;     //D2

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  pinMode(Relay3, OUTPUT);

  digitalWrite(Relay1,LOW);
  digitalWrite(Relay2,LOW);
  digitalWrite(Relay3,LOW);

}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
  digitalWrite(Relay1,HIGH);
  digitalWrite(Relay2,LOW);
  digitalWrite(Relay3,HIGH);
```

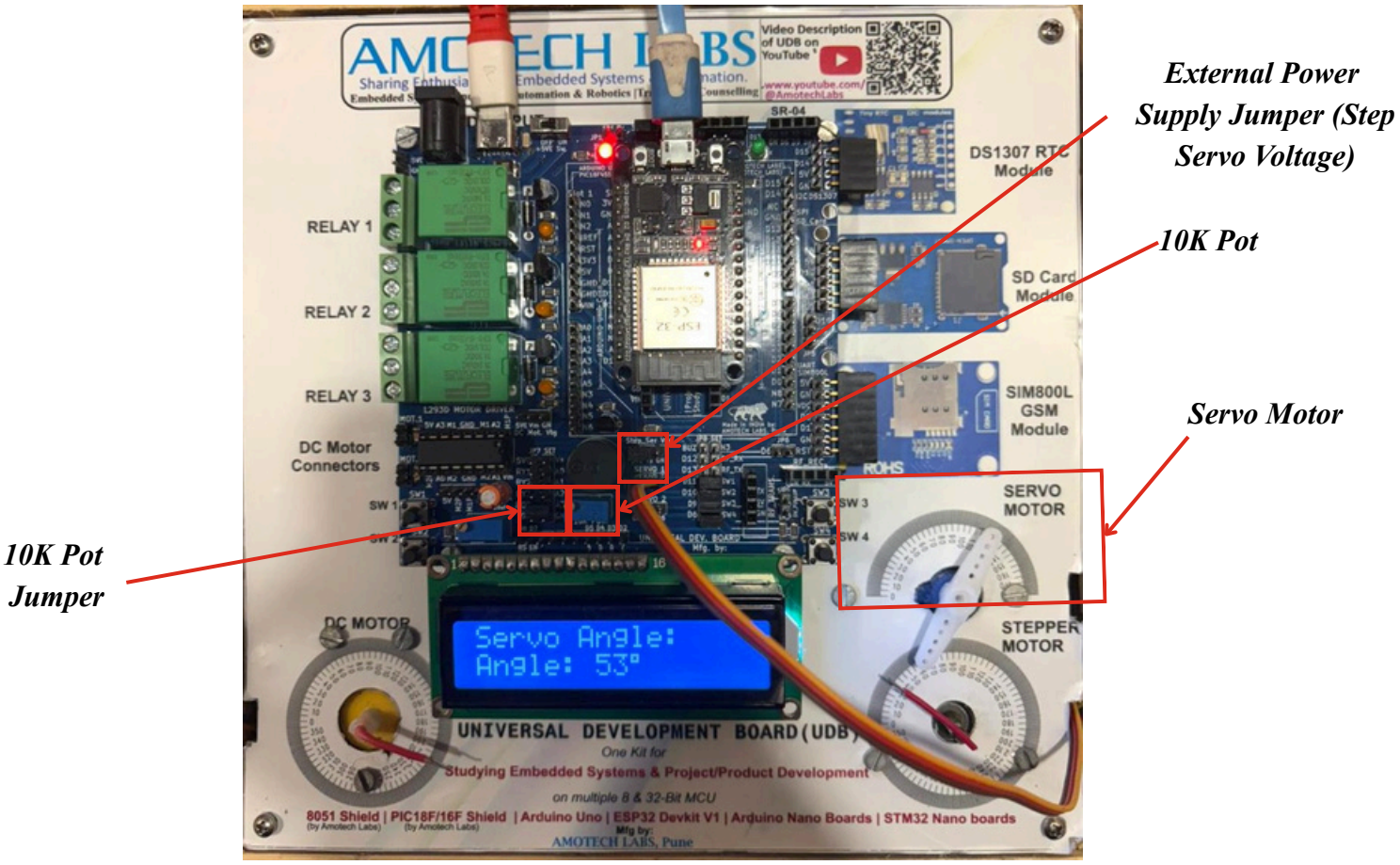
```
  delay(2000);
  digitalWrite(Relay1,LOW);
  digitalWrite(Relay2,HIGH);
  digitalWrite(Relay3,LOW);

  delay(2000);
  digitalWrite(Relay1,HIGH);
  digitalWrite(Relay2,HIGH);
  digitalWrite(Relay3,LOW);
  delay(2000);
  digitalWrite(Relay1,LOW);
  digitalWrite(Relay2,LOW);
  digitalWrite(Relay3,LOW);
  delay(2000);
}
```

Experiment 4

Servo Motor Interfacing

Aim: To study the interfacing Servo Motor using ESP32 with UDB and observe the output.

Circuit Dig:



Procedure:

1. Connect the servo motor signal pin to a suitable GPIO pin of the ESP32 Dev Kit as shown in the circuit diagram.
2. Provide an external 5 V power supply to the servo motor and connect the ground to ESP32 GND.
3. Connect the ESP32 Dev Kit to the computer using a USB cable and open the Arduino IDE.
4. Select ESP32 Dev Module and the correct COM port from the Tools menu.
5. Open the servo motor control program.
6. Compile and upload the program to the ESP32 Dev Kit.
7. Observe the servo motor movement after successful upload.
8. Verify the angular position and direction of rotation.

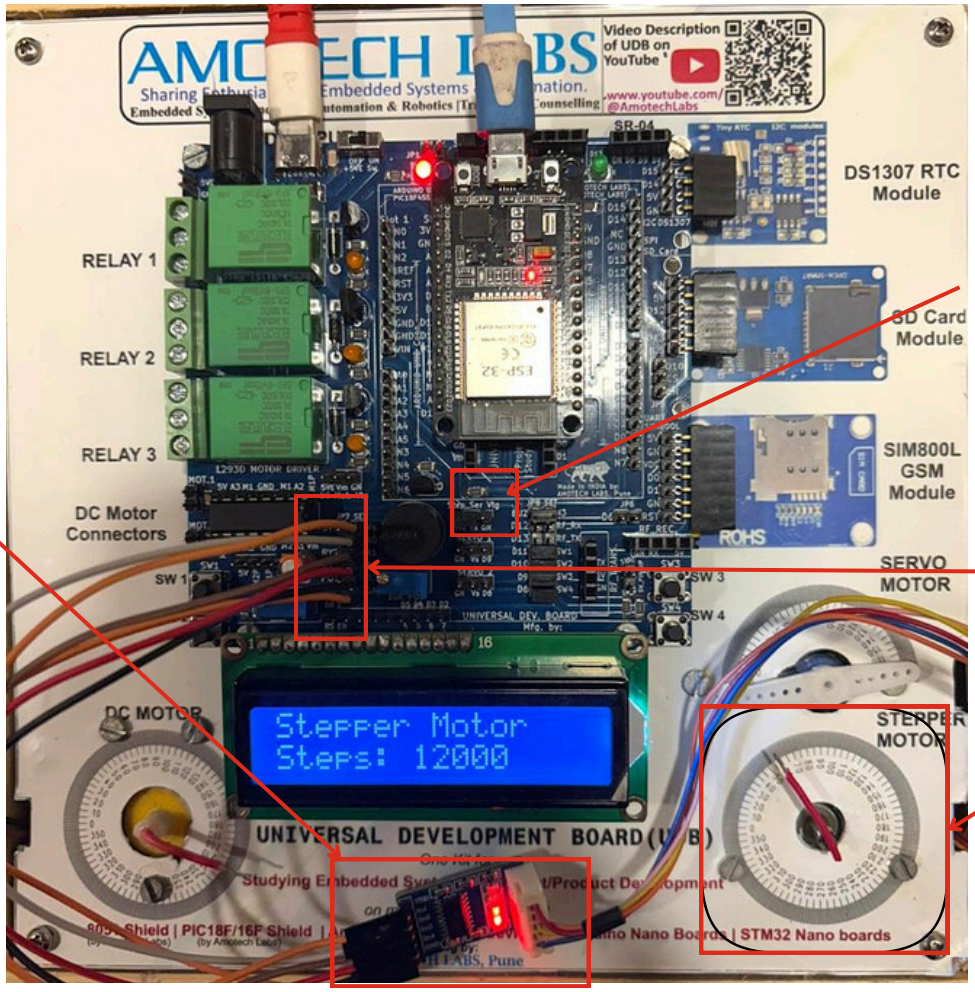
Program:

```
#include <ESP32Servo.h>
#include <LiquidCrystal.h>

/* ---------- LCD ---------- */
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
long stepCount = 0;

Servo myservo;  /* create servo object to control a servo */

int potpin = 4;  /* analog pin used to connect the potentiometer */
int val;    /*  variable to read the value from the analog pin */

void setup() {
  Serial.begin(9600);
  myservo.attach(33);  /* attaches the servo on pin 9 to the servo object */

  // -------- LCD INIT (ADDED) --------
  lcd.begin(16, 2);
  lcd.print("Servo Angle:");
}

void loop() {
  val = analogRead(potpin);            /* reads the value of the potentiometer (value between 0 and 1023) */
  Serial.print("Analog Value : ");
  Serial.print(val);
  Serial.print("\n");

  val = map(val, 0, 1023, 0, 180);     /* scale it to use it with the servo (value between 0 and 180) */
  Serial.print("Mapped Value : ");
  Serial.print(val);
  Serial.print("\n\n");

  myservo.write(val);                  /* sets the servo position according to the scaled value */

  // -------- LCD UPDATE (ADDED) --------
  lcd.setCursor(0, 1);
  lcd.print("Angle: ");
  lcd.print(val);
  lcd.print((char)223);   // degree symbol
  lcd.print("   ");       // clear leftover characters

  delay(200);                          /* waits for the servo to get there */
}

```

Experiment 5
Stepper Motor Interfacing

Aim: To study the interfacing Stepper Motor Control using ESP32 with UDB and observe output

Circuit Dig:



Procedure:

1. Connect the stepper motor to the stepper motor driver module as shown in the circuit diagram.
2. Connect the driver control pins to the appropriate GPIO pins of the ESP32 Dev Kit.
3. Provide the required external power supply to the stepper motor driver.
4. Connect the ESP32 Dev Kit to the computer using a USB cable and open the Arduino IDE.
5. Select ESP32 Dev Module and the correct COM port from the Tools menu.
6. Open the stepper motor control program.
7. Compile and upload the program to the ESP32 Dev Kit.
8. Observe the stepper motor rotation after successful upload.
9. Verify the direction of rotation and step movement.

Program:

```
#include <LiquidCrystal.h>

/* ---------- LCD ---------- */
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
long stepCount = 0;

/*
#define 4 - 15
        5   2
        6   32
        7   4
*/

void setup() {
  pinMode(15, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(32, OUTPUT);
  pinMode(4, OUTPUT);

  // LCD init (ADDED)

  lcd.begin(16, 2);
  lcd.print("Stepper Motor");
}

void loop() {

  /* Rotation in one direction */
  for(int i = 0; i < 100; i++)
  {
    digitalWrite(4, HIGH);
    digitalWrite(32, LOW);
    digitalWrite(2, LOW);
    digitalWrite(15, LOW);
    delay(10);
    stepCount++;

    digitalWrite(4, HIGH);
    digitalWrite(32, HIGH);
    digitalWrite(2, LOW);
    digitalWrite(15, LOW);
    delay(10);
    stepCount++;

```

```
47      digitalWrite(4, LOW);
48      digitalWrite(32, HIGH);
49      digitalWrite(2, LOW);
50      digitalWrite(15, LOW);
51      delay(10);
52      stepCount++;
53
54      digitalWrite(4, LOW);
55      digitalWrite(32, HIGH);
56      digitalWrite(2, HIGH);
57      digitalWrite(15, LOW);
58      delay(10);
59      stepCount++;
60
61      digitalWrite(4, LOW);
62      digitalWrite(32, LOW);
63      digitalWrite(2, HIGH);
64      digitalWrite(15, LOW);
65      delay(10);
66      stepCount++;
67
68      digitalWrite(4, LOW);
69      digitalWrite(32, LOW);
70      digitalWrite(2, HIGH);
71      digitalWrite(15, HIGH);
72      delay(10);
73      stepCount++;
74
75      digitalWrite(4, LOW);
76      digitalWrite(32, LOW);
77      digitalWrite(2, LOW);
78      digitalWrite(15, HIGH);
79      delay(10);
80      stepCount++;
81
82      digitalWrite(4, HIGH);
83      digitalWrite(32, LOW);
84      digitalWrite(2, LOW);
85      digitalWrite(15, HIGH);
86      delay(10);
87      stepCount++;
88    }
89
90    lcd.setCursor(0, 1);
91    lcd.print("Steps: ");
92    lcd.print(stepCount);
93    lcd.print("    ");
94
95    digitalWrite(4, HIGH);
96    digitalWrite(32, LOW);
97    digitalWrite(2, LOW);
98    digitalWrite(15, LOW);
99    delay(10);
```

```
100
101   /* Rotation in opposite direction */
102   for(int j = 0; j < 100; j++)
103   {
104     digitalWrite(4, LOW);
105     digitalWrite(32, LOW);
106     digitalWrite(2, LOW);
107     digitalWrite(15, HIGH);
108     delay(10);
109     stepCount++;
110
111     digitalWrite(4, LOW);
112     digitalWrite(32, LOW);
113     digitalWrite(2, HIGH);
114     digitalWrite(15, HIGH);
115     delay(10);
116     stepCount++;
117
118     digitalWrite(4, LOW);
119     digitalWrite(32, LOW);
120     digitalWrite(2, HIGH);
121     digitalWrite(15, LOW);
122     delay(10);
123     stepCount++;
124
125     digitalWrite(4, LOW);
126     digitalWrite(32, HIGH);
127     digitalWrite(2, HIGH);
128     digitalWrite(15, LOW);
129     delay(10);
130     stepCount++;
131
132     digitalWrite(4, LOW);
133     digitalWrite(32, HIGH);
134     digitalWrite(2, LOW);
135     digitalWrite(15, LOW);
136     delay(10);
137     stepCount++;
138
139     digitalWrite(4, HIGH);
140     digitalWrite(32, HIGH);
141     digitalWrite(2, LOW);
142     digitalWrite(15, LOW);
143     delay(10);
144     stepCount++;
145
146     digitalWrite(4, HIGH);
147     digitalWrite(32, LOW);
148     digitalWrite(2, LOW);
149     digitalWrite(15, LOW);
```

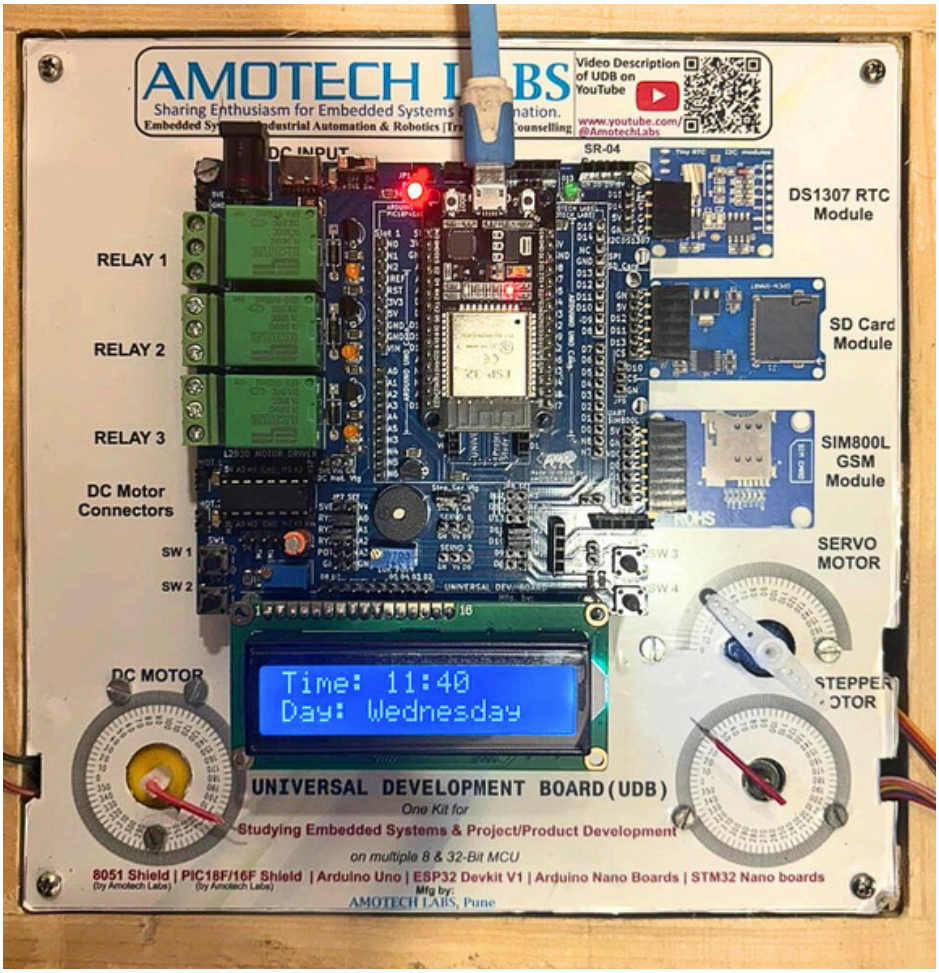
```
150       delay(10);
151       stepCount++;
152
153       digitalWrite(4, HIGH);
154       digitalWrite(32, LOW);
155       digitalWrite(2, LOW);
156       digitalWrite(15, HIGH);
157       delay(10);
158       stepCount++;
159     }
160
161     lcd.setCursor(0, 1);
162     lcd.print("Steps: ");
163     lcd.print(stepCount);
164     lcd.print("    ");
165
166     digitalWrite(4, LOW);
167     digitalWrite(32, LOW);
168     digitalWrite(2, LOW);
169     digitalWrite(15, HIGH);
170     delay(10);
171   }
172
```

Experiment 6

Wifi Time Update Interfacing

Aim: To study the Wifi time and Day update interfacing with ESP32 dev-kit and observe the output.

Circuit Dig:



Procedure:

1. Connect the ESP32 Dev Kit to the computer using a USB cable.
2. Open the Arduino IDE and select ESP32 Dev Module and the correct COM port from the Tools menu.
3. Open the program for Wi-Fi–based time synchronization (NTP).
4. Enter the Wi-Fi network SSID and password in the program.
5. Configure the time server and update interval in the code.
6. Compile and upload the program to the ESP32 Dev Kit.
7. Open the Serial Monitor after successful upload.
8. Observe the current date and time updated at scheduled intervals.
9. Verify that time is correctly synchronized via Wi-Fi.

Program:

```
#include <WiFi.h>
#include "time.h"
#include <LiquidCrystal.h>

// LCD pin configuration
LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

// Wi-Fi credentials
const char* ssid = "Amotech Labs";
const char* password = "123456789";

// NTP settings
const char* ntpServer = "in.pool.ntp.org";
const long gmtOffset_sec = 19800;      // IST
const int daylightOffset_sec = 0;

char timeH_M[6];
char dayStr[10];

void setup() {
  Serial.begin(115200);

  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Connecting WiFi");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
    lcd.setCursor(0, 1);
    lcd.print("Please wait...");
  }

  Serial.println("WiFi connected");
  lcd.clear();
  lcd.print("WiFi Connected");
  delay(1000);
  lcd.clear();

  // Initialize NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

```

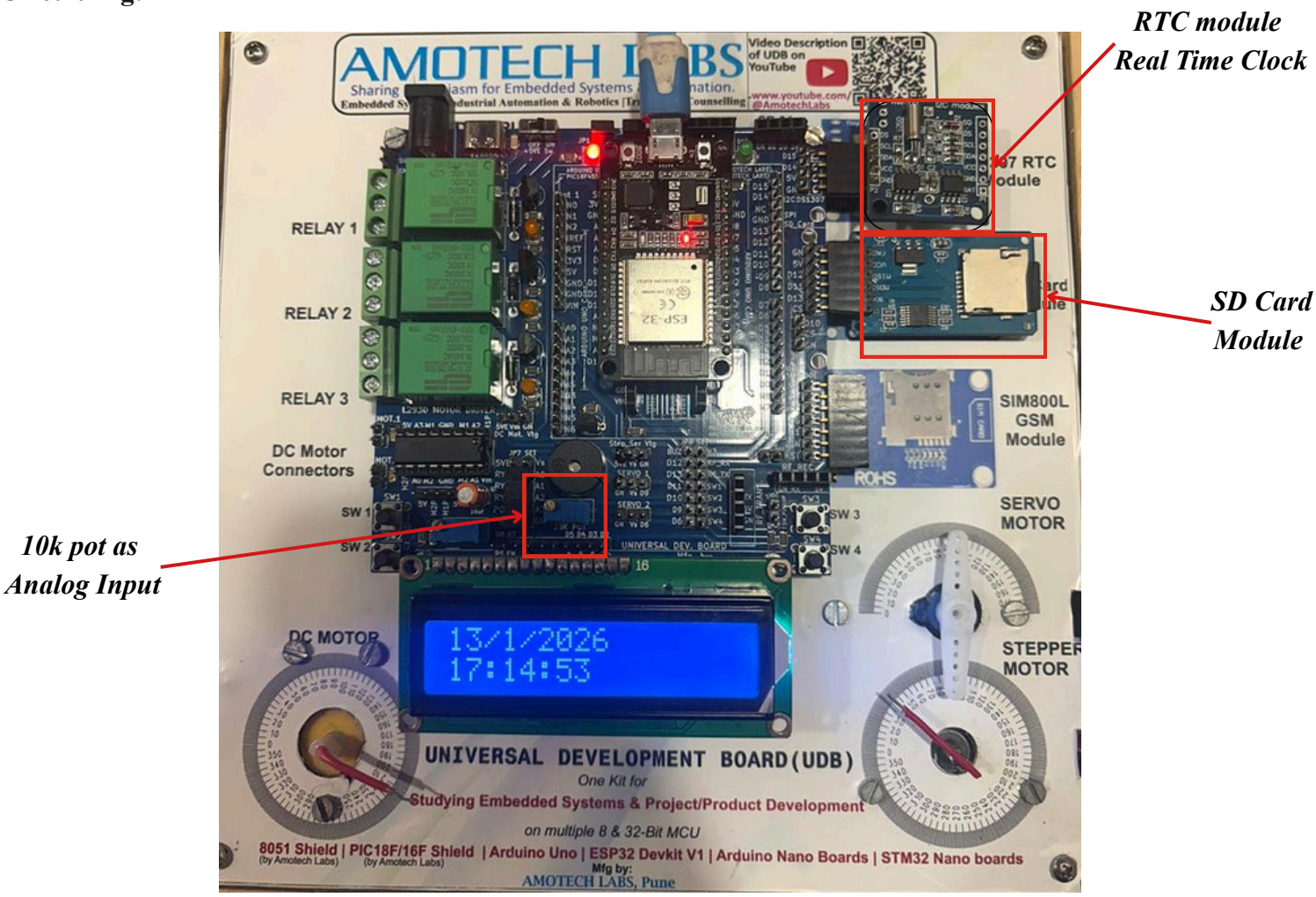
```
46  void loop() {
47    printLocalTime();
48    delay(1000);   // update every second
49  }
50
51  void printLocalTime() {
52    struct tm timeinfo;
53
54    if (!getLocalTime(&timeinfo)) {
55      Serial.println("Failed to get time");
56      lcd.clear();
57      lcd.print("Time Error");
58      return;
59    }
60
61    // Format time HH:MM
62    strftime(timeH_M, sizeof(timeH_M), "%H:%M", &timeinfo);
63
64    // Format day name (Monday, Tuesday, etc.)
65    strftime(dayStr, sizeof(dayStr), "%A", &timeinfo);
66
67    // Display on LCD
68    lcd.clear();
69    lcd.setCursor(0, 0);
70    lcd.print("Time: ");
71    lcd.print(timeH_M);
72
73    lcd.setCursor(0, 1);
74    lcd.print("Day: ");
75    lcd.print(dayStr);
76
77    // Serial debug
78    Serial.print("Time: ");
79    Serial.print(timeH_M);
80    Serial.print(" | Day: ");
81    Serial.println(dayStr);
82  }
83
```

Experiment 7

RTC, SD card, ADC Interfacing

Aim: To Study Real-Time Data Logging Using RTC, SD Card, and 10k pot for Analog Input Interfaced with ESP32 Using UDB.

Circuit Dig:



Procedure:

1. Connect the RTC module, SD card module, and 10 kΩ potentiometer to the Universal Development Board (UDB) as shown in the circuit diagram.
2. Connect the ESP32 Dev Kit to the UDB and provide the required power supply.
3. Connect the ESP32 Dev Kit to the computer using a USB cable and open the Arduino IDE.
4. Select ESP32 Dev Module and the correct COM port from the Tools menu.
5. Open the program for RTC, SD card, and analog input interfacing.
6. Compile and upload the program to the ESP32 Dev Kit.
7. After successful upload, rotate the 10 kΩ potentiometer and open the Serial Monitor.
8. Observe the real-time clock data and corresponding analog input values.
9. Verify that the data is correctly logged into the SD card.

Program:

```
/*
 Aim:
 To study real-time data logging using RTC, SD card,
 and 10k potentiometer for analog input using ESP32 Dev Kit
*/

#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <RTClib.h>
#include <LiquidCrystal.h>

// ---------------- Pin Definitions ----------------
#define SD_CS       5        // SD card CS pin
#define POT_PIN     34       // ADC pin for 10k pot (ESP32)

// LCD pins: RS, EN, D4, D5, D6, D7
const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// ---------------- Objects ----------------
File myFile;
RTC_DS3231 rtc;

// ---------------- Setup ----------------
void setup() {
  Serial.begin(9600);
  delay(1000);

  // LCD initialization
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("System Ready");
  delay(1500);
  lcd.clear();

  // I2C for RTC
  Wire.begin();

  // Initialize RTC
  if (!rtc.begin()) {
    lcd.print("RTC Error");
    Serial.println("Couldn't find RTC");
    while (1);
  }

```

```
46
47    // Set RTC time if power was lost
48    if (rtc.lostPower()) {
49      rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
50    }
51
52    // Initialize SD card
53    lcd.print("SD Init...");
54    Serial.print("Initializing SD card...");
55    if (!SD.begin(SD_CS)) {
56      lcd.clear();
57      lcd.print("SD Failed");
58      Serial.println("Initialization failed!");
59      while (1);
60    }
61    Serial.println("Initialization done.");
62
63    lcd.clear();
64    lcd.print("SD Ready");
65
66    // Create file and write header
67    myFile = SD.open("DATA.txt", FILE_WRITE);
68    if (myFile) {
69      myFile.println("Date Time Analog_Value");
70      myFile.close();
71    }
72
73    delay(1000);
74    lcd.clear();
75  }
76
77  // ---------------- Log Time ----------------
78  void loggingTime() {
79    DateTime now = rtc.now();
80
81    myFile = SD.open("DATA.txt", FILE_WRITE);
82    if (myFile) {
83      myFile.print(now.year());
84      myFile.print('/');
85      myFile.print(now.month());
86      myFile.print('/');
87      myFile.print(now.day());
88      myFile.print(' ');
89      myFile.print(now.hour());
90      myFile.print(':');
91      myFile.print(now.minute());
```

```
    myFile.print(':');
    myFile.print(now.second());
    myFile.print("  ");
    myFile.close();
  }

  // LCD display - Date & Time
  lcd.setCursor(0, 0);
  lcd.print(now.day());
  lcd.print('/');
  lcd.print(now.month());
  lcd.print('/');
  lcd.print(now.year());

  lcd.setCursor(0, 1);
  lcd.print(now.hour());
  lcd.print(':');
  lcd.print(now.minute());
  lcd.print(':');
  lcd.print(now.second());

  // Serial Monitor
  Serial.print(now.year());
  Serial.print('/');
  Serial.print(now.month());
  Serial.print('/');
  Serial.println(now.day());
  Serial.print(now.hour());
  Serial.print(':');
  Serial.print(now.minute());
  Serial.print(':');
  Serial.println(now.second());
}

// ---------------- Log Analog Value ----------------
void loggingAnalog() {
  int analogValue = analogRead(POT_PIN);

  Serial.print("Analog Value: ");
  Serial.println(analogValue);

  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    myFile.println(analogValue);
```

```
135       myFile.println(analogValue);
136       Serial.println("open with success");
137       myFile.close();
138     }
139
140     // LCD display - Analog value
141     lcd.clear();
142     lcd.setCursor(0 , 0);
143     lcd.print("Analog Value");
144     lcd.setCursor(0, 1);
145     lcd.print(analogValue);
146   }
147
148   // ---------------- Loop ----------------
149   void loop() {
150     loggingTime();
151     delay(2000);
152
153     loggingAnalog();
154     delay(3000);   // Total logging interval = 5 seconds
155     lcd.clear();
156   }
```