Universal Development Board(UDB)

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

+

Arduino Nano / UNO

Reference Manual

AMOTECH LABS

Embedded systems | IndustrialAutomation | Robotics

Scan to download So Copy of Manual & Online Purchase

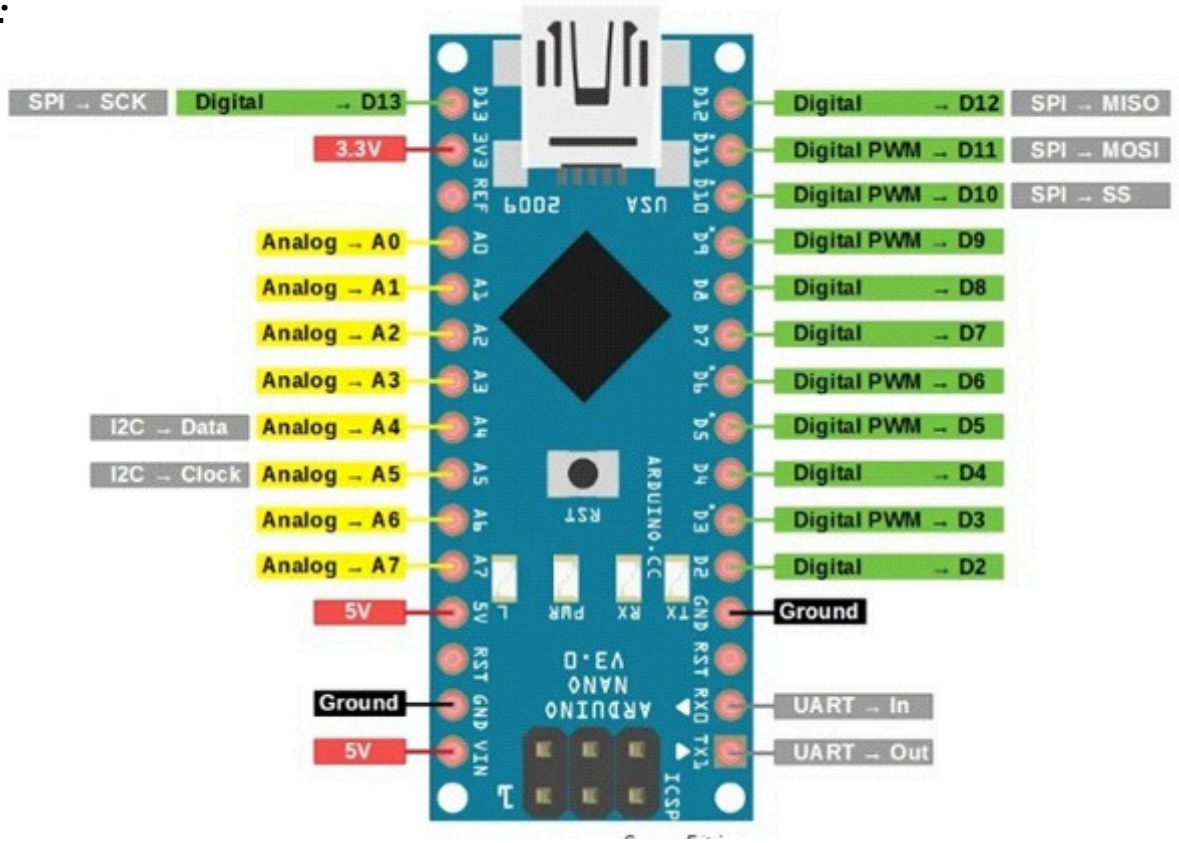
Scan for Video Tutorials

Table of Contents

Introduction

Embedded System Arduino Nano Microcontroller :

Overview :



Arduino Nano is a small, complete, and breadboard-friendly development board based on the ATmega328 (Arduino Nano 3.0) or ATmega168 (Arduino Nano 2.x). It has more or less the same functionality of the Arduino UNO, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The connectivity is the same as the Arduino UNO board Features ofArduinoNano

FEATURES	DESCRIPTION
Microcontroller Operating	ATmega328P
Voltage Input	5V
Voltage (Recommended) Input	7-12V
Voltage (Limit)	6-20V
Digital I/O Pins	14 (of which 6 can provide PWM output)
Analog Input Pins	8 (A0-A7)
DC Current per I/O Pin	40mA
Flash Memory	32KB (of which 2KB used by bootloader)
SRAM	2KB
EEPROM	1KB
Clock Speed	16 MHz
USB Interface	Mini-B USB (for programming and power)
Dimensions	45mm x 18mm

:

Universal Development Board(UDB)

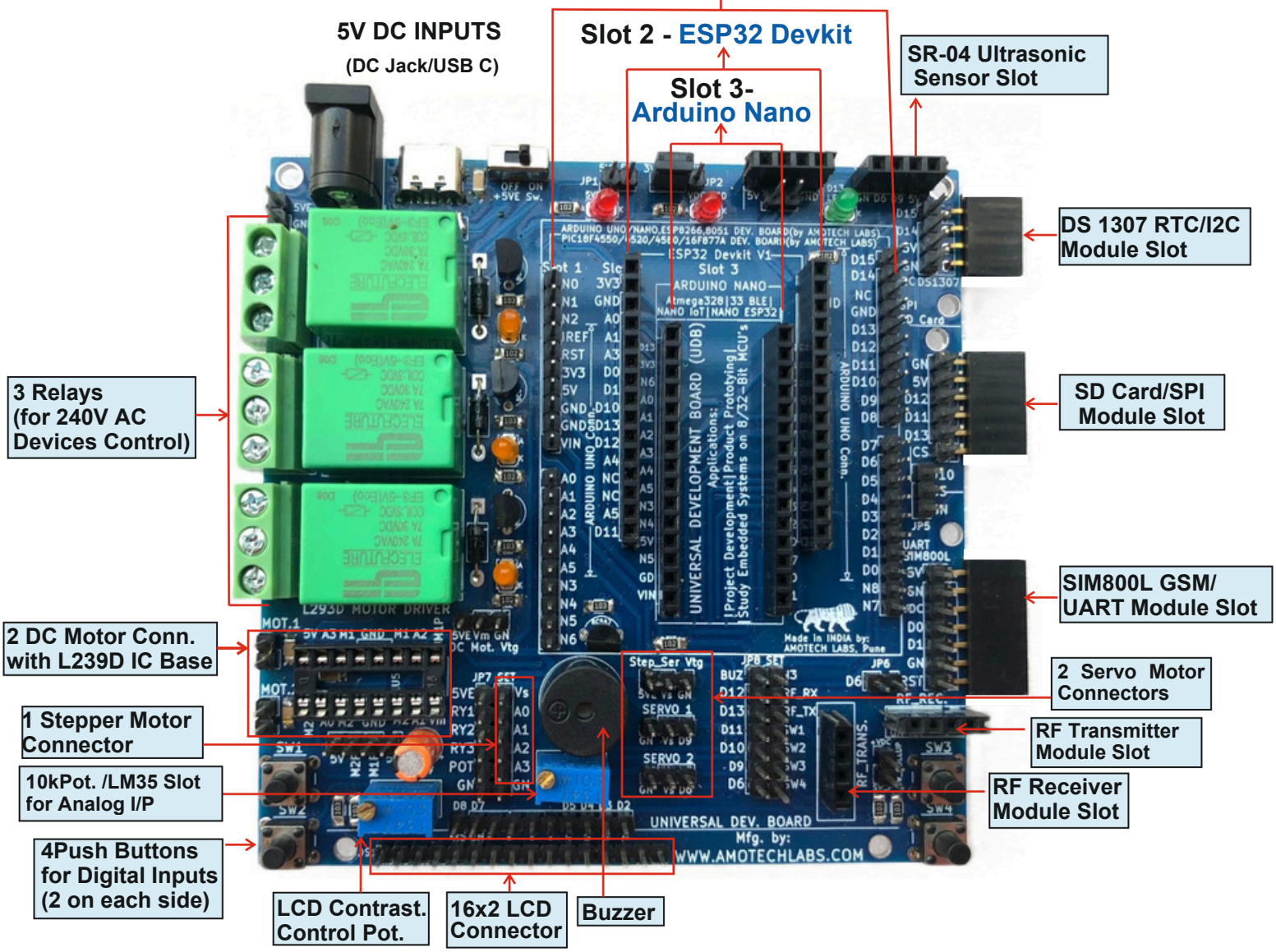
One Development for

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

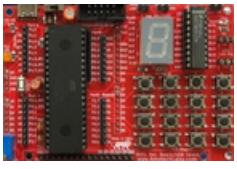
Slot 1

Arduino Uno

STM32F0/F4 NucleoBoards

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

8051 & PIC18F/16 Dev. Boards by Amotech Labs



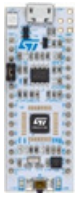
Slot 2

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

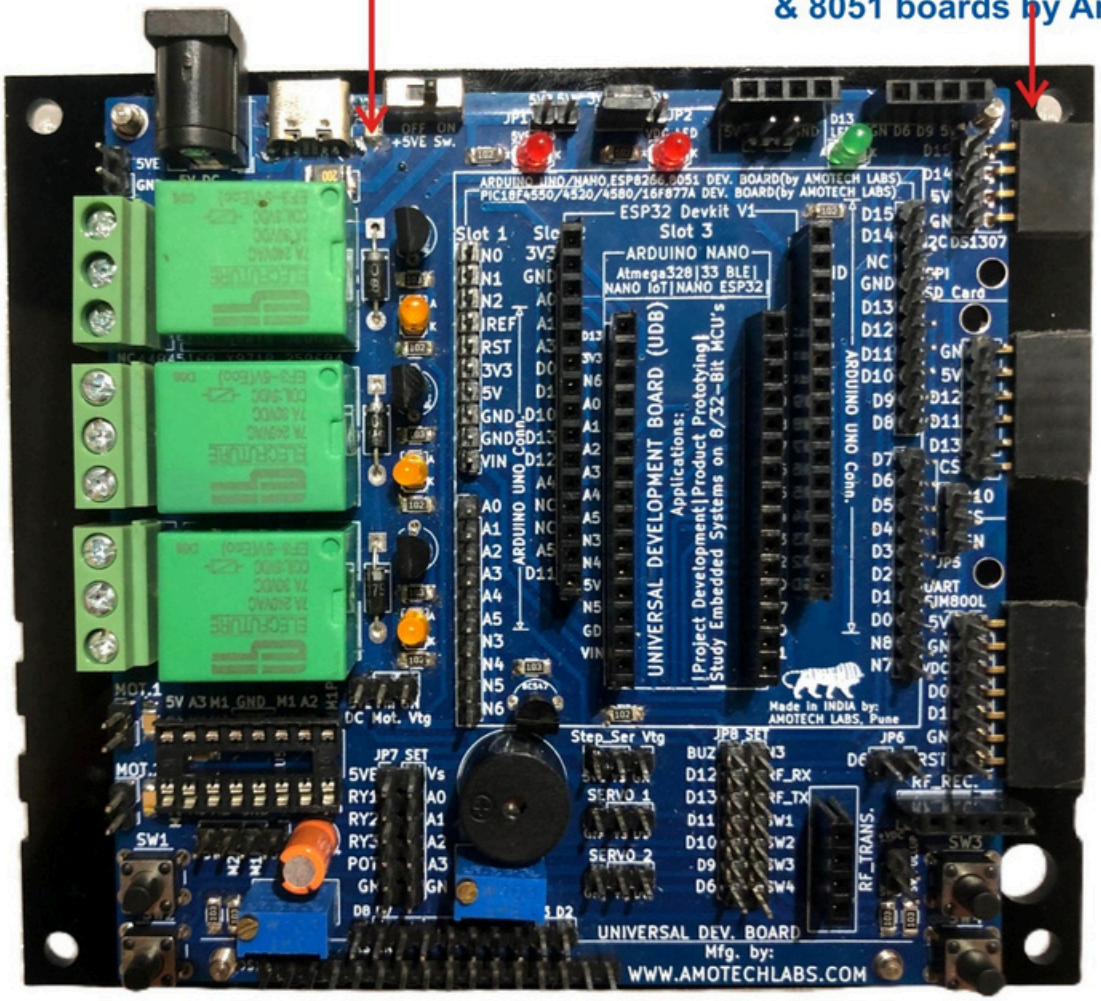
For Students

Universal Development Board(UDB)
+
UDB 2 Wheel Robot Acrylic Sheet

This includes below two items as shown in the below

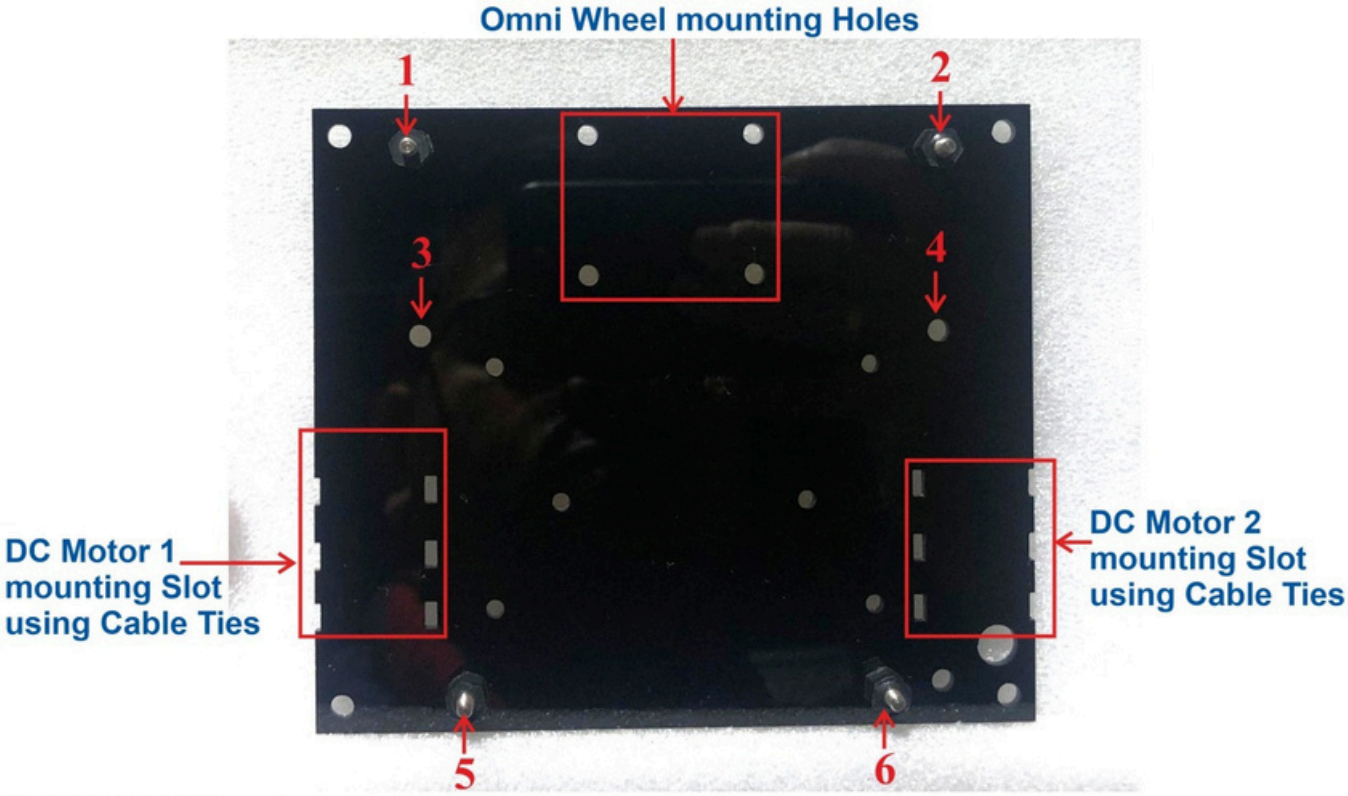
1- Universal Development Board(UDB)

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



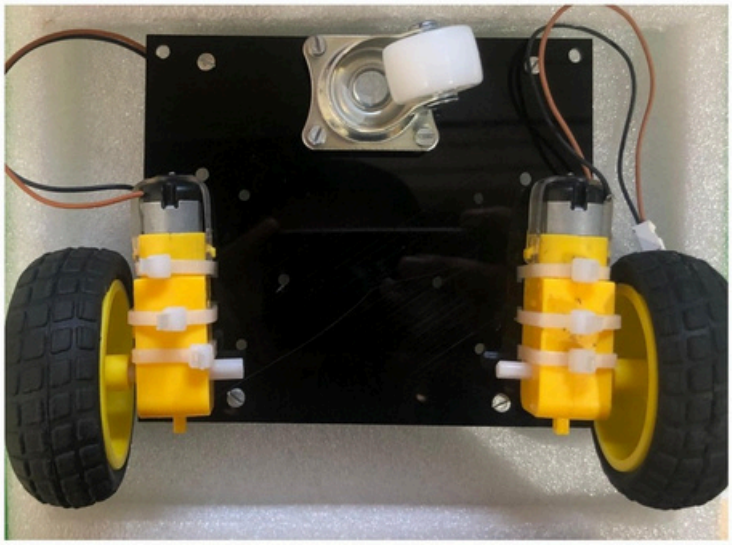
Note: 'UDB 2 Wheel Robot Acrylic Sheet' can be used to hold UDB and mount it on any other platform to use it for multiple Applications. It can be also converted into Robot by attaching 2 DC motors with wheels at it's back and 1 omni wheel at front. Refer UDB Manual or our Reference YouTube videos to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

UDB 2 Wheel Robot Acrylic Sheet

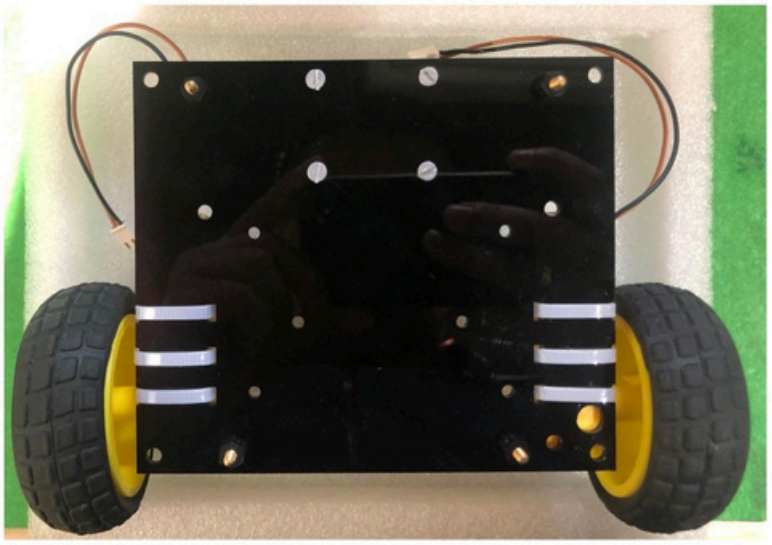


UDB & PIC/8051 Development Boards Mounting Instructions :
Universal Development Board : Use holes 1,2,5 & 6 for mounting UDB on above Acrylic Sheet.
PIC & 8051 Dev. Boards: Use holes 3,4,5 & 6 for mounting PIC18F/16F & 8051 Development Boards.

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



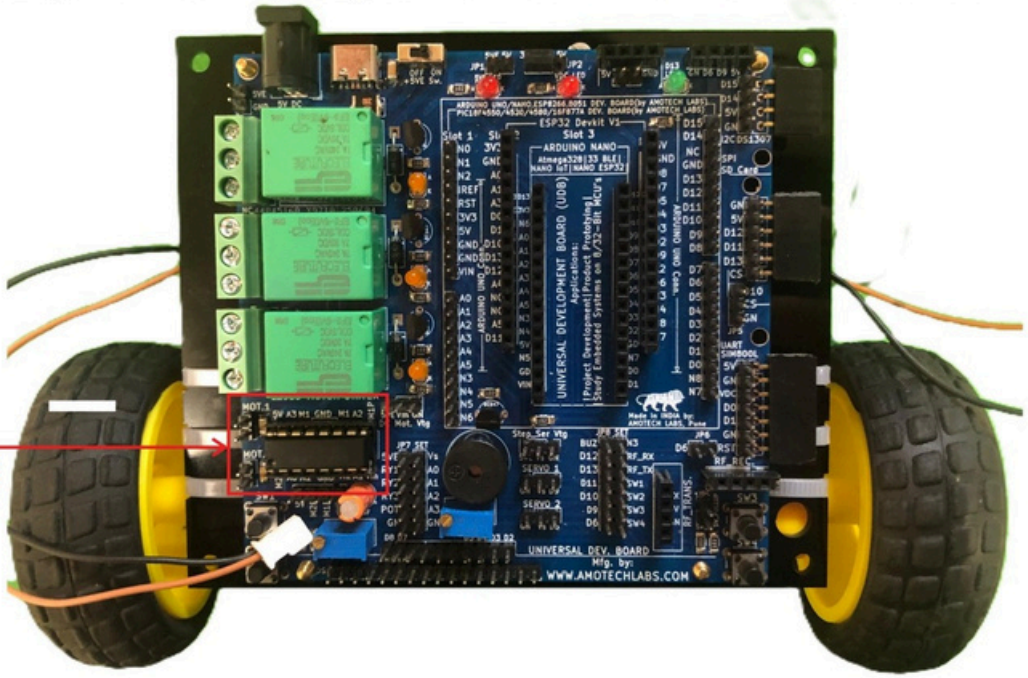
Backside View of Acrylic Sheet after mounting Omni Wheel and 2 DC Motors with their Wheels. Omni wheel is mounted using 10mm spacers with M3 nuts and DC motors can be mounted using Cable ties.



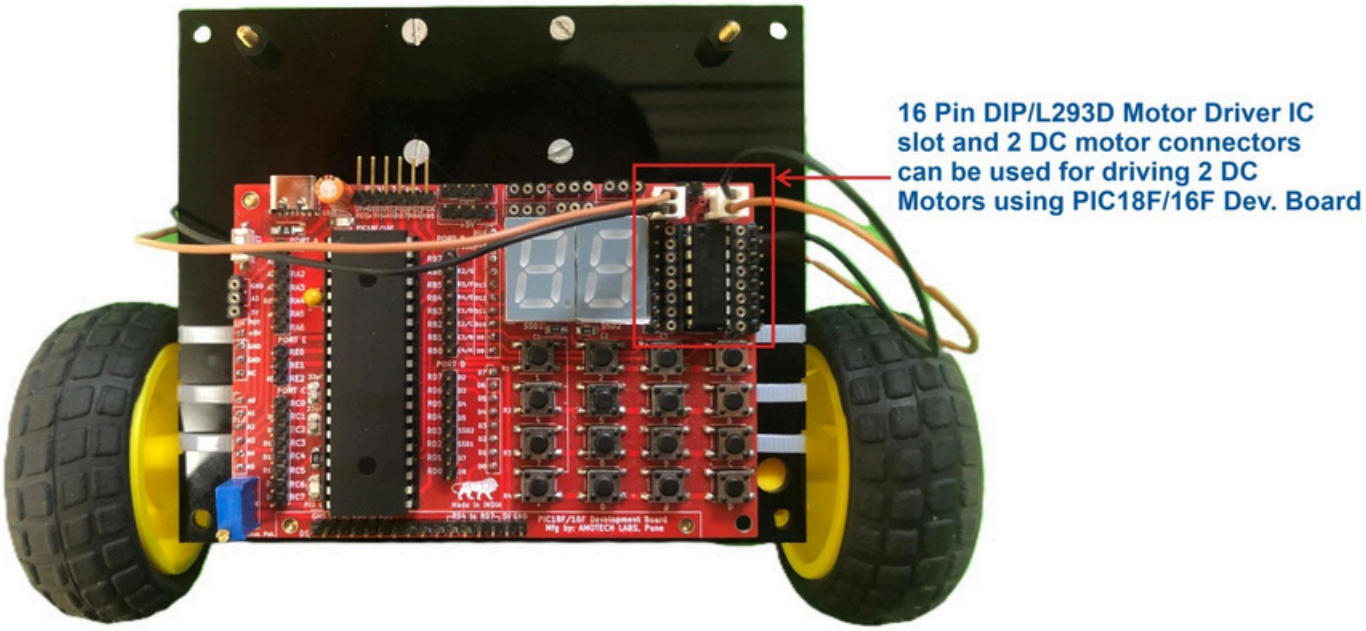
Front View of Acrylic Sheet after mounting Omni Wheel and 2 DC Motors with their Wheels. Now, on top 'UDB' and PIC/8051 Development Boards can be mounted.

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

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



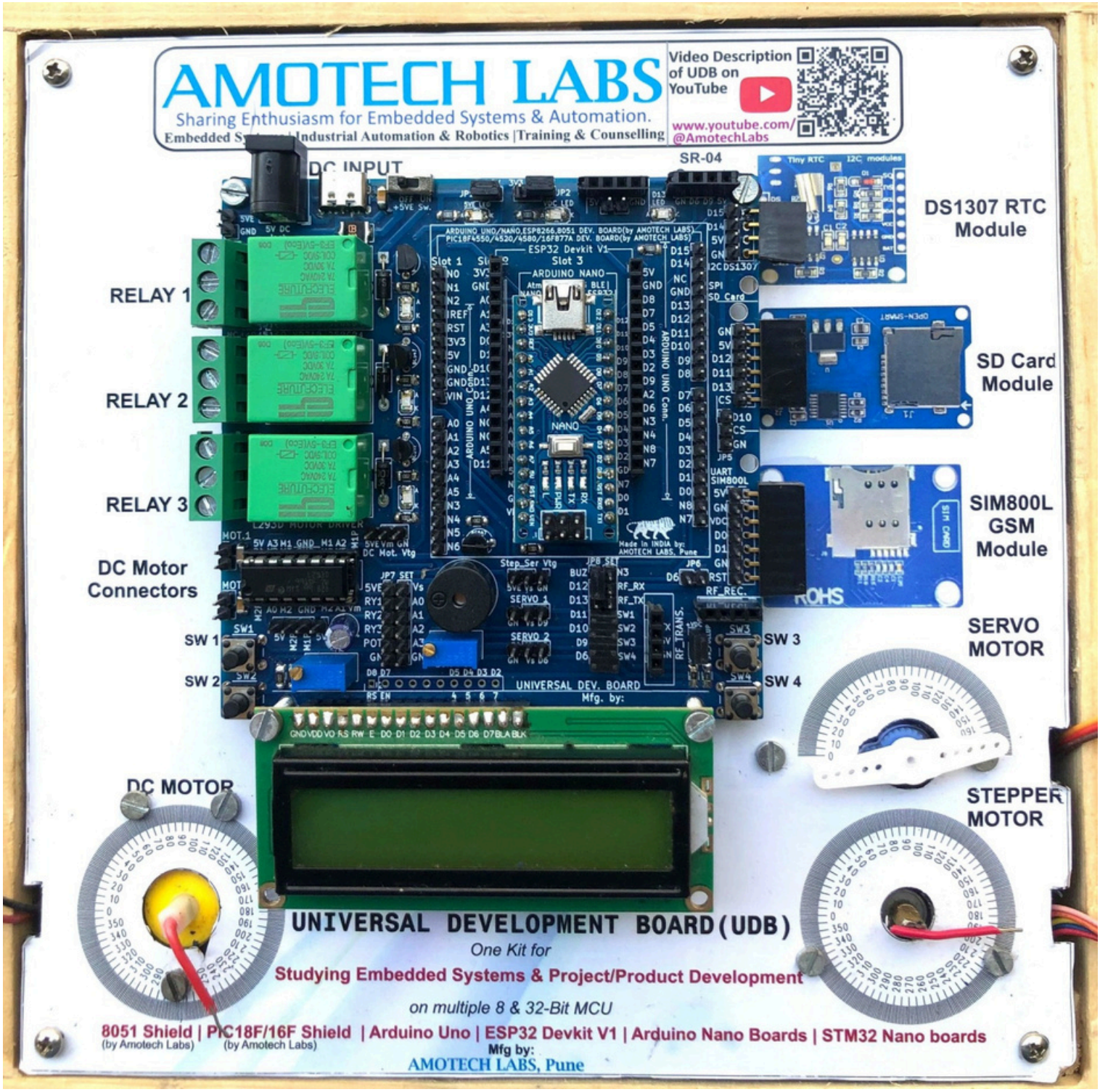
'PIC18F/16F Development Board' mounted on assembled Robot Sheet



Universal Development Board(UDB) kit

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

Universal Development Board(UDB) + Arduino Nano



In above Kit, UDB is mounted on a sheet in a Wooden Enclosure Box along with DC, Servo and Stepper Motors. The Arduino Nano must be attached to UDB to it's Slot 3 as seen in the Picture above.

To study & understand the 'Universal Development Board(UDB)', visit our website for its Reference Material, and also refer to our YouTube channel for Video tutorials on UDB

Website Link:

https://www.amotechlabs.com/

For Youtube Tutorial:

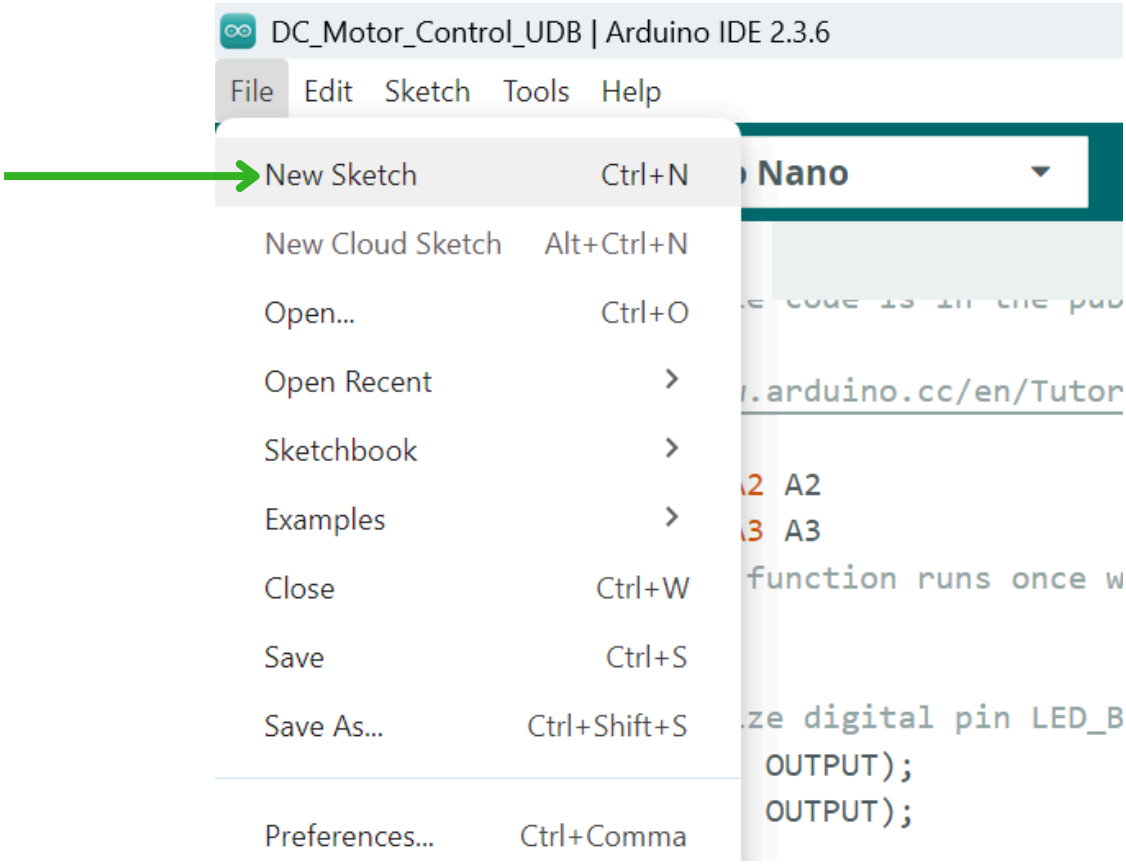
Scan for Video Tutorials

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

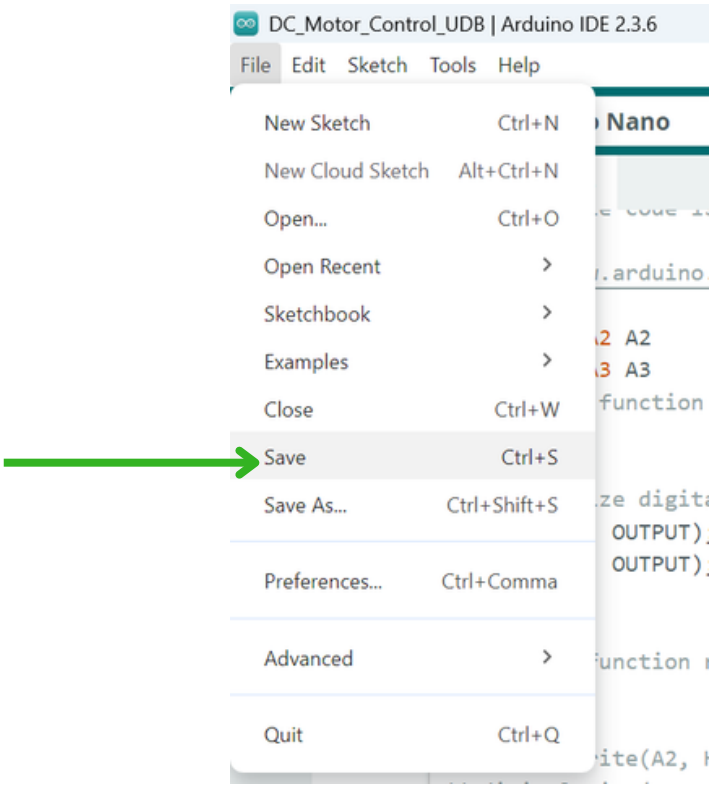
UDB Software User Guide : Arduino IDE Usage Steps

1. Double click on Arduino IDE icon on the desktop

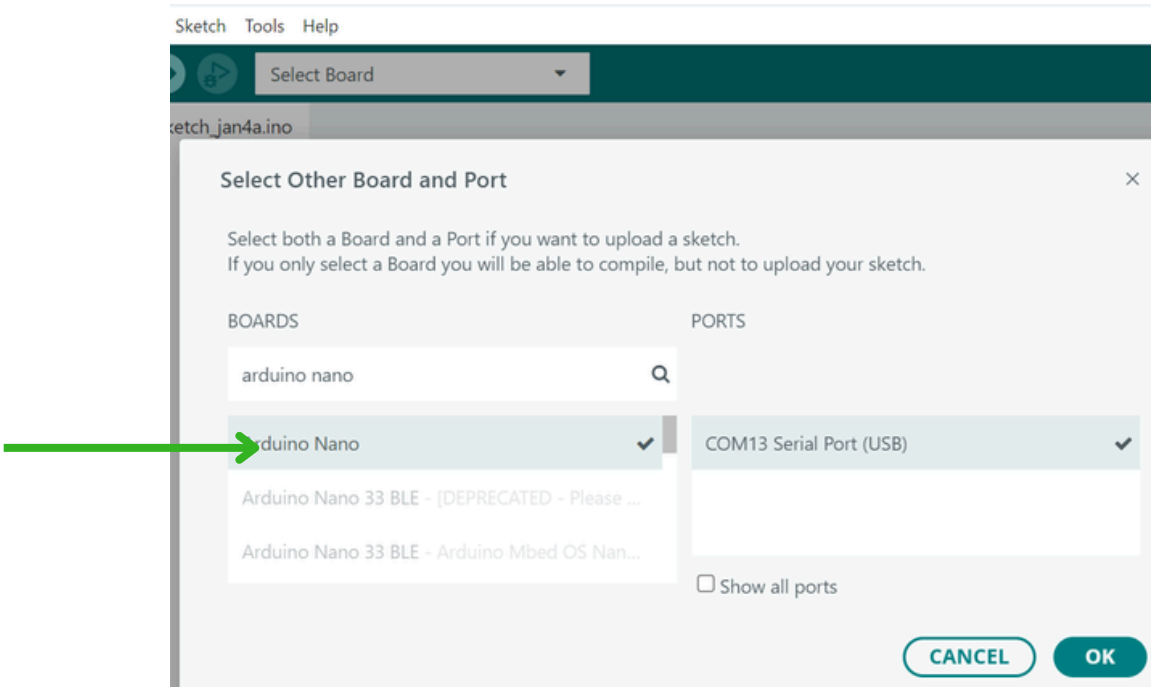
2. To create new project. Select File click on New Sketch



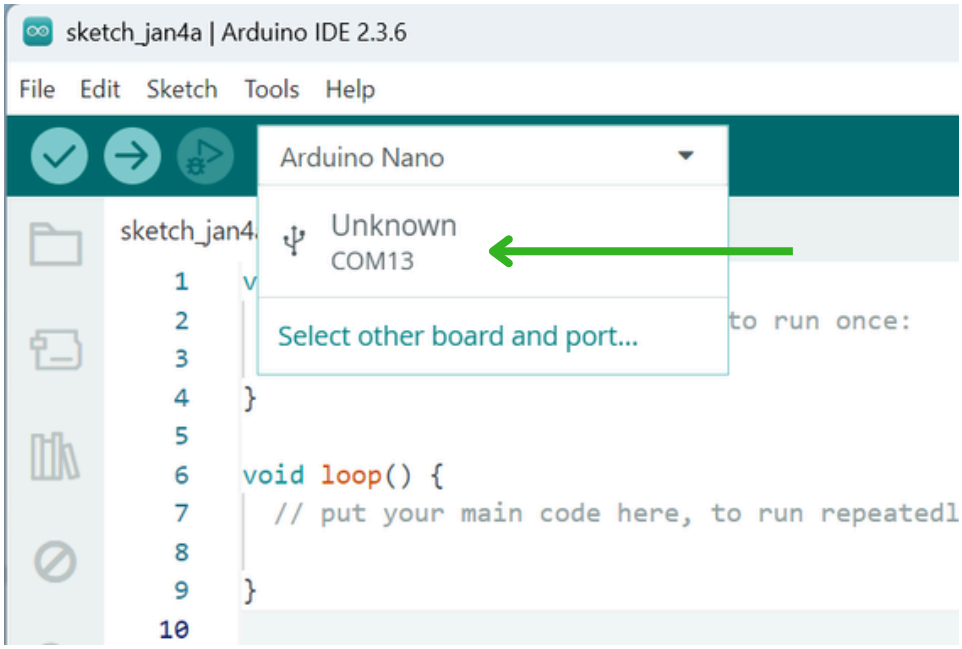
3. Save the sketch : Go to Menu → File → Save the file.



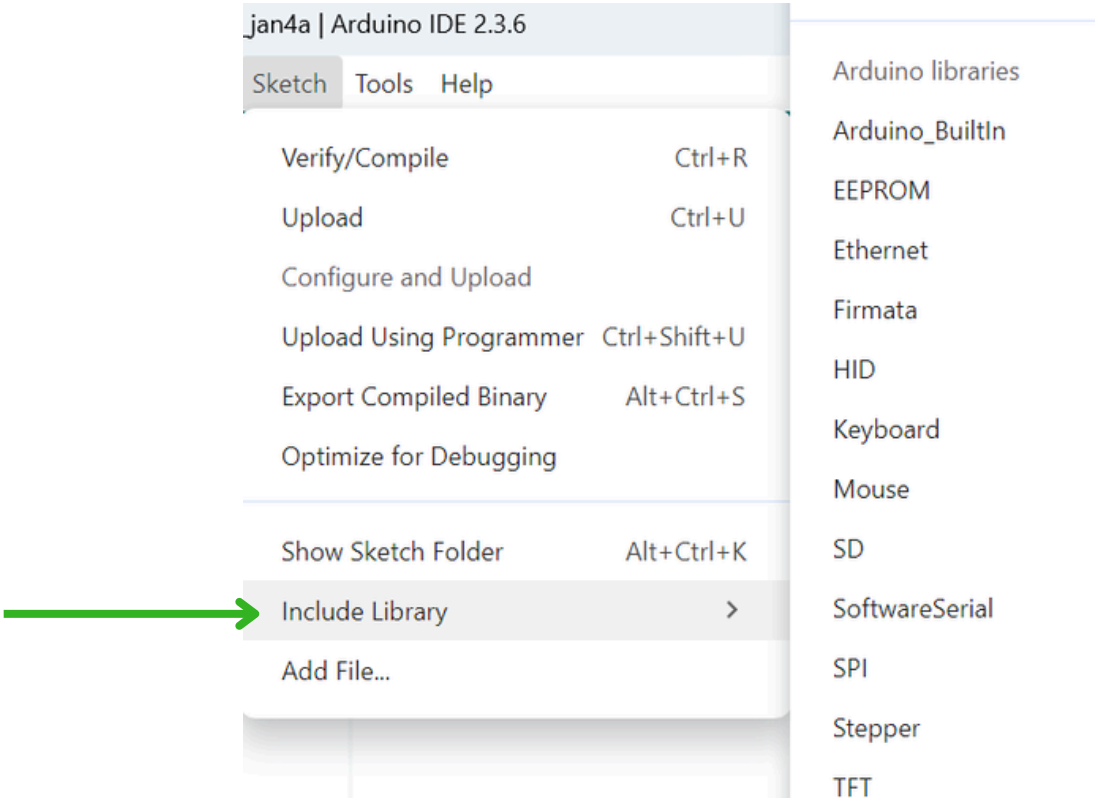
4. Select the board
Tools → Board → {Your board type, e.g. Arduino Nano}.



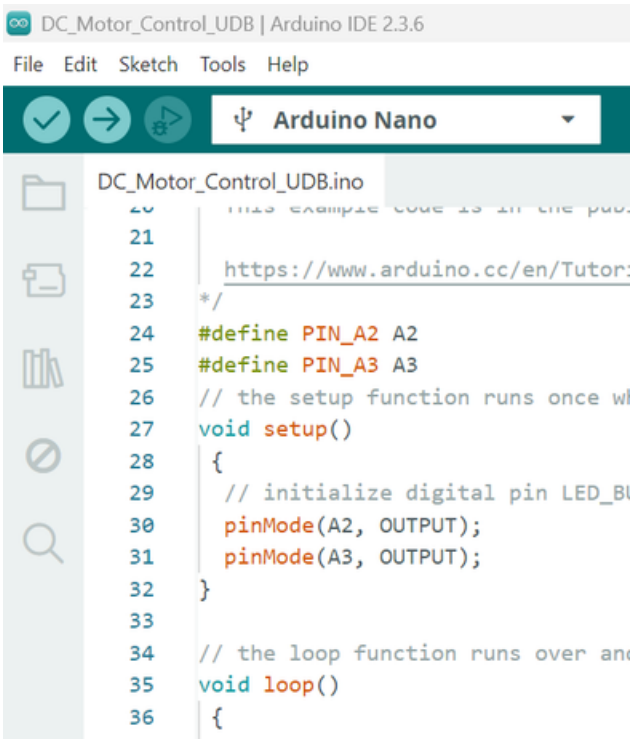
5. Select the port and Verify the code in the device manager if any problem occurs.
   Tools → Port → COMx



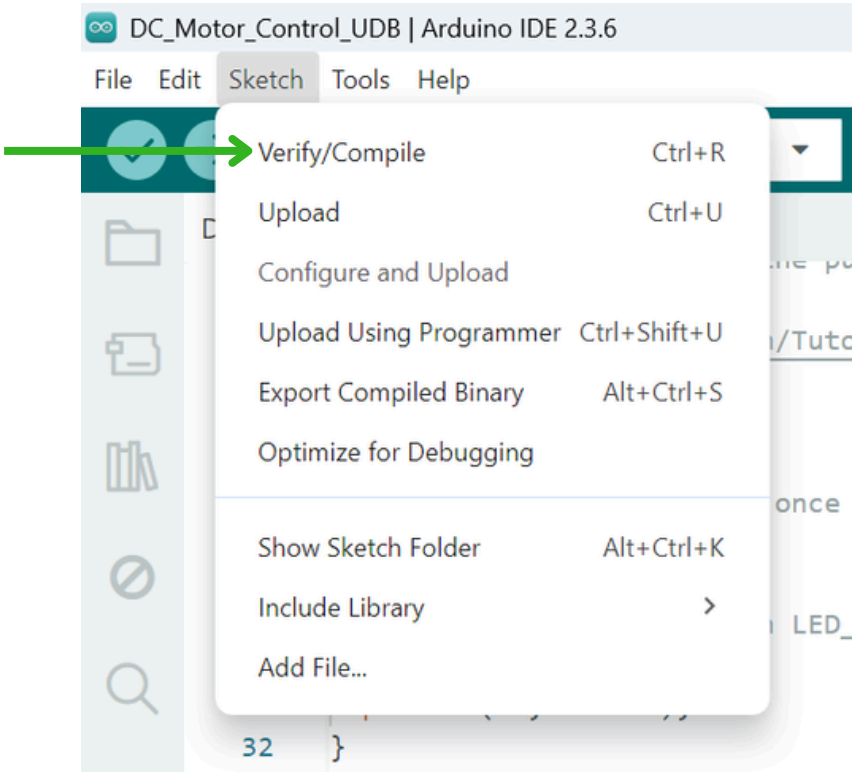
6. Install/Include libraries (if needed) Sketch → Include Library → Manage Libraries



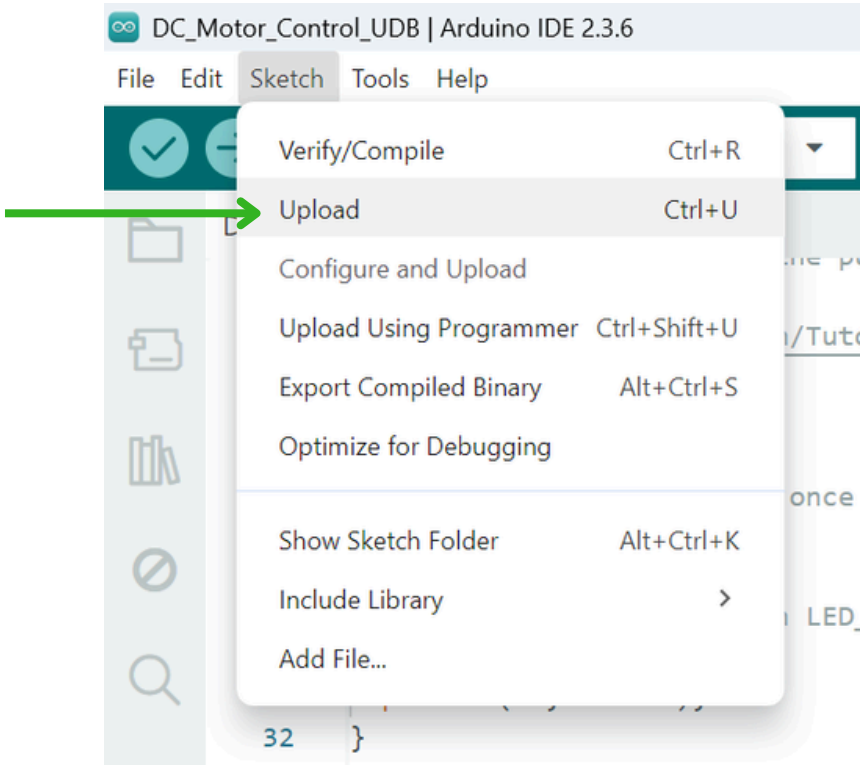
7. Write or paste code in the IDE



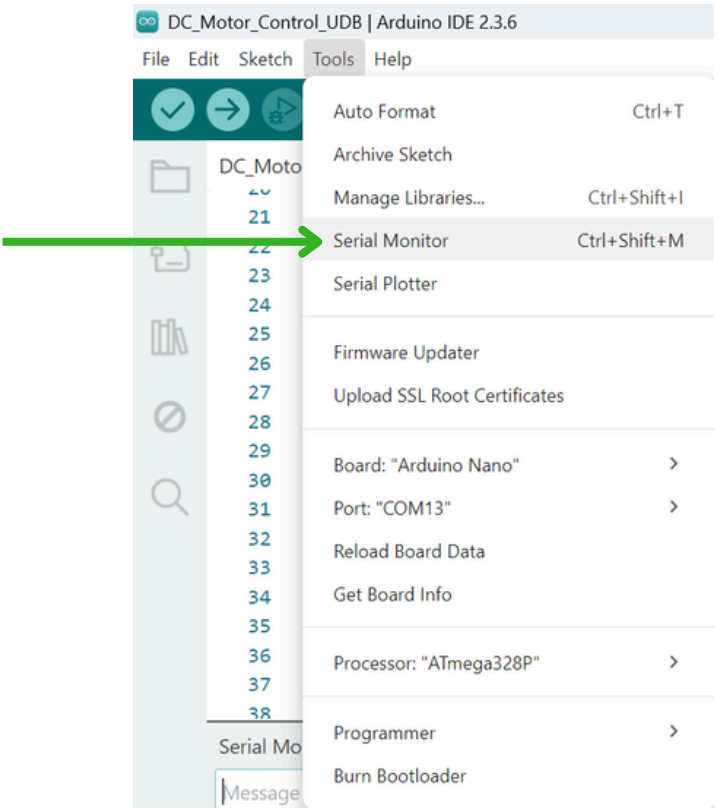
8. Verify code is correct or not (compile)



9. Upload program into the board (Wait until the upload process completes successfully.)



10. Open the Serial Monitor by selecting Tools → Serial Monitor or pressing Ctrl + Shift + M



Experiment 1

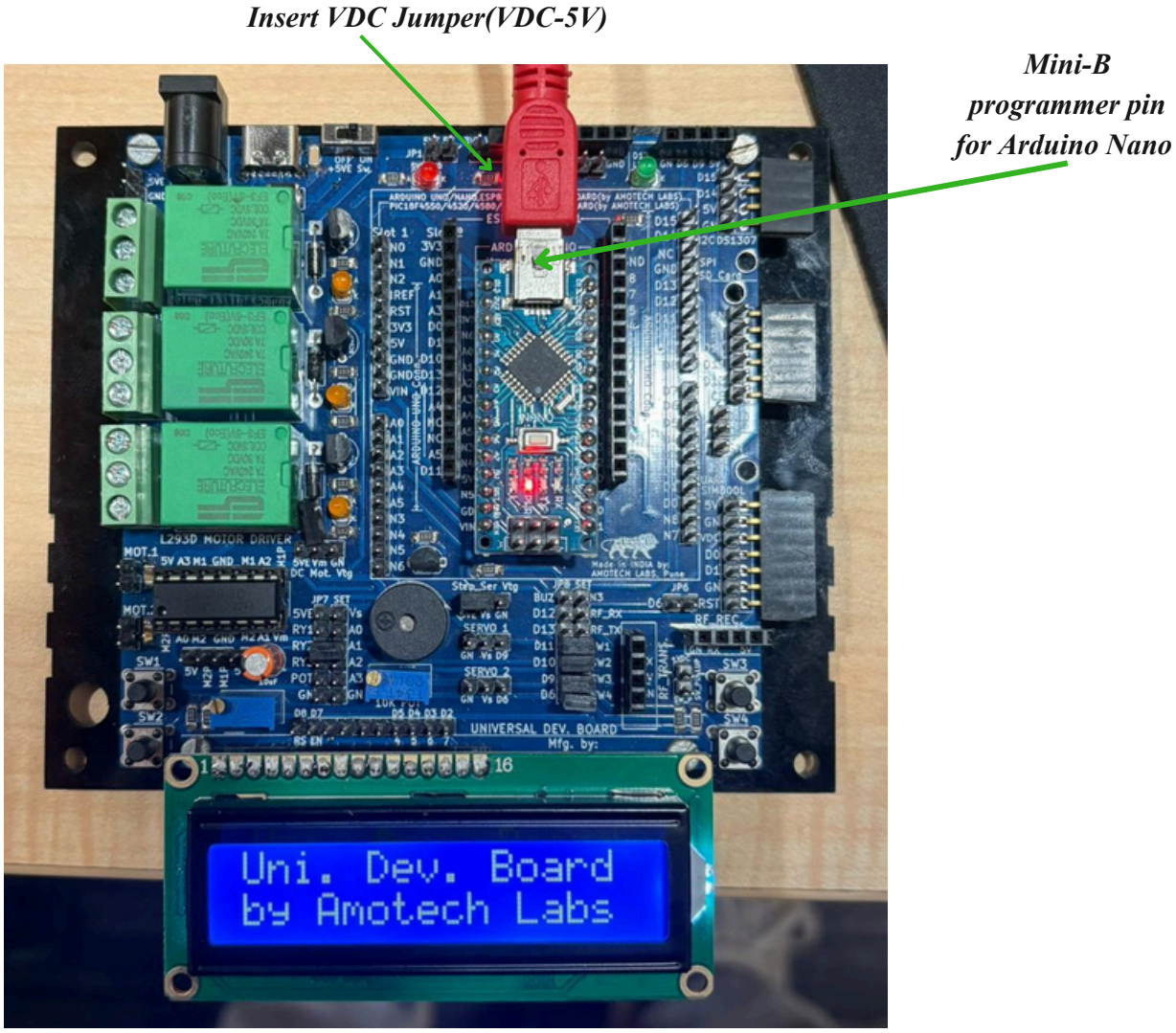
LCD Interfacing

Aim : LCD Interfacing with Arduino Nano using UDB.

Instructions :

1. Open the Arduino IDE and follow the procedure mention in above section Steps for use Arduino IDE.
2. Use a Mini-B USB programming cable to connect the Arduino Nano to the computer. This connection is used for both programming and communication.
3. ***Insert the VDC jumper:*** Select 5V with common VDC, as the Arduino Nano operates at 5V. By Selecting 5V we are setting the pull-up voltage level and maximum analog voltage level.
4. Connect the LCD module to the Universal Development Board as shown in the below diagram verify the Power and Ground connections are secure.
5. Compile and upload the provided Arduino program to the Arduino Nano using the Arduino IDE. Confirm that the upload completes without errors.
6. After successful upload, observe the LCD display. The programmed message should appear on the LCD, confirming correct interfacing and operation.

Figure :



Program :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 7, 5, 4, 3, 2);
void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Uni. Dev. Board");
  lcd.setCursor(0,1);
  lcd.print("by Amotech Labs");
}
void loop() {
}
```

Experiment 2

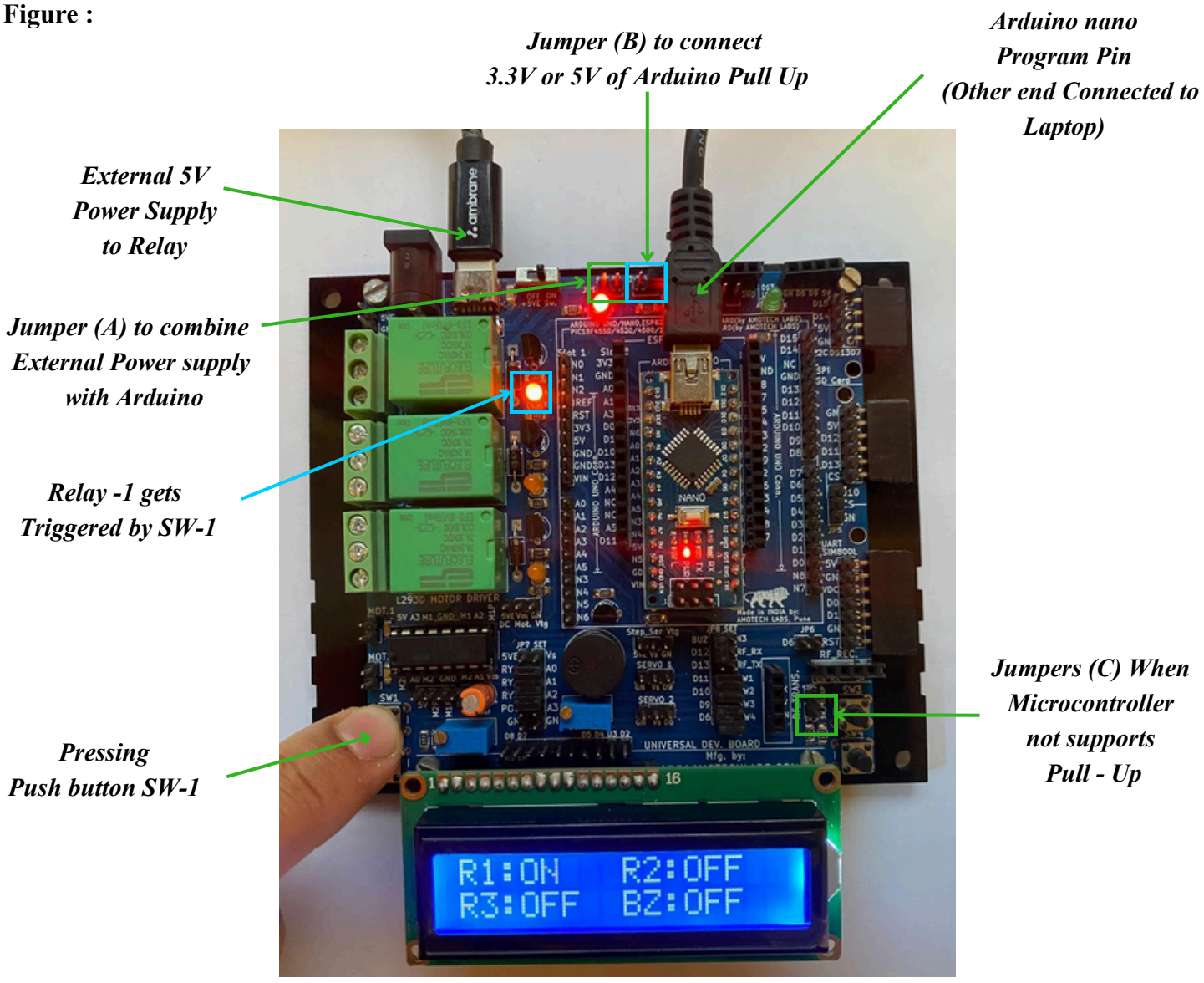
Inputs & Outputs

Aim : To study the Inputs & Outputs of Arduino Nano using UDB..

Procedure :

1. Open the Arduino IDE and follow the procedure mention in above section Steps for use Arduino IDE.
2. Connect the power supply to board i.e. 5V. and turn ON the external Power Supply switch. and Keep connected Arduino nano programming Pin.
3. Make all necessary connections as per above diagram. if we need to combine external power supply to Arduino then, connect jumper (A) as shown in figure.
4. If the Microcontroller Supports Pull up then no need to connected Jumper (C). If Microcontroller do not supports pull up then it is necessary to connect Jumpers (C).
5. Download and run the program.
6. Observe the output - When we press push Button then relay & Buzzer gets triggered & outputs we see Output on LCD Display. SW1 → RLY1, SW2 → RLY2, SW3 → RLY3, SW4 → BUZZER.

Figure :



Program :

Digital_io_UDB.ino

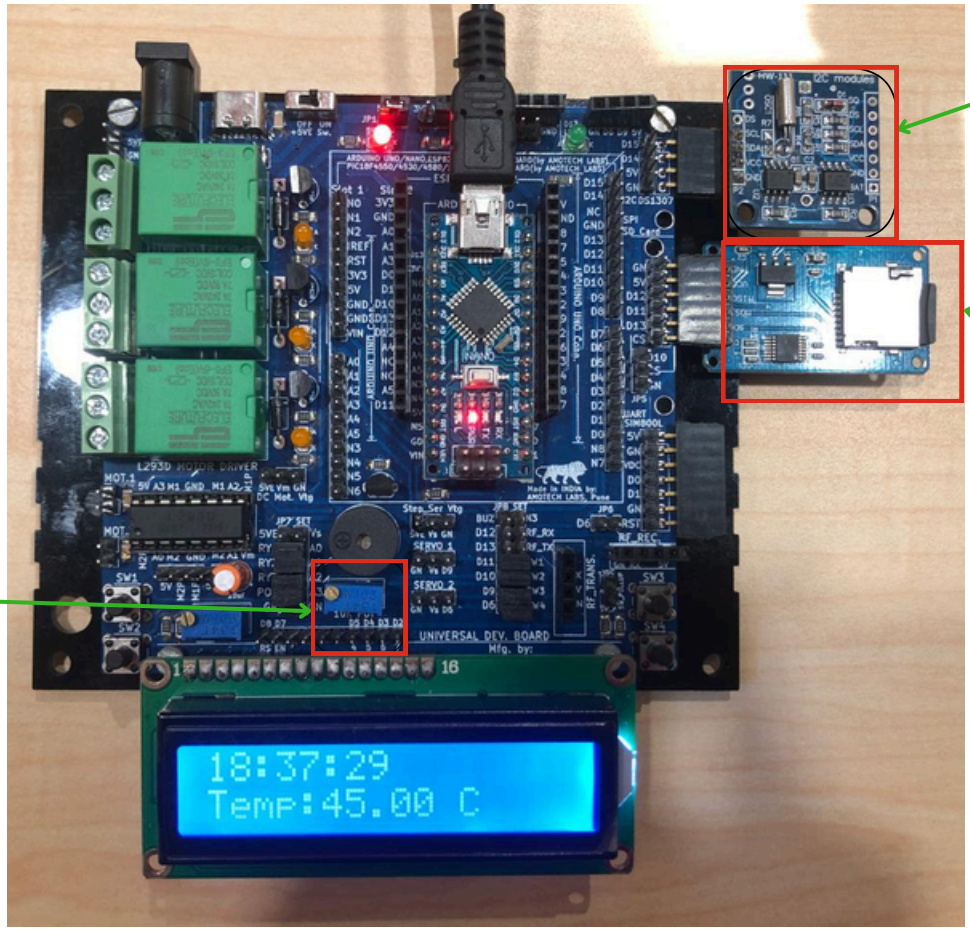
```
#include <LiquidCrystal.h>

// LCD pins: RS, EN, D4, D5, D6, D7
LiquidCrystal lcd(8, 7, 5, 4, 3, 2);

// Relay pins
#define RELAY1 A0
#define RELAY2 A1
#define RELAY3 A2

// Buzzer pin
#define BUZZER 12

// Push button pins
#define SW1 11
#define SW2 10
#define SW3 9
#define SW4 6

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

  // Set outputs
  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  pinMode(RELAY3, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  // Set inputs with pull-up
  pinMode(SW1, INPUT_PULLUP);
  pinMode(SW2, INPUT_PULLUP);
  pinMode(SW3, INPUT_PULLUP);
  pinMode(SW4, INPUT_PULLUP);

  // Turn everything OFF initially
  digitalWrite(RELAY1, LOW);
  digitalWrite(RELAY2, LOW);
  digitalWrite(RELAY3, LOW);
  digitalWrite(BUZZER, LOW);
}

void loop() {

  // -------- RELAY 1 --------
  if (digitalRead(SW1) == LOW) {
    digitalWrite(RELAY1, HIGH);
    lcd.setCursor(0, 0);
    lcd.print("R1:ON  ");
  } else {
    digitalWrite(RELAY1, LOW);
    lcd.setCursor(0, 0);
    lcd.print("R1:OFF ");
  }

  // -------- RELAY 2 --------
  if (digitalRead(SW2) == LOW) {
    digitalWrite(RELAY2, HIGH);
    lcd.setCursor(8, 0);
    lcd.print("R2:ON ");
  } else {
    digitalWrite(RELAY2, LOW);
    lcd.setCursor(8, 0);
    lcd.print("R2:OFF");
  }

  // -------- RELAY 3 --------
  if (digitalRead(SW3) == LOW) {
    digitalWrite(RELAY3, HIGH);
    lcd.setCursor(0, 1);
    lcd.print("R3:ON  ");
  } else {
    digitalWrite(RELAY3, LOW);
    lcd.setCursor(0, 1);
    lcd.print("R3:OFF ");
  }

  // -------- BUZZER --------
  if (digitalRead(SW4) == LOW) {
    digitalWrite(BUZZER, HIGH);
    lcd.setCursor(8, 1);
    lcd.print("BZ:ON ");
  } else {
    digitalWrite(BUZZER, LOW);
    lcd.setCursor(8, 1);
    lcd.print("BZ:OFF");
  }

  delay(200); // small delay for stability
}
```

Experiment – 3

RTC, SD card, ADC Interfacing

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

Figure :



Instructions :

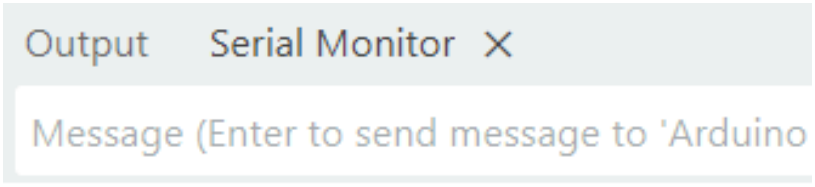
1. Connect the RTC module, SD card module, and 10k pot to the UDB as per the figure.
2. Connect the Arduino Nano to the computer using a USB cable & Open the Arduino IDE.
3. Select the correct Board: Arduino Nano and the appropriate Processor.
4. Select the correct COM Port from the Tools menu.
5. Open the program file for RTC, SD card, interfacing.
6. Compile and upload the code to the Arduino Nano and After successful uploading rotate 10k pot and open the Serial Monitor.
7. Observe the real-time clock data, temperature, and humidity values on the serial monitor.
8. Verify that the data is successfully logged into the SD card.

Program :

SD_Temp_RTC.ino

```
/* In this Program we sense a Analog Signal at Pin A3.
   The Analog Input is varied using a 10K pot.
   The sensed Analog Value is stored in a SD Card with Time
   stamp. The Real Time is fetched from DS1307 RTC*/
#include <SPI.h> //for the SD card module
#include <SD.h> // for the SD card
#include <RTClib.h> // for the RTC

const int chipSelect = 10;
File myFile; // Create a file to store the data
RTC_DS1307 rtc; // For RTC

void setup()
{
  Serial.begin(9600); //initializing Serial monitor
  // setup for the RTC
  while(!Serial);
    if(! rtc.begin()) {
      Serial.println("Couldn't find RTC");
      while (1);
    }
    else {
    //ets the RTC to the date & time this sketch was compiled
      rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    }
    if(! rtc.isrunning()) {
      Serial.println("RTC is NOT running!");
    }
  // setup for the SD card
  Serial.print("Initializing SD card...");
  if(!SD.begin(chipSelect)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  myFile=SD.open("DATA.txt", FILE_WRITE);//open file
  // if the file opened ok, write to it:
  if (myFile) {
    Serial.println("File opened ok");
    // print the headings for our data
    myFile.println("Date,Time,Temperature ºC");
  }
  myFile.close();
}

void loggingTime() {
  DateTime now = rtc.now();
  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    myFile.print(now.year(), DEC);
    myFile.print('/');
    myFile.print(now.month(), DEC);
    myFile.print('/');
```

```
    myFile.print('/');
    myFile.print(now.day(), DEC);
    myFile.print(',');
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print(",");
  }
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.println(now.day(), DEC);
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.println(now.second(), DEC);
  myFile.close();
  delay(1000);
}
void loggingTemperature(){
  float t = analogRead(A3);
  //Consider Analog Input is a Temperature Reading
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
  myFile = SD.open("DATA.txt", FILE_WRITE);
  if (myFile) {
    Serial.println("open with success");
    myFile.print(t);
    myFile.println(",");
  }
  myFile.close();
}
void loop() {
  loggingTime();
  loggingTemperature();
  delay(5000);
}
```

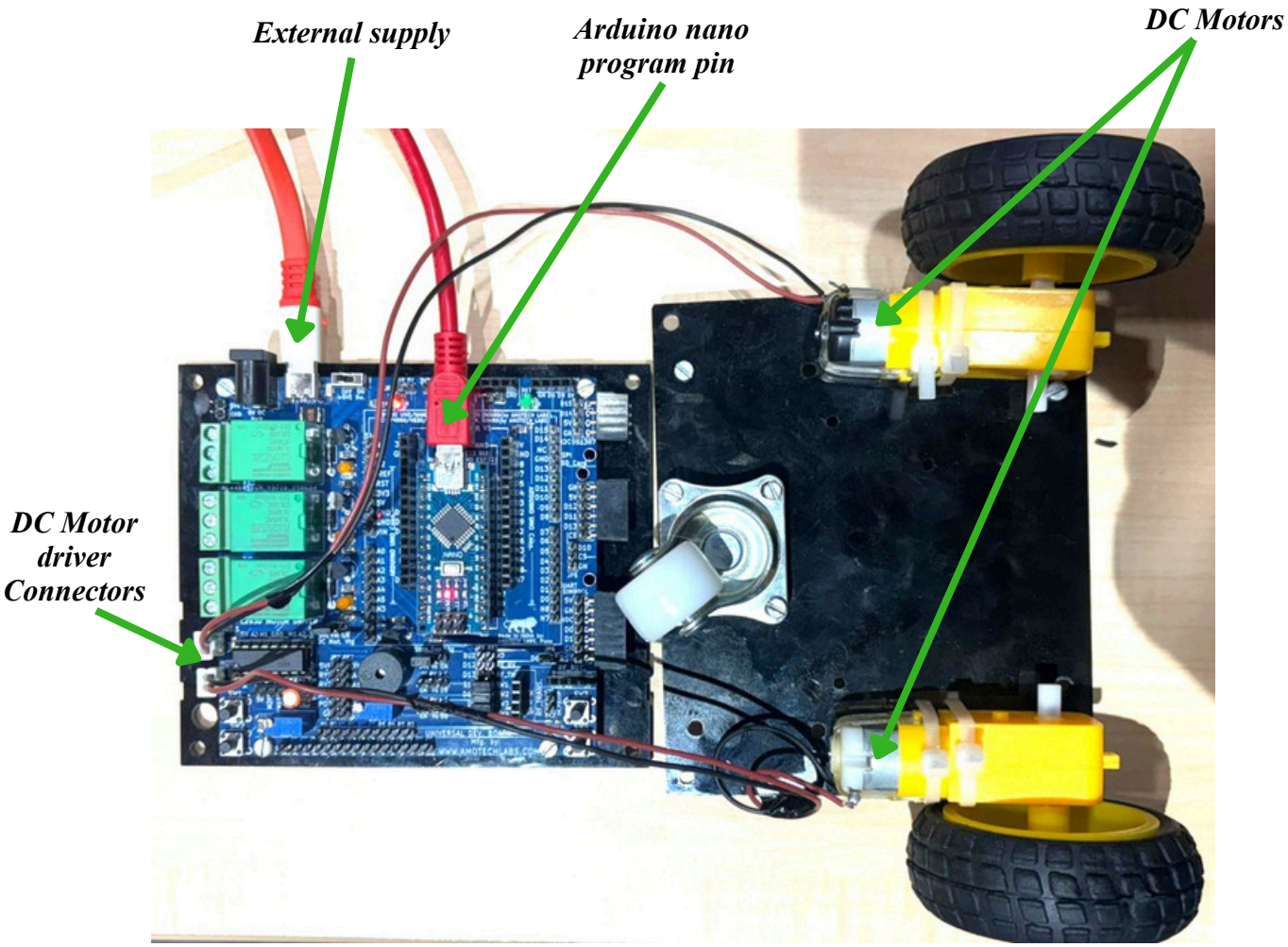



```
2026/1/5
18:27:47
Temperature: 44.00 *C
open with success
2026/1/5
18:27:53
Temperature: 44.00 *C
open with success
2026/1/5
18:27:59
Temperature: 44.00 *C
open with success
2026/1/5
18:28:5
Temperature: 44.00 *C
open with success
2026/1/5
18:28:11
Temperature: 44.00 *C
open with success
2026/1/5
18:28:17
Temperature: 44.00 *C
open with success
2026/1/5
18:28:24
Temperature: 44.00 *C
open with success
2026/1/5
18:28:30
```

Experiment 4
DC Motor Interfacing

Aim : To Study DC Motor Interfacing with Arduino Nano Using UDB..

Figure :



Instruction :

1. Connect the DC motor to the motor driver as shown in the figure.
2. Connect the motor driver pins MOT1 and MOT2 on the UDB to the DC motor.
3. Connect the Arduino Nano to the computer using a USB cable and open the Arduino IDE.
4. Compile and upload the DC motor control program to the Arduino Nano.
5. Observe the DC motor operation after successful upload.
6. Verify the direction of rotation and speed control.

Program :

```
// -------- MOTOR DRIVER PINS --------
// Left motor
const int L_IN1 = A0;
const int L_IN2 = A1;
const int L_EN = 3;  // PWM

// Right motor
const int R_IN1 = A2;
const int R_IN2 = A3;
const int R_EN = 5;  // PWM

void setup() {
  pinMode(L_IN1, OUTPUT);
  pinMode(L_IN2, OUTPUT);
  pinMode(R_IN1, OUTPUT);
  pinMode(R_IN2, OUTPUT);
  pinMode(L_EN, OUTPUT);
  pinMode(R_EN, OUTPUT);

  // Motors OFF initially
  analogWrite(L_EN, 0);
  analogWrite(R_EN, 0);
}

void loop() {

  // -------- BOTH MOTORS FORWARD --------
  digitalWrite(L_IN1, HIGH);
  digitalWrite(L_IN2, LOW);
  digitalWrite(R_IN1, HIGH);
  digitalWrite(R_IN2, LOW);
  analogWrite(L_EN, 180);
  analogWrite(R_EN, 180);
  delay(1500);

  // -------- BOTH MOTORS BACKWARD --------
  digitalWrite(L_IN1, LOW);
  digitalWrite(L_IN2, HIGH);
  digitalWrite(R_IN1, LOW);
  digitalWrite(R_IN2, HIGH);
  analogWrite(L_EN, 180);
  analogWrite(R_EN, 180);
  delay(1500);
```

Program :

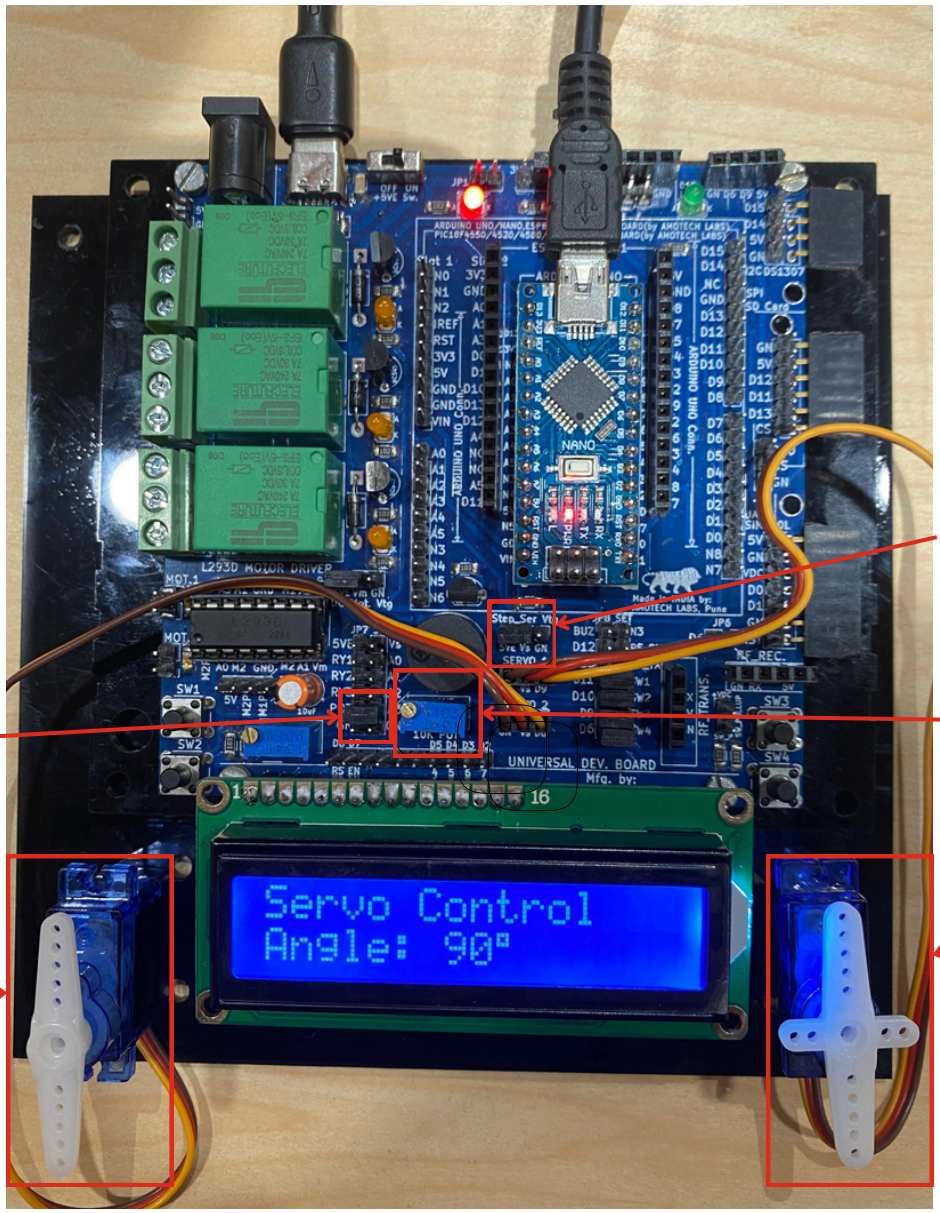
```
  // -------- LEFT MOTOR ONLY --------
  analogWrite(R_EN, 0);  // Right OFF

  // Left anticlockwise
  digitalWrite(L_IN1, HIGH);
  digitalWrite(L_IN2, LOW);
  analogWrite(L_EN, 180);
  delay(1500);

  // Left clockwise
  digitalWrite(L_IN1, LOW);
  digitalWrite(L_IN2, HIGH);
  delay(1500);

  // -------- RIGHT MOTOR ONLY --------
  analogWrite(L_EN, 0);  // Left OFF

  // Right anticlockwise
  digitalWrite(R_IN1, HIGH);
  digitalWrite(R_IN2, LOW);
  analogWrite(R_EN, 180);
  delay(1500);

  // Right clockwise
  digitalWrite(R_IN1, LOW);
  digitalWrite(R_IN2, HIGH);
  delay(1500);

  // -------- ALL MOTORS OFF BEFORE REPEAT --------
  analogWrite(L_EN, 0);
  analogWrite(R_EN, 0);
  delay(1500);
}
```

Experiment 5

Survo Interfacing

Aim : To Study Servo Motor Interfacing with Arduino Nano Using UDB.

Figure :



External Power Supply Jumper (Step Servo Voltage)

10K Pot Jumper

10k Pot

Servo 1 (with 90 degree Rotation)

Servo 2 (with 90 degree Rotation)

Instructions :

1. Connect the Servo motor jumper (Pot Jumper) on UDB as per the Figure. (Pot is connected to A3)
2. Connect the Servo pins (SERVO 1, SERVO 2) on UDB to the Survo Motor.
3. Connect jumpers near buzzer named as Step servo voltage, so our servo is connected to external Power Supply which is given by USB C - Type Cable.
4. Connect the Arduino Nano to the computer using a USB cable & Open the Arduino IDE.
5. After successful uploading, observe the Servo operation.
6. Verify the direction of rotation and Angle displays on LCD.

Program :

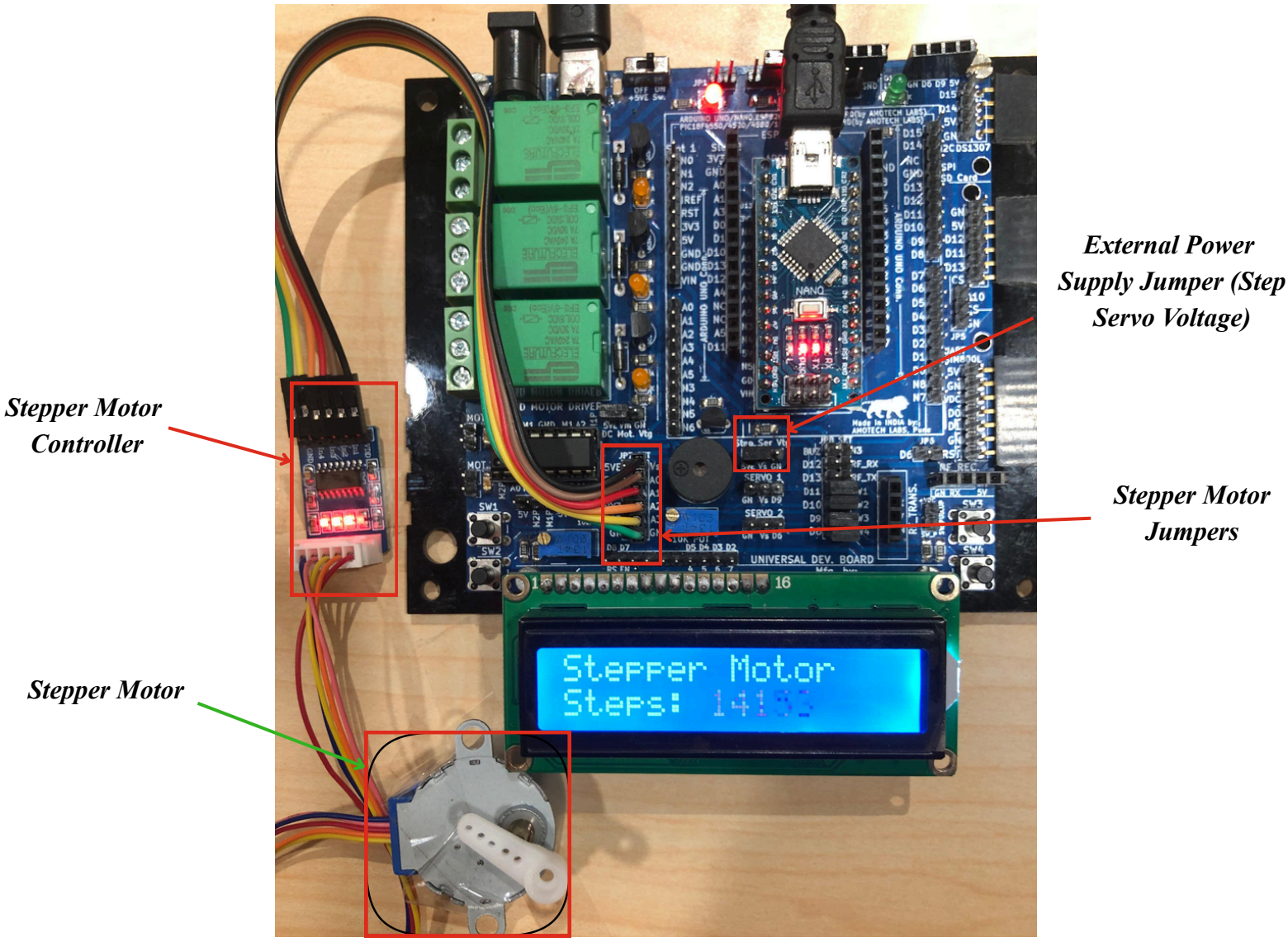
Servo_motors.ino

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

Servo servo1;   // Servo on D6
Servo servo2;   // Servo on D9

int potPin = A3; // ----- Potentiometer -----
int potValue = 0;
int angle = 0;

LiquidCrystal lcd(8, 7, 5, 4, 3, 2); // ----- LCD pins: RS, EN, D4, D5, D6, D7
void setup()
{
  servo1.attach(6);   // Servo 1 on D6
  servo2.attach(9);   // Servo 2 on D9

  lcd.begin(16, 2);
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print("Servo Control");

  lcd.setCursor(0, 1);
  lcd.print("Angle: ");
}
void loop()
{
  potValue = analogRead(potPin);          // Read potentiometer
  angle = map(potValue, 0, 1023, 0, 180); // Map to 0–180°

  servo1.write(angle);
  servo2.write(angle);

  lcd.setCursor(7, 1);   // After "Angle: "
  lcd.print("    ");     // Clear old value
  lcd.setCursor(7, 1);
  lcd.print(angle);
  lcd.print((char)223);  // Degree symbol
  lcd.print(" ");

  delay(100);
}

```

Experiment 6
Stepper Motor Interfacing

Aim : To Study Stepper Motor Interfacing with Arduino Nano Using UDB.

Figure :



Instructions :

1. Connect the stepper motor to the UDB as shown in the figure.
2. Connect the Arduino Nano to the computer using a USB cable and open the Arduino IDE.
3. Insert the Step / Servo Voltage jumper near the buzzer to select external power supply for the stepper motor (via USB Type-C).
4. Compile and upload the program to the Arduino Nano.
5. Observe the stepper motor operation.
6. Verify the direction of rotation.

Program :

stepper_motor.ino

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

const int stepsPerRevolution = 200;  // change this to fit the

int pin1 = A0;
int pin2 = A1;
int pin3 = A2;
int pin4 = A3;
Stepper myStepper(stepsPerRevolution, pin1, pin2, pin3, pin4);

int stepCount = 0;  // number of steps the motor has taken

LiquidCrystal lcd(8, 7, 5, 4, 3, 2); // LCD pins: RS, EN, D4, D5, D6, D7

void setup() {
  // initialize the serial port:
  Serial.begin(9600);

  // initialize LCD
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Stepper Motor");
  lcd.setCursor(0, 1);
  lcd.print("Steps: ");
}

void loop() {

  myStepper.step(1); // step one step:

  Serial.print("steps:");
  Serial.println(stepCount);

  lcd.setCursor(7, 1);   // after "Steps: "
  lcd.print("      ");  // clear old value
  lcd.setCursor(7, 1);
  lcd.print(stepCount);

  stepCount++;
  delay(5);
}
```

Experiment 7

GSM SIM800L Interfacing

Aim : To study the interfacing of GSM SIM800L with Arduino Nano and send and receive the sms Using UDB..

Figure :

Instructions :

1. Connect the GSM Module (800L) to the UDB as per the Figure.
2. Connect the GSM Module (800L) female header pins on UDB to the GSM Module (800L).
3. Connect the Arduino Nano to the computer using a USB cable & Open the Arduino IDE.
4. Select the correct Board: Arduino Nano and the appropriate Processor.
5. Select the correct COM Port from the Tools menu.
6. Compile and upload the program to the Arduino Nano.
7. After successful uploading, observe the sending & receiving of message on mobile phone.

Program : (Sending)

IM800L_Msg_Send.ino

```
#include <SoftwareSerial.h>
//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(14, 15); //SIM800L Tx & Rx is
                                 //connected to Arduino #3 & #2
void setup()
{
  Serial.begin(9600);//Begin serial communication \
                     //with Arduino and Arduino IDE (Serial Monitor)
  mySerial.begin(9600); //Begin serial communication with Arduino & SIM800L
  delay(1000);    //Serial.println("Initializing...");
  Serial.println("AT"); //Once the handshake test is successful, it will back to OK
  delay(500);   //updateSerial();
  Serial.println("AT+CMGF=1"); // Configuring TEXT mode
  delay(500);   //updateSerial();
 Serial.println("AT+CMGS=\"+91xxxxxxxxxx\"");
 delay(500);  // updateSerial();
  Serial.print("Universal Dev Board by AMOTECH LABS");
  delay(500);        //updateSerial();
  Serial.write(26);
  delay(5000);
}
void loop()
{
  delay(500);
}
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
  }
}
```

Program : (Receiving)

/800L_Msg_Receive.ino

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2
void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);
  mySerial.begin(9600);//Begin serial communication with Arduino and SIM800L

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly
  updateSerial();                        //arrived SMS messages should be handled
}
void loop()
{
  updateSerial();
}
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
  }
}
```