N.B.K.R INSTITUTE OF SCIENCE & TECHNOLOGY::VIDYANAGAR

(AUTONOMOUS)

VIDYANAGAR-524413, TIRUPATI DIST, AP.

(Accredited by NBA: Accredited 'A' Grade by NAAC)

CERTIFICATE

Certified that this is the Bona-fide record of the practical work done by

Mr. /Ms. ____bearing the

Roll. No.____of _________Year,

B.Tech ____

in the_______Semester for the Academic year__________________________ has completed the

____Laboratory.

No. of experiments completed:

Staff member Incharge Signature:

External Examiner Signature with date:

S.No.	**Title of the Experiment**	**Date**	**Page No.**	**Grade**	**Faculty Signature**
1	Make your own parallel and series circuits using breadboard for any application of your choice.				
2	Design and 3D print a Walking Robot				
3	Design and 3D Print a Rocket				
4	Temperature & Humidity Monitoring System (DHT11 + LCD)				
5	Water Level Detection and Alert System				
6	Automatic Plant Watering System				
7	Bluetooth-Based Door Lock System				
8	Smart Dustbin Using Ultrasonic Sensor				
9	Fire Detection and Alarm System				
10	Soil Moisture-Based Irrigation				
11	Smart Helmet for Accident Detection				
12	Temperature-Controlled Chemical Reactor				
13	ECG Signal Acquisition and Plotting				

Note: The students can also design and implement their own ideas, apart from the list of experiments mentioned above.

- Note: A minimum of 8 to 10 experiments must be completed by the students.

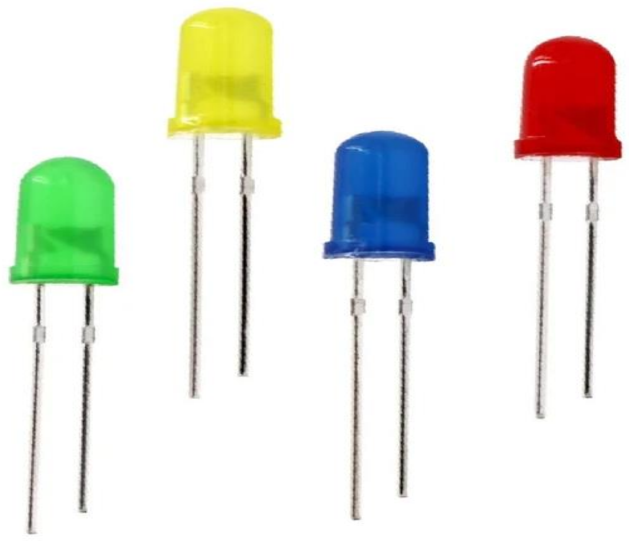
Prerequisites of Tinkering Laboratory

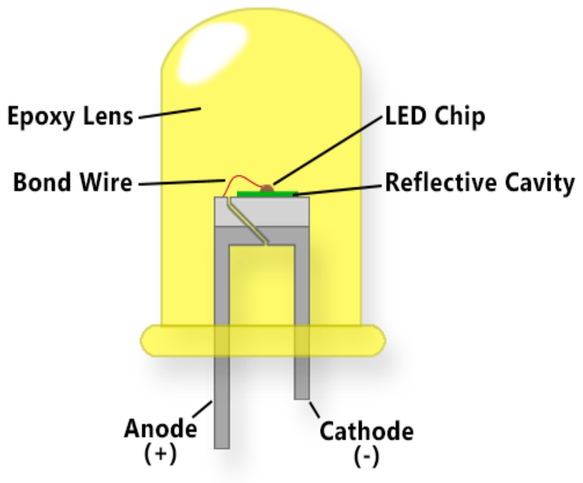
The concept of “tinkering” has gained significant importance both globally and in India, particularly in the context of education, innovation, and skill development. Tinkering involves hands-on learning, experimentation, and problem-solving through practical experiences. So that studying various components is mandatory to explore some of the experiments to have a live experience in the domain of Engineering. Hence the following list of components and supporting briefing is inevitable to gain practical hands on experience.

LED

LED (Light Emitting Diode) is a diode which when connected in forward bias generates light. The light generated depends on the colour for which the LED is designed.

A LED contains two leads - positive terminal (longer one) and negative terminal (shorter one). LED will glow when positive pin of LED is connected to +ve terminal of the battery and negative pin of LED is connected to -ve terminal of the battery. The operating voltage of an LED is between 1.8 and 3.3 volts.


LEDs



LED internal structure

How to Use:

To make a simple connection, we can connect the LED with a battery to make it glow. The positive lead of the LED needs to be connected to the positive terminal of the battery, and negative lead to the negative terminal. We need to make sure that the voltage rating matches the battery voltage. Otherwise, we can use resistors to provide the required voltage.

Coin Cell/Rechargeable 9V batteries

Coin cells and rechargeable 9V batteries are commonly used power sources in various low-power and portable electronic devices. **Coin cells**, also known as button cells, are small, round, flat batteries typically used in watches, calculators, hearing aids, and small sensors. They are usually non-rechargeable (e.g., CR2032), although rechargeable variants like LIR2032 exist. These batteries provide a stable voltage (typically 3V) and are valued for their compact size and long shelf life.

On the other hand, **rechargeable 9V batteries** are used in applications requiring higher voltage and capacity, such as smoke detectors, handheld instruments, multimeters, and wireless microphones. These batteries come in different chemistries, including NiMH (Nickel Metal Hydride) and Li-ion (Lithium-ion), offering the advantage of being reused multiple times, thereby reducing waste and long-term cost.

Rechargeable 9V battery

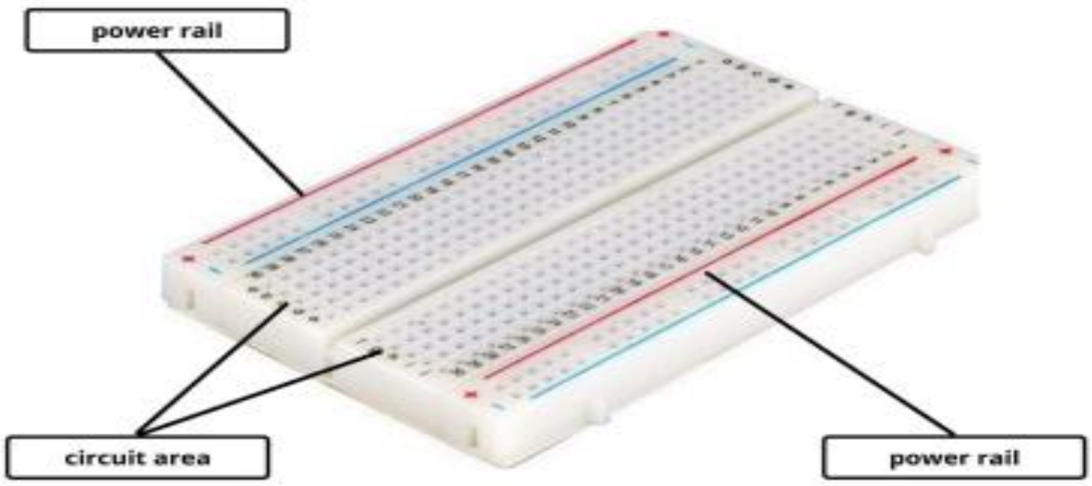
Coin cells

How to Use:

- **Coin Cell**: Insert the coin cell into the battery holder or compartment with the correct polarity (usually marked with "+" and "–"). Ensure secure contact. Non-rechargeable coin cells should not be recharged to avoid leakage or explosion. If using rechargeable types, only use a compatible charger designed for that specific cell type.
- **Rechargeable 9V Battery**: Charge the battery fully using a proper 9V rechargeable battery charger before first use. Insert the battery into the device, aligning the terminals correctly. Recharge the battery once its performance drops or the device indicates low power. Avoid overcharging or exposing it to extreme temperatures.

Breadboards

Breadboard is a thin plastic board which is used to hold components to test out the circuit temporarily before making the actual final circuit connections. A breadboard has internal connections which makes it very handy to connect different components based on the requirements. There are two major types of size in which breadboards are available:



Breadboard

Normal size: It has 800 pins in it to be used for making connections. It is suitable for making complex circuit prototypes.

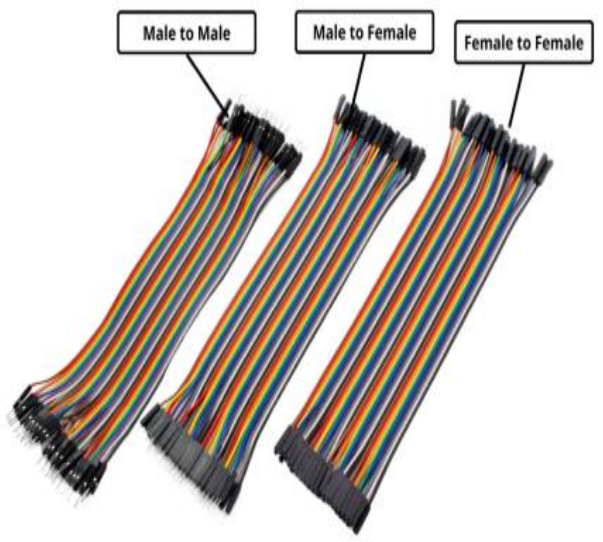
Mini Breadboard size: It has 400 pins in it to be used for making connections. It is suitable for comparatively smaller and simpler circuit designs.

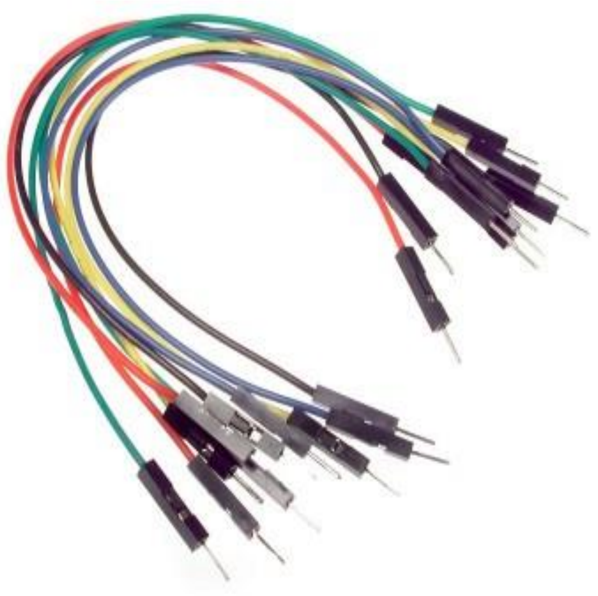
How to Use:

To use a breadboard, one should first have an idea regarding which pins are short circuit and which are open circuit. In a breadboard, usually, when held horizontally, the upper two rows and lower two rows are divided into 4 short-circuited connections each. The other Rows have pins short circuited vertically. Once identified, you can utilize pins of your choice to put components for making connections, and then test them accordingly. A very simple example could be to connect a LED light with a battery and switch.

Jumper Cables

Jumper cables are used very often in making connections. They're very simple to use as compared to normal wires, because you don't need to strip the wire, make connection and then twist it to make it intact. Just plugging in to the breadboard does the job in case of jumper wires. There are majorly two ends in a jumper wires, such as male and female. These jumper cables can be of three types such as male-male, male-female, female-female.





Jumper Cables

How to Use:

A jumper wire is used when a connection is being made. The male connector of jumper wire can fit into the breadboard easily by just plugging it in.

Multimeter

A **multimeter** is a versatile electronic measuring instrument commonly used to test and troubleshoot electrical and electronic circuits. It combines several measurement functions in one unit, typically including the ability to measure voltage (volts), current (amperes), and resistance (ohms). Some advanced digital multimeters (DMMs) also include additional features such as continuity testing, diode testing, capacitance measurement, and temperature sensing. Multimeters come in two main types: **analog** (with a moving needle display) and **digital** (with a numeric LCD display). They are widely used by electricians, technicians, and engineers for both domestic and industrial applications.

Analog

Digital

How to Use:

Turn on the Multimeter **– Set the dial to the desired measurement (e.g., voltage, current, resistance).**

1. **Connect the Probes** – Insert the black probe into the COM (common) port and the red probe into the port labeled for the specific measurement (e.g., VΩmA for voltage or resistance).
2. **Measure Voltage** – For DC voltage (e.g., battery), set the dial to DCV and place the probes across the terminals. For AC voltage (e.g., wall outlet), set it to ACV.
3. **Measure Current** – Move the red probe to the port marked for current (A or mA). Break the circuit and connect the multimeter in series to measure current flow.
4. **Measure Resistance** – Set the dial to the Ω symbol. Touch the probes across the resistor or component to read resistance.
5. **Continuity Test** – Select the continuity mode (usually marked with a sound wave symbol). If there is a complete path, the meter beeps.
6. **Safety Tips** – Always start with the highest range available, never test resistance or continuity on a live circuit, and check probe condition before use.

Wire Stripper

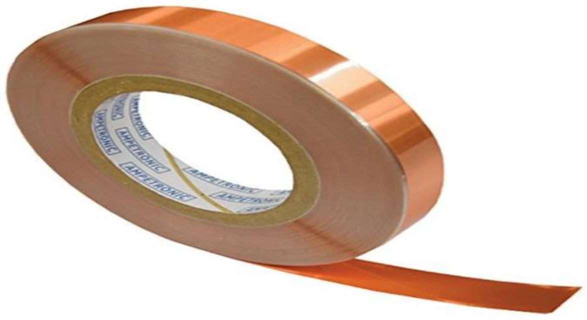
A **wire stripper** is a hand-held tool used by electricians and technicians to remove the protective insulation from electric wires, enabling a proper connection or termination. It is an essential tool for electrical work, ensuring safe and efficient stripping without damaging the internal conductor. Wire strippers come in various forms, including manual, automatic, and adjustable types, each suited for different wire gauges and insulation materials. Most wire strippers feature multiple notches for different wire sizes, a cutting blade, and sometimes crimping capabilities for terminals.

How to Use:

1. **Select the Right Notch:** Identify the wire gauge and select the corresponding notch on the stripper.
2. **Insert the Wire:** Place the wire into the notch, positioning it where you want to remove the insulation.
3. **Squeeze the Handles:** Gently squeeze the handles together. The blades will cut through the insulation without damaging the wire.
4. **Pull the Insulation Off:** While still squeezing, pull the stripper towards the end of the wire to slide off the insulation cleanly.
5. **Inspect the Wire:** Ensure the copper or aluminium conductor is clean and undamaged for proper electrical contact.

Conductive (Copper) Tape

Conductive (copper) tape is a thin strip of copper with adhesive backing that is often used in electronics, DIY projects, and electromagnetic shielding applications. The adhesive used is typically electrically conductive, allowing electricity to pass through the tape even when it's attached to a surface. Copper tape is valued for its excellent electrical conductivity, flexibility, and ease of use. It is commonly used for making low-voltage circuits, repairing printed circuit boards (PCBs), creating capacitive touch sensors, grounding in electronic devices, and reducing electromagnetic interference (EMI) in enclosures or cables. Additionally, it finds applications in educational kits and artistic projects involving interactive electronics.



Copper Tape

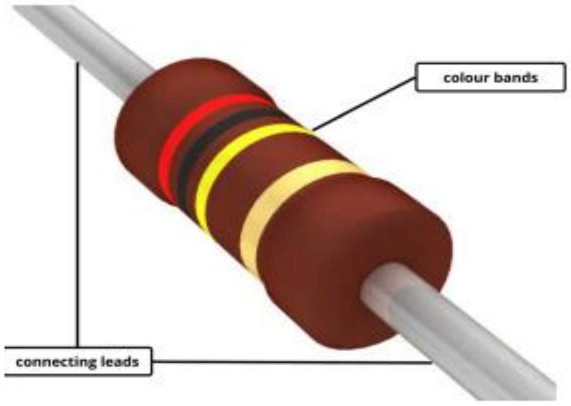
How to Use:

Cut the Required Length: Use scissors to cut a piece of copper tape to the desired length for your application.

1. **Peel Off the Backing:** Carefully peel off the paper or plastic liner to expose the adhesive side.
2. **Apply to Surface:** Stick the tape onto the desired clean and dry surface, such as paper, plastic, wood, or circuit board.
3. **Ensure Good Contact:** Press down firmly to ensure solid contact, especially if you’re connecting it to another conductive component.
4. **Overlap for Connectivity:** When joining two strips of copper tape, slightly overlap them to maintain electrical continuity.
5. **Soldering (Optional):** For a more permanent or reliable connection, you can solder components like LEDs or resistors directly onto the tape.

Resistor

A resistor is such a device which finds use in almost everything related to electronics. A resistor is used in lamps, fans, toaster, microwave, etc. Technically, a resistor is a two terminal component that is used to obtain electrical resistance in circuits. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages and more. Resistors are usually made of metals or other components that have a very high resistivity as compared to common connecting metals like copper and aluminium. These metals are then coated with an insulating material.



Resistor

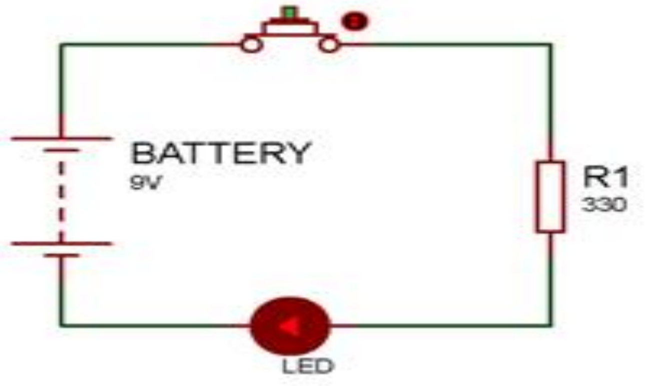
How to Use:

To use a resistor, just select a resistor of appropriate ratings and put it in the circuit. Resistors don't have any polarity of terminals so they can be connected either way in the circuit and made to work. The connections once made can be soldered to keep them intact.

Push button

A **push button** is a simple electromechanical switch used to control a circuit. When pressed, it either completes (closes) or interrupts (opens) an electrical circuit, depending on its configuration. Push buttons are commonly used in electronic devices, control panels, household appliances, and embedded systems like Arduino projects. They are typically momentary, meaning they only stay activated while being pressed, returning to their default state when released.

How to Use a Push Button:



Push button with LED circuit

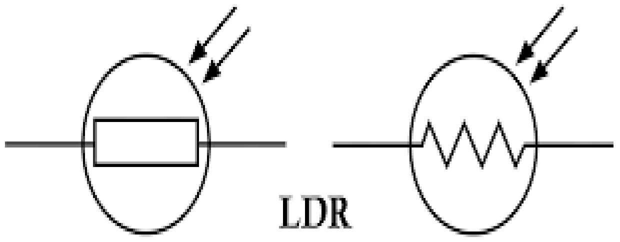
To use a push button in a circuit:

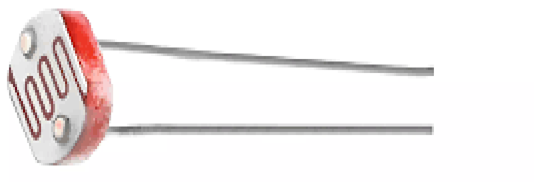
1. **Connect one leg of the push button** to a digital input pin on a microcontroller (e.g., Arduino) or to the control input of another electronic component.
2. **Connect the opposite leg** to the ground (GND) or to a pull-up resistor and voltage supply, depending on the logic needed.
3. **Use pull-up or pull-down resistors** to ensure the pin reads a stable HIGH or LOW signal when the button is not pressed.
4. In programming, **monitor the input pin's state** (HIGH or LOW) to detect button presses and trigger desired actions like turning on an LED, sending a signal, or activating a motor.

LDR

An LDR, or Light Dependent Resistor, is a type of resistor whose resistance varies depending on the amount of light falling on it. It's also known as a photoresistor or photocell. When light shines on an LDR, its resistance decreases. Conversely, when it's dark, the resistance increases. This property makes LDRs useful in various light-sensitive applications. Here's a more detailed explanation:

- LDRs are made of semiconductor materials that exhibit photoconductivity.
- Photoconductivity means that when light strikes the material, it increases the material's electrical conductivity, thereby lowering its resistance.
- In the dark, the resistance of an LDR can be very high (even millions of ohms), and in bright light, it can be quite low (a few hundred ohms).





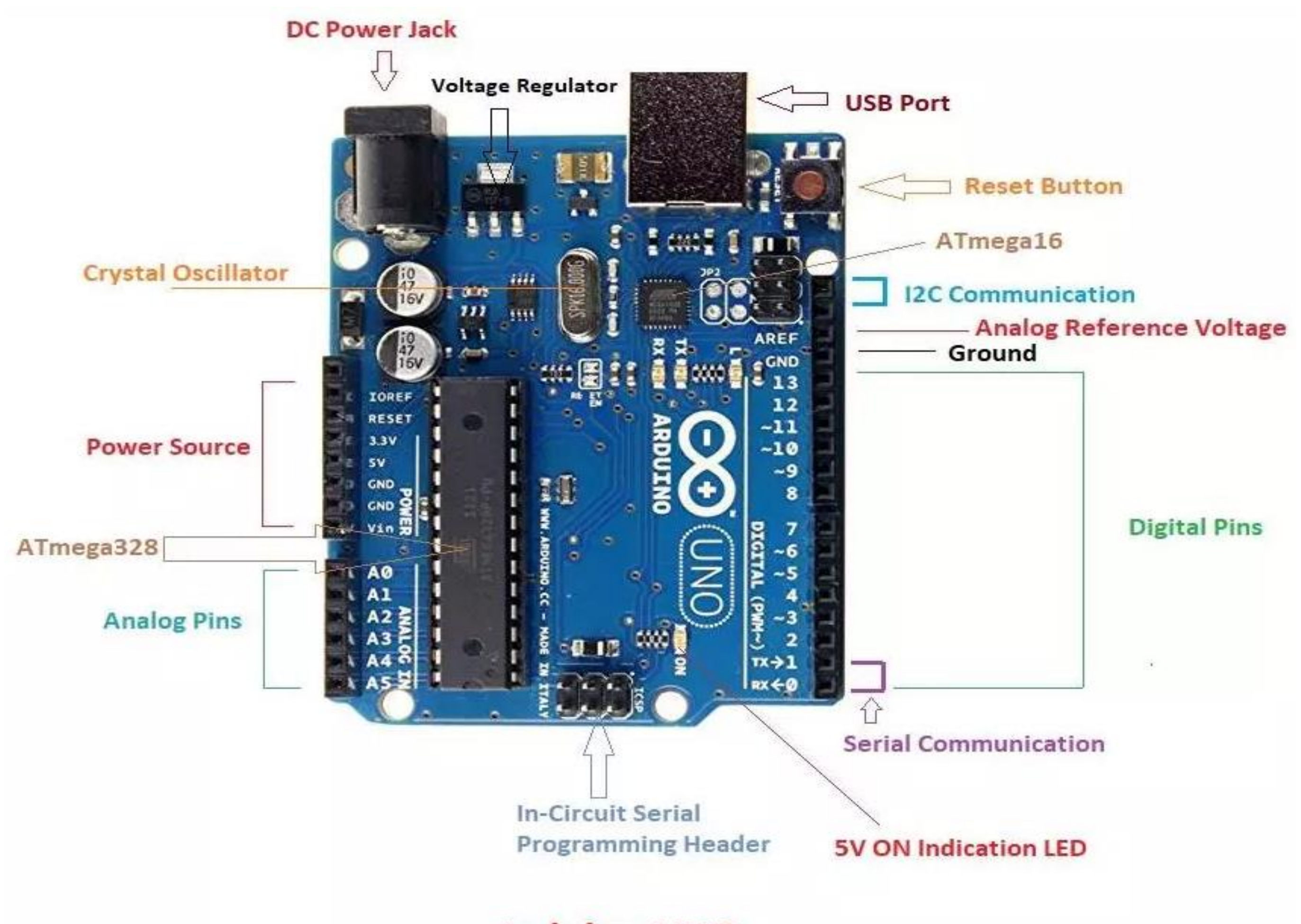
LDRs are used in a wide array of applications, including:

- **Automatic Lighting Control:** Turning streetlights on at dusk and off at dawn.
- **Light Meters:** Measuring light levels for photography or other purposes.
- **Burglar Alarms:** Triggering an alarm when a light beam is broken.
- **Clocks:** Some alarm clocks use LDRs to adjust the brightness of the display.
- **Camera Exposure Systems:** LDRs can help to determine the correct exposure settings.
- **Consumer Electronics:** LDRs are used in screens and displays to adjust brightness automatically.
- **Industrial Applications:** Monitoring light intensity in manufacturing, process control, and quality assurance.

Ardino Uno Board

all Arduino boards differ from each other, there are several key components that can be found on practically any Arduino. Let's take a look at the image below:

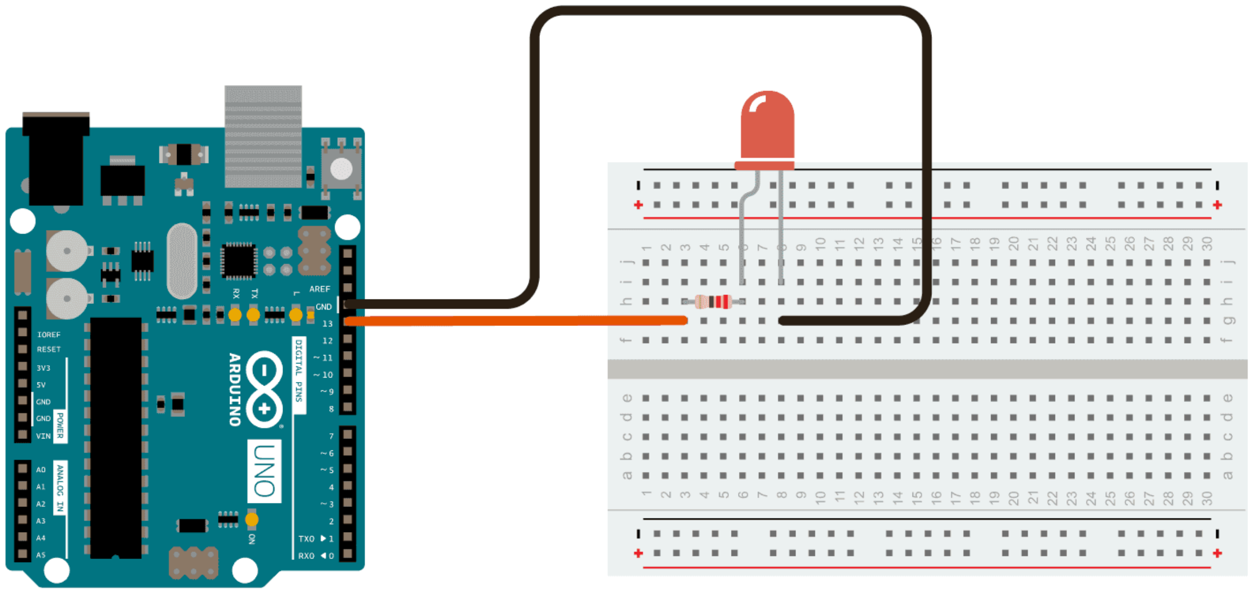
- Microcontroller - this is the brain of an Arduino, and is the component that we load programs into. Think of it as a tiny computer, designed to execute only a specific number of things.
- USB port - used to connect your Arduino board to a computer.
- USB to Serial chip - the USB to Serial is an important component, as it helps translating data that comes from e.g. a computer to the on-board microcontroller. This is what makes it possible to program the Arduino board from your computer.
- Digital pins - pins that use digital logic (0, 1 or LOW/HIGH). Commonly used for switches and to turn on/off an LED.
- Analog pins - pins that can read analog values in a 10 bit resolution (0-1023).
- 5V / 3.3V pins- these pins are used to power external components.
- GND - also known as ground negative or simply, is used to complete a circuit, where the electrical level is at 0 volt.
- VIN - stands for Voltage In, where you can connect external power supplies.



Arduino UNO

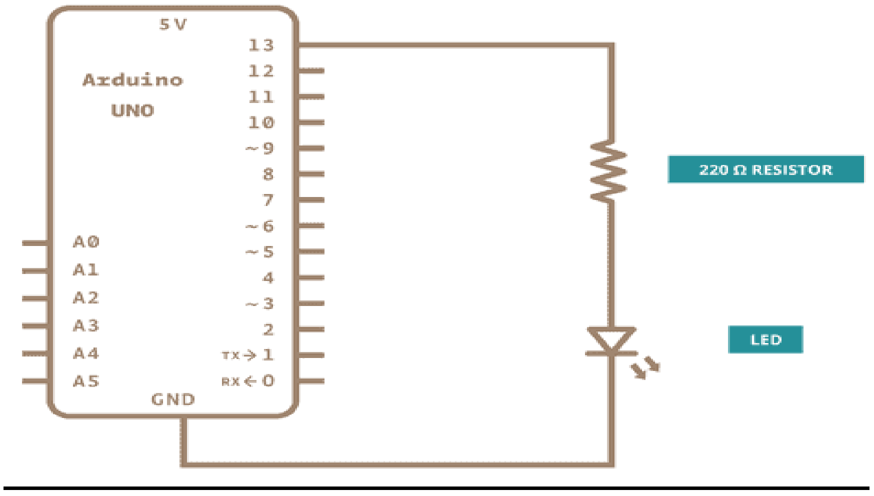
Basic Operation:

Most Arduino boards are designed to have a single program running on the microcontroller. This program can be designed to perform one single action, such as blinking an LED. It can also be designed to execute hundreds of actions in a cycle. The scope varies from one program to another.



The program that is loaded to the microcontroller will start execution as soon as it is powered. Every program has a function called "loop". Inside the loop function, you can for example:

- Read a sensor.
- Turn on a light.
- Check whether a condition is met.
- All of the above.



The speed of a program is incredibly fast, unless we tell it to slow down. It depends on the size of the program and how long it takes for the microcontroller to execute it, but it is generally in microseconds (one millionth of a second).

Circuit Basics:

Circuits consist of at least one active electronic component, and a conductive material, such as wires, so that current can pass through. When working with an Arduino, you will in most cases build a circuit for your project.

A simple example of a circuit, is an LED circuit. A wire is connected from a pin on the Arduino, to an LED via a resistor (to protect the LED from high current), and finally to the ground pin (GND). When the pin is set to a HIGH state, the microcontroller on the Arduino board will allow an electric current to flow through the circuit, which turns on the LED. When the pin is set to a LOW state, the LED will turn off, as an electric current is not flowing through the circuit. Circuits are typically represented as schematics, which are the blueprints for your circuit. The image below shows the schematic's representation of the same circuit shown in the image above.

ESP32

ESP32 is a chip that provides Wi-Fi and (in some models) Bluetooth connectivity for embedded devices – in other words, for IoT devices. While ESP32 is technically just the chip, the modules and development boards that contain this chip are often also referred to as "ESP32" by the manufacturer. The original ESP32 chip had a single core Tensilica Xtensa LX6 microprocessor. The processor had a clock rate of over 240 MHz, which made for a relatively high data processing speed.

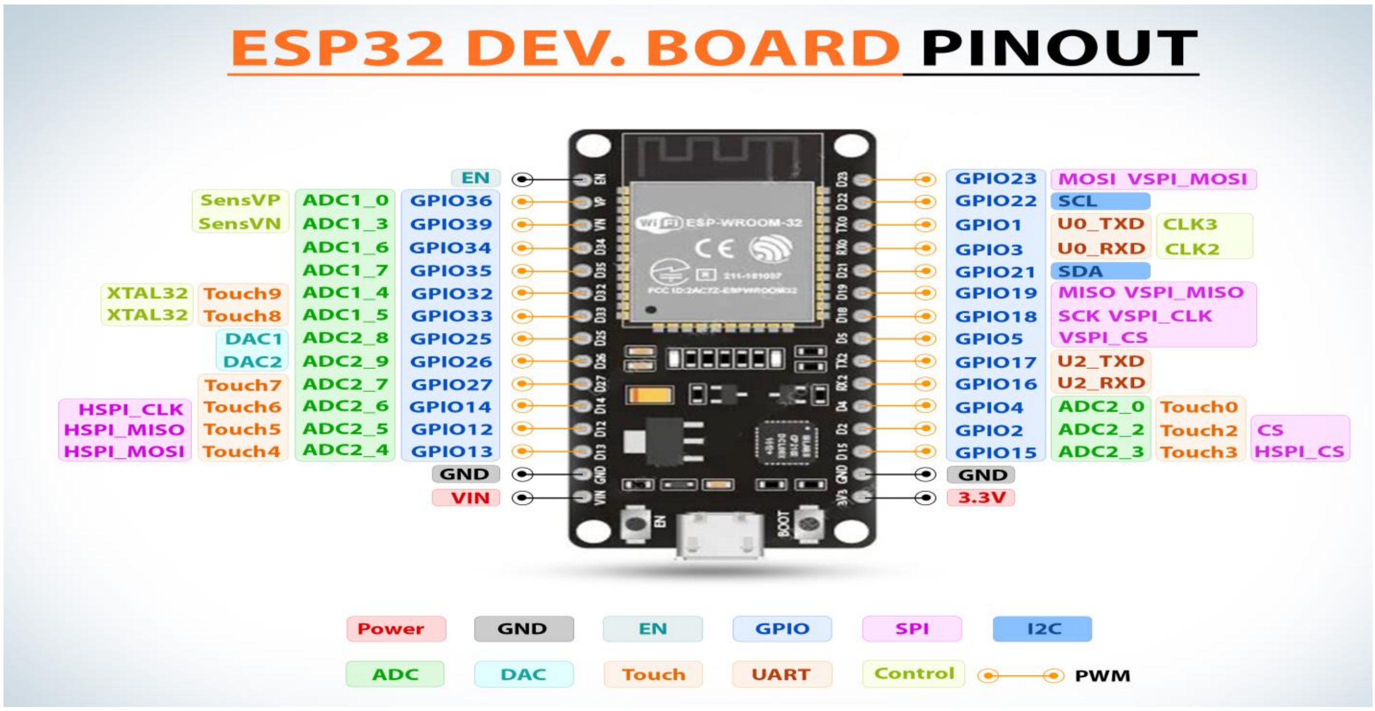
More recently, new models were added, including the ESP32-C and -S series, which include both single and dual core variations. These two series also rely on a Risc-V CPU model instead of Xtensa. The newer models are available with combined Wi-Fi and Bluetooth connectivity, or just Wi-Fi connectivity. There are several different chip models available, including:

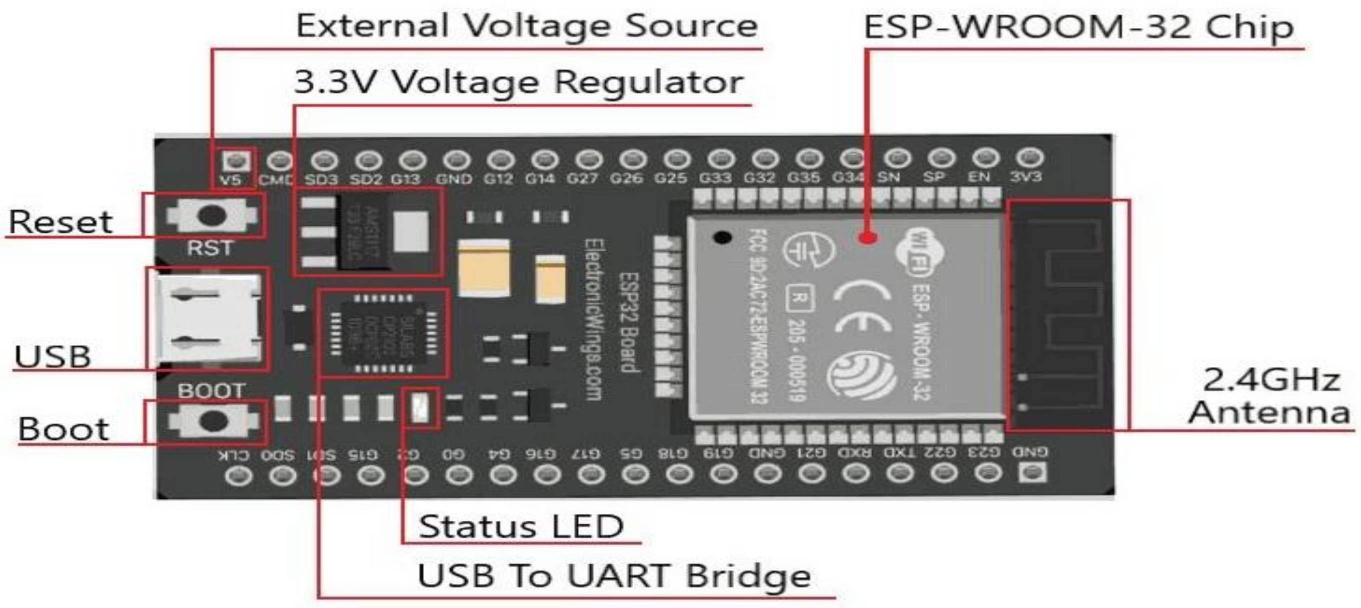
- ESP32-D0WDQ6 (and ESP32D0WD)
- ESP32-D2WD
- ESP32-S0WD
- System in package (SiP) – ESP32-PICO-D4
- ESP32 S series
- ESP32-C series
- ESP32-H series

The ESP32 is most commonly used for mobile devices, wearable tech, and IoT applications, such as Nabto Edge. Moreover, since Mongoose OS introduced an ESP32 IoT Starter Kit, the ESP32 has gained a reputation as the ultimate chip for hobbyists and IoT developers. It's suitable for commercial IoT, and its capabilities and resources have grown impressively over the past four years.

ESP32 features and specifications:

ESP-32	DESCRIPTION
Core	2
Architecture	32 bits
Clock	Tensilica Xtensa LX106 160-240MHz
WiFi	IEEE802.11 b/g/n
Bluetooth	Yes - classic & BLE
RAM	520KB
Flash	External QSPI - 16MB
GPIO	22
DAC	2
ADC	18
Interfaces	SPI-I2C-UART-I2S-CAN





And here’s a more detailed summary:

- **Processors** – The ESP32 uses a Tensilica Xtensa 32-bit LX6 microprocessor. This typically relies on a dual core architecture, with the exception of one module, the ESP32-S0WD, which uses a single-core system.
- **Wireless connectivity** – The ESP32 enables connectivity to integrated Wi-Fi through the 802.11 b/g/n/e/i/. Moreover, Bluetooth connectivity is made possible with the v4.2 BR/EDR, and the series also features Bluetooth low energy (BLE).
- **Memory** – Internal memory for the ESP32 is as follows. ROM: 448 KB (for booting/core functions), SRAM: 520 KB (for data/instructions),
- **External flash and SRAM** – ESP32 supports up to four 16 MB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers’ programs and data.
- **Security** – The ESP32 supports all IEEE 802.11 standard security features, including WFA, WPA/WPA2 and WAPI. Moreover, ESP32 has a secure boot and flash encryption.

ESP32 functions: ESP32 has many applications when it comes to IoT. Here are just some of the IoT functions the chip is used for**:**

- **Networking**: The module's Wi-Fi antenna and dual core enables embedded devices to connect to routers and transmit data.
- **Data processing**: Includes processing basic inputs from analog and digital sensors to far more complex calculations with an RTOS or non-OS software development kit (SDK).
- **P2P connectivity**: Creates direct communication between different ESPs and other devices using IoT P2P connectivity.
- **Web server**: Provides access to pages written in HTML or development languages.

ESP32 Applications: The ESP32 modules are commonly found in the following IoT devices:

- Smart industrial devices, including programmable logic controllers (PLCs)
- Smart medical devices, including wearable health monitors
- Smart energy devices, including HVAC and thermostats
- Smart security devices, including surveillance cameras and smart locks

Tinkercad

Tinkercad is a beginner-friendly, browser-based CAD tool that has become a favourite among educators, students, and hobbyists. Its intuitive drag-and-drop interface makes it ideal for anyone curious about 3D design—especially those just starting out or exploring 3D printing for the first time. Tinkercad makes bringing creative ideas to life easy, allowing users to experiment with basic shapes and concepts without needing advanced design skills. Its simplicity and accessibility make it perfect for hands-on learning and creative exploration.

Tinkercad provides keyboard and mouse functionalities that are common across much of the software we use day to day. Familiar actions like undoing work by pressing Ctrl + z and resizing boxes by dragging their corners are present in Tinkercad.

Scalability and interoperability set Autodesk CAD software apart. Beginners can start with Tinkercad's easy drag-and-drop interface, then transition to professional-grade tools like Fusion and AutoCAD as their skills grow. A few links are available to explore the features and practice the Tinkercad software as per the given below:

1. https://www.tinkercad.com/learn
2. https://www.tinkercad.com/blog/official-guide-to-tinkercad-circuits
3. https://promoambitions.com/tinkercad/
4. https://www.tinkercad.com/blog/official-guide-to-tinkercad-circuits
5. https://www.vla.org/assets/Conference_Session_Docs_Slideshows/2017_VLAAnnual/PresenterMaterials/tinkering%20with%20tinkercad%20-%20a%20beginners%20guide%20to%20creating%203d%20printer%20designs%20-%20michael%20hibben.pdf
6. https://www.podareducation.org/uploads/nagpurbesa/documents/3dprintingnotes2019/v_3d%20printing_2019_20.pdf

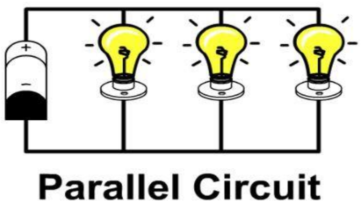
Expt. No.: 1	PARALLEL AND SERIES CIRCUITS USING BREADBOARD FOR ANY APPLICATION USING TINKERCAD

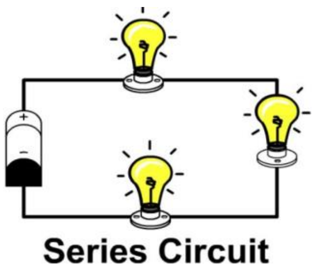
Aim: To make parallel and series circuits using breadboard for any application of your choice.

<u>Components Required (Tinkercad)</u> :

- LED bulbs
- Coin Cell/Rechargeable 9 V batteries
- Breadboard
- Jumper wires
- Multimeter
- Wire Stripper
- Conductive (Copper) Tape
- Resistors (330 ohms)
- Push button

CIRCUIT CONNECTIONS:



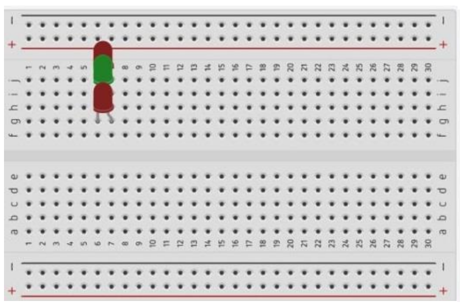


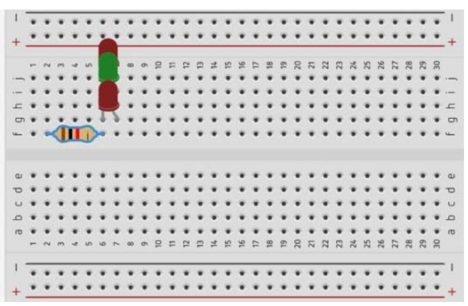
Steps to make a Parallel circuit:

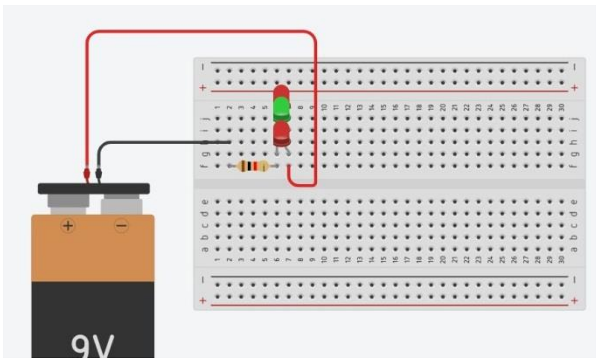
Step 1: Insert a LED bulbs into the breadboard as shown in the image

Step 2: Insert the Resistor into the breadboard and connect it to the negative terminal of LED

Step 3: Connect the battery Red wire (+) to the positive side of the LED and connect Black wire (-) to the other end of the resistor.





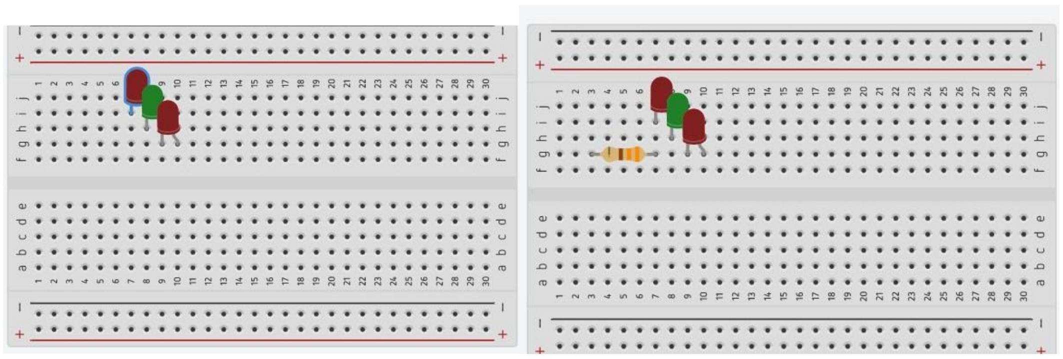


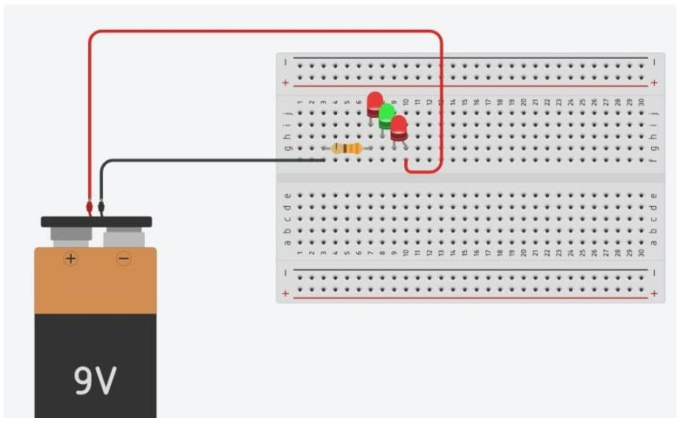
Steps to make a Series circuit:

Step 1: Insert a LED bulbs into the breadboard as shown in the image

Step 2: Insert the Resistor into the breadboard and connect it to the negative terminal of LED

Step 3: Connect the battery Red wire (+) to the positive side of the LED and connect Black wire (-) to the other end of the resistor.





Procedure:

1. Log in to Tinkercad.

2. Click on "Circuits" > "Create new Circuit".

3. From the Components panel, drag and drop the following to the workspace.

4. Connect the Components in the bread board using different coloured wires as shown in the Figures.

5. Click "Start Simulation".

6. Note down the Voltage, Current across the connections using Multimeter.

Result:

Hence the Voltage and Current readings were observed by making Series and Parallel Connections with LED Lights on Bread Board using Tinkercad.

Exp No.: 02	DESIGN AND 3D PRINT A WALKING ROBOT

Aim: To design, model, and 3D print a simple walking robot mechanism using Tinkercad, and understand the principles of mechanical linkages and basic robotics.

Tools and Materials Required:

- Computer with internet access
- Tinkercad account (https://www.tinkercad.com)

Theory:

A **walking robot** typically moves using leg mechanisms rather than wheels. Many simple walking robots use **crank-slider** or **four-bar linkage** mechanisms to imitate walking motion.

Tinkercad allows creating mechanical models using basic shapes. Additionally, the **Tinkercad Circuits** platform can be used for simulating motion (optional extension).

Procedure:

1. Create a New Tinkercad Design

- Visit https://www.tinkercad.com, login, and click "Create New Design."

2. Design the Robot Body

- Drag a **box** shape for the body (approx. 80mm x 40mm x 10mm).
- This will serve as the main chassis.

3. Design Legs

Each leg can be made using 2 parts:

- **Upper leg** (e.g., rectangle 30mm x 10mm x 4mm)
- **Lower leg** (e.g., rectangle 40mm x 10mm x 4mm)

Steps:

- Create a **circular joint** using a **cylinder** (diameter: 5mm).
- Place the joint between upper and lower leg sections.
- Use **duplicate + mirror** to make symmetrical legs.
- Create **at least 2 legs**, depending on the design.

4. Attach Legs to Body

- Use **cylindrical joints** to connect legs to the side of the body.

- Ensure spacing and alignment allow free movement.
- You may add holes for screws (3mm) or axle pins (for motor shaft).

5. Design Feet

- Use flattened **box shapes** or **rounded pads** to create stable feet.
- Attach to the bottom of the lower leg.

6. Motor Mount (Optional)

- Create a small bracket on the body to hold a **DC motor** or **servo**.
- Include screw holes or slots to fit the motor securely.

7. Assemble All Parts

- Group and align parts using Tinkercad tools.
- Use **holes** and **pegs** for movable joints.

8. Label Parts (Optional)

- Add text (e.g., "Left Leg", "Body", etc.) using the **Text** tool.

Result:

A functional walking robot was successfully designed using Tinkercad. The assembled model demonstrates mechanical movement through legs.

Exp No.: 03	DESIGN AND 3D PRINT A ROCKET

Aim: To learn the basics of 3D modelling using Tinkercad by designing a simple rocket and exporting the model for 3D printing.

Tools and Materials Required:

- Computer with internet access
- Tinkercad account (https://www.tinkercad.com)

Theory:

Tinkercad is an online 3D modelling tool that uses simple shapes to create complex models. It is perfect for beginners and educators.

In this lab, you will:

- Design a model rocket using basic shapes.
- Learn how to group, align, and manipulate objects.

Procedure: Designing the Rocket in Tinkercad

1. **Login to Tinkercad**
   Go to https://www.tinkercad.com and sign in or create a free account.
2. **Start a New Design**
   - Click “Create New Design.”
3. **Create the Rocket Body**
   - Drag a **cylinder** from the shape panel to the workplane.
   - Adjust the **height** (e.g., 80 mm) and **diameter** (e.g., 20 mm) from the shape inspector.
4. **Add the Nose Cone**
   - Drag a **cone** shape and place it on top of the cylinder.
   - Resize it to match the cylinder’s diameter (20 mm) and a height of around 30 mm.
   - Use the align tool to center the cone on top of the cylinder.
   - Group both shapes.
5. **Add Fins**
   - Use a **wedge** or **box** shape to design a fin.

    - Resize and rotate to attach it to the lower part of the rocket.
    - Duplicate the fin and place them symmetrically (e.g., 3 or 4 fins using copy + rotate).
    - Group all fins with the body.
6. **Optional: Add a Hollow Chamber (for Parachute)**
    - Use a **cylinder hole** shape to subtract from the body.
    - Align and group to subtract the hole.
7. **Label or Add Features**
    - Add text or other features if desired.
8. **Export the Model**
    - Click on **Export > STL** to download your 3D model.

Results:

The rocket was successfully designed in Tinkercad. All components (nose, body, fins) were dimensionally accurate and properly assembled.

Exp No.: 04	TEMPERATURE & HUMIDITY MONITORING SYSTEM (DHT11 + LCD)

Aim:

to create a temperature and humidity monitoring system using an Arduino, a DHT11 sensor, and an LCD display within the Tinkercad environment

Components/tools needed:

- Arduino Uno
- DHT11 Temperature and Humidity Sensor
- 16x2 LCD Display
- Breadboard
- Jumper Wires
- 220Ω Resistors (for the LCD)
-

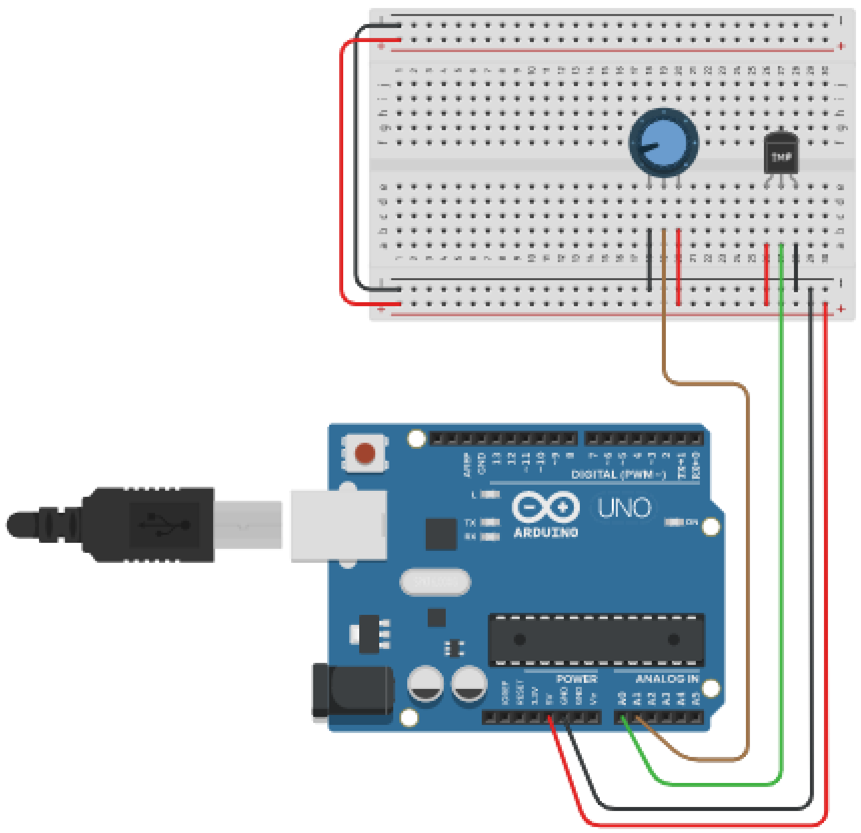
Setting up the circuit in Tinkercad:

- Arduino and Breadboard: Open Tinkercad and create a new circuit. Drag and drop an Arduino Uno and a breadboard onto the workspace.
- DHT11 Sensor: Connect the DHT11 sensor to the Arduino as follows:
  - VCC pin to the Arduino's 5V pin.
  - GND pin to the Arduino's GND pin.
  - DATA pin to a digital pin on the Arduino (e.g., Digital Pin 2). Consider using a 10k resistor between the DATA pin and the 5V pin for stability, according to Instructables.
- LCD Display: Connect the LCD to the Arduino as shown below (you may need to adjust the specific digital pins based on your code and available pins):
  - VSS to GND
  - VDD to 5V
  - VO to a potentiometer for contrast adjustment (if available in Tinkercad) or a resistor for fixed contrast.
  - RS to Digital Pin 12
  - RW to GND
  - EN to Digital Pin 11
  - D4 to Digital Pin 5
  - D5 to Digital Pin 4
  - D6 to Digital Pin 3
  - D7 to Digital Pin 2
  - A (backlight anode) to 5V
  - K (backlight cathode) to GND

Writing the Arduino code:

```
/*
This code records the temperature through testing the mV put out by the sensor.
It records in both Celcius and Fahrenheit.
It can only detect from -40 degrees C to 125 degrees C or -40 degrees F to 257 degrees F
The Humidity is simulated by a potentiometer by being mapped into percentages
*/

const int analogIn = A0;
int humiditysensorOutput = 0;
// Defining Variables
int RawValue= 0;
double Voltage = 0;
double tempC = 0;
double tempF = 0;

void setup(){
  Serial.begin(9600);
  pinMode(A1, INPUT);
}

void loop(){

  RawValue = analogRead(analogIn);
  Voltage = (RawValue / 1023.0) * 5000; // 5000 to get millivots.
  tempC = (Voltage-500) * 0.1; // 500 is the offset
  tempF = (tempC * 1.8) + 32; // convert to F
  Serial.print("Raw Value = " );
  Serial.print(RawValue);
  Serial.print("\t milli volts = ");
  Serial.print(Voltage,0); //
  Serial.print("\t Temperature in C = ");
  Serial.print(tempC,1);
  Serial.print("\t Temperature in F = ");
  Serial.println(tempF,1);
  humiditysensorOutput = analogRead(A1);
  Serial.print("Humidity: "); // Printing out Humidity Percentage
  Serial.print(map(humiditysensorOutput, 0, 1023, 10, 70));
  Serial.println("%");

  delay(5000);  //iterate every 5 seconds

}
```

Circuit Diagram:



Link : https://www.tinkercad.com/things/jCoseKCUo3i-20-dht11-temperature-and-humidity-sensor

Result: Successfully created a functional temperature and humidity monitoring system using Arduino, a DHT11 sensor, and an LCD display in Tinkercad

Exp No.: 05	WATER LEVEL DETECTION AND ALERT SYSTEM

Aim: To create a virtual water level indicator using an ultrasonic sensor and LCDs

Components needed:

- Arduino Uno board
- Potentiometer for LCD brightness
- 16*2 LCD display
- DC motor water pump
- Ultrasonic distance Sensor
- Jumper Wires

Setting up the circuit in Tinkercad

- Place the Arduino and breadboard on the Tinkercad workspace.
- Connect the ultrasonic sensor's VCC and GND to the Arduino's 5V and GND pins, respectively.
- Connect the sensor's Trig and Echo pins to the Arduino's digital pins (e.g., 10 and 11).
- Connect the LCD's VCC and GND to the Arduino's 5V and GND, and the SDA and SCL pins to the Arduino's A4 and A5 pins.
- Connect the LEDs (if used) to the Arduino's digital pins (e.g., 4, 5, 6) through resistors.
- Connect the buzzer (if used) to the Arduino's digital pin 8 and GND.

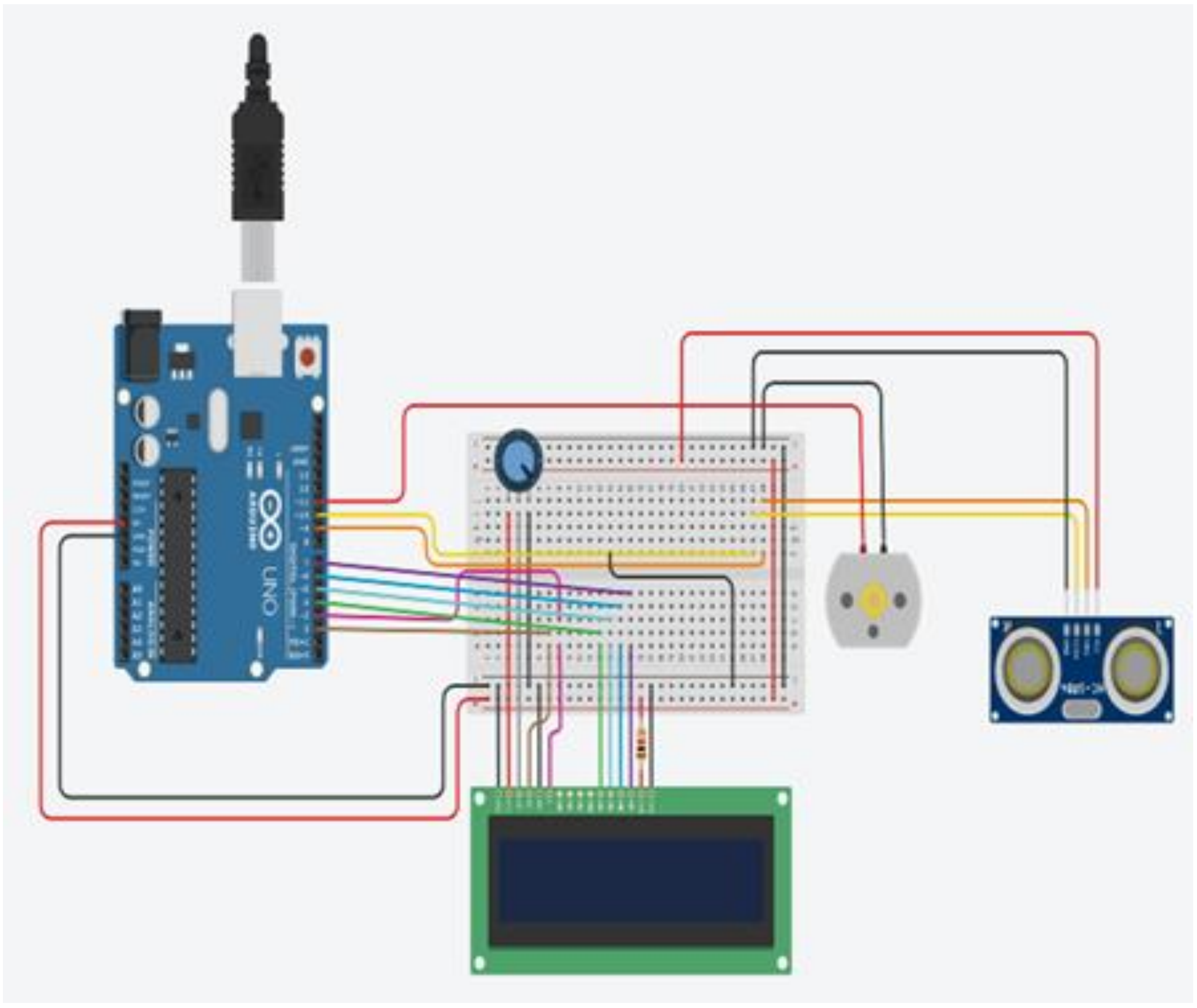
Tinkercad Code:

- **Include necessary libraries:** For the LCD and potentially other components.
- **Define pins:** Assign Arduino pins to the sensor, LEDs, and buzzer.
- **Initialize the sensor and LCD:** Set up the sensor for distance measurement and initialize the LCD.
- **Read sensor data:** Use the ultrasonic sensor to measure the distance to the water surface.
- **Convert distance to water level:** Calculate the water level based on the measured distance and the tank dimensions.
- **Display water level:** Output the calculated water level on the LCD or control the LEDs/buzzer based on the level.
- **Code is given below:**

```
 //for 600 Litre Water Tank
/*#define trigPin 9;
#define echoPin 10;*/
long duration;

#include<LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);

long UltrasonicDistance(int trigPin, int echoPin)
```

```
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  //float dist;
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(2);
  digitalWrite(trigPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  return pulseIn(echoPin, HIGH);

}

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.clear();
  lcd.print("Water Level(%)");

}
void loop()
{
  float dist = 0.01723 * UltrasonicDistance(9,10);
  int level = map(dist, 334 , 0, 0 , 100);
  lcd.setCursor(0,1);
  lcd.print(level);
  lcd.setCursor(3,1);
  lcd.print("%");
  int speed=10;

  if( level >=99)
  {
   analogWrite(11,0);
  }
  else if( level <95)
  {
    analogWrite(11,speed);
  }

  else
  {
   analogWrite(11,0); // this is for safety in case.
  }

}
```

Circuit Board:



Link: https://www.tinkercad.com/things/9DZbcf79kT8-

Simulation and Testing:

- Start the simulation in Tinkercad.
- Observe the sensor data on the LCD and the LEDs (if used).
- Adjust the simulated water level to test the functionality of the indicator.

Result: Successfully created a functional water level indicator in Tinkercad

Exp No.: 06	AUTOMATIC PLANT WATERING SYSTEM

Aim: To design an automatic plant watering system controlled by a temperature sensor in Tinkercad.

Description:
An automatic plant watering system based on temperature in Tinkercad can be created by combining an Arduino Uno, a temperature sensor, a soil moisture sensor, a relay module, a water pump, and a power supply. The system will monitor the temperature and soil moisture levels, and activate the water pump when both exceed predefined thresholds, ensuring plants are watered only when necessary.

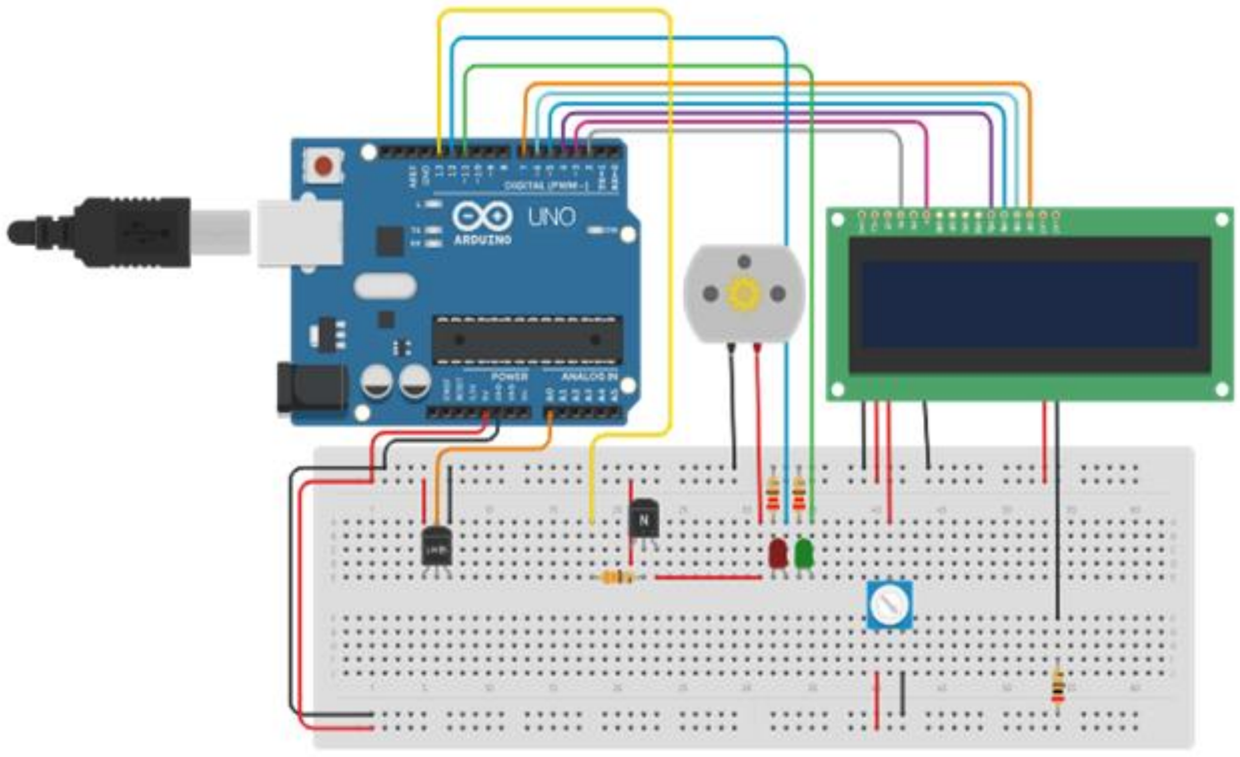
Components needed:
- Arduino Uno R3
- Temperature Sensor (e.g., TMP36)
- DC Water Pump (simulated by a DC motor)
- NPN Transistor (e.g., 2N2222)
- Diode (1N4007)
- Resistors (e.g., 10k-ohm and 220-ohm)
- Breadboard
- Jumper Wires
- Water Reservoir (to hold water for the pump)
- soil moisture sensor,
- relay module
- Power Supply (for the water pump, if it requires more power than the Arduino can provide)
- LCD

Setting up the circuit in Tinkercad:
1. Open Tinkercad, create a new circuit, and add the necessary components from the list above.
2. Connect the temperature sensor (TMP36) to the Arduino, linking VCC to 5V, GND to GND, and the Analog Output to an analog pin (like A0).
3. Set up the DC motor (simulating the pump) with the NPN transistor acting as a switch. Connect the motor and transistor to the breadboard and Arduino, including a diode for protection and resistors for proper operation.

Writing the code:
1. Open the Code Editor and select Text mode.
2. Write the Arduino code to read the temperature sensor, convert the reading to Celsius, and control the motor based on a predefined temperature threshold.
3. The code should define the pins used for the sensor and motor, set the temperature threshold, and use `analogRead` and `digitalWrite` functions to manage the system.
4. A delay should be included in the loop to space out readings.
5. Code is given below:

```
#include <LiquidCrystal.h>
const int LM35 = A0;
const int motor = 13;
const int LedRed = 12;
const int LedGreen = 11;
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("Automated Plant");
  lcd.setCursor(0,1);
  lcd.print("Watering System!");
  pinMode(motor, OUTPUT);
  pinMode(LedRed, OUTPUT);
  pinMode(LedGreen, OUTPUT);
  delay(2000);
  lcd.clear();
  lcd.print("Temp= ");
  lcd.setCursor(0,1);
  lcd.print("WaterPump= ");
}
void loop() {
  int value = analogRead(LM35);
  float Temperature = value * 500.0 / 1023.0;
  lcd.setCursor(6,0);
  lcd.print(Temperature);
  lcd.setCursor(11,1);
   if (Temperature > 50){
    digitalWrite(motor, HIGH);
    digitalWrite(LedRed, HIGH);
    digitalWrite(LedGreen, LOW);
    lcd.print("ON ");
  }
  else {
    digitalWrite(motor, LOW);
    digitalWrite(LedRed, LOW);
    digitalWrite(LedGreen, HIGH);
    lcd.print("OFF");
  }
   delay(1000);
}
```

Circuit board:



Simulating the circuit in Tinkercad:
Start the simulation and test the system by adjusting the temperature sensor's value to see how the motor (pump) responds to different temperatures.

Important notes:
Using temperature alone may not be sufficient for optimal plant watering; combining it with a soil moisture sensor is recommended.

Link to get the simulation: https://www.tinkercad.com/things/2cTfQT7fAno-automated-plant-watering-system

Result: The project of Automatic Plant Watering system in implemented in Tinkercad successfully.

Exp No.: 07	BLUETOOTH-BASED DOOR LOCK SYSTEM

Aim: To design and simulate an Arduino-based door lock system.

Components needed:

- Arduino Uno R3
- Keypad 4x4
- Positional Micro Servo
- LCD 16 x 2
- 1 kΩ Resistor

Circuit design (in Tinkercad)

- Arduino and Keypad: Connect the keypad's row and column pins to the Arduino's digital pins as specified in the keypad library or chosen for your project. For example, the keypad's columns might connect to D2-D5 and rows to D6-D9.
- Arduino and Servo Motor: Connect the servo motor's signal pin to an Arduino PWM pin (e.g., pin 9 or 10), the power pin to 5V, and the ground pin to GND.
- Arduino and LCD (Optional): Connect the LCD's SDA pin to A4 and SCL to A5 if using an I2C module. Connect the LCD's power (VCC) to 5V and ground to GND.
- Arduino and LEDs/Buzzer (Optional): Connect the positive terminals of the LEDs/buzzer to digital pins (e.g., Red LED to pin 12 and Green LED to pin 13, and Buzzer to pin 11) and the negative terminals to GND, potentially through a resistor for LEDs.

Arduino code:

- Include Libraries: Include necessary libraries for the keypad (`Keypad.h`) and servo motor (`Servo.h`). If using an LCD with I2C, include `LiquidCrystal_I2C.h`.
- Define Connections: Define the Arduino pins connected to the keypad, servo, LCD, LEDs, and buzzer.
- Define Password: Set the default password (e.g., using a character array).
- `setup()` function:
    - Initialize the serial monitor (for debugging, optional).
    - Initialize the LCD (if used).
    - Attach the servo to its defined pin.
    - Set the initial servo position to lock the door.
    - Set the LED pins as outputs and set the initial locked state (e.g., red LED ON, green LED OFF).

- `loop()` function:

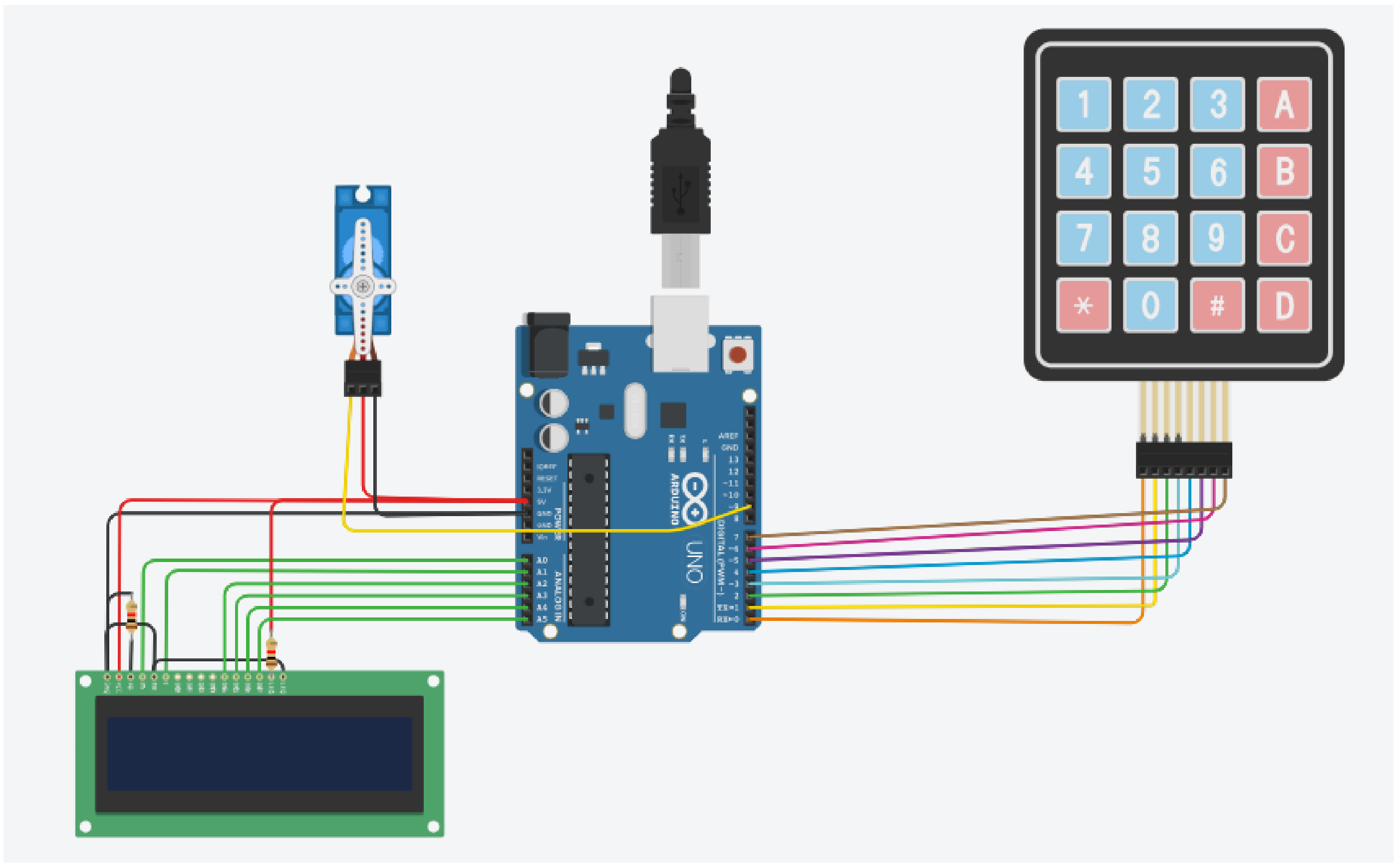
    - Keypad Input: Use `keypad.getKey()` to read input from the keypad.
    - Password Check: Compare the entered password with the predefined password.
    - Conditional Actions:

    - If the correct password is entered:
        - Change the servo position to unlock the door (e.g., rotate to 0 degrees).
        - Update the LCD display (if used) with an "Access granted" message.
        - Change the LED indicators (if used) (e.g., green LED ON, red LED OFF).
        - Optionally, re-lock the door automatically after a time delay.
    - If the incorrect password is entered:
        - Update the LCD display (if used) with a "Wrong password" message.
        - Sound the buzzer (if used) to alert the user.
        - Keep the door locked (servo position unchanged).

Code:

```
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#define Password_Length 5
Servo myservo;
LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
int pos = 0;
char Data[Password_Length];
char Master[Password_Length] = "1234";
byte data_count = 0, master_count = 0;
bool Pass_is_good;
bool door = false;
char customKey;

/*---preparing keypad---*/
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {0, 1, 2, 3};
byte colPins[COLS] = {4, 5, 6, 7};
Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
/*--- Main Action ---*/
void setup()
{
  myservo.attach(9, 2000, 2400);
  ServoClose();
  lcd.begin(16, 2);
  lcd.print("Protected Door");
  loading("Loading");
  lcd.clear();
}
void loop()
{
  if (door == true)
  {
    customKey = customKeypad.getKey();
    if (customKey == '#')
    {
      lcd.clear();
      ServoClose();
      lcd.print("Door is closed");
      delay(3000);
      door = false;
    }
  }
  else
    Open();
}
void loading (char msg[]) {
  lcd.setCursor(0, 1);
  lcd.print(msg);
  for (int i = 0; i < 9; i++) {
    delay(1000);
    lcd.print(".");
  }
}
void clearData()
{
  while (data_count != 0)
  {
    Data[data_count--] = 0;
  }
  return;
}
void ServoClose()
{
  for (pos = 90; pos >= 0; pos -= 10) {
    myservo.write(pos);
  }
}
void ServoOpen()
{
  for (pos = 0; pos <= 90; pos += 10) {
    myservo.write(pos);
  }
}
void Open()
{
  lcd.setCursor(0, 0);
  lcd.print("Enter Password");
  customKey = customKeypad.getKey();
  if (customKey)
  {
    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print(Data[data_count]);
    data_count++;
  }
  if (data_count == Password_Length - 1)
  {
    if (!strcmp(Data, Master))
    {
      lcd.clear();
      ServoOpen();
      lcd.print(" Door is Open ");
      door = true;
      delay(5000);
      loading("Waiting");
      lcd.clear();
      lcd.print(" Time is up! ");
      delay(1000);
      ServoClose();
      door = false;
    }
    else
    {
      lcd.clear();
      lcd.print(" Wrong Password ");
      door = false;
    }
    delay(1000);
    lcd.clear();
    clearData();
  }
}
```

Circuit Board:



Link: https://www.tinkercad.com/things/cMDBAUzuXqA-door-lock

Result: Successfully simulated an Arduino-based door lock system in Tinkercad

Exp No.: 08	SMART DUSTBIN USING ULTRASONIC SENSOR

Aim: to design a basic Arduino Smart Dustbin circuit in Tinkercad, focusing on automatic lid opening

Components needed:

- Arduino Uno
- HC-SR04 Ultrasonic Distance Sensor
- Servo Motor (e.g., SG90)
- Breadboard
- Jumper Wires
- Power Source (e.g., USB connection to computer)

Circuit setup in Tinkercad:

- Place Arduino Uno: Drag and drop the Arduino Uno onto the Tinkercad workspace.
- Connect the Ultrasonic Sensor (HC-SR04):
  - Place the ultrasonic sensor on the breadboard.
  - Connect VCC to 5V, GND to GND, Trig to a digital pin (e.g., pin 9), and Echo to another digital pin (e.g., pin 10) on the Arduino.

- Connect the Servo Motor:

  - Connect the servo's signal wire to a PWM-capable digital pin (e.g., pin 9), GND wire to the Arduino's GND, and positive wire to the Arduino's 3.3V pin.

- Power the Circuit:

  - Connect a power source like a USB cable to your Arduino.

Code:

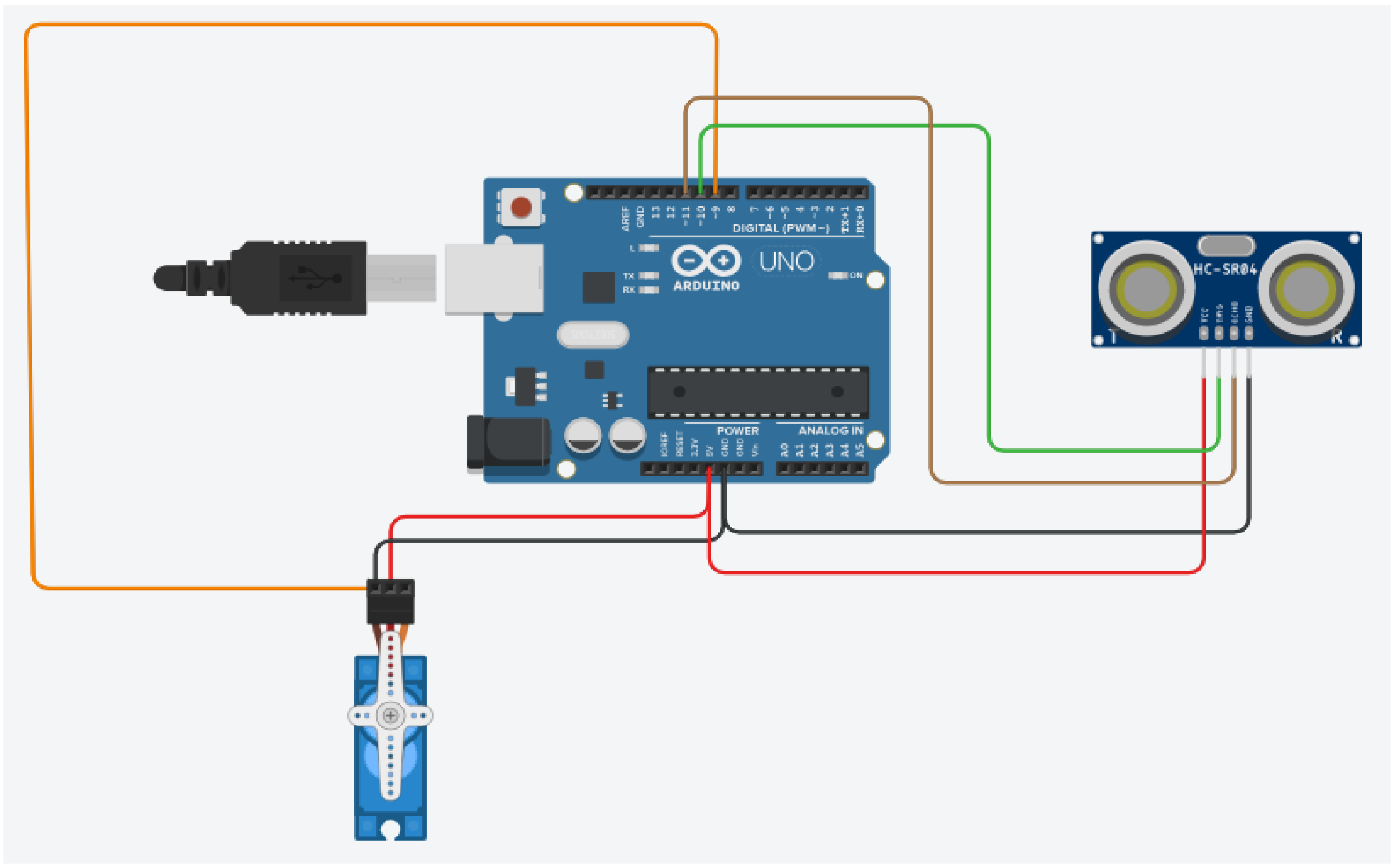
```
#include <Servo.h>
Servo servoMain; // Define our Servo
int trigpin = 10;
int echopin = 11;
int distance;
float duration;
float cm;
void setup()
{
  servoMain.attach(9); // servo on digital pin 10
   pinMode(trigpin, OUTPUT);
   pinMode(echopin, INPUT);
}
void loop()
{
```

```
digitalWrite(trigpin, LOW);
delay(2);
digitalWrite(trigpin, HIGH);
delayMicroseconds(10);
digitalWrite(trigpin, LOW);
duration = pulseIn(echopin, HIGH);
cm = (duration/58.82);
distance = cm;
  if(distance<30)
{
  servoMain.write(180);  // Turn Servo back to center position (90 degrees)
 delay(3000);
}
 else{
   servoMain.write(0);
   delay(50);
  }
}
```

Basic functionality:

The ultrasonic sensor detects objects. When an object is within a set distance, the Arduino signals the servo to open the lid. After a delay, the servo closes the lid.

Circuit board:



Link: https://www.tinkercad.com/embed/kxwoGlM7BNY?editbtn=1

Result: Basic Arduino Smart Dustbin circuit in Tinkercad is executed successfully

Exp No.: 09	FIRE DETECTION AND ALARM SYSTEM

Aim: to create a fire detection and alarm system in Tinkercad using an Arduino

Components needed:

- Arduino Uno R3
- Breadboard
- LM-35 Temperature Sensor
- (or TMP36)
- MQ2 Gas Sensor
- Piezo Buzzer
- LED (Light Emitting Diode)
- Resistors (e.g., 220Ω, 1kΩ, 22kΩ)
- Jumper Wires

Circuit connections:

- Power and Ground: Connect the Arduino's 5V pin to the positive rail of the breadboard and the Arduino's ground (GND) pin to the negative rail of the breadboard.
- Temperature Sensor:
  - Connect the Vs (Supply) pin of the LM-35 (or TMP36) to the positive rail of the breadboard.
  - Connect the Ground pin to the negative rail of the breadboard.
  - Connect the Vout (Output) pin to an analog pin on the Arduino (e.g., A1).
- Gas Sensor:
  - Connect three of the gas sensor's pins (B1, H2, and B2 if using a 6-pin sensor) to the positive rail of the breadboard.
  - Connect another pin (e.g., A1) to an analog pin on the Arduino (e.g., A0).
  - Connect another pin (e.g., H1) to the negative rail of the breadboard.
  - Connect the remaining pin (e.g., A2) through a resistor (e.g., 22kΩ) to the ground rail.
- Buzzer:
  - Connect one terminal of the piezo buzzer to the negative rail of the breadboard.
  - Connect the other terminal of the buzzer to a digital pin on the Arduino (e.g., pin 7).
- LED:
  - Connect the LED's cathode (shorter leg) to the Arduino's GND pin.
  - Connect the LED's anode (longer leg) through a resistor (e.g., 220Ω) to a digital pin on the Arduino (e.g., pin 13).

How it works:

- The temperature sensor measures the ambient temperature, according to LearnElectronics India.
- The gas sensor detects the presence and concentration of various gases like methane, alcohol, carbon monoxide, or carbon dioxide, says Tinkercad.
- The Arduino reads the analog outputs from both sensors.
- If the temperature or gas levels exceed predefined thresholds, the Arduino triggers the alarm (LED and/or buzzer).
- The Serial Monitor displays the sensor readings, allowing you to monitor the values in real-time.

Source Code:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

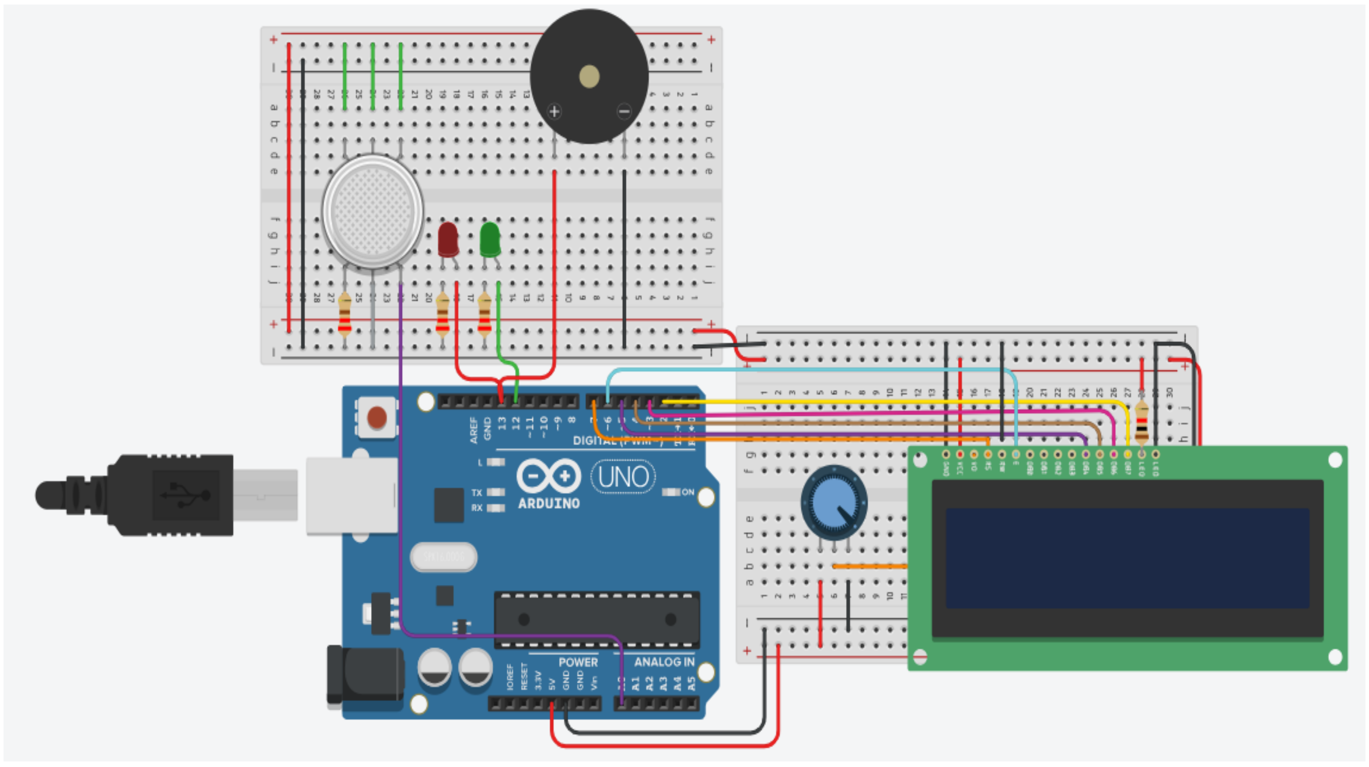
```
int Smoke = 0;
int r =0;
void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

}
void loop()
{
  Smoke = analogRead(A0);
  Serial.println(analogRead(A0));
  if (Smoke >=25)
  {
    digitalWrite(13, HIGH);
    digitalWrite(12, LOW);
    tone(13, 523); // play tone 60 (C5 = 523 Hz)
    lcd.setCursor(0,0);
         lcd.print("Emergency exit is right to the elevator");
         delay(100);
         lcd.setCursor(0,1);
         lcd.print("Fire and Rescue Dial 101 immediately");
         lcd.setCursor(1,0);
    for(r=0;r<36;r++)
     {
       lcd.scrollDisplayLeft();
         delay(100);
     }
    }
  else
  {
    digitalWrite(13, LOW);
    digitalWrite(12, HIGH);
    lcd.clear();
    noTone(13);
    lcd.setCursor(6,0);
    lcd.print("GOOD");
    lcd.setCursor(6,1);
    lcd.print("DAY:)");
    delay(1000);
  }
  delay(10); // Delay a little bit to improve simulation performance
}
```

Simulation in Tinkercad:

- Access Tinkercad: Go to the Tinkercad website and log in or create an account.
- Create a New Circuit: Select "Circuits" and then "Create new Circuit".
- Add Components: Drag and drop the necessary components (Arduino, breadboard, sensors, LED, buzzer, resistors, jumper wires) from the component library onto the workspace.

- Connect Components: Make the circuit connections as described above, ensuring the correct pins are connected.
- Add Code: Click on the "Code" button and select "Text" to input the Arduino code.
- Start Simulation: Click on "Start Simulation" to run the circuit and observe its behaviour.
- Test the System: Simulate fire conditions (e.g., increase the temperature or gas concentration values on the virtual sensors) to see if the alarm activates correctly.
- Monitor Output: Check the Serial Monitor for the temperature and gas sensor readings.
- By following these steps, you can build and simulate a fire detection and alarm system in Tinkercad, learning about sensor interfacing and Arduino programming in the process

Circuit Diagram:



Link: https://www.tinkercad.com/things/2X0n2iTvhY2-fire-alarm-system

Result: Successfully created a fire detection and alarm system in Tinkercad using an Arduino.

Exp No.: 10	SOIL MOISTURE-BASED IRRIGATION

Aim: To simulate and design a smart irrigation systems using Arduino.

Components needed:

- Ambient Light Sensor [Phototransistor]
- DC Motor
- H-bridge Motor Driver
- Arduino Uno R3
- Temperature Sensor [TMP36]
- PCF8574-based, 32 (0x20) LCD 16 x 2 (I2C)
- Soil Moisture Sensor

How it works:

A smart irrigation system on Tinkercad typically involves:

1. Soil Moisture Sensor: This component measures the moisture level in the soil. It outputs an analog value – a higher value indicates drier soil, and a lower value indicates wetter soil.
2. Arduino Uno: This acts as the brain of the system, processing the data from the sensor and controlling the water pump.
3. Relay Module: Since a real water pump can't be simulated in Tinkercad, an LED is used as a stand-in. The relay module acts as a switch to control the LED (representing the pump) based on the Arduino's commands.
4. Water Pump (Simulated with an LED): As mentioned, an LED represents the water pump. It turns on when the Arduino determines the soil needs watering.

Building a smart irrigation system in Tinkercad:

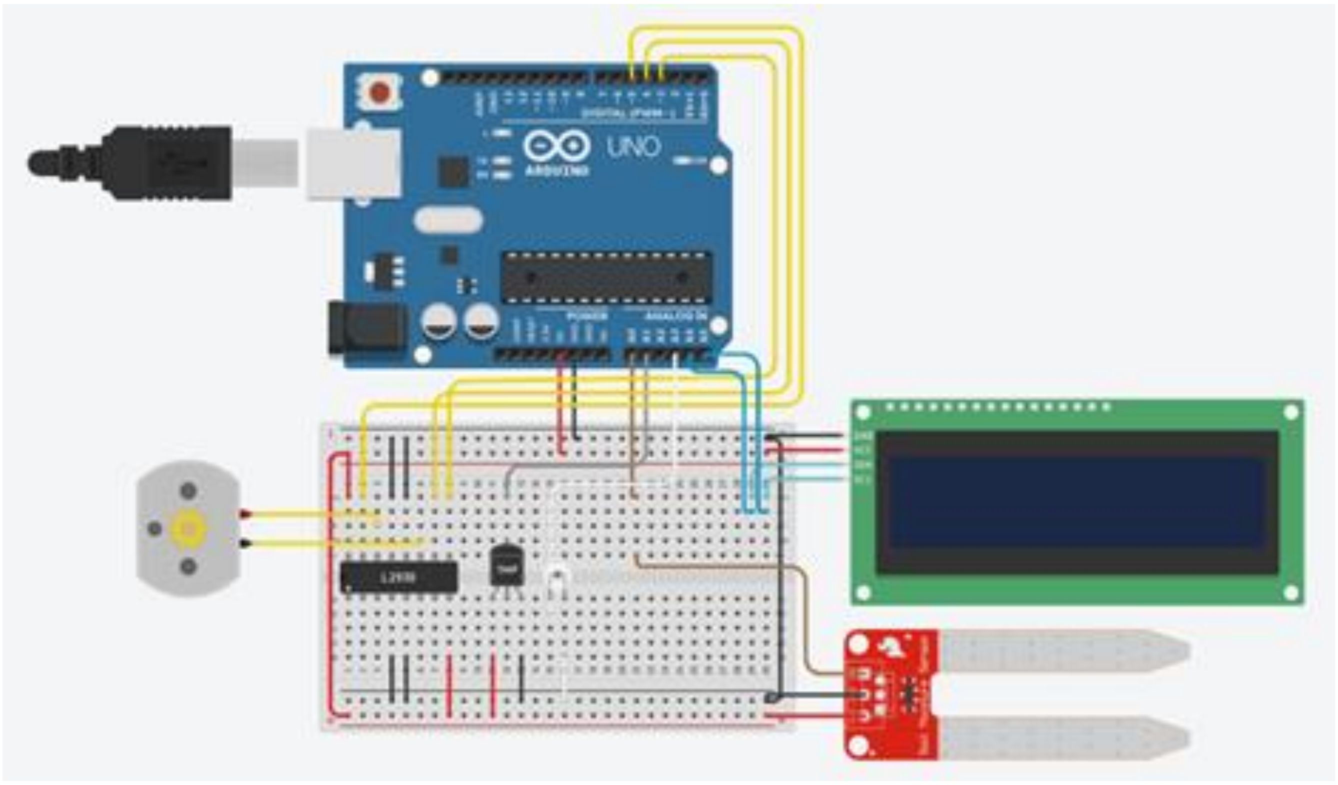
1. Set up the circuit
    - Arduino and Breadboard: Start by placing an Arduino Uno and a breadboard on the workspace.
    - Soil Moisture Sensor: Connect its VCC pin to 5V on the Arduino, GND to GND, and SIG to analog pin A0.
    - Relay Module: Connect VCC to 5V on the Arduino, GND to GND, and IN to digital pin 7.
    - Water Pump (LED): Connect the LED's anode (longer leg) to the Normally Open (NO) pin on the relay and its cathode (shorter leg) to GND via a 220Ω resistor.
2. Write the Arduino code
    - The code reads the soil moisture sensor, compares it to a threshold, and activates the relay (LED) if the soil is dry.

Code:

```
 #include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Define the pins for the motor control
int enA = 5;  // Enable pin for the L293D motor driver
int in1 = 3;  // Input pin 1 for the L293D motor driver
int in2 = 4;  // Input pin 2 for the L293D motor driver
```

```
LiquidCrystal_I2C lcd(32,16,2);
// Define the pins for the sensors
int tempPin = A1;     // Temperature sensor is connected to A1
int lightPin = A2;    // Light sensor is connected to A2
int moisturePin = A0; // Soil moisture sensor is connected to A0

void setup() {
 // Initialize the Serial Monitor
            lcd.begin(16,2);
            lcd.init();
            lcd.backlight();
            Serial.begin(9600);

  // Set the motor control pins as outputs
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
}

void loop() {
  // Read the sensor values
  int tempValue = analogRead(tempPin);
  int lightValue = analogRead(lightPin);
  int moistureValue = analogRead(moisturePin);

  // Convert the sensor values to their corresponding units
  float temp = (((tempValue * 5.0) / 1024.0)-0.5)*100;
  float light = (lightValue * 5.0) / 1024.0;
  float moisture = (moistureValue * 5.0) / 1024.0;

  Serial.print("Temp ");
  Serial.print(temp);
  Serial.println(" C");
  Serial.print("Light:");
  Serial.print(light);
  Serial.println("V");
  Serial.print("Moist: ");
  Serial.print(moisture);
  Serial.println("V");

   // Display the sensor values on the LCD screen
  lcd.clear(); // Clear the LCD screen
  lcd.setCursor(0, 0); // Set the cursor to the top-left corner
  lcd.print("Temp:"); // Print the temperature label
  lcd.print(temp); // Print the temperature value
  lcd.print("C"); // Print the temperature unit
  lcd.setCursor(0, 1); // Set the cursor to the second row
  lcd.print("Light:"); // Print the light label
  lcd.print(light); // Print the light value
  lcd.print("V"); // Print the light unit
  lcd.setCursor(8, 1); // Set the cursor to the second row, ninth column
  lcd.print("Moist:"); // Print the moisture label
  lcd.print(moisture); // Print the moisture value
  lcd.print("V"); // Print the moisture unit
  // Control the motor based on the sensor readings
  if((temp > 30 || light < 2)&& moisture < 4 ) {
    // If it's hot, the soil is dry, and it's morning, turn on the motor
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    analogWrite(enA, 255); // Set the motor speed to maximum
    Serial.println("Turning on the motor...");
  } else {
    // Otherwise, turn off the motor
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    analogWrite(enA, 0); // Set the motor speed to 0
    Serial.println("Turning off the motor...");
  }

  // Wait for 1 second before taking another reading
  delay(1000);
}
```

Circuit diagram:



Simulate the circuit: Start the simulation to see the LED (pump) respond to simulated soil moisture levels.

Link: https://www.tinkercad.com/things/eKtHRRkOZoA-

Enhancements and further possibilities:
You can enhance the system by adding components like an LCD for displaying data, an RTC for scheduling, or a Wi-Fi module for remote access. The system can also be expanded with additional sensors and actuators. This project is a good introduction to IoT and smart home automation using Arduino.
Result: The simulation and design of a smart irrigation systems using Arduino has been implemented successfully.

Exp No.: 11	SMART HELMET FOR ACCIDENT DETECTION

Aim: To design a smart helmet for accident detection and simulation it in Tinkercad

Components needed in Tinkercad:

- Arduino Uno: The brain of the system, processing sensor data and controlling outputs.
- Breadboard: To easily connect the components without soldering.
- Accelerometer (MPU-6050): Detects sudden changes in movement (impacts), indicating a potential accident.
- GPS module (NEO-6M): Tracks the helmet's location (latitude and longitude).
- GSM module (SIM800L): Sends emergency SMS alerts with the location data to predefined numbers.
- Bluetooth module (HC-05): Optionally, for real-time data monitoring on a smartphone app.
- Buzzer or LED: Provides immediate alerts to the rider upon impact detection.
- Battery: Powers the entire system.
- Connecting wires: To link the components together.

Setting up the Tinkercad circuit:

- Open Tinkercad and create a new circuit project.
- Drag and drop an Arduino Uno and a breadboard onto the workspace.
- Add the necessary components (accelerometer, GPS, GSM, Bluetooth, buzzer/LED) to your workspace.
- Wiring: Connect the components to the Arduino board based on the circuit diagram. For example:
    - Connect the accelerometer's power (VCC) and ground (GND) pins to the Arduino's 5V and GND pins, respectively.
    - Connect the accelerometer's data pins to appropriate analog input pins on the Arduino.
    - Similarly, connect the GPS and GSM modules to the Arduino using UART communication (Rx and Tx pins).
    - Connect the buzzer's positive lead to a digital pin (e.g., pin 10) and the negative lead to GND on the Arduino.
    - Connect the LED's anode (long leg) to a digital pin (e.g., pin 12) through a resistor and the cathode (short leg) to GND.

Writing the Arduino code:

- Open the Code Editor in Tinkercad.
- Write the Arduino code to program the smart helmet’s functionality.
    - Initialize the accelerometer, GPS, and GSM modules.
    - Read data from the accelerometer to detect sudden changes in movement (impacts).
    - If an accident is detected, retrieve the current location from the GPS module.
    - Construct an emergency message containing the location data.
    - Send the emergency message via the GSM module to pre-programmed numbers.
    - Activate the buzzer or LED to alert the rider.

Code:

```
#include <Servo.h>
Servo servo;
42ons tint PIRpin=7;
42ons tint buzzerPin=6;
42ons tint ledPin=5;
42ons tint blue=10;
42ons tint green=11;
42ons tint red=8;
```

```
43ons tint temperaturePin = A0;
43ons tint gas_pin = A1;
43ons tint LEFT = 4, RIGHT=3, UP=2,DOWN=1;
void setup()
{
   servo.attach(9);
  servo.write(0);
  pinMode(PIRpin, INPUT);
  pinMode (temperaturePin , INPUT);
  pinMode (gas_pin, INPUT);
  pinMode(LEFT, INPUT);
  pinMode(RIGHT, INPUT);
  pinMode(UP, INPUT);
  pinMode(DOWN, INPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(blue, OUTPUT);
  pinMode(green, OUTPUT);
  Serial.begin(9600);
}

void loop()
{ float degreesC;
  degreesC=analogRead(0);
  degreesC=degreesC*0.0048828125;
  degreesC=(degreesC-0.5)*100;
  int sensorValue;
  sensorValue = analogRead(gas_pin);

  int right,left,up,down;
  right = digitalRead(RIGHT);
  left = digitalRead(LEFT);
  up = digitalRead(UP);
  down = digitalRead(DOWN);
   if((right == LOW && left == LOW)||(up == LOW && down == LOW)&& sensorValue > 720
&& degreesC>37)
  {
    digitalWrite(red,HIGH);
    delay(500);
    digitalWrite(red,LOW);
    delay(500);
    digitalWrite(red,HIGH);
    delay(500);
    digitalWrite(red,LOW);
    delay(500);
    Serial.println(“Accident Detected”);
    Serial.println(“Sending Alert”);
    delay(1000);
  }
  else if( sensorValue > 720 && degreesC>37)
  {
    tone(buzzerPin, 1000, 500);
    delay(1000);

    digitalWrite(blue,HIGH);
    delay(500);
    digitalWrite(blue,LOW);
    delay(500);
    digitalWrite(blue,HIGH);
    delay(500);
    digitalWrite(blue,LOW);
    delay(500);
    Serial.println(“Gas is detected And Teperature Is Increased:”);
    delay(10000);
  }
else if(degreesC>37) {
    tone(buzzerPin, 1000, 200);
    delay(1000);
    digitalWrite(green,HIGH);
    delay(500);
    digitalWrite(green,LOW);
    digitalWrite(green,HIGH);
    delay(500);
    digitalWrite(green,LOW);
    delay(500);
    Serial.println(“Teperature Is Increased:”);
    delay(10000);
  }
  else
  {
    noTone(buzzerPin);
  }

  if(digitalRead(PIRpin)==1)
  {
    tone(buzzerPin, 1000, 2000);
    delay(1000);
    digitalWrite(ledPin,HIGH);
    delay(500);
    digitalWrite(ledPin,LOW);
    delay(500);
    digitalWrite(ledPin,HIGH);
    delay(500);
    digitalWrite(ledPin,LOW);
    delay(500);
    servo.write(90);
    delay(5000);
    servo.write(0);
  }
```

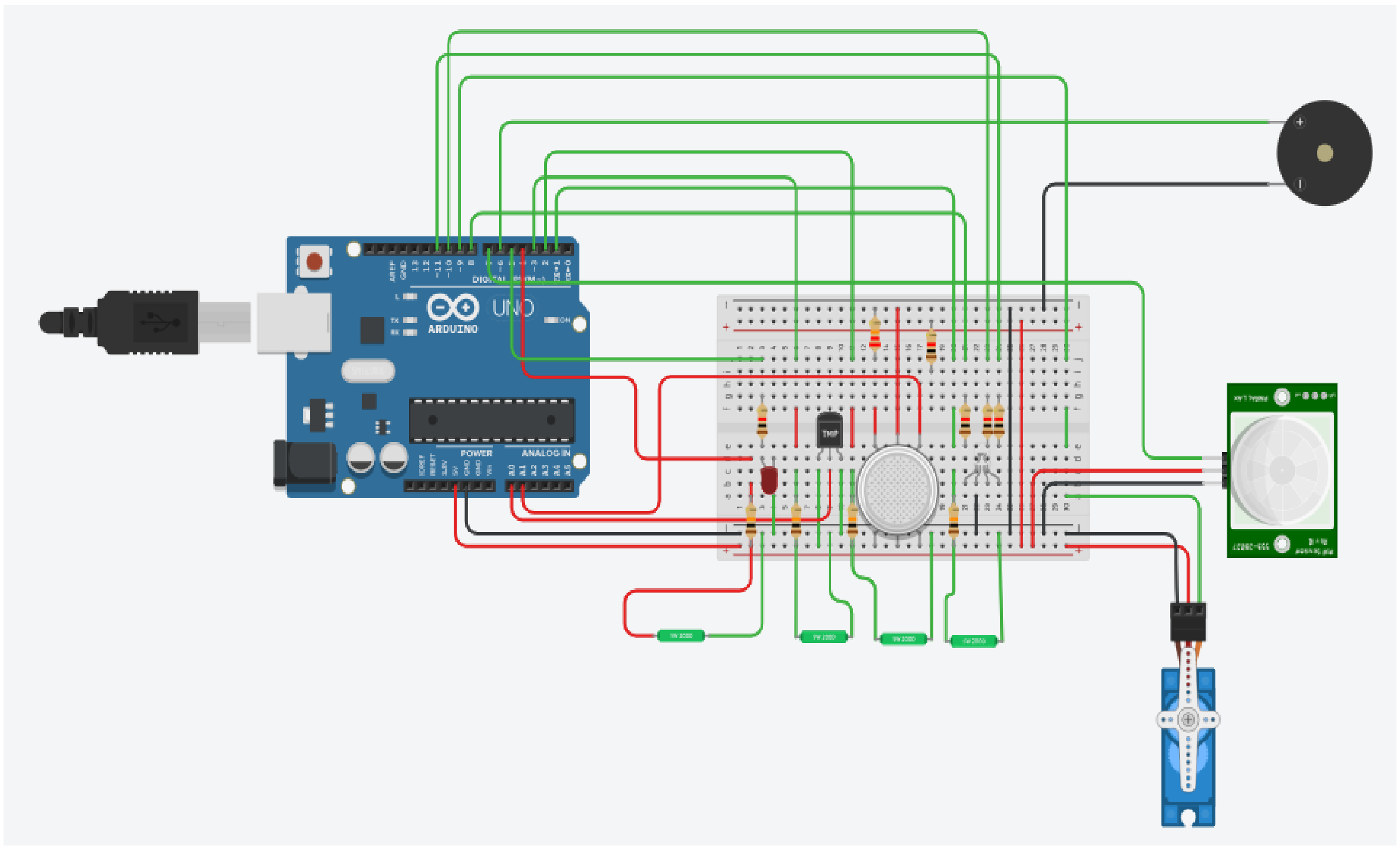
Simulating the circuit and code:

- Click "Start Simulation" in Tinkercad.
- Testing Accident Detection: Simulate an accident by adjusting the accelerometer's values to mimic a sudden impact.
- Verifying Alerts: Observe if the buzzer sounds or the LED flashes, indicating a detected accident.
- Checking Location and Messaging: If your Tinkercad simulation includes simulated GPS and GSM functionality, check the serial monitor for the location data and verify that the emergency message is triggered or sent (if simulating sending messages directly through the simulator).

Important Notes:

- Tinkercad simulates the hardware and software behaviour, allowing you to test the logic before building a physical circuit.
- The actual hardware and real-world implementation might require careful consideration of factors like battery life, antenna placement, and signal strength.
- Ensure a clear view of the sky for the GPS module to receive signals in a real-world scenario.
- For a complete accident detection system, consider adding other features like fall detection or helmet wear detection. according to Techatronic.

Circuit Board:



Link: https://www.tinkercad.com/things/h7qvsrFRx9G-accident-prevention-and-detection

Result: Successfully created the design of a smart helmet for accident detection and simulation it in Tinkercad

Exp No.: 12	TEMPERATURE-CONTROLLED CHEMICAL REACTOR

Aim: To design and simulate an automatic room temperature controller system in Chemical Reactor using Arduino

Components needed:
- Arduino Uno R3
- Breadboard (small).
- Temperature sensor (e.g., TMP36 or LM35).
- DC motor (to simulate a fan).
- 10k-ohm resistor.
- NPN transistor (e.g., 2N2222).
- Diode (e.g., 1N4007).
- Jumper wires.
- (Optional) LCD 16x2 Display to show temperature readings and system status.
- (Optional) Potentiometer to adjust the temperature threshold.

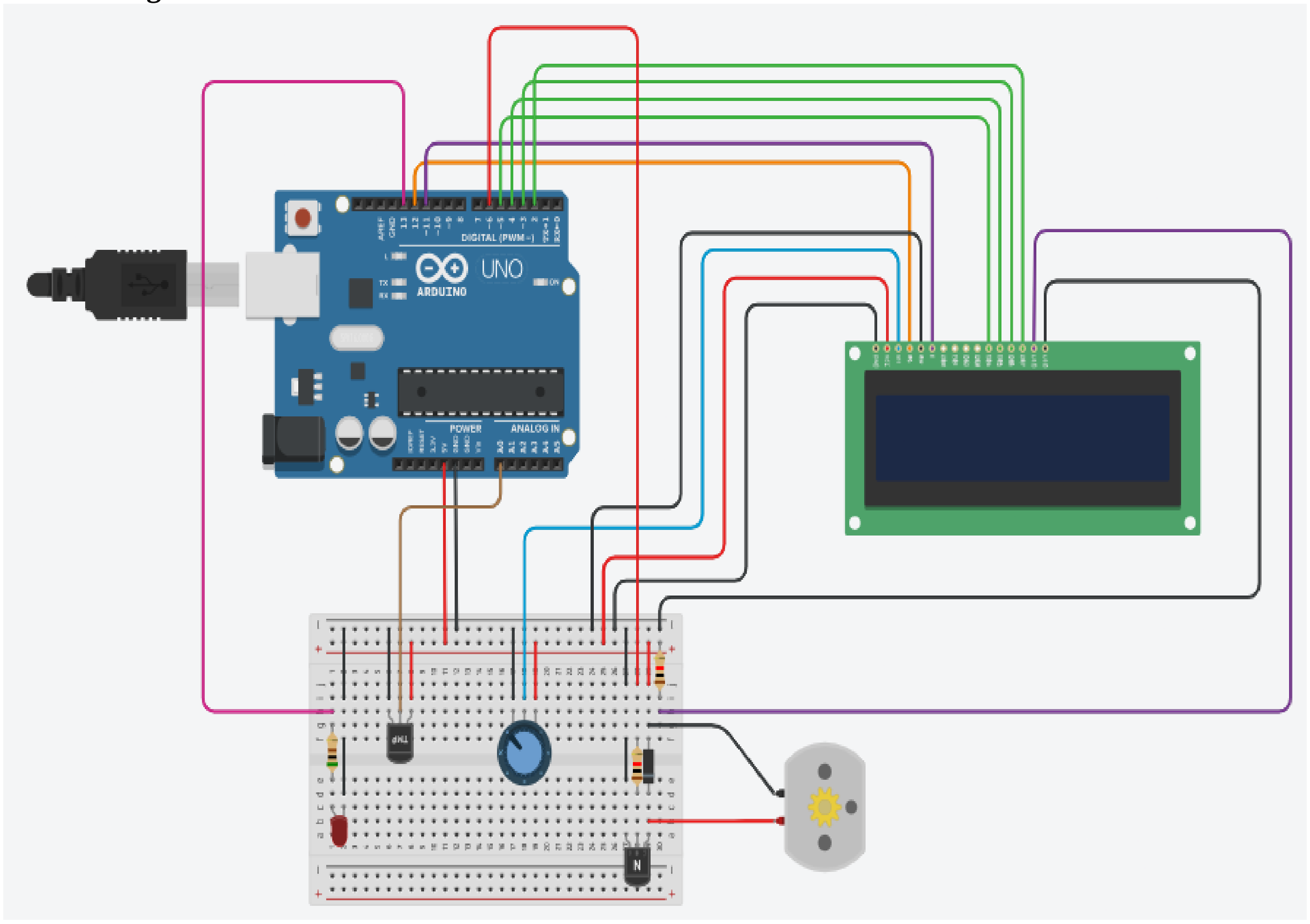
Building the circuit:
1. Set up the breadboard and Arduino: Place the breadboard and Arduino Uno R3 on your Tinkercad workspace.
2. Connect the temperature sensor: Connect the temperature sensor to the breadboard, linking its output to an analog Arduino pin (e.g., A0) and connecting power and ground to the breadboard's 5V and GND rails, respectively. For a TMP36, the left pin goes to 5V, the center to A0, and the right to GND.
3. Connect the fan (DC motor): Use an NPN transistor on the breadboard to control the DC motor. Connect one motor terminal to 5V and the other to the transistor's collector. A diode should be placed across the motor terminals with the cathode on the 5V side. The transistor's emitter connects to GND, and its base connects to a digital Arduino pin (e.g., pin 9) through a 10k-ohm resistor for PWM control.
4. Optional additions: Connect an LCD display and/or a potentiometer for temperature display and threshold adjustment.

Writing the code (Arduino sketch):
The code will read the temperature, compare it to a threshold (or potentiometer value), activate the fan using PWM when the temperature is too high, and optionally display information on the LCD. The final code is given below:

```
const int temp_trans_pin = A0;
const int heater_pin = 13;
const int fan_pin = 6;
// set the range of desire temp
float MinTemp = 20, MaxTemp = 25; // room temp (20,25)
// Include the LCD library code
#include<LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal LCD (12, 11, 5, 4, 3, 2);
void setup(){
  // set up the LCD's number of columns and rows:
  LCD.begin(16,2);
  pinMode(heater_pin, OUTPUT);
  pinMode(fan_pin, OUTPUT);
  // Display the desired range of temperature
  LCD.print("Room Temp(C):");
  LCD.setCursor(2,1);
  LCD.print(MinTemp); LCD.print("-"); LCD.print(MaxTemp);
  delay(2000);
}
void loop(){
  // Read voltage and convert to temperature (Celsius)
  float Equ_volt = analogRead(temp_trans_pin)*5.0/1023;
  float SensorTemp = 100.0*Equ_volt-50.0;
  // Display the sensor reading
  LCD.clear();
  LCD.print("Sensor Reading: ");
  LCD.setCursor(2,1);
  LCD.print(SensorTemp); LCD.print(" C");
  delay(2000);
   if (SensorTemp>MaxTemp){
  LCD.clear();
   //Turn off HEATER(LED) in case if its on
    digitalWrite(heater_pin, 0);
   //Turn on FAN (dc motor) to regulate the temp
    LCD.print("Temp is HIGH");
    LCD.setCursor(0,1); LCD.print("Turn on FAN");
    for (int i=0; i<255; i++){
    analogWrite(fan_pin, i);
    }
    delay(2000);
 }
  else if (SensorTemp<MinTemp){
  LCD.clear();
     //Turn off FAN (dc motor) in case if it is on
    LCD.clear();
    for (int i=255; i>=0; i--){
    analogWrite(fan_pin, i);
    }
   LCD.print("Temp is LOW");
    LCD.setCursor(0,1); LCD.print("Turn on HEATER");
    //Turn on HEATER(LCD)
    digitalWrite(heater_pin, 1);
    delay(3000);
   }
  else if (SensorTemp>MinTemp && SensorTemp<MaxTemp){
  LCD.clear();
    //Turn of all in case it is on
    for (int i=255; i>=0; i--){
    analogWrite(fan_pin, i);
    }
     digitalWrite(heater_pin, 0);
     LCD.print("Temp is normal");//Normal Room temperature
    LCD.setCursor(2,1);
    LCD.print("Turn off all");
    delay(3000);
    LCD.clear();
 }
  else {
  LCD.clear();
    LCD.print("Something went Wrong");
```

```
      LCD.setCursor(2,1);
      LCD.print("wrong in circuit");
      delay(1000);
      LCD.clear();
    }
    delay(1000);
  }
```

Circuit Diagram:



Link: https://www.tinkercad.com/things/fndGND1saJ4-automatic-room-temperature-controller

Simulating and testing:

Start the simulation in Tinkercad and observe temperature readings and adjust the sensor's temperature or the potentiometer. Confirm that the fan responds by changing speed.

Result: The design and simulate an automatic room temperature controller system in Chemical Reactor using Arduino is completed successfully.

Exp No.: 13	ECG SIGNAL ACQUISITION AND PLOTTING

Aim: To design and simulate a basic portable ECG circuit using the Tinkercad platform

Components needed:

- kΩ Resistor
- 20 kΩ Resistor
- 10 kΩ Resistor
- 100 kΩ Resistor
- 150 Ω Resistor
- 500 kΩ Resistor
- 1 kΩ Resistor
- 10 kΩ Potentiometer
- 20 kΩ Potentiometer
- Arduino Uno R3
- 10 uF, 16 V Polarized Capacitor
- 15 , 1 Power Supply
- 741 Operational Amplifier
- 8 Pin Header
- Voltage Multimeter
- 100 ms Oscilloscope
- 5 Hz, 3 V, 0 V, Square Function Generator

Key elements and considerations

- ECG Signal Acquisition and Conditioning: The core of such a project often involves the AD8232 ECG sensor, which extracts, amplifies, and filters the faint electrical signals generated by the heart.
- Microcontroller: An Arduino Uno is a popular choice for processing the analog ECG signal from the sensor and converting it into a digital format that can be further processed and displayed.
- Display/Visualization: The processed ECG data can be viewed on a serial monitor in Tinkercad, or potentially with an external display component within the simulation, according to The Science and Information (SAI) Organization.
- Portability Aspects: While Tinkercad simulations focus on the circuit design, a physical portable ECG system would require compact components, battery power, and a suitable enclosure.

Building a simulated portable ECG in Tinkercad Circuits

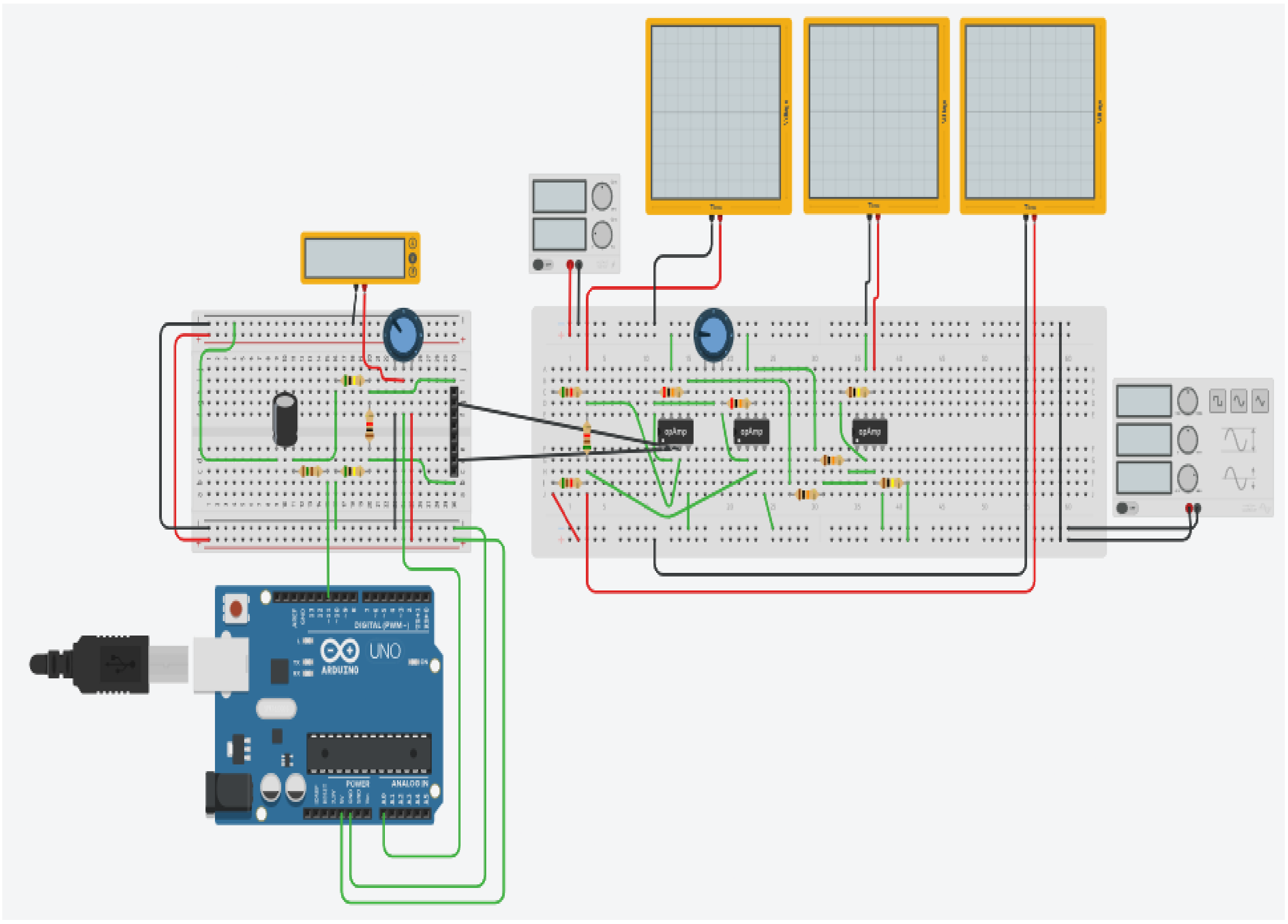
- Start a new circuit: In Tinkercad, create a new circuit design.
- Add components: Include an Arduino Uno, a breadboard, resistors, capacitors, and, if available as a simulated component, an AD8232 sensor module or simulate the signal with a function generator.
- Wire the circuit: Connect the components according to a schematic or instructional guide. Pay close attention to connections like the AD8232's output to an analog input on the Arduino, and the necessary power and ground connections.
- Code the Arduino: Program the Arduino to read the analog signal from the ECG sensor (or simulated signal) and display the values on the serial monitor or other output components.
- Simulate and verify: Run the simulation in Tinkercad and observe the output to verify that the circuit is functioning as expected and an ECG-like waveform is being displayed.

Code:

```
#define NumPts 256    // num point is one beat
#define minBPM 40     // min allowable HR
```

```
#define maxBPM 120    // max allowable HR
#define NOISE  50     // amplitude random noise
#define BPMVAR 2      //BPM variability

const uint8_t PROGMEM ECG[] = {
  43,43,43,44,44,46,47,48,50,52,54,55,57,59,
  61,63,65,67,69,70,71,72,73,74,74,74,74,74,
  73,72,71,70,69,67,65,63,61,59,57,55,54,52,
  50,48,47,46,44,44,43,43,43,43,43,43,43,43,
  43,43,43,43,43,43,43,43,43,43,43,43,43,43,
  43,43,43,34,26,17,9,0,11,21,32,43,69,96,
  122,149,175,202,228,255,228,202,175,149,
  122,96,69,43,32,21,11,0,9,17,26,34,43,43,
  43,43,43,43,43,43,43,43,43,43,43,43,43,43,
  43,43,43,43,43,43,43,43,44,44,44,44,45,45,
  45,46,46,47,48,49,49,50,51,52,54,55,56,58,
  59,61,63,64,66,68,70,72,74,76,78,80,82,84,
  86,88,89,91,92,93,94,95,95,96,96,96,95,95,
  94,93,92,91,89,88,86,84,82,80,78,76,74,72,
  70,68,66,64,63,61,59,58,56,55,54,52,51,50,
  49,49,48,47,46,46,45,45,45,44,44,44,44,43,
  43,43,43,43,43,43,43,43,43,43,43,43,43,43,
  43,43,43,43,43,43,43,43,43,43,43,43,43,43,
  43,43,43,43,43,43,43,43
};
unsigned long oneHzSample, waitTime;
byte i      = 0;   // index
int fgPin   = 11;  // pin for output of signal
int ratePin = A0;  // Channel for HR control
int bpm;           // Actual heartrate to output
int Ai0 = 0;
int baud = 9600;
int led = 7;
void setup()
{
  oneHzSample = 1000000/NumPts;  // sample for the 1Hz signal expressed in microseconds
  pinMode(fgPin, OUTPUT);
  randomSeed(analogRead(5));
  Serial.begin(baud);
  pinMode(led, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop()
{
  // Read desired HR and allow to vary randomly
  bpm = map(analogRead(ratePin), 0, 1023, minBPM, maxBPM) + random(BPMVAR);
  // Calculate the time between beats
  waitTime = oneHzSample * (60/bpm);
  // Output signal one point at a time
  analogWrite(fgPin, pgm_read_byte_near(ECG+i) + random(50));  // write the selected waveform
  // Increment counter to read next point in wave
  i++;
  // Reset the counter to repeat the wave
  if(i == NumPts) i = 0;
  // Hold the sample value for the sample time
  delayMicroseconds(waitTime);
   int val = analogRead(Ai0);
    Serial.println(bpm);
    if(bpm >= 70.0)
  {
    digitalWrite(13, HIGH);
    delay(100);
  }
  else
  {
     digitalWrite(13, LOW);
    delay(100);
  }
}
```

Circuit Diagram:



Link: https://www.tinkercad.com/things/hbXyygH4jbr-portable-electrocardiograph-ecg

Note: Tinkercad Circuits offers a great environment for exploring the fundamentals of ECG circuit design and simulation. However, it's essential to remember that this is a simulation for educational purposes. Real-world ECG devices require specialized electrodes, advanced signal processing, and medical-grade calibration and safety standards.

Result: designed and simulated a basic portable ECG circuit in Tinkercad successfully.