

LAKKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(Autonomous & Affiliated to JNTUK, Kakinada & Approved by AICTE, New Delhi,)

NAAC Accredited with 'A' grade, accredited by NBA,
Certified by ISO 9001:2018, 500001:2018, 14001:2015.
L B Reddy Nagar, Mylavaram-521 230, NTR District, Andhra Pradesh.

DEPARTMENT OF AEROSPACE ENGINEERING

B. Tech III Year - I Semester

R23 Regulation Tenkering Lab-23ME01



Prepared by
Dr. Y. Raghu Vamsi

M. Tech, Ph. D
Sr. Assistant Professor, EEE
Mr. G V Surya Narayana,
M. Tech.
Assistant Professor, ASE

Name:
Roll Number:
Semester:
Academic year:

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)
L.B. REDDY NAGAR, MYLAVARAM,
N.T.R. DIST-521 230 ANDHRA PRADESH



Certificate

Certificate that this is a bonafied record of Practical work done in _____ laboratory by _____ with
Regd. No. _____ of _____ B. Tech course
(_____ Semester) in _____ branch
during the academic year _____.

No. of Experiments held _____

No. of Experiments done _____

Date:

TEACHER – IN – CHARGE

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

Exp. No.	Name of the Experiment	Date	Page No.	Faulty Signature
1.	Make your own parallel and series circuits using breadboard for any application of your choice			
2.	Demonstrate a traffic light circuit using breadboard.			
3.	Build and demonstrate automatic Street Light using LDR			
4.	Simulate the Arduino LED blinking activity in Tinkercad.			
5.	Build and demonstrate an Arduino LED blinking activity using Arduino IDE.			
6.	Interfacing IR Sensor and Servo Motor with Arduino.			
7.	Blink LED using ESP32.			
8.	LDR Interfacing with ESP32.			
9.	Control an LED using Mobile App.			
10.	Design and 3D print a Walking Robot			
11.	Design and 3D Print a Rocket.			
12.	Build a live soil moisture monitoring project, and monitor soil moisture levels of a remote plan in your computer dashboard.			

III Year I Semester	ENGINEERING SCIENCE TINKERING LAB	L	T	P	C
		0	0	2	1

Course Educational Objectives (CEO): The aim of tinkering lab for engineering students is to provide a hands-on learning environment where students can explore, experiment, and innovate by building and testing prototypes. These labs are designed to demonstrate practical skills that complement theoretical knowledge.

Course Outcomes (COs): At the end of the course - students will be able to:

CO1: Use Arduino boards for controlling various applications. **(Apply-L3)**

CO2: Control various applications using ESP32. **(Apply-L3)**

CO3: Design of different real time applications using breadboard, Mobile App and 3D printer. **(Apply-L3)**

List of Experiments:

- 1) Make your own parallel and series circuits using breadboard for any application of your choice.
- 2) Demonstrate a traffic light circuit using breadboard.
- 3) Build and demonstrate automatic Street Light using LDR.
- 4) Simulate the Arduino LED blinking activity in Tinkercad.
- 5) Build and demonstrate an Arduino LED blinking activity using Arduino IDE.
- 6) Interfacing IR Sensor and Servo Motor with Arduino.
- 7) Blink LED using ESP32.
- 8) LDR Interfacing with ESP32.
- 9) Control an LED using Mobile App.
- 10) Design and 3D print a Walking Robot
- 11) Design and 3D Print a Rocket.
- 12) Build a live soil moisture monitoring project, and monitor soil moisture levels of a remote plan in your computer dashboard.
- 13) Demonstrate all the steps in design thinking to redesign a motor bike.

Students need to refer to the following links:

- 1) <https://aim.gov.in/pdf/equipment-manual-pdf.pdf>
- 2) <https://atl.aim.gov.in/ATL-Equipment-Manual/>
- 3) <https://aim.gov.in/pdf/Level-1.pdf>
- 4) <https://aim.gov.in/pdf/Level-2.pdf>
- 5) <https://aim.gov.in/pdf/Level-3.pdf>

1. Make your own parallel and series circuits using breadboard for any application of your choice

Aim:

To construct and compare **series** and **parallel circuits** using a breadboard for a basic application — **LED lighting** — and to observe how the current and voltage behave in each configuration.

Apparatus

- Breadboard
- 3x LEDs (same color or different)
- 3x Resistors (330Ω or 220Ω, depending on LED type)
- Jumper wires
- 9V battery + battery clip (or 5V regulated power supply)
- Multimeter (optional, for measurement)

Theory:

1. Series Circuits

- **Definition:** Components are connected end-to-end, forming a single path for current to flow.
- **Key Features:**
 - **Current (I):** Same through all components.
 - **Voltage (V):** Divided among components.
 - **Resistance (R):** Total resistance is the sum of all resistors.

$$R_{\text{total}} = R_1 + R_2 + R_3 + \dots$$

-
- **Example:** Christmas lights—if one bulb goes out, the entire string stops working.

2. Parallel Circuits

- **Definition:** Components are connected across the same two points, forming multiple paths for current.
- **Key Features:**
 - **Voltage (V):** Same across all branches.
 - **Current (I):** Divided among branches.
 - **Resistance (R):** Total resistance is **less** than the smallest individual resistor.

$$\frac{1}{R_{\text{total}}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots$$

○

- **Example:** Household wiring—if one device is off, others still work.

Application

LED room lighting system:

- **Series circuit** simulates a linear LED strip where one failure breaks the whole circuit.
- **Parallel circuit** simulates independent lights in a room where one can fail without affecting others.

Circuit Designs

A. Series Circuit

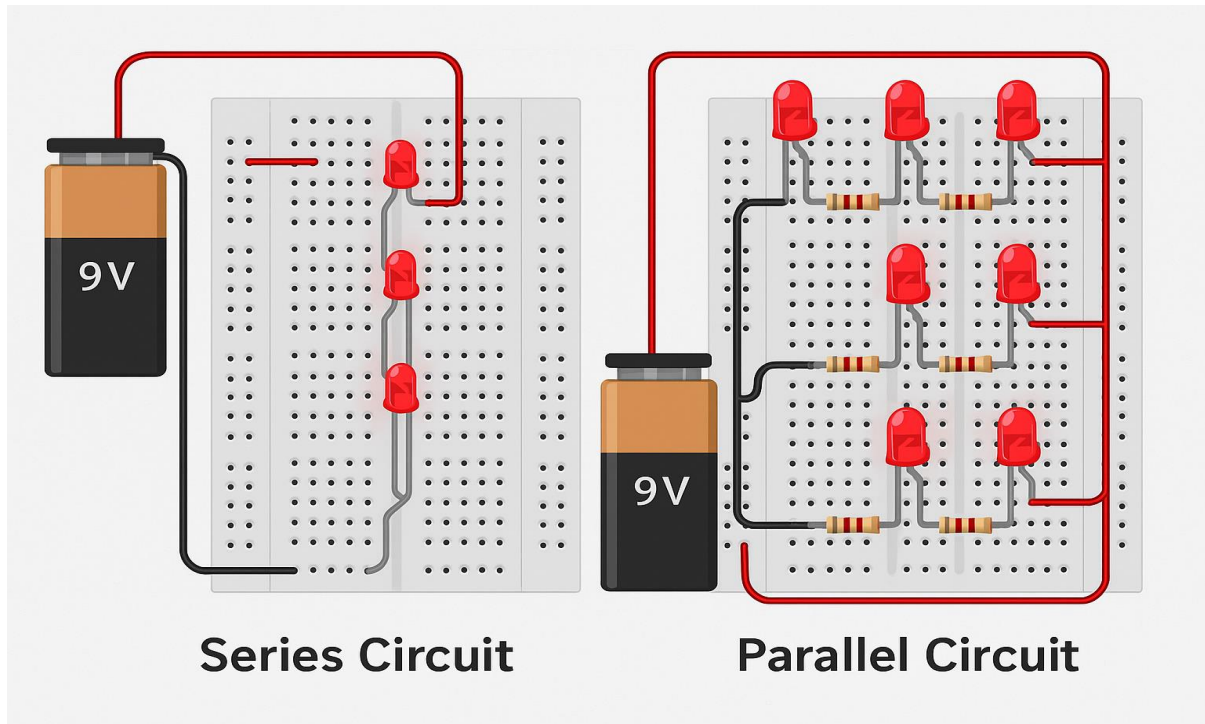
- Connect the **positive terminal of the battery** to one end of the first **resistor**.
- From the resistor, connect to the **anode (long leg)** of the first LED.
- Connect the **cathode** of the first LED to the anode of the **second LED**.
- Continue the same to the **third LED**.
- Finally, connect the cathode of the third LED to the **negative terminal of the battery**.

All LEDs are in a **single path**: current flows through each in sequence.

B. Parallel Circuit

- Connect the **positive rail** of the breadboard to the **battery's positive terminal**.
- Connect the **negative rail** to the **battery's negative terminal**.
- Place each **resistor** in series with an **individual LED**, and connect each LED-resistor pair **between the same power rails**.

Each LED gets the **full voltage** independently.



Procedure

1. Assemble the **series circuit** on the breadboard following the design.
2. Power the circuit and observe LED brightness.
3. Measure voltage across each LED using a multimeter.
4. Break one LED connection and observe the effect (all LEDs go off).
5. Now build the **parallel circuit** on the breadboard.
6. Power the circuit and observe LED brightness.
7. Disconnect one LED — the others stay on.
8. Measure current drawn using a multimeter if possible.

Observations

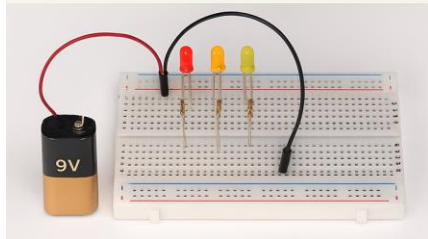
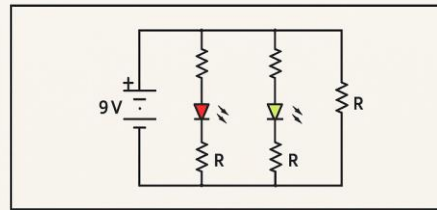
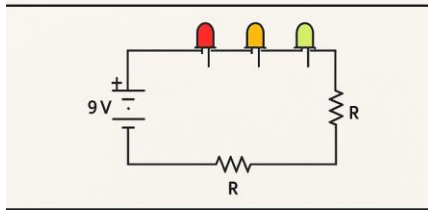
Parameter	Series Circuit	Parallel Circuit
LED Brightness	Dimmer	Brighter
Voltage per LED	Shared (drops)	Full supply voltage
Effect of 1 LED failure	All LEDs turn off	Others remain on
Total Current Draw	Lower	Higher

Conclusion

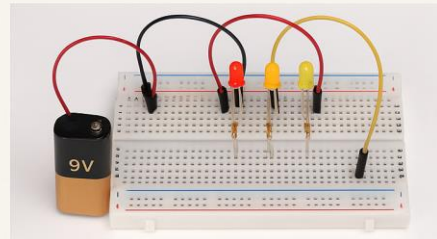
- In **series**, LEDs depend on one another — failure in one affects the whole circuit.
- In **parallel**, LEDs operate independently and are brighter due to full voltage availability.

- For practical applications like room or decorative lighting, **parallel circuits** are more reliable.

LAB EXPERIMENT:



SERIES CIRCUIT



PARALLEL CIRCUIT

2. Demonstrate a traffic light circuit using breadboard

Aim:

To design and implement a basic **traffic light control circuit** on a breadboard using **LEDs, resistors** traffic light sequence (Red → Yellow → Green).

Apparatus:

- Breadboard
- 1x Red LED
- 1x Yellow LED
- 1x Green LED
- 3x 220Ω or 330Ω resistors
- Jumper wires
- SPDT switch (or multiple push buttons)
- 5V–9V battery or regulated power supply

Theory:

Application

- Traffic control systems at road intersections — to manage vehicle movement safely using signal timing logic.

Circuit Diagram Description

- Connect the **common ground** to the cathodes (short leg) of all three LEDs.
- Connect each **anode (long leg)** through a **resistor** to one **switch terminal**.
- The other side of each switch goes to **Vcc**.
- Pressing each switch lights up the respective LED (Red, Yellow, or Green).

Procedure

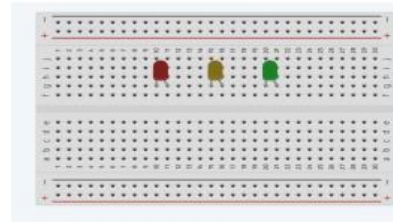
1. Place LEDs on the breadboard.
2. Connect resistors in series with each LED.
3. Wire push buttons or switches to control each LED.
4. Connect battery to the circuit.
5. Press each button to simulate traffic light manually.

Observations

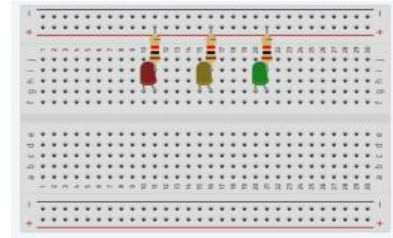
LED Color	Function	Duration	Condition
Green	Go	5 sec	ON
Yellow	Caution	2 sec	ON
Red	Stop	5 sec	ON

Steps to make a traffic light

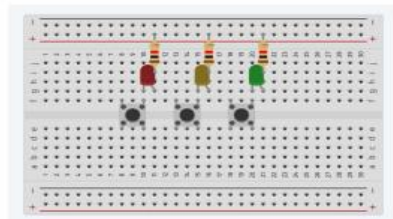
Step 1: Insert a LED into the breadboard.



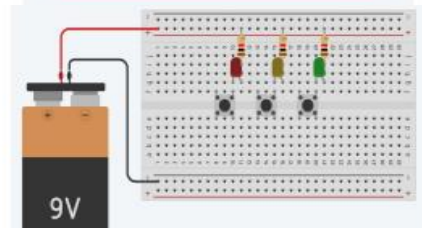
Step 2: Insert the Resistor into the breadboard and connect it to the positive terminal of LED.



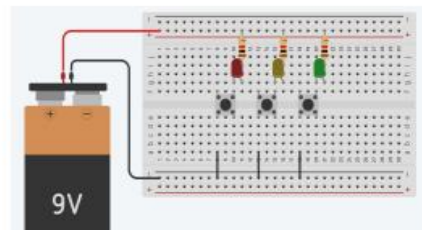
Step 3: Add the push button and connect it to the negative side of LED like as shown in the picture.



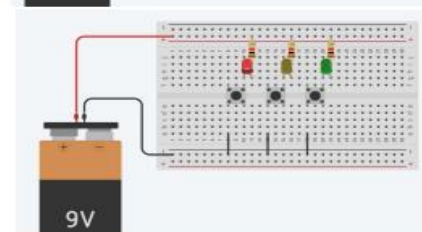
Step 4: Connect the 9v battery to the circuit.



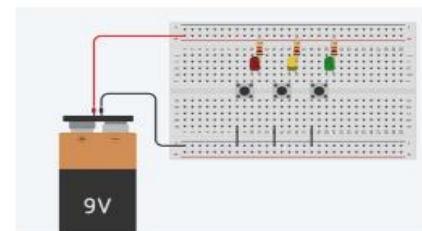
Step 5: Connect the pushbutton and neutral of the battery by using the connecting wires.



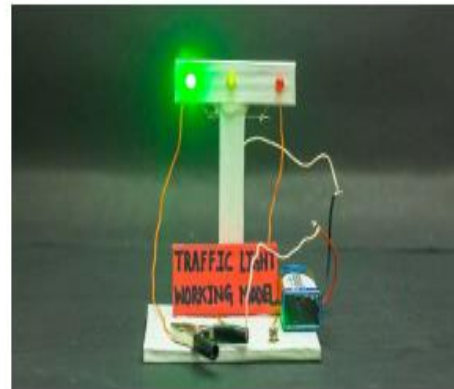
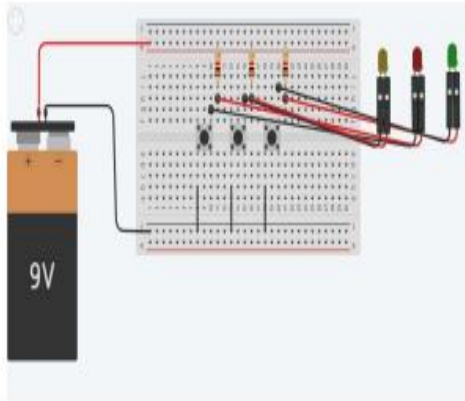
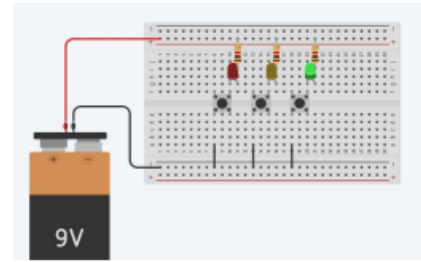
Step 6: To turn on red light, press the push button that is connected to the Red LED.



Step 7: To turn on yellow light, press the push button that is connected to the Yellow LED



Step 8: To turn on green light, press the push button that connected to the Green LED



Conclusion

The traffic light circuit demonstrates the sequencing and timing required for a simple traffic control system. Using an Arduino allows automation, making it more practical and adaptable to real-world systems.

Reflection and Learnings:

1. Which type of circuit is a Traffic Light. Series or Parallel?
2. How will you apply these skills outside class?
3. What did you or your team struggle with while doing this activity and why? How can you improve next time?
4. How did you contribute to the team and how do you think it helped? How else could you have contributed?
5. What communication challenges did you and your team face? How can your team communicate better next time?

3. Build and demonstrate automatic Street Light using LDR

Aim: To build and demonstrate automatic Street Light using LDR

Components Required:

- LED
- LDR
- Coin Cell/Rechargeable 9 V batteries
- Breadboard
- Jumpers
- Multimeter
- Resistors

Theory:

Automatic Street Light is such a kind of lighting system where the light will automatically turn ON and OFF implemented for usage to reduce power consumption. It is applied extensively for street lights which light the parking areas and other outdoor locations, thus improving security requirements in a locality as well lowering energy consumption. This system uses an important principle LDR (Light Dependent Resistor) is used as a sensor to detect the light surrounding in that environment. Even when light levels fall, for instance at nightfall the system activates. The system is programmed to switch them off when daylight returns.

Importance of Automation in Street Lighting Systems

Street lighting automation leads to several benefits such as:

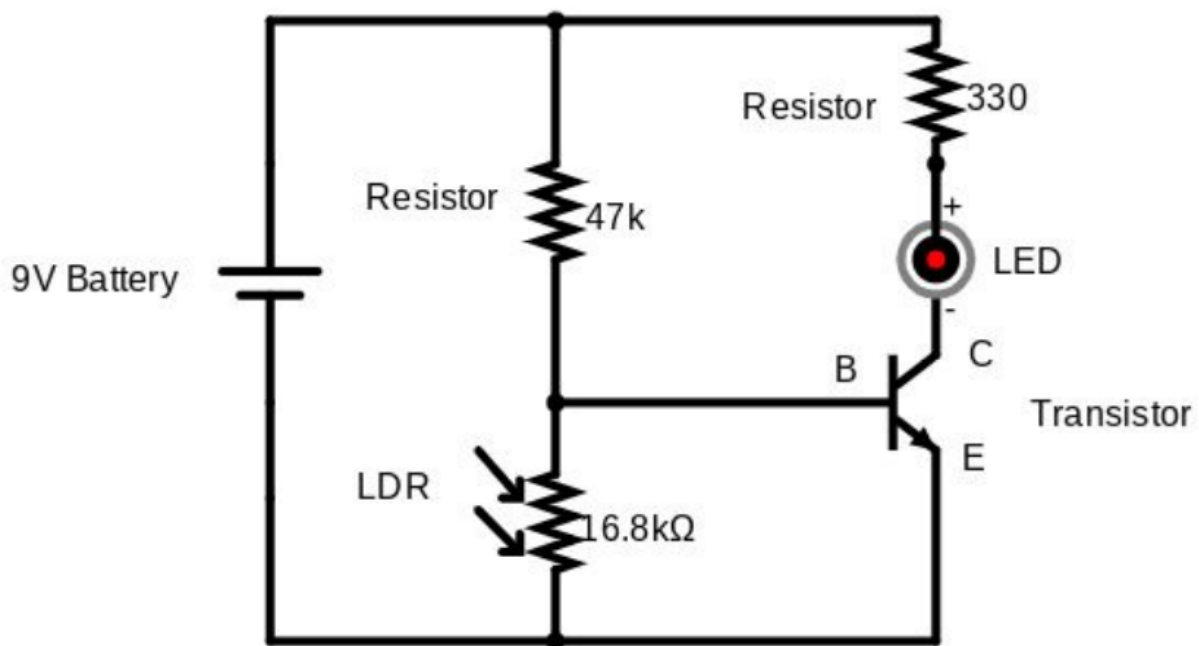
- **Energy efficiency:** The street light automatically turns on low wattage when only few pedestrians move across the road and this helps in reducing energy.
- **Cost-effectiveness:** Less resource consumption leads to lower electricity bills, basically making it an economic solution at the end of the day.
- **Increased safety:** Automatic street lights can reduce accidents and help keep communities safe by ensuring that areas remain well lit throughout the night.

They are commonly used in public infrastructure, residential areas and commercial properties. The automatic street light project detailed in this guide

is a basic model and comes with the option to scale it up, depending on your specific requirements hence being well suited for small applications all the way up to more intricate setups.

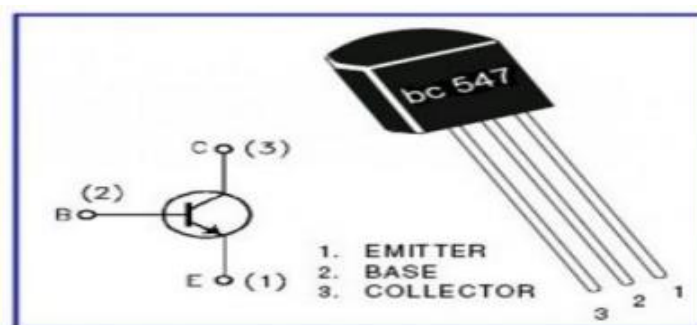
Circuit Diagram:

When sufficient light falling on LDR the resistance of LDR is very low, as a result, all current is flowing from resistor R2 and LDR so LED D1 is not glowing, When there is darkness, and no light is falling on LDR, so the LDR resistance became very high so, current will flow to the base of transistor Q1 BC547, so transistor became turn on and LED D1 to glow.

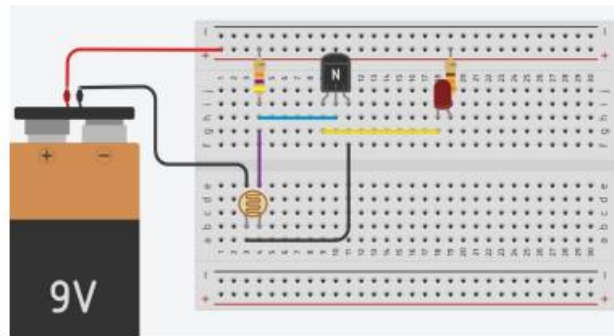


Procedure:

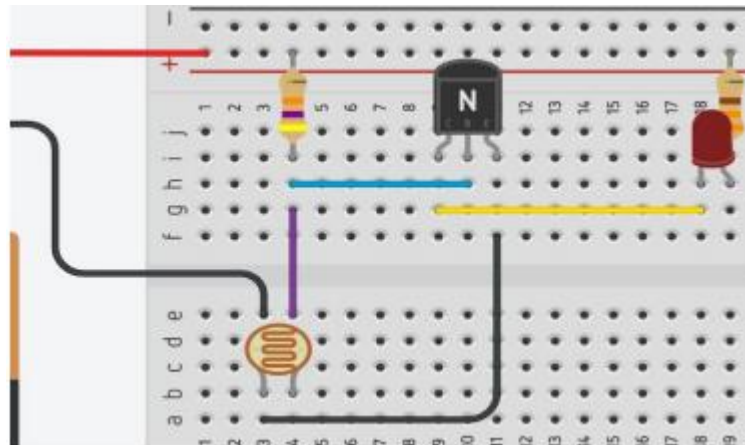
- Identify the terminals of transistor as shown in the image.



- Connect positive and negative terminal of the battery to breadboard as shown in the image.



- Place the LED, Resistors, Transistor and LDR in breadboard.



- Connect each components carefully as per the circuit.
- Check the working of circuit by placing the circuit in a dark area

Reflection and Learnings:

1. Take the LDR Circuit to a dark place. Write down your observation in the space below.
2. What is the function of a LDR?
3. What did you or your team struggle with while doing this activity and why? How can you improve next time?
4. How did you contribute to the team and how do you think it helped? How else could you have contributed?
5. What communication challenges did you and your team face? How can your team communicate better next time?

Try it yourself!

Using the concepts you have learnt so far, Try creating a Street light system with number of LEDs.

4. Simulate the Arduino LED blinking activity in Tinkercad

Aim:

To simulate the blinking LED activity using Arduino in Tinkercad software.

Components required in Tinkercad software:

- Arduino UNO
- USB Cable
- 220Ω Resistor
- LED
- Breadboard
- Jumper Wires

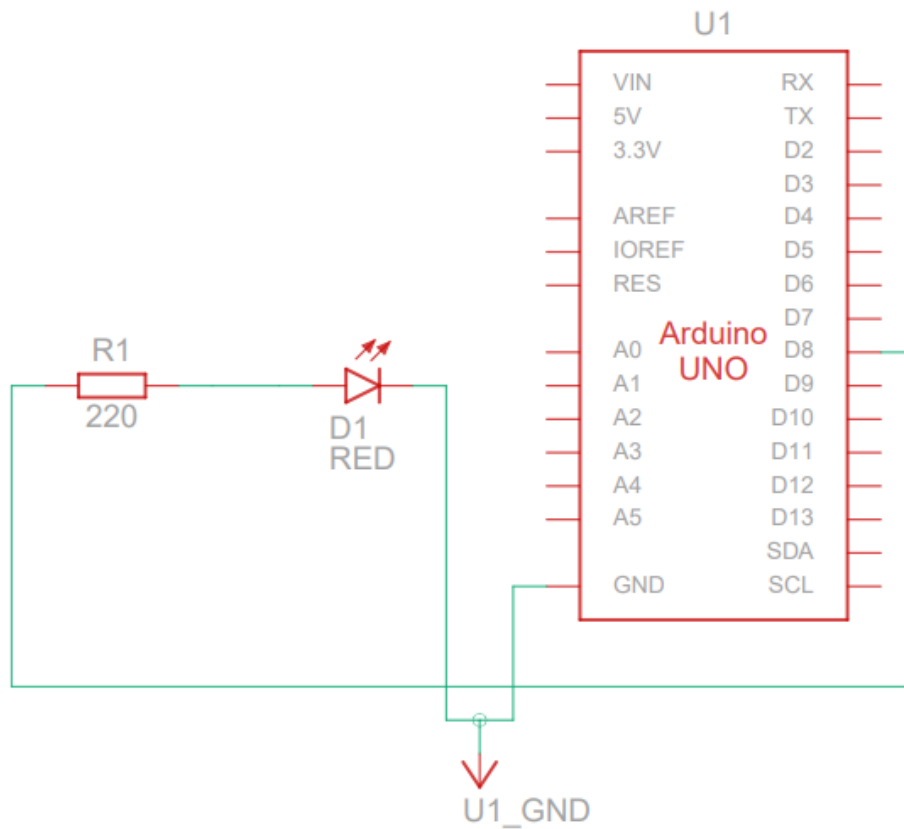
Theory:

What is Tinkercad?

Tinkercad Circuits is an online software which is the easiest way to get students started with learning electronics. Using an interactive circuit editor, students can explore, connect, and code virtual projects with simulated components. Tinkercad helps students facilitate their learning virtually to understand and breakdown complex concepts.

In this work, we will program the Arduino's GPIO output high level (+5V) and low level (0V), and then make the LED which is connected to the Arduino's GPIO flicker with a certain frequency.

1. What is the LED? The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes: a positive electrode and a negative one. It lights up only when a forward current passes, and it can flash red, blue, green, yellow, etc. The color of the light depends on the material it is made. In general, the drive current for LED is 5-20mA. Therefore, in reality it usually needs an extra resistor for current limitation so as to protect the LED.
2. What is resistor? The main function of the resistor is to limit currents. In the circuit, the character 'R' represents resistor, and the unit of resistor is ohm(Ω). A band resistor is used in this experiment. It is a resistor with a surface coated with some particular color through which the resistance can be identified directly. There are two methods for connecting LED to pins of an Arduino board:
1 As shown in the schematic diagram, the anode of the LED is connected to Arduino's GPIO via a resistor, and the cathode to the ground (GND). When the GPIO outputs high level, the LED is on; when it outputs low, the LED is off.



Blinking LED Schematic diagram

Introduction to Arduino: Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects, You can connect various sensors to their input pin and get the output accordingly. The Arduino hardware and software were designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smartphone or your TV!

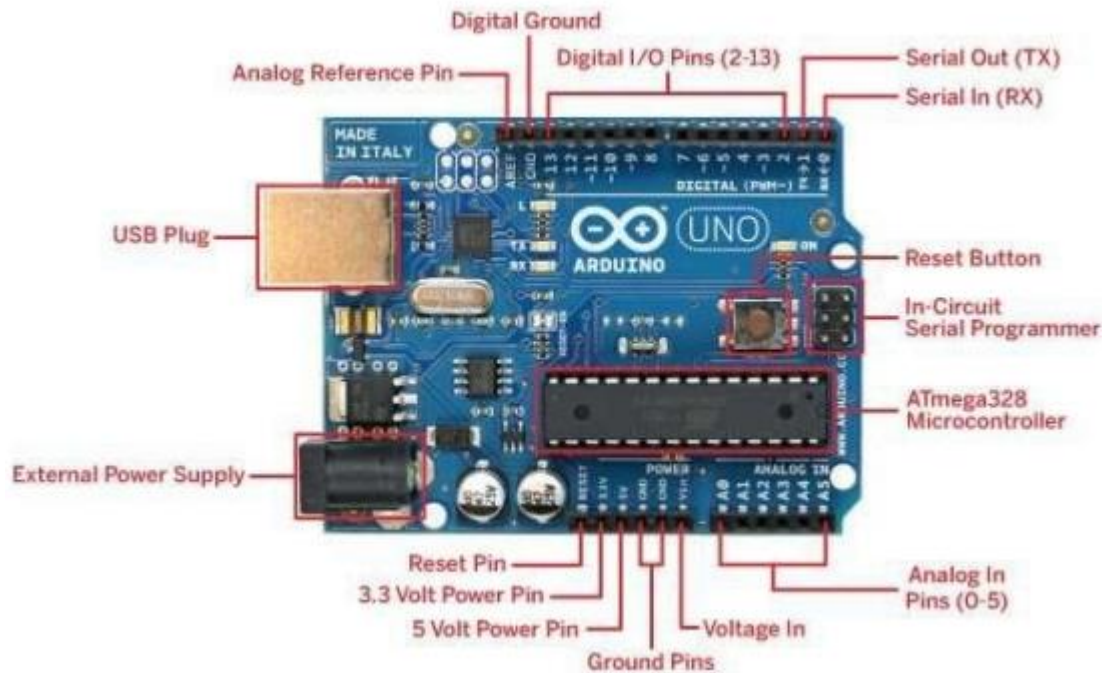
Types of Arduino:

Arduino's boards are available in different sizes, form factors, and different no. of I/O pins etc. Some commonly known and frequently used Arduino boards are Arduino UNO, Arduino Mega, Arduino Nano, Arduino Micro, and Arduino Lilypad. The Uno is one of the more popular boards in the Arduino family and a great choice for beginners. Let's understand how to use Arduino using a small activity.

Arduino Uno:

Arduino UNO is a basic and inexpensive Arduino board and is the most popular of all the Arduino boards. Arduino UNO is considered to be the best prototyping board for beginners in electronics and coding. Arduino Uno is a microcontroller board based on an 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists of other components such as a crystal oscillator, serial communication, voltage regulator, etc.

to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header, and a reset button.

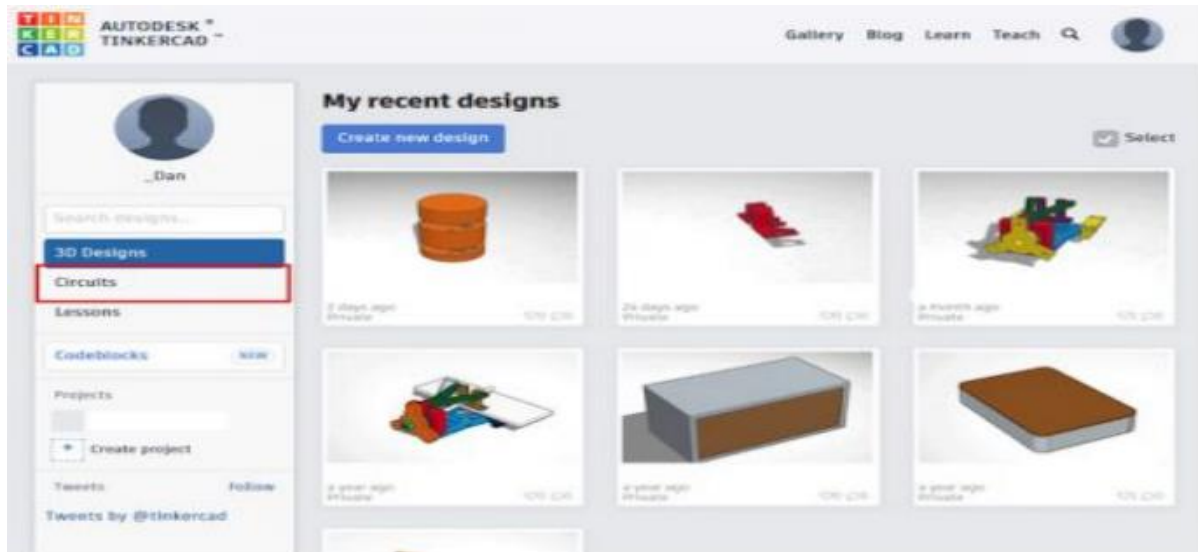


Key functions of Arduino:

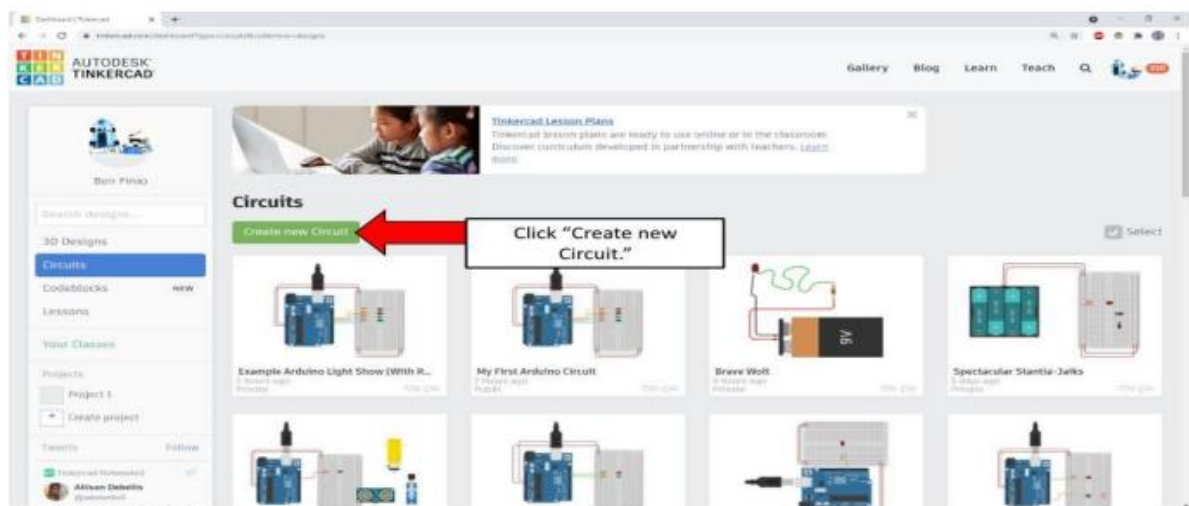
- **setup():** The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.
- **loop():** After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.
- **pinMode():** Configures the specified pin to behave either as an input or an output. As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.
- **digitalWrite():** Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.
- **delay():** Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Procedure:

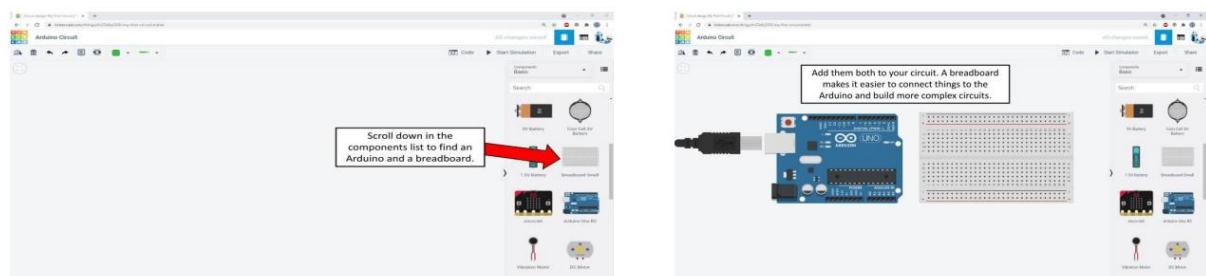
1. Visit the Tinkercad website (<https://tinkercad.com>) and create an account or log into an existing one. Then select Circuits on the left side of the screen:



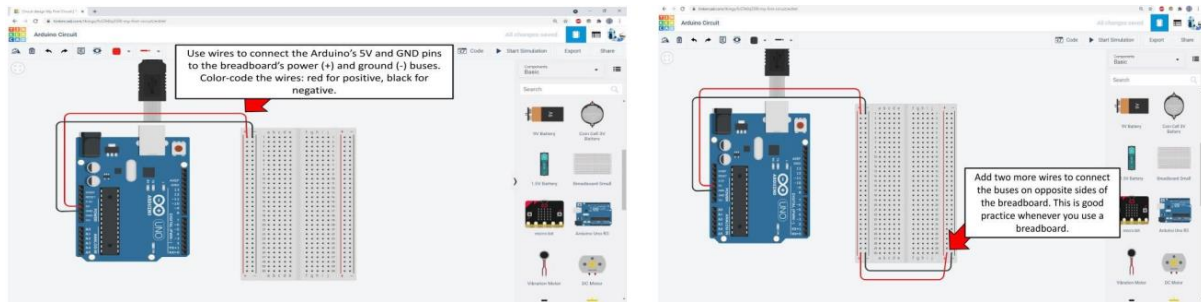
2. Click on “Create new Circuit”.



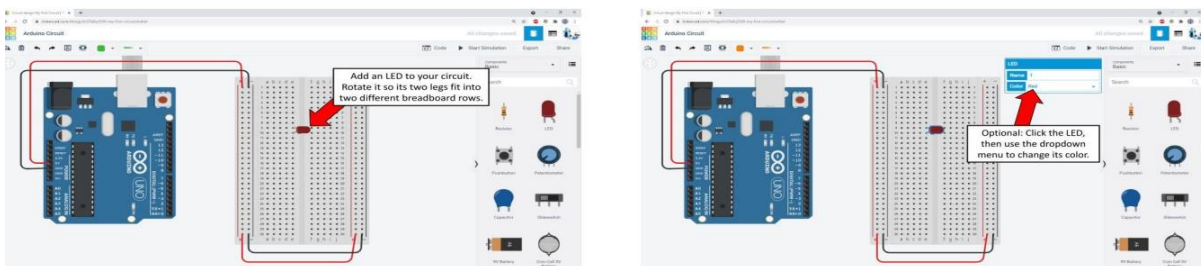
3. Add the Breadboard and Arduino Uno.



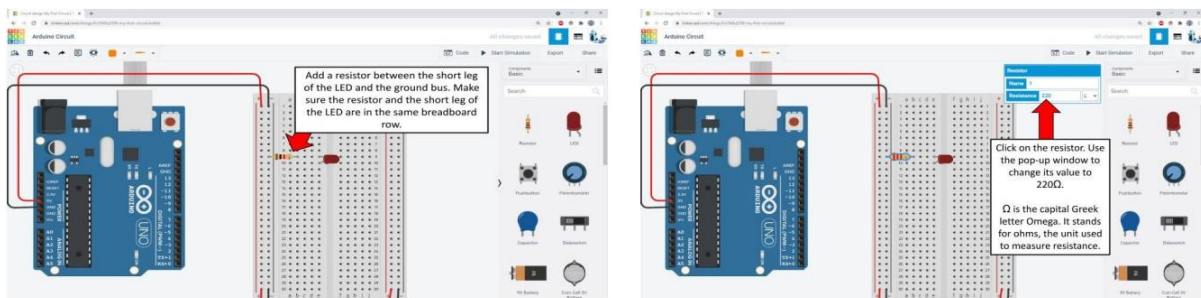
4. Connect +ve and -ve wires to the breadboard.



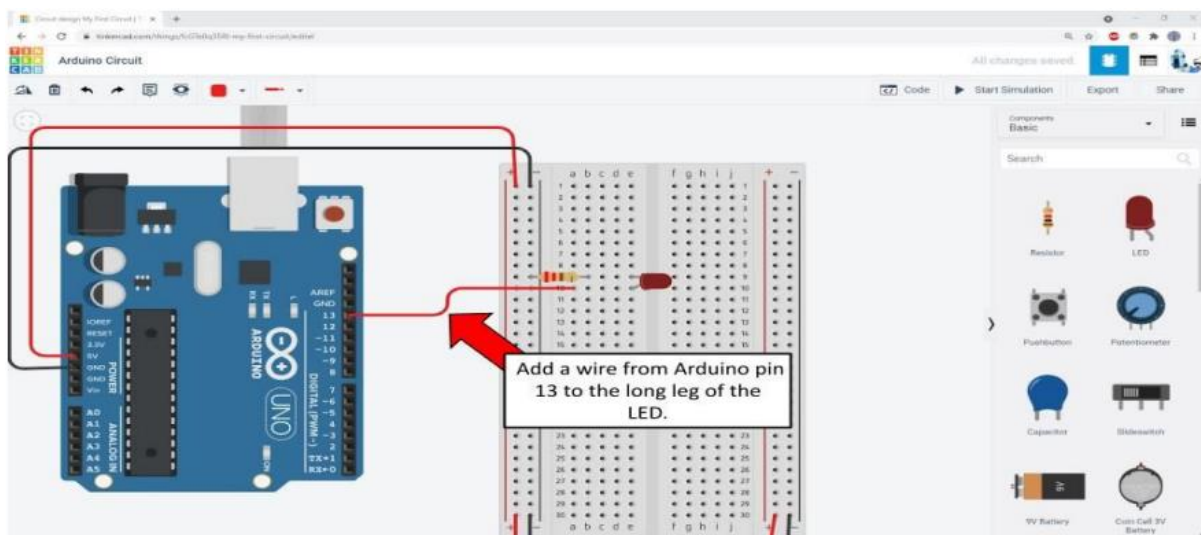
5. Add an LED.



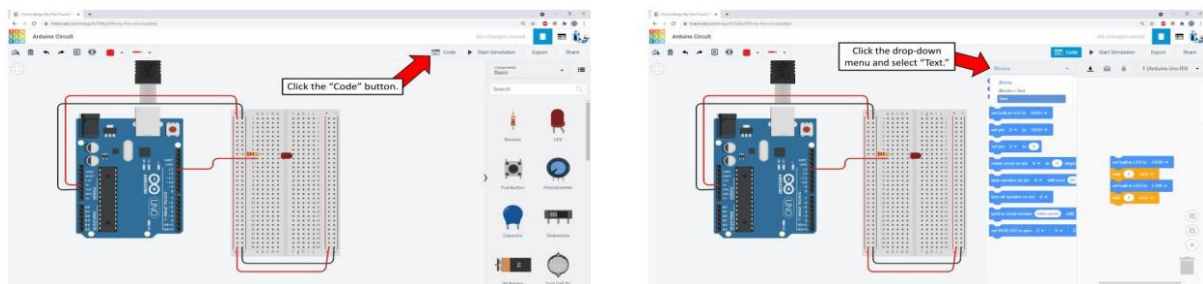
6. Add a Resistor.



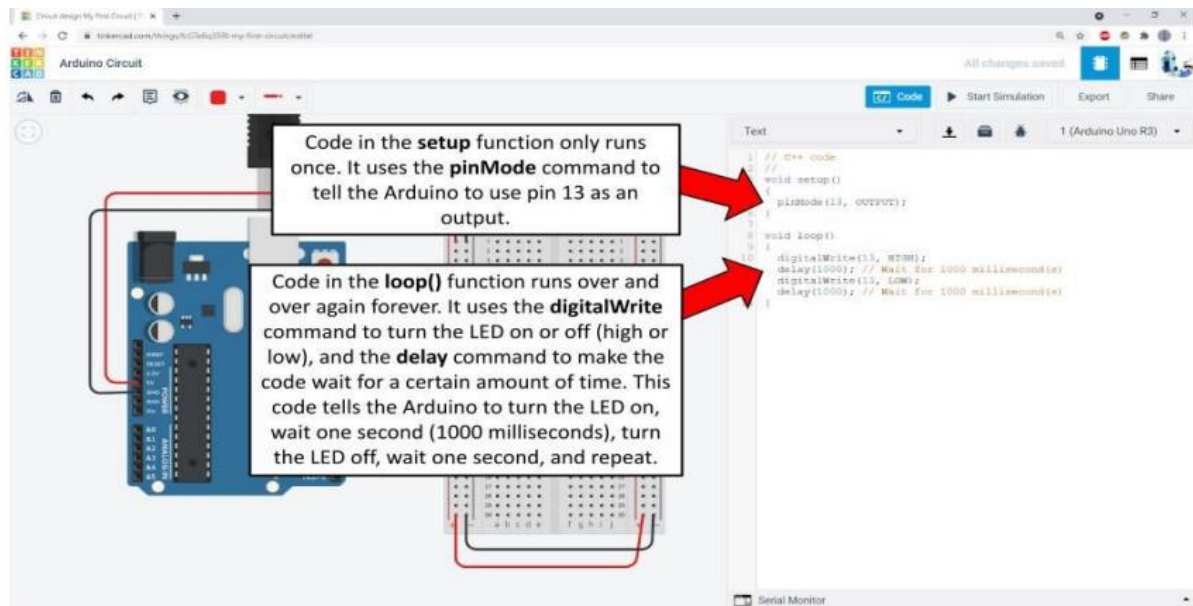
7. Add a wire from Arduino Pin 13 to the Anode of LED.



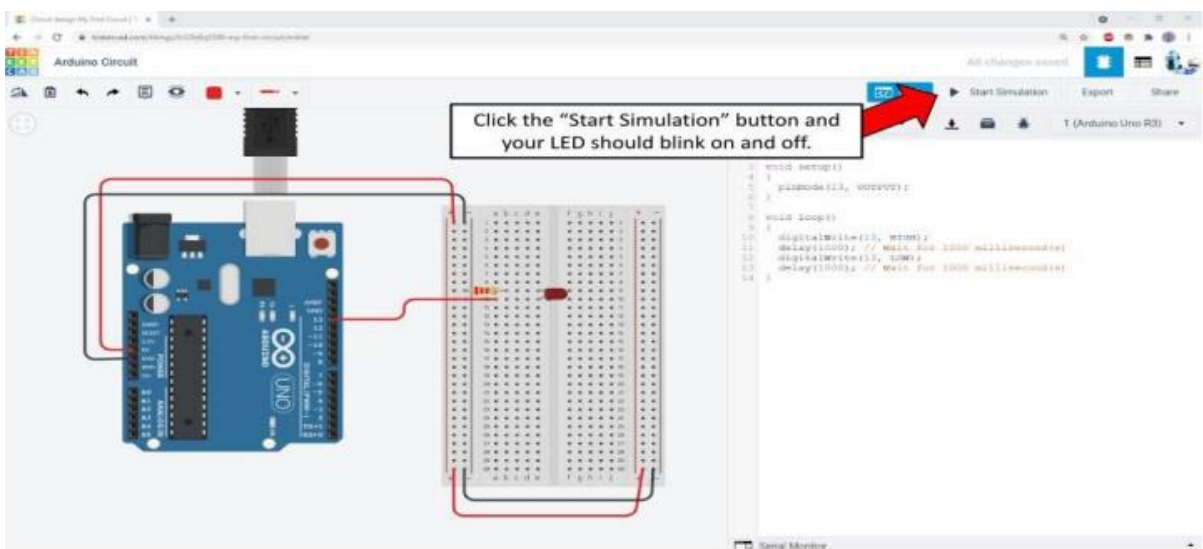
8. Open the Coding Window.



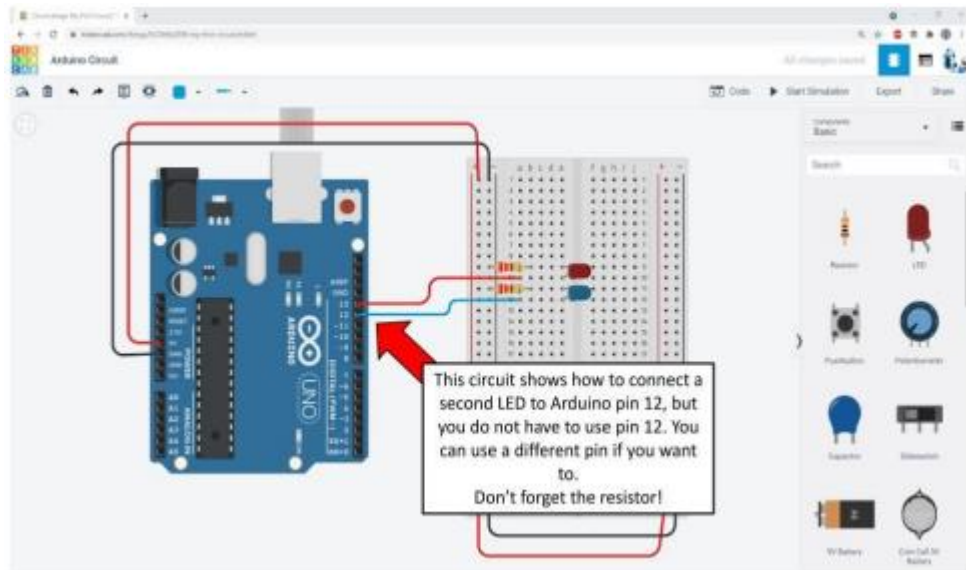
9. Write the Code. Refer Arduino code-1



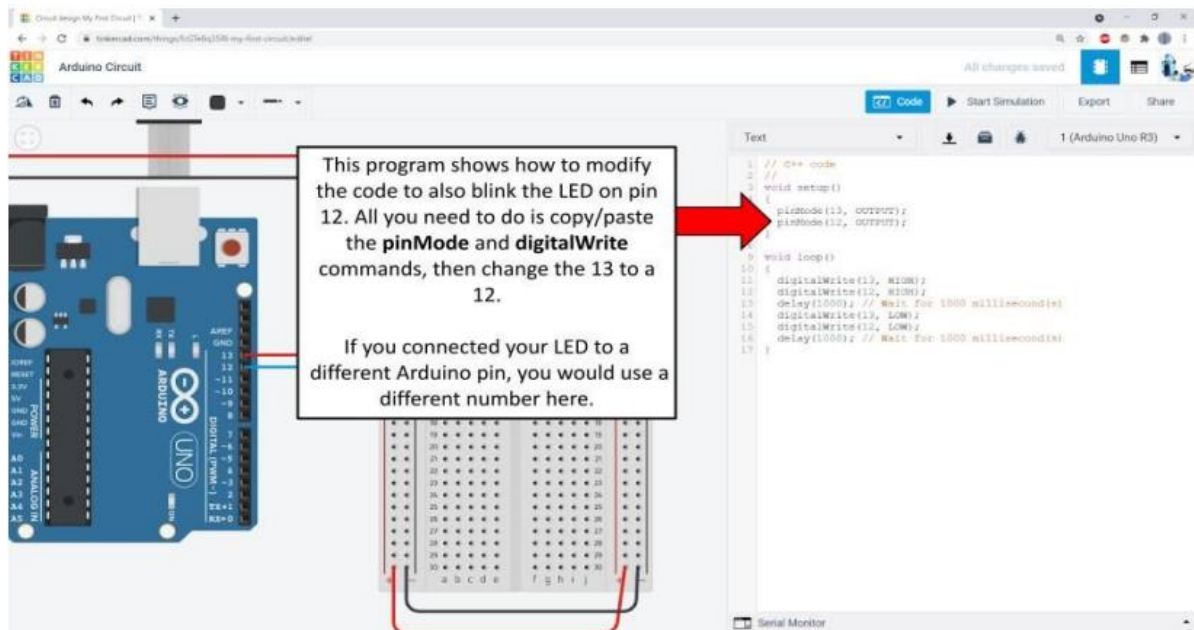
10. Start the Simulation - Your LED should blink on and off, If your LED does not blink, carefully double-check your breadboard wiring.



11. Add the second LED.



12. Write the code for 2 LED circuit (you can modify your existing program) Refer Arduino code-2.



Arduino Code-1:

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
```

```

delay(1000); // Wait for 1000 millisecond(s)
}

```

Arduino Code-2:

```

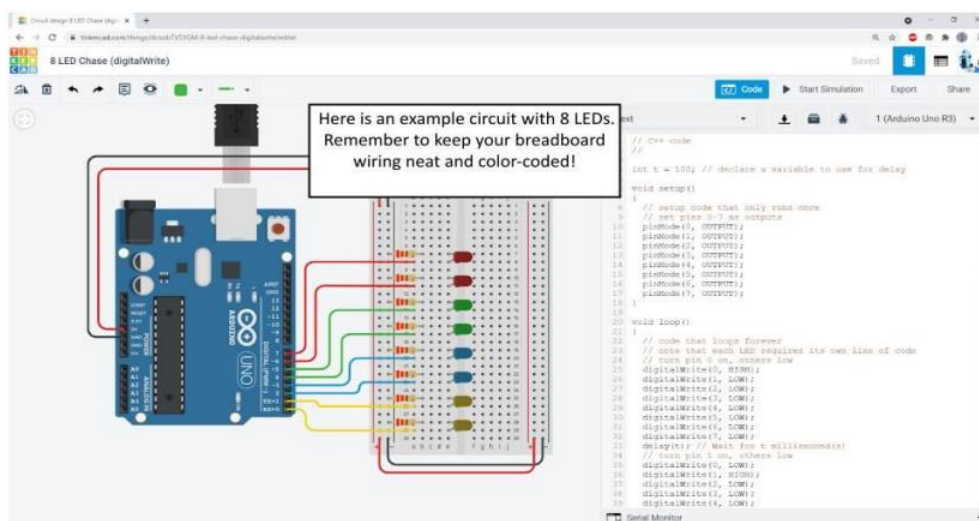
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}

```

Challenge:

- Design your light show
- Use what you have learned to program your own LED light show!
- You can connect up to 14 LEDs to your Arduino
- Remember: -Use the pinMode command to set pins as outputs. -Use the digitalWrite command to turn pins on and off. -Use the delay command to make the program wait. You can use variables to make it easier to adjust your delay times.



Code :

```
int t = 100; // declare a variable to USE FOR DELAY
void setup()
{
  // setup code that only runs once
  // set pins 0-7 as outputs
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}
void loop()
{
  // code that loops forever
  // note that each LED requires its own line of code
  // turn first LED on, others low
  digitalWrite(0, HIGH);
  digitalWrite(1, LOW);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  delay(t); // wait for t millisecond(s)
  // turn pin 1st and 2nd LED on, others low
  digitalWrite(0, HIGH);
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
```

```

digitalWrite(7, LOW);
delay(t); // wait for t millisecond(s)
// turn 1st,2nd,3rd and 4th LED on, others low
digitalWrite(0, HIGH);
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
delay(t); // wait for t millisecond(s)
//turn 1st,2nd,3rd,4th and 5th LED on, others low
digitalWrite(0, HIGH);
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
delay(t); // wait for t millisecond(s)
// turn 1st,2nd,3rd,4th,5th and 6th LED on, others low
digitalWrite(0, HIGH);
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
delay(t); // wait for t millisecond(s)
// turn 1st,2nd,3rd,4th,5th,6th and 7th LED on, others low
digitalWrite(0, HIGH);
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);

```

```

digitalWrite(4, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, LOW);
delay(t); // wait for t millisecond(s)
// All LEDs ON
digitalWrite(0, HIGH);
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
delay(t); // wait for t millisecond(s)
// OFF all LEDs
digitalWrite(0, LOW);
digitalWrite(1, LOW);
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
digitalWrite(8, LOW);
delay(t); // wait for t millisecond(s)
}

```

Reflection and Learnings:

1. Share at least 3 skills or lessons that you learned today.
2. Why do you think we use resistors while connecting the LED? What will happen if we don't use resistors.
3. What did you or your team struggle with while doing this activity and why? How can you improve next time?
4. How did you contribute to the team and how do you think it helped? How else could you have contributed?
5. Research and find out the differences between an Integrated Circuit (IC) and a Microcontroller.

5. Build and demonstrate an Arduino LED blinking activity using Arduino IDE

Aim:

To build and demonstrate blinking LED activity using Arduino IDE.

Components required:

1. Arduino IDE (Online Software)
2. Arduino Uno Board
3. LEDs
4. 220 Ohm Resistor
5. Breadboard
6. Jumper Wires

Theory:

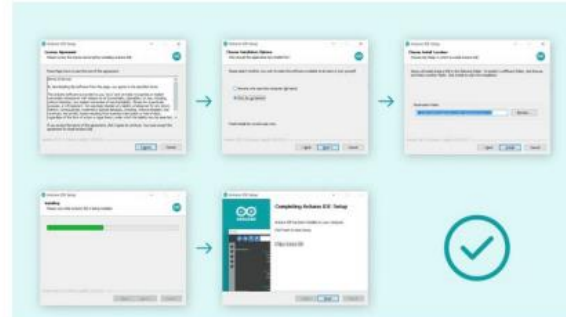
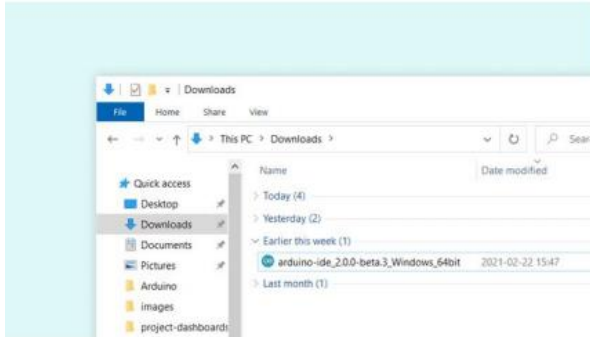
Arduino IDE is an open source software for programming Arduino boards, We already tried arduino simulation in Tinkercad now we are going to do the same activities using physical arduino board, we need to download and install Arduino IDE 2,0 software for programming arduino. The Arduino IDE 2.0 The Arduino IDE 2.0 is an open-source project. It is a big step from its sturdy predecessor, Arduino IDE 1.x, and comes with revamped UI, improved board & library manager, debugger, autocompletes feature, and much more.

Steps to build an Arduino LED blinking activity using Arduino IDE:

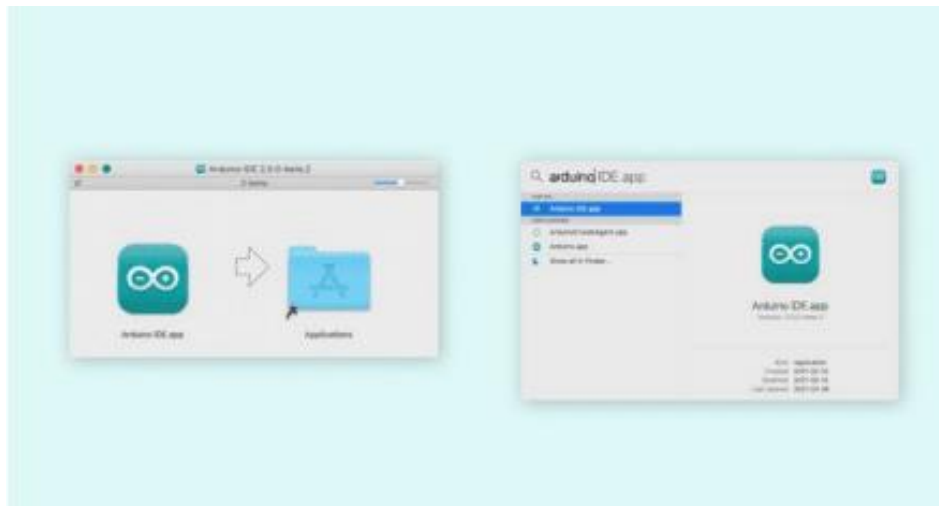
1. Download & Install Arduino IDE 2.0 on your computer (If you already have Arduino software on your PC skip this step) Goto <https://www.arduino.cc/en/software> and Download the appropriate version.



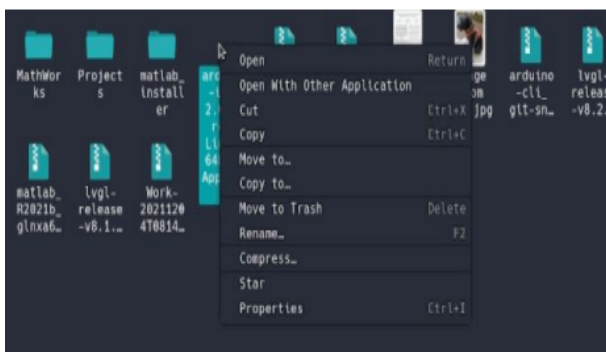
Windows Installation: To install the Arduino IDE 2.0 on a Windows computer, simply run the file downloaded from the software page.



macOS Installation: To install the Arduino IDE 2.0 on a macOS computer, simply copy the downloaded file into your application folder.

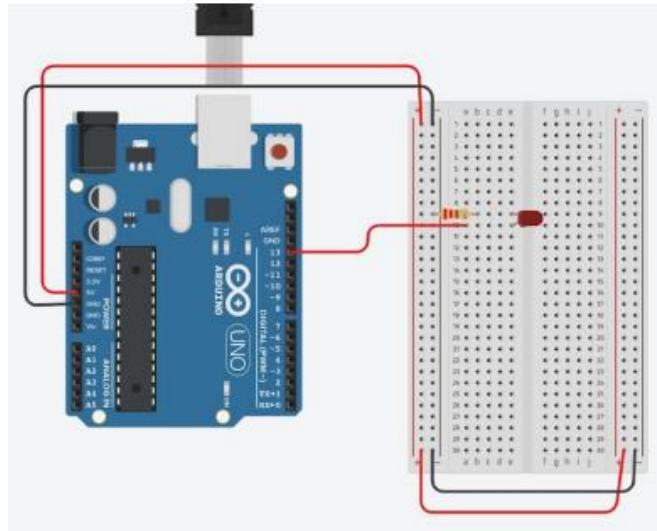


Linux Installation: To install the Arduino IDE 2.0 on Linux, first download the AppImage 64 bits (X86-64). Before we launch the editor, we need to first make it an executable file. right-click the file -> choose Properties -> select Permissions tab -> click the Allow executing file as program box.

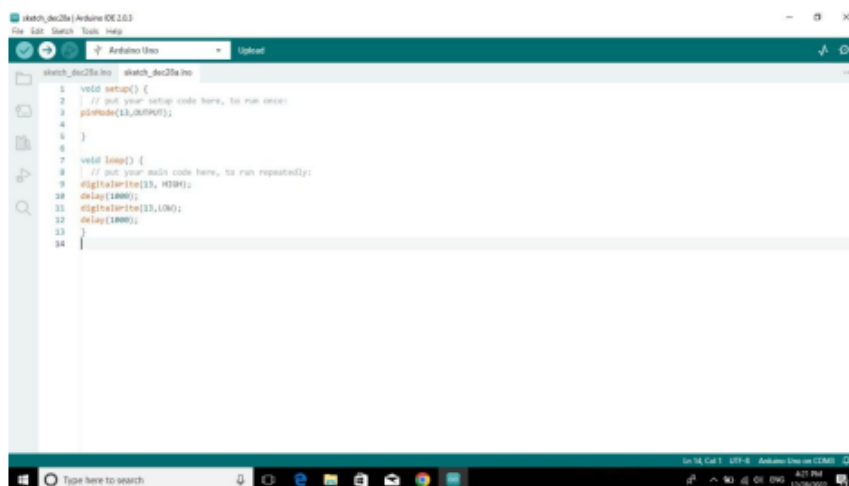


You can now double click the file to launch the Arduino IDE 2 on your Linux machine.

2. Connect circuit for a single LED blinking as per the circuit diagram.
 - Positive (Long Leg) of LED Connected to Pin no 13 of arduino.
 - Negative (Short Leg) of LED to GND via resistor.



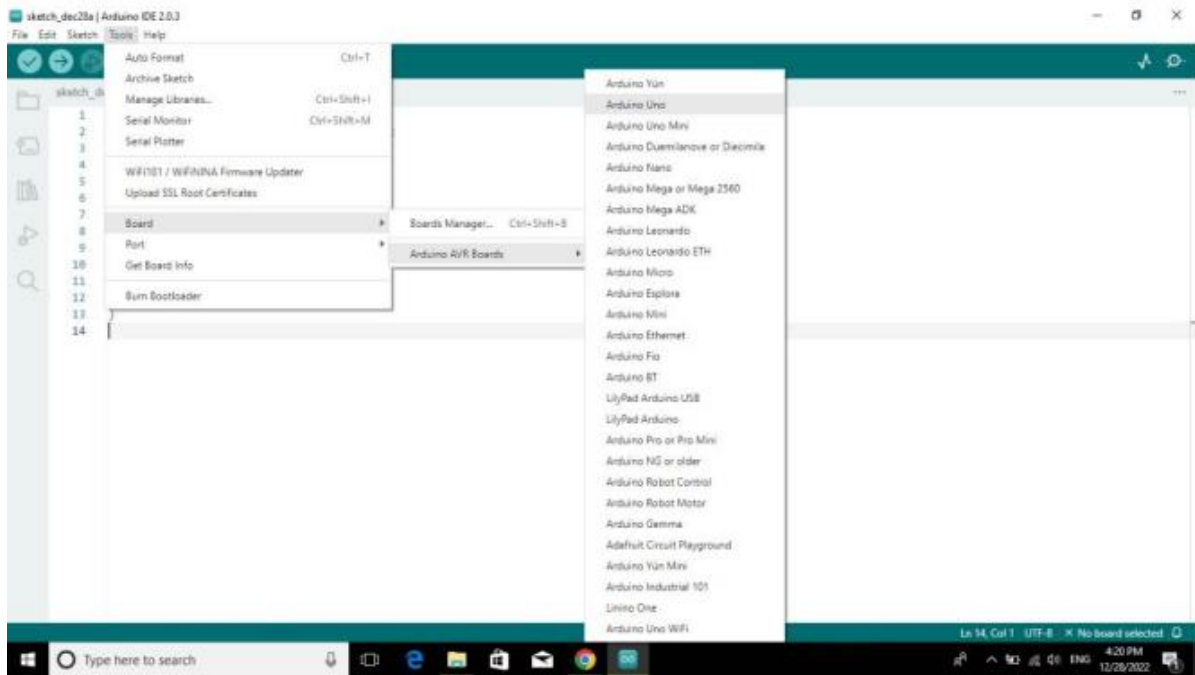
3. Open Arduino software and write the code for LED Blinking.



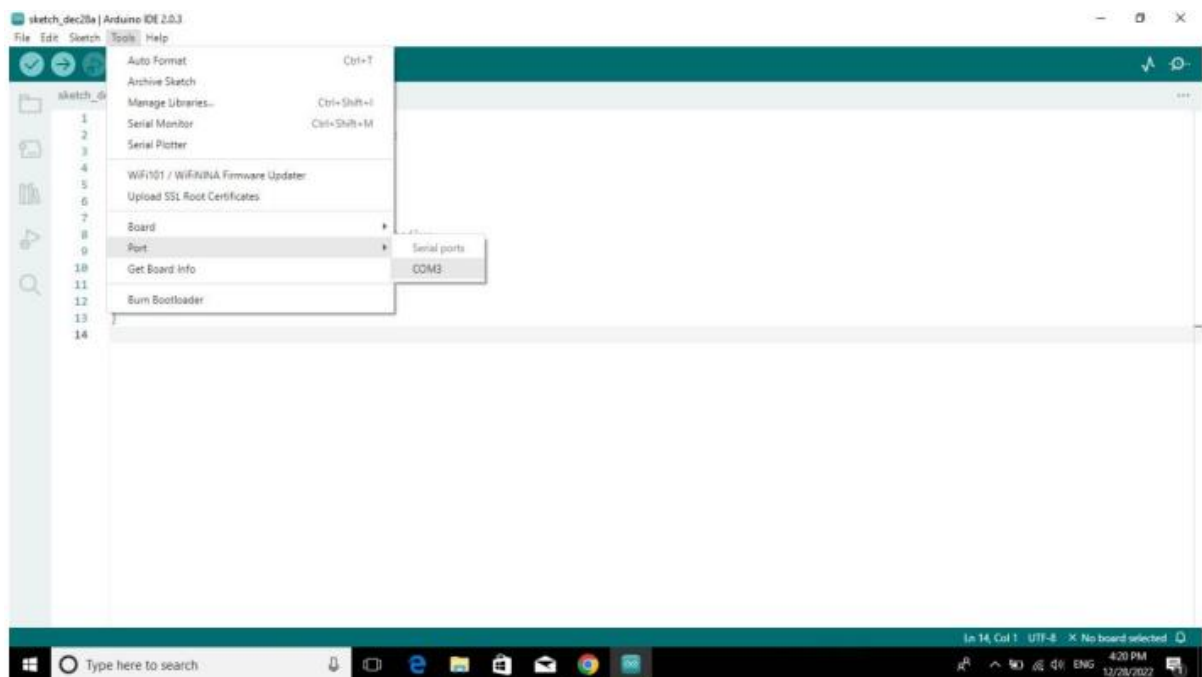
4. Connect Arduino board to the computer USB via Arduino USB cable.



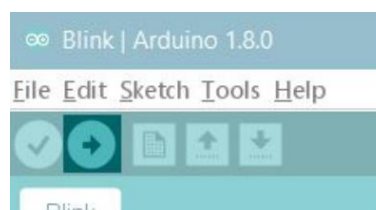
5. Select Board, Go to Tools>Boards>Arduino AVR Boards>Arduino Uno.



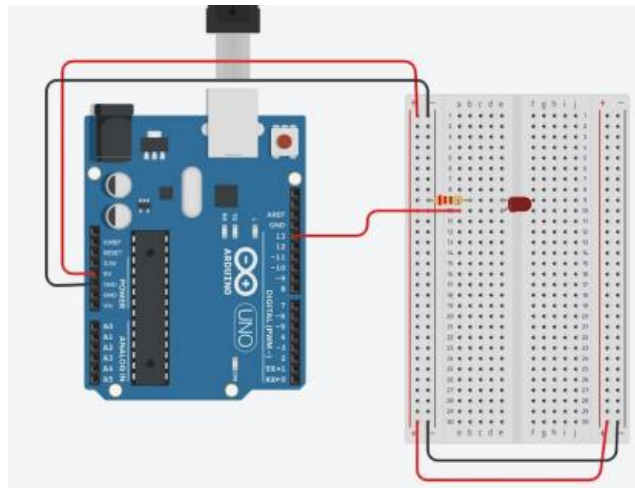
6. Select Port, Go to Tools and select the COM Port (COM Port number will vary in my case it is COM3).



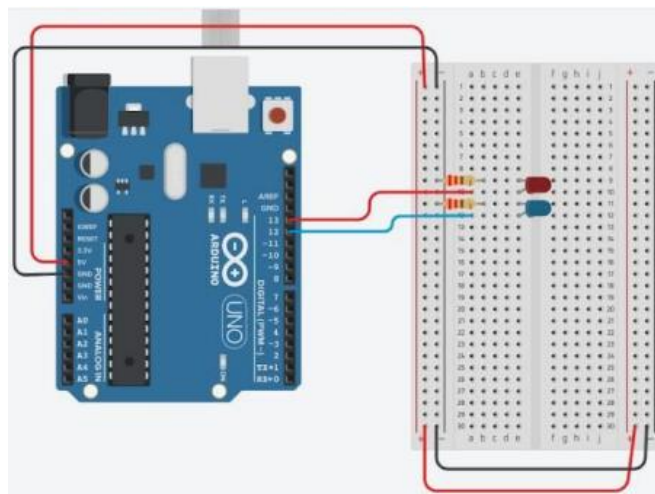
7. Upload the code by clicking the Upload button in Arduino and wait for a minute, you will get a notification when the upload is completed.



8. Check the Working, LED should blink in a 1 second interval.



9. Add the second LED to the circuit as per the circuit diagram.



10. Write the code in arduino IDE and upload the code.

Try it yourself: Light show with 8 LEDs

Here is an example circuit with 8 LEDs. Remember to keep your breadboard wiring neat and color-coded!

```

// C++ code
//
int t = 100; // declare a variable to use for delay

void setup()
{
  // setup code that only runs once
  // set pins 0-7 as outputs
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop()
{
  // code that loops forever
  // note that each LED requires its own line of code
  // turn pin 0 on, others low
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  digitalWrite(8, LOW);
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  delay(t); // Wait for t milliseconds
  // turn pin 0 on, others low
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  digitalWrite(8, LOW);
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  }

```

Reflection and Learnings:

1. What do you think is the purpose of a code? Can a controller work without a code?
2. How will you apply these skills outside class?
3. What did you or your team struggle with while doing this activity and why? How can you improve next time?
4. How did you contribute to the team and how do you think it helped? How else could you have contributed?
5. What communication challenges did you and your team face? How can your team communicate better next time?

6. Interfacing IR Sensor and Servo Motor with Arduino

Aim:

To interface IR sensor and servo motor using Arduino IDE.

Components required:

1. Arduino IDE (Online Software)
2. Arduino Uno Board
3. IR Sensor
3. LEDs
4. 220 OHM Resistor
5. Breadboard
6. Jumper Wires

Theory:

a. IR Sensor:

The IR sensor is used to sense if there is some sort of obstacle in its line of sight. It does that by sending an IR (infrared) light from the transmitter end, which isn't visible to the normal eyes, and then, if there's some sort of obstacle, then the light is reflected back and is detected by the receiver. The sending process is continuous and as soon as the obstacle is detected, we can use it to perform our required output. The Detection Range of an IR sensor is 2 to 20cm & the Detection Angle is 30° to 40°.

The **working of the IR sensor module** is very simple, it consists of two main components: the first is the IR transmitter section and the second is the IR receiver section. In the transmitter section, **IR led** is used and in the receiver section, a **photodiode** is used to receive infrared signal and after some signal processing and conditioning, you will get the output.

An IR proximity sensor works by applying a voltage to the onboard **Infrared Light Emitting Diode** which in turn emits infrared light. This light propagates through the air and hits an object, after that the light gets reflected in the photodiode sensor. If the object is close, the reflected light will be stronger, if the object is far away, the reflected light will be weaker. If you look closely toward the module. When the sensor becomes active it sends a corresponding **Low signal** through the output pin that can be sensed by an Arduino or any kind of microcontroller to execute a particular task. The one cool thing about this module is that it has two onboard LEDs built-in, one of which lights on when power is available and another one turns on when the circuit gets triggered.

An IR sensor module typically has three pins: VCC, GND, and OUT. Here's a detailed

explanation of each pin:

VCC (Power Supply Pin): This pin is used to provide power to the IR sensor module. It is connected to the positive terminal of the power supply, typically 5V.

GND (Ground Pin): This pin is connected to the ground (negative terminal) of the power supply. It completes the electrical circuit by providing a common reference point for the sensor module. Connecting this pin to the ground of the Arduino or any other microcontroller ensures a common ground, which is essential for the proper functioning of the sensor.

OUT (Output Pin): This pin provides the digital output signal from the IR sensor module. The output signal indicates whether an object is detected or not.



An IR obstacle avoider sensor contains: -

1. Transmitter LED
2. Receiver LED
3. Detection distance adjuster
4. Power LED
5. Obstacle detection LED
6. Pins (power, ground, output)

b. Servo Motor:

Servo motors are a specific kind of motors that are used to push or rotate parts of machines with high precision. They are used at angular or linear positions to provide a specific acceleration and velocity to the rotor.



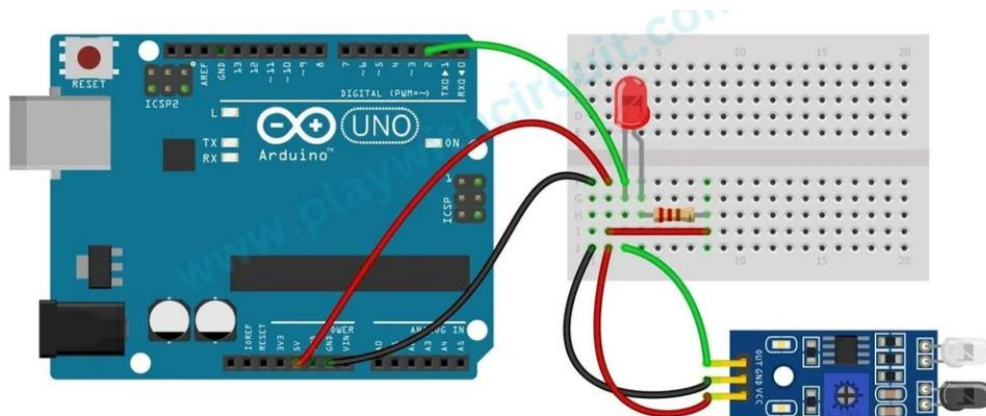
A servo motor typically has three pins, each serving a specific purpose for its operation.

Here's a breakdown of a common three-pin servo motor pinout:

1. **Power Pin (Vcc):** This pin is used to supply the voltage required to power the servo motor. For most standard hobby servos, the voltage range is typically between 4.8V to 6V. This pin is usually connected to a positive power supply.
2. **Ground Pin (GND):** This pin is the ground connection for the servo motor. It is connected to the ground of the power supply. Ensuring a common ground between the servo and the controlling device (such as a microcontroller) is essential for proper operation.
3. **Control Pin (Signal):** This pin receives the control signal from the controller (like an Arduino or another microcontroller). The control signal is a Pulse Width Modulation (PWM) signal, which determines the position of the servo motor. The duration of the pulse typically ranges from 1ms to 2ms, where 1ms usually corresponds to one extreme of the servo's range and 2ms to the other extreme. A 1.5ms pulse generally represents the neutral or centre position.

Procedure:

a) IR Sensor Interfacing with Arduino:

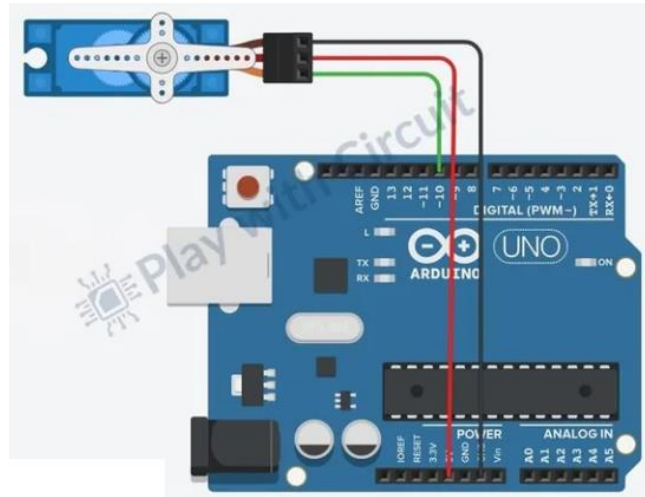


Connection diagram

- Connect the VCC pin of the IR sensor module to the 5V pin on the Arduino.
- Connect the GND pin of the IR sensor module to one of the GND pins on the Arduino.
- Connect the OUT pin of the IR sensor module to a digital input pin on the Arduino, for example, digital pin 7.

<https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino>

b) Servo Motor Interfacing with Arduino:



Connection diagram

- **Power Connection:** The red wire from the servo motor is connected to the 5V pin on the Arduino UNO. This provides the necessary power to the servo motor.
- **Ground Connection:** The black or brown wire from the servo motor is connected to one of the GND (ground) pins on the Arduino UNO. This establishes a common ground for the servo and the Arduino.
- **Signal Connection:** The yellow wire from the servo motor is connected to pin 10 on the Arduino UNO. This pin sends the PWM signal to control the servo motor's position.

<https://www.instructables.com/Interfacing-Servo-Motor-With-Arduino/>

Code for IR Sensor Interfacing with Arduino:

```
// Define the pin connections
const int irSensorPin = 7; // IR sensor output pin connected to digital pin 7
const int ledPin = 13;    // LED connected to digital pin 13 (optional)
void setup()
{
  pinMode(irSensorPin, INPUT); // Set IR sensor pin as input
  pinMode(ledPin, OUTPUT);     // Set LED pin as output (optional)
  Serial.begin(9600);          // Begin serial communication for debugging
}
void loop()
{
  int sensorValue = digitalRead(irSensorPin); // Read the value from the IR sensor
  if (sensorValue == LOW)
  {
    // Obstacle detected
    digitalWrite(ledPin, HIGH); // Turn on LED (optional)
  }
}
```

```

    Serial.println("Obstacle detected!");
}
else
{
    // No obstacle
    digitalWrite(ledPin, LOW); // Turn off LED (optional)
    Serial.println("No obstacle.");
}
delay(100); // Small delay for stability
}

```

Code for Servo Motor Interfacing with Arduino:

```

#include <Servo.h>
#define SERVO_PIN 10

Servo myservo; // create servo object to control a servo
int pos = 0; // variable to store the servo position

void setup()
{
    myservo.attach(SERVO_PIN); // attaches the servo on pin 9 to the servo object
    myservo.write(pos); // go to position zero first
    delay(2000); // wait for some time
}

void loop()
{
    delay(1000); // delay of 1 second before start
    for (pos = 0; pos <= 180; pos += 1)
    { // goes from 0 degrees to 180 degrees in steps of 1 degree
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15 ms for the servo to reach the position
    }
    delay(1000); // delay of 1 second before changing direction of rotation
    for (pos = 180; pos >= 0; pos -= 1)
    { // goes from 180 degrees to 0 degrees
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15 ms for the servo to reach the position
    }
}

```

Precautions:

1. Don't manually rotate the rotor of the servo motor beyond its limit.
2. Always use compatible voltage with the servo motor.
3. Never touch any electrical equipment wet hands.
4. Your work area must always be kept dry.
5. If you smell anything burning, immediately disconnect the power and examine your power supply to find out what went wrong.

Common Applications of IR sensor:

1. In obstacle avoider robots.
2. In aid for visually impaired.
3. Line following robots
4. Security Alarm systems
5. Object/Vistor counting applications
6. Distance measurement (2-10cm range)

Common Applications of Servo Motor:

1. In robotic arms
2. In industrial robots because of their smooth switching on and off and accurate positioning.
3. In the aerospace industry to maintain hydraulic fluid in their hydraulic systems.
4. Radio controlled toys.
5. In automobiles to maintain the speed of vehicles
6. Electronic devices such as DVDs or Blue ray Disc players to extend or replay the disc trays.

7. Blink LED using ESP32

Aim:

To perform blinking LED activity using ESP32 module.

Components required:

1. ESP32 Development Board
2. Breadboard
3. LED
4. 220 OHM Resistor
5. Jumper Wires

Theory:

The Internet of Things, often referred to as IoT, are everyday objects that connect to the internet. These connected devices can be activated using voice commands, or controlled by downloading and using an app or via a Bluetooth or Wifi connection.

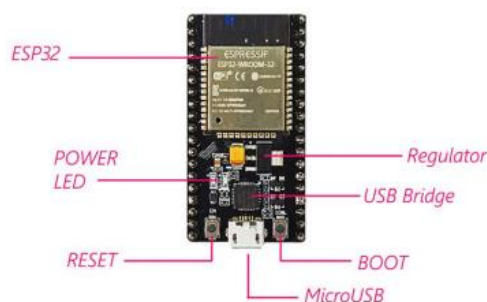
Examples of the Internet of Things include:

- Smart speakers
- Smart meters (for home electricity and water)
- Wearables such as smartwatches or fitness trackers

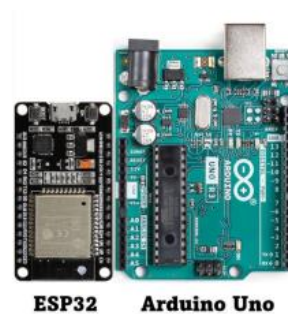
For Instance, Imagine you are going for a trip for a month and we need to monitor our plants? Do you think we can do that from our mobile app? Yes, we can build an IoT project with ESP32 microcontroller and a soil moisture sensor and this device will update live status of soil moisture level on your mobile app! We can use Arduino based microcontroller boards like ESP8266, Nodemcu, ESP32 etc for experimenting with Internet of things.

Hardware for IoT:

ESP32 is a popular low-power microcontrollers with integrated Wi-Fi and Bluetooth, making it suitable for designing and prototyping IoT solutions. This module covers various basic tutorials and projects to start with ESP32 NodeMCU based IoT projects.



Source: <https://electropeak.com>



Source: <https://linuxhint.com>

Introduction to ESP32:

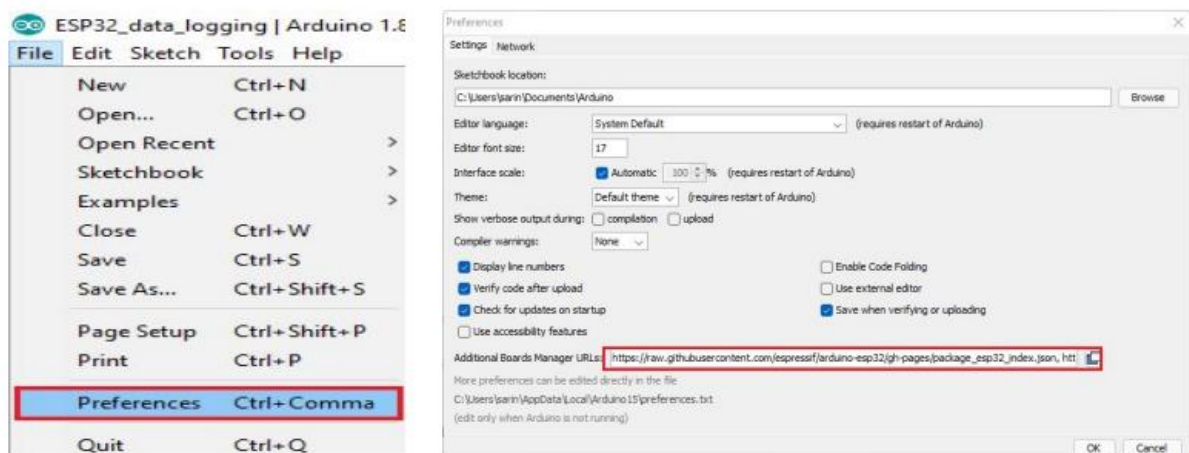
ESP32 is a microcontroller development board like arduino uno which have inbuilt Wifi and bluetooth etc. It's mainly used for IoT projects because of its inbuilt wifi connectivity feature, we can connect ESP32 to nearby wifi hotspots and can communicate with remote devices in all over the world using the internet. We can program ESP32 using the same arduino IDE which we used for programming arduino Uno before. In this module we are starting with LED blink using ESP32 board after that we will do sensor Integration and Mobile App Interfacing etc.

Procedure:

1. Download ESP32 Board Manager

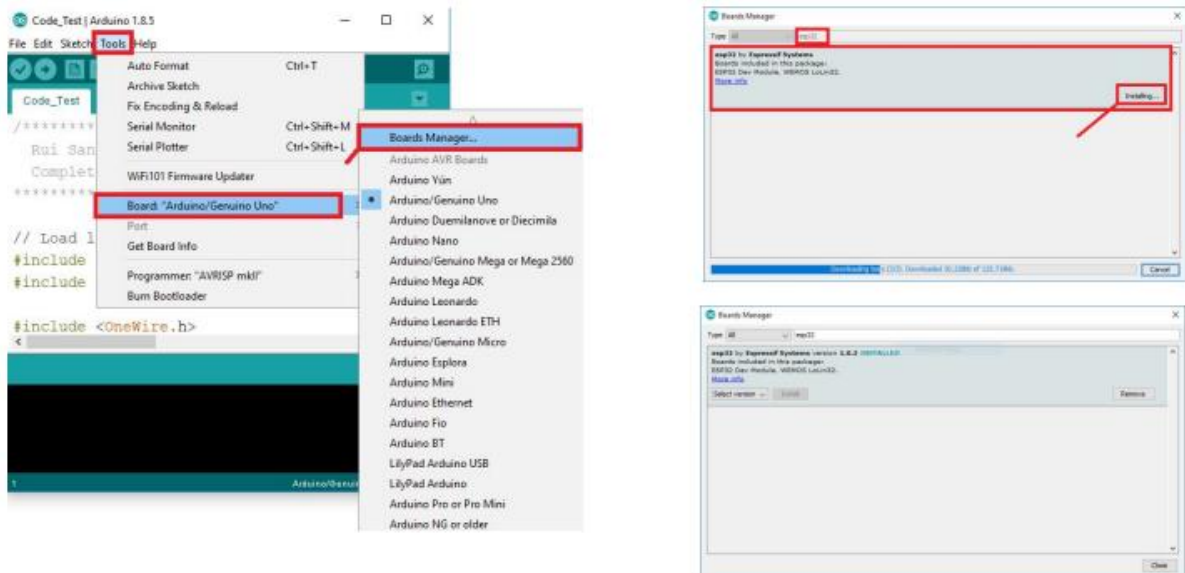
To install the ESP32 board in your Arduino IDE, follow these next instructions

- In your Arduino IDE, go to **File> Preferences**
- Enter the following into the “**Additional Board Manager URLs**” field:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- Then Click OK

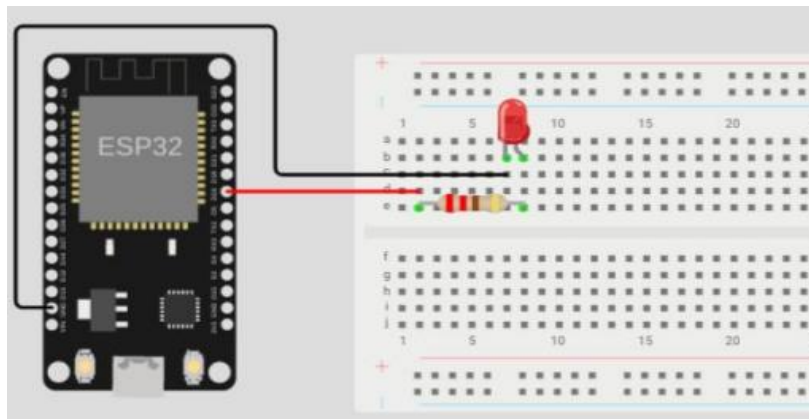


2. Install ESP32 Board Manager

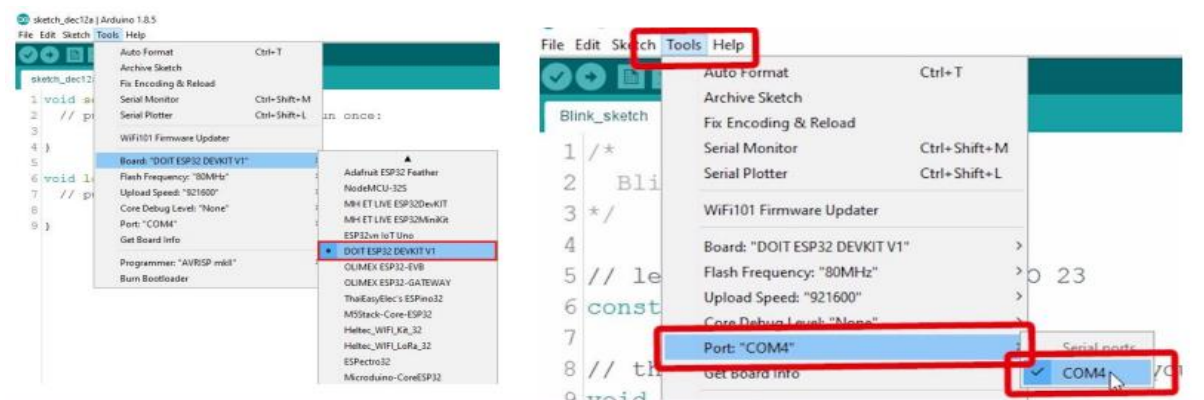
- Open the Boards Manager. Go to **Tools > Board > Boards Manager**
- Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**”:
- That’s it. It should be installed after a few seconds.



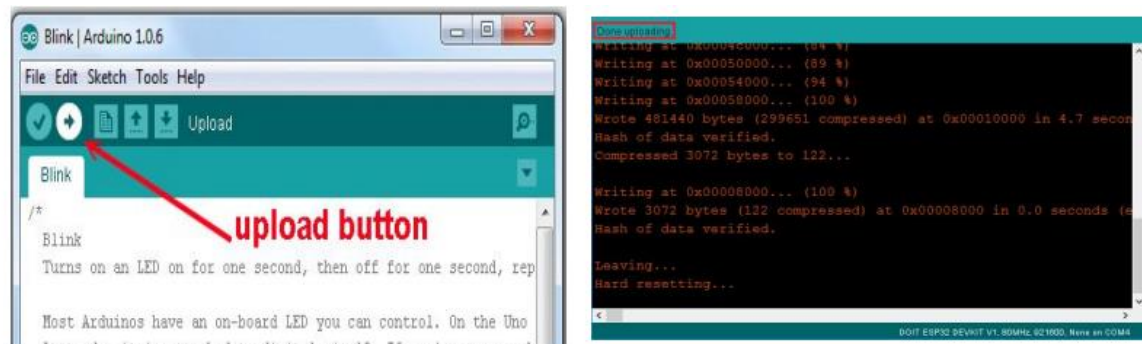
3. **Connect LED and Resistor to the ESP32:** Connect Anode of LED to Pin 18 Of ESP32 through the resistor and Cathode of LED to GND of ESP32.



4. Write the code in Arduino IDE.
5. Select ESP32 Board & Port
 - Select your Board in **Tools > Board menu**
 - Select the Port



6. Press the Upload button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board, if everything went as expected, you should see a "Done uploading." message.



Then LED on the breadboard will blink in 0.5 seconds Interval , you can adjust the delay and pin number as we did in arduino uno.

Code:

```
void setup()
{
  // initialize digital pin GPIO18 as an output.
  pinMode(18, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(18, HIGH); // turn the LED on
  delay(500); // wait for 500 milliseconds
  digitalWrite(18, LOW); // turn the LED off
  delay(500); // wait for 500 milliseconds
}
```

Precautions:

1. Never touch any electrical equipment wet hands.
2. Your work area must always be kept dry.
3. If you smell anything burning, immediately disconnect the power and examine your power supply to find out what went wrong.

Reflection and Learnings:

1. How do you think you can use IoT in a real-world scenario?
2. What do you think are pros and cons of using IoT
3. Have you ever seen an IoT device? If yes, what is that?
4. What did you or your team struggle with while doing this activity and why? How can you improve next time?
5. How did you contribute to the team and how do you think it helped? How else could

you have contributed?

Try it yourself!

Using the concepts you have learnt so far, Try creating a simple blinking LED circuit with multiple LEDs Use the space to draw or write your reflections and learnings.

8. LDR Interfacing with ESP32

Aim:

To perform LDR interfacing using ESP32 module.

Components required:

1. ESP32 Development Board
2. Breadboard
3. LED
4. 220 OHM Resistor
5. Jumper Wires
6. LDR Sensor Module
7. Software: Arduino IDE

Theory:

The main purpose of sensors is to collect data from the surrounding environment. Sensors, or 'things' of the IoT system, form the front end. These are connected directly or indirectly to IoT networks after signal conversion and processing. But all sensors are not the same and different IoT applications require different types of sensors. One of the main functions of sensors that has the Internet of Things is the monitoring of the variables of an environment or the factors that determine success in the performance of an activity, such as the control of the operation of a machine in a company. Sensors are the important part of Internet of Things, we need to use sensors for different applications such as Temperature Monitor, Light Monitor, Human Presence Detection, Water Level detection etc. Today we are going to use an LDR (Light Sensor) with ESP32.



LDR Module have 4 Pins, they are

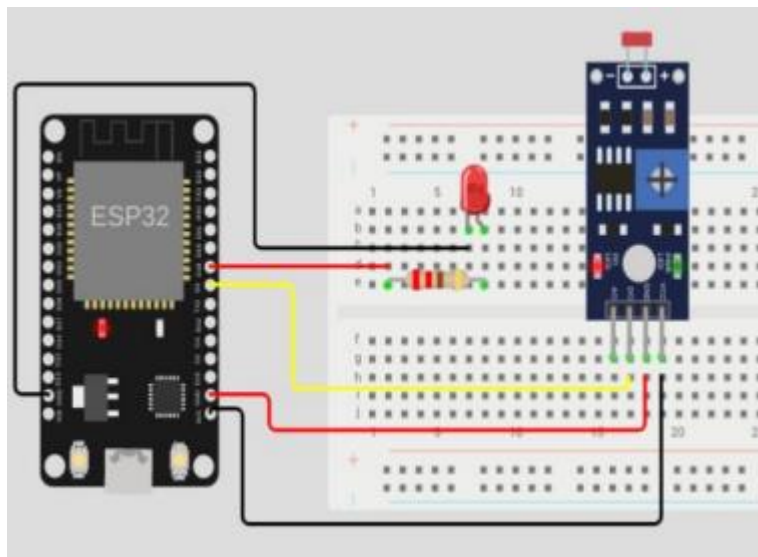
- 1)VCC - 5V
- 2)GND - GND
- 3)D0 - Digital Output
- 4)A0-Analog Output

We are going to use VCC,GND and Digital Output only now

The ESP32 is an advanced IoT microcontroller board possessing WiFi and Bluetooth Low Energy capabilities, as well as limited compatibility with the Arduino Core. ESP32 is a low-powered, low-cost microcontroller (MCU) board, with both Wifi and Bluetooth built in, and is based on a dual-core processor mechanism. Due to its low-cost, multiple-component support design, built-in Wifi and Bluetooth and ready compatibility with Arduino and many other boards, ESP32 is becoming a fast popular choice for IoT projects. Engineered for mobile devices, wearable electronics and IoT applications, ESP32 achieves ultra-low power consumption with a combination of several types of proprietary software. ESP32 also includes state-of-the-art features, such as fine-grained clock gating, various power modes and dynamic power scaling.

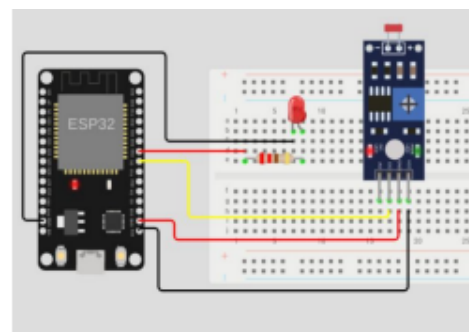
Procedure:

1. Place LDR, Resistor and LED as per the circuit diagram.



2. Connect the circuit.

VCC of LDR Module - 3.3V of ESP32
GND of LDR Module - GND of ESP32
D0 of LDR Module - D5 of ESP32
Negative of LED - GND of ESP32
Positive of LED - 220 OHM Resistor-D18 of ESP32



3. Upload the following code to the ESP32 as you have done in the previous session.

Code:

```
void setup()
{
  // initialize digital pin GPIO18 as an output.
  pinMode(18, OUTPUT);
  pinMode(5, INPUT);
} // the loop function runs over and over again forever
void loop()
{
  if(digitalRead(5)==HIGH) //Check LDR module output
  {
    digitalWrite(18, HIGH);
    // turn the LED on }
  else
  {
    digitalWrite(18, LOW); // turn the LED on
  }
}
```

Precautions:

1. Never touch any electrical equipment wet hands.
2. Your work area must always be kept dry.
3. If you smell anything burning, immediately disconnect the power and examine your power supply to find out what went wrong.

Conclusion:

Without light the LDR Module output(D0) will be LOW and in bright light D0 will be HIGH We are reading output of LDR Module using digitalRead and checking the output using an if condition If reading from LDR module is HIGH it will turn on the LED and if the reading in LDR module is LOW then the LED will be turned off.

Reflection and Learnings:

1. What is a Sensor?
2. How sensors plays an important role in IoT currently?
3. How can you identify on what terms the LED stays on and and goes off?
4. What do you think is the role of ESP32 in an IoT circuit?
5. What do you think is the difference between Artificial Intelligence and IoT?

Try it yourself!

Using the concepts you have learnt so far, Try creating a Street light system as shown in the image

9. Control an LED using Mobile App

Aim:

To control an LED using Mobile App.

Components required:

1. ESP32 Development Board
2. Breadboard
3. LED
4. 220 OHM Resistor
5. Jumper Wires

Theory:

We already learned to connect sensors and LEDs with the ESP32 board now we need to experience real IoT use cases, today we are going to control an LED using mobile app from remote location. We need a server for our project but now we are not setting up our own server instead of that we are going to use an IoT platform called Blynk.

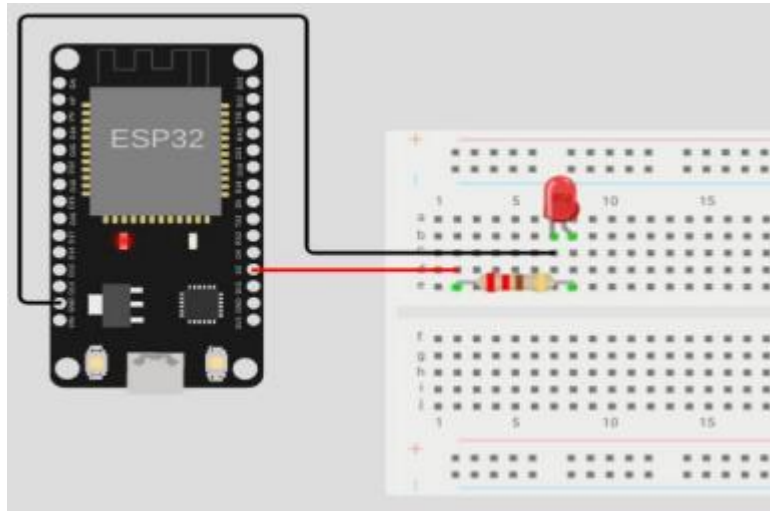
Blynk Mobile App:

Blynk is a highly accessible smartphone-based application available for both Android and iOS operating systems. It provides an interactive dashboard where the user can build their own IoT projects by placing components. This will as a result create a mobile app through which the user will be able to control microcontrollers connected through WIFI. You can control LEDs, relays, electric motors and many more. No internal programming is required to build your project in Blynk. You only need to drag and place your electronic components and it is as easy as that. Thus, building your IoT projects through Blynk is extremely easy and requires very little effort.



Procedure:

1. Place the LED and Resistor as per the circuit diagram.
2. Connect the cathode of LED to GND of ESP 32 and Anode of LED to D2 Pin of ESP32 through a 220 OHM Resistor.



3. Go to <https://blynk.cloud/dashboard/register> on your browser and create an account.

Sign Up

Welcome! Fill in your email address and we will send an account activation link.

EMAIL

☐ I agree to [Terms and Conditions](#) and accept [Privacy Policy](#)

[Sign Up](#)

[Back to Login](#)

4. After creating and account a window will open, Click on create new template.

Start by creating your first template

Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.

[+ New Template](#)

5. Click on New Template. Select a Template name, Select hardware as esp32, Select connection type as WiFi. Click on done.

Create New Template

NAME
myIoT

HARDWARE
ESP32

CONNECTION TYPE
WiFi

DESCRIPTION
This is my template

19 / 128

Cancel Done

After that the BLYNK_TEMPLATE_ID, BLYNK_DEVICE_NAME and AUTH_TOKEN will appear on screen. These will required while do programming in esp32.

TEMPLATE NAME
myIoT

HARDWARE
ESP32

CONNECTION TYPE
WiFi

DESCRIPTION
This is my template

19 / 128

TEMPLATE ID
TMPLH2P0yZ

MANUFACTURER
My organization: 8348EJ

OFFLINE IGNORE PERIOD
00 hrs 00 mins 00 secs

HOTSPOT PREFIX
Hotspot Prefix

myIoT

TEMPLATE IMAGE (OPTIONAL)
Add image
Upload from computer or drag-n-drop .png or .jpg, minimum width 500px

FIRMWARE CONFIGURATION

```

define BLYNK_TEMPLATE_ID "TMPLH2P0yZ"
define BLYNK_DEVICE_NAME "myIoT"

```

Template ID and Device Name should be included at the top of your main firmware

Cancel Save

Info Metadata **Datstreams** Events Automations Web Dashboard Mobile Dashboard

6. Create a Data stream.

Datstreams

Datstreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.

[+ New Datstream](#)

Select DataStream tab
Click on New DataStream and select Virtual pin.
Enter a Name. Select the virtual pin V1, & select datatype be Integer. Click on Create.

Virtual Pin Datstream

NAME
v1

ALIAS
v1

PIN
V1

DATA TYPE
Integer

UNITS
None

MIN
0

MAX
1

DEFAULT VALUE
0

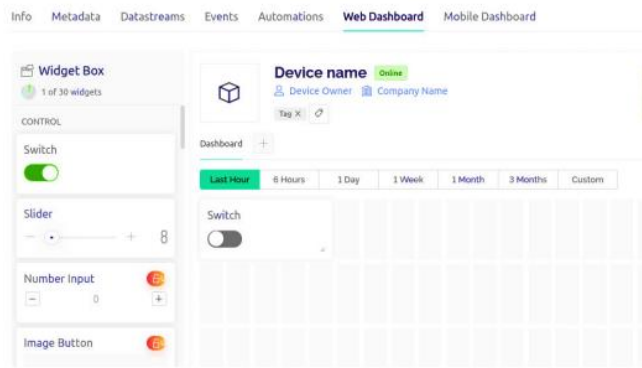
ADVANCED SETTINGS

Cancel Create

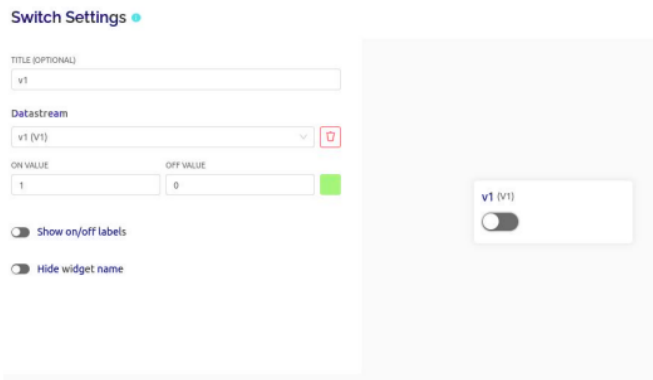
7. Setting up Blynk web Dashboard.

Step 7: Setting up Blynk web Dashboard.

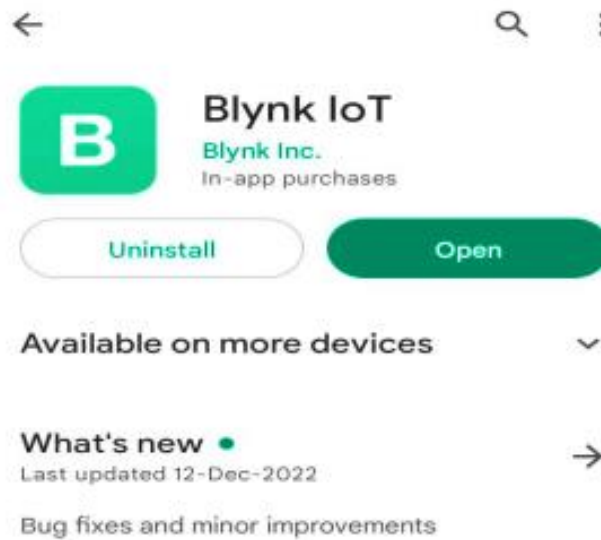
Go to Web Dashboard Drag and Drop Switch Widget



Go to Settings of Switch Widget and Select Data Stream

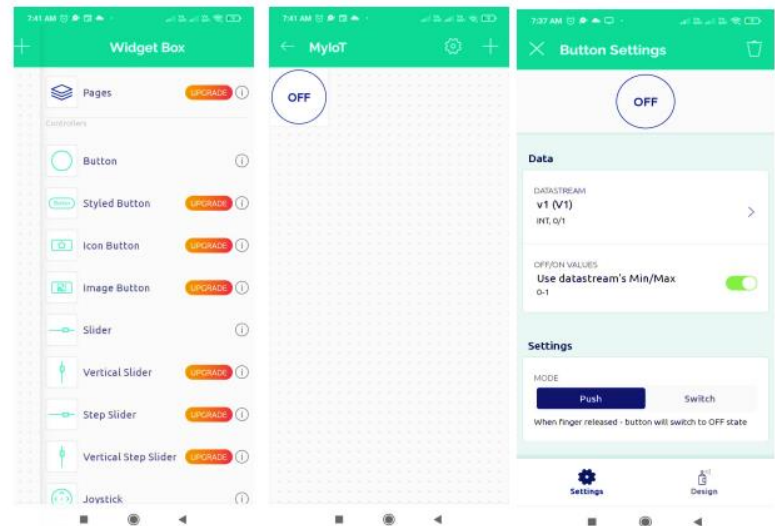


8. Install Blynk IoT app from Play Store.



Download the blynk lot app from google play store.
Log in with the same account that you have created before.
Select developer option.
Select the template that you have created before.
Select the Hamburger button on the left side of your screen.
The widget box will open.

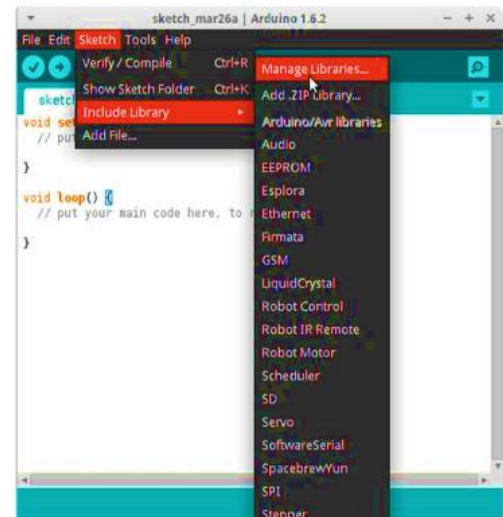
Select Button and Tap on Datastream and Select Datastream v1



9. Install Blynk Library in Arduino IDE

Step 9: Install Blynk Library in Arduino IDE

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. Search for Blynk library and in the version selection choose the latest version to date

Finally click on Install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.

10.Code ESP 32



Note : We need to Copy paste TEMPLATE ID & DEVICE NAME from Blynk Dashboard

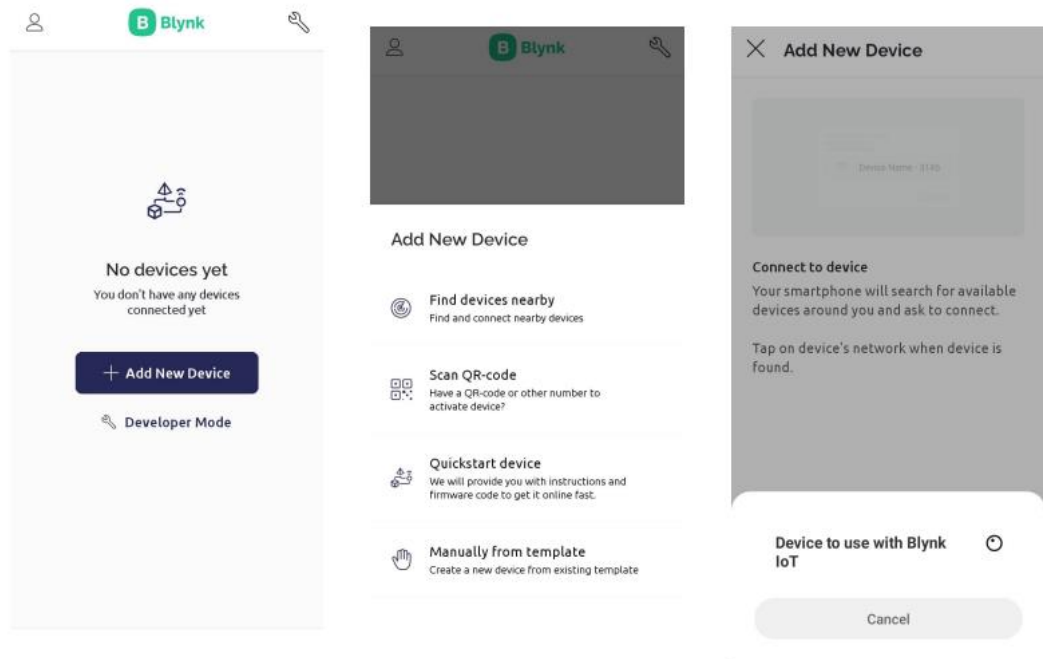
FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPL5Qk88T_2"  
#define BLYNK_DEVICE_NAME "myIoT"
```

Template ID and Device Name should be included at the top of your main firmware

11.Upload the Code to ESP32 Board.

12.Open Blynk IoT Mobile App and Add your ESP32 Device.



App will scan your nearby wireless devices and it will detect your ESP32 , you need to add that.

13.Testing

Step 13: Testing

We can Turn on and off LED using the button we have already created in mobile app

This is the simplest IoT project we can add more components and features in this project using the same Blynk App

Code:

```
#define BLYNK_TEMPLATE_ID
"TMPLxxxxxx"
#define BLYNK_DEVICE_NAME
"Device"
#define BLYNK_FIRMWARE_VERSION
"0.1.0"
#define BLYNK_PRINT Serial
// #define BLYNK_DEBUG
#define APP_DEBUG
#include "BlynkEdgent.h"
BLYNK_WRITE(V1)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
  digitalWrite(2,pinValue); // LED Connected to pin2 (D2) of ESP32
}
void setup()
{
  pinMode(2, OUTPUT);
  Serial.begin(115200);
  delay(100);
  BlynkEdgent.begin();
}
void loop()
{
  BlynkEdgent.run();
}
```

Precautions:

1. Never touch any electrical equipment wet hands.
2. Your work area must always be kept dry.
3. If you smell anything burning, immediately disconnect the power and examine your power supply to find out what went wrong.

Reflection and Learnings:

1. What is the purpose of using a Blynk App in this activity?
2. What do you think is the difference between a smart bulb and a normal bulb?
3. What did you or your team struggle with while doing this activity and why? How can

you improve next time?

4. How did you contribute to the team and how do you think it helped? How else could you have contributed?

5. What communication challenges did you and your team face? How can your team communicate better next time? Use the space to draw or write your reflections and learnings

10. Design and 3D print a Walking Robot

Aim:

To design a walking robot using 3D printing.

Components required:

1. Toy motor
2. Battery pack
3. Screws
4. Soldering kit
5. 3D printer
6. Tinkercad software

Theory:

Robotics is the branch of science focused on learning about and creating robots or machines that can do work. Engineers, mathematicians, and computer scientists work together to build robots that move, have power and sensors, and complete tasks.



Source:CHEP youtube channel

We are going to design and fabricate a movable robot using Tinkercad, we are using the following additional materials to build this robot.



Toy Motor



Battery Pack



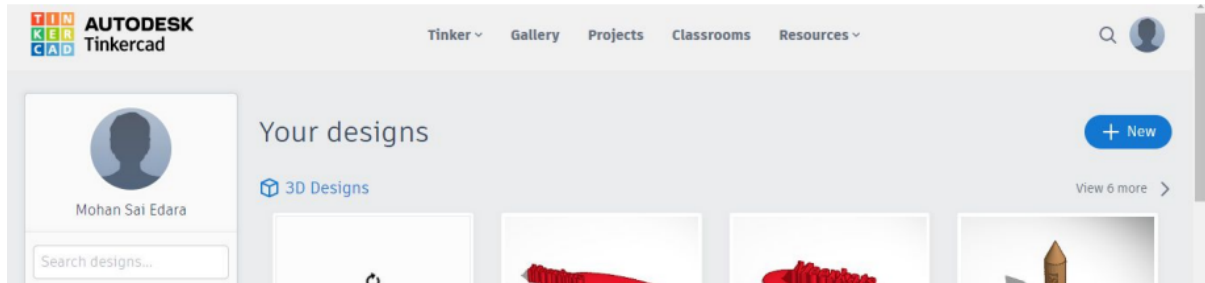
Screws



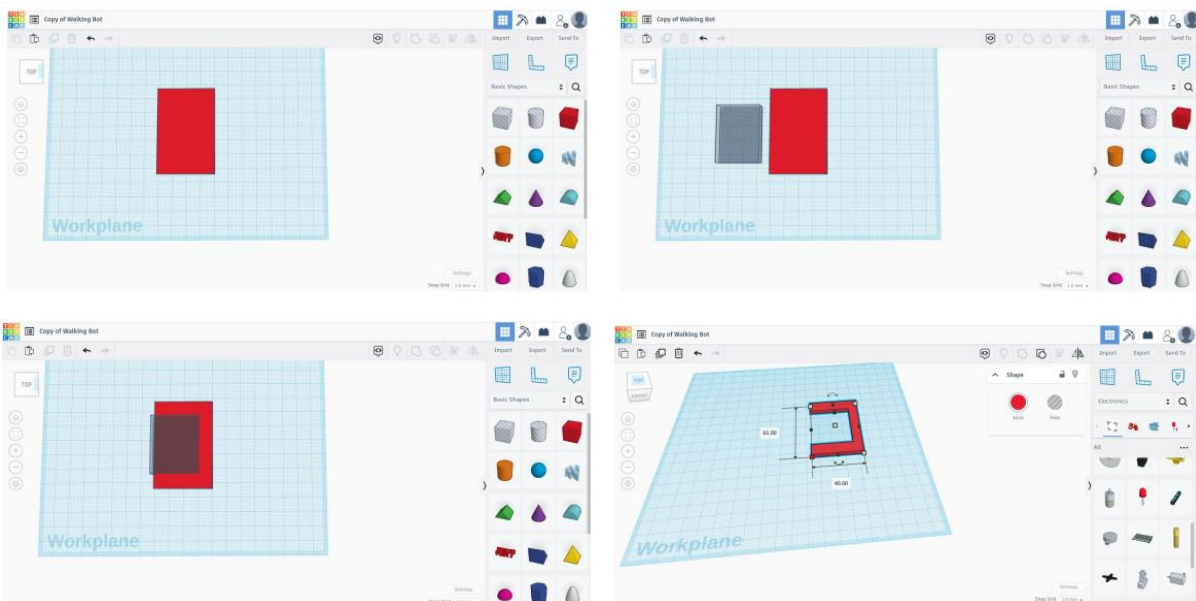
Soldering Kit

Procedure:

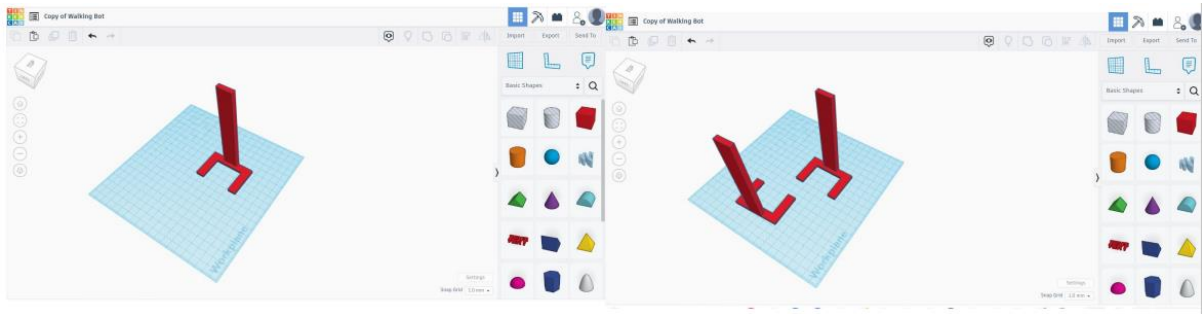
1. Open a browser and type tinkercad in the url and press enter. Click on the first result www.tinkercad.com and you will be redirected to a tinkercad website. Click on sign in and use your google account or autodesk account to sign in to the tinkercad website.
2. After logging in, click on “+ New” to start a new design.



3. First we need to create the base of the Robot leg
 - Insert a solid box and change its dimensions to 60MMX40MMX3MM
 - insert another hollow box with 40MMX30MMX5MM
 - Group this two shapes and will get base of leg.

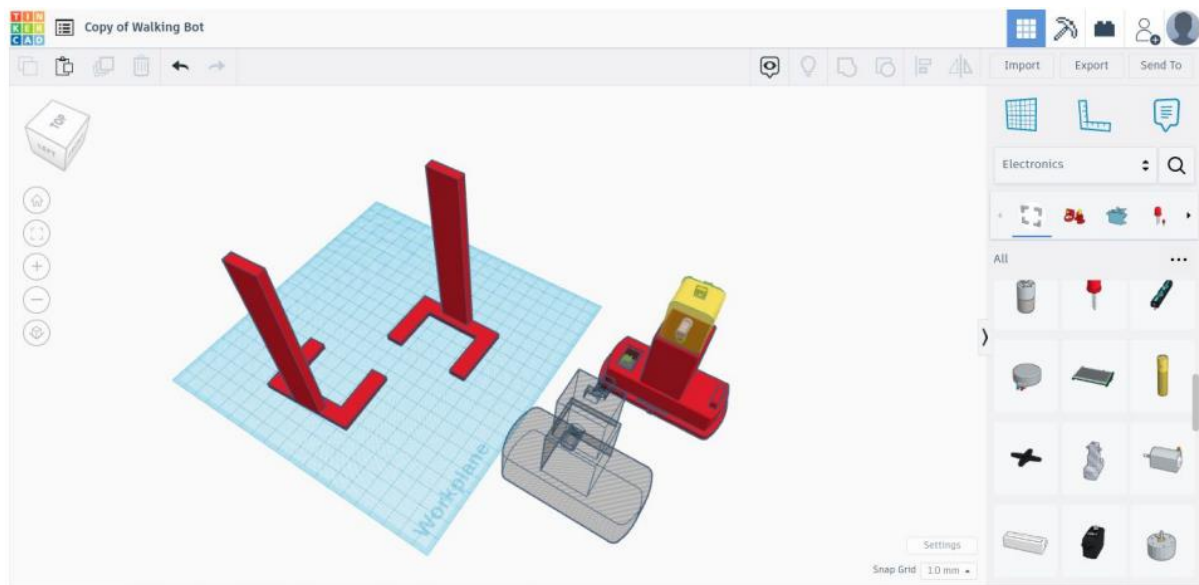


4. ➤ Insert a solid box, change the dimensions 10MMX5MMX120MM and place it as shown in the image.
 - Group the two parts of leg.
 - Copy and paste the leg, we need two legs.



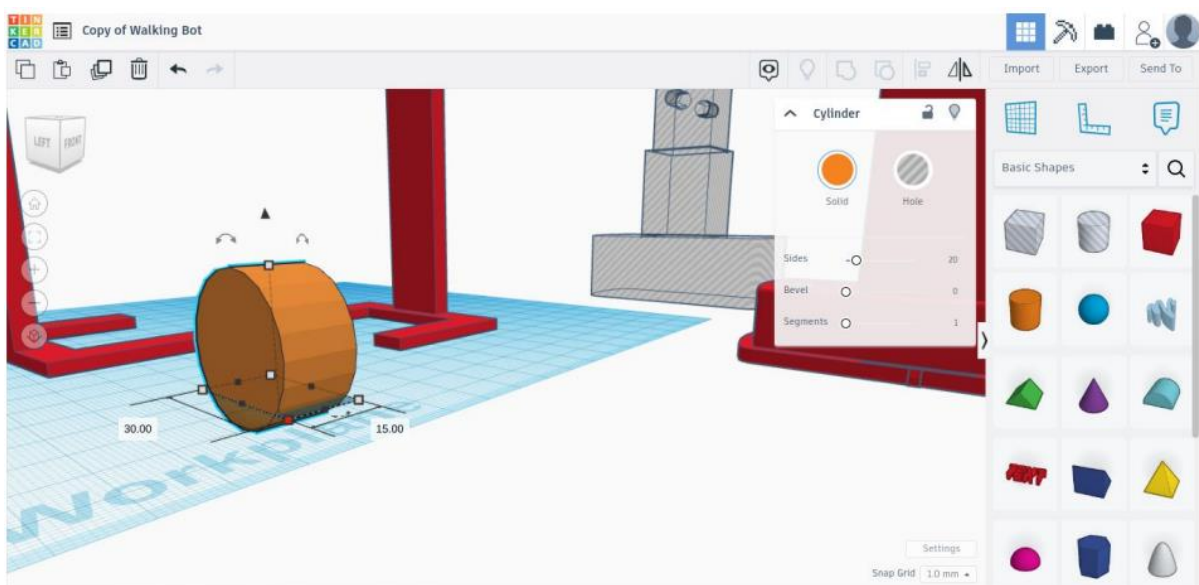
5. Insert the motor part from the tinkercad assembly library for getting accurate measurements, we will need both the 3D Model and the Hollow cut part.

- Go to Toolbar (Right Side) -> Select Electronics -> Scroll Down and Select Spin
- Similarly insert Spin Cutout



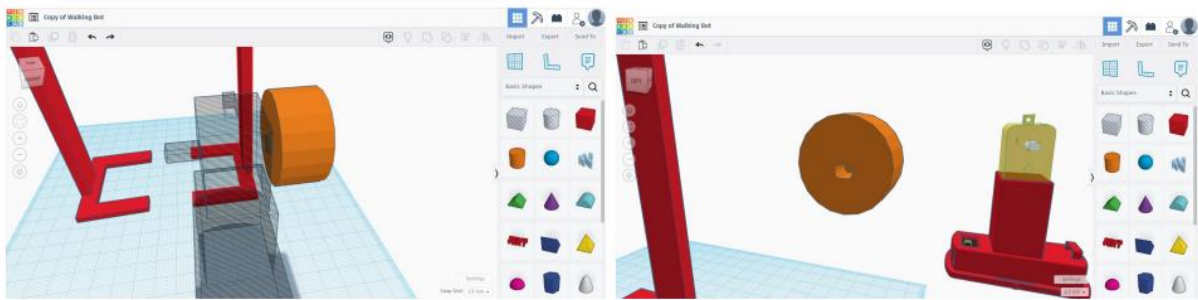
6. Design an attachment for connecting the robot leg and motor.

- Insert a cylinder and change dimensions (Diameter 30MM, Height 15 MM)

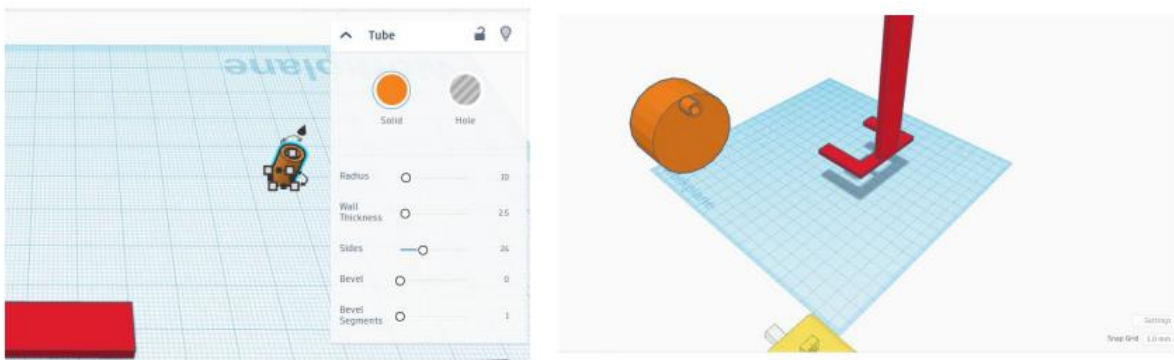


7. Design an attachment for connecting the robot leg and motor.

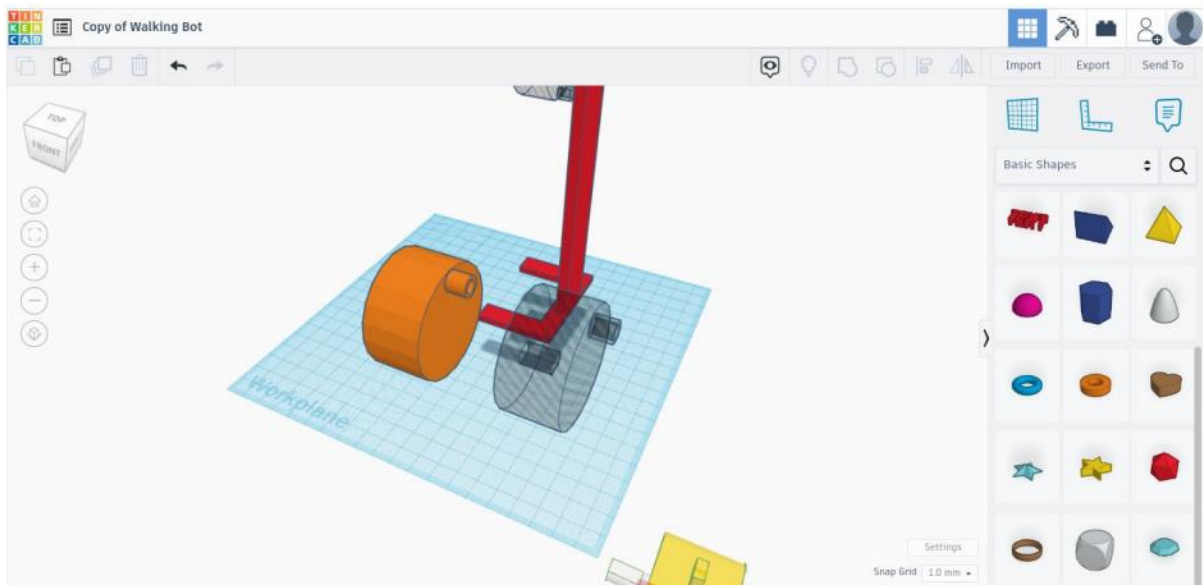
- First we need to take the hollow cut out model of motor for making holes.
- Group motor hollow part and cylinder for getting a hole for holding motor.



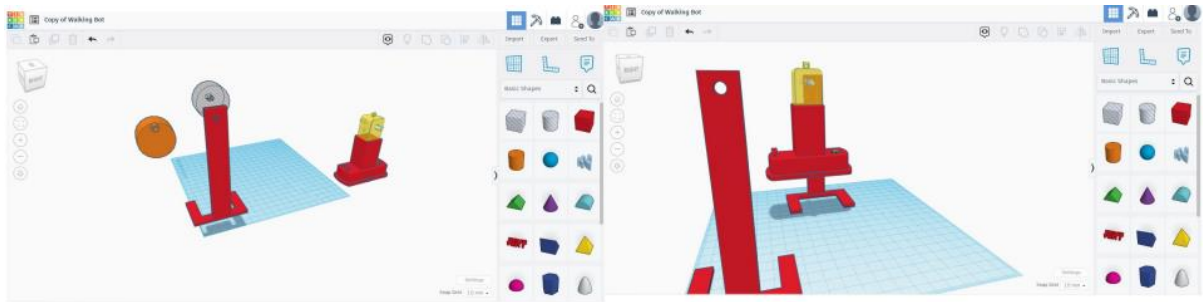
8. Add a small cylinder to hold the attachment and leg.
 - Insert a tube with a 5MM radius and 5MM length.
 - Attach the tube to the cylinder already created and group them.



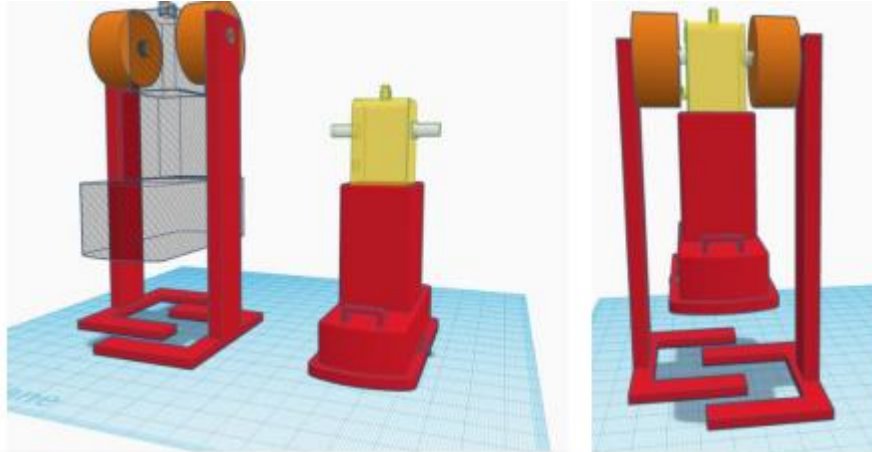
9. Make a copy of the newly created attachment and make it hollow (we need a hollow model to make a hole in the robot leg).



10. Group robot leg & the hollow attachment to make a hole as shown in the image given below.



11. Make two copies of leg & cylinder attachment and assemble the robot.



12. Export all STL files and 3D print it.

13. Assemble the robot using 3d printed parts and motors using a screw.



Image Source : CHEP Youtube Channel

14. Take battery box and solder positive and negative terminals of battery to toy motor (Optionally you can add a switch otherwise connect battery terminals directly to battery).



Image Source : CHEP youtube channel

Precautions:

1. Put your 3D printer in an area which isn't easily accessible
2. Wear gloves when handling your 3D printer.
3. Keep a mental note in your head that your 3D printer gets very hot.
4. Only reach for your printer when you are certain it's off

Reflection and Learnings:

1. What do you think is the difference between a Robot and Human ?
2. What will happen when you use the "Hole" option in TinkerCad?
3. Imagine you got a chance to 3D print an item that solves a problem in your locality, what would you 3D print and how will it solve the problem?
4. What did you or your team struggle with while doing this activity and why? How can you improve next time?

11. Design and 3D Print a Rocket

Aim:

To design a rocket using 3D printing.

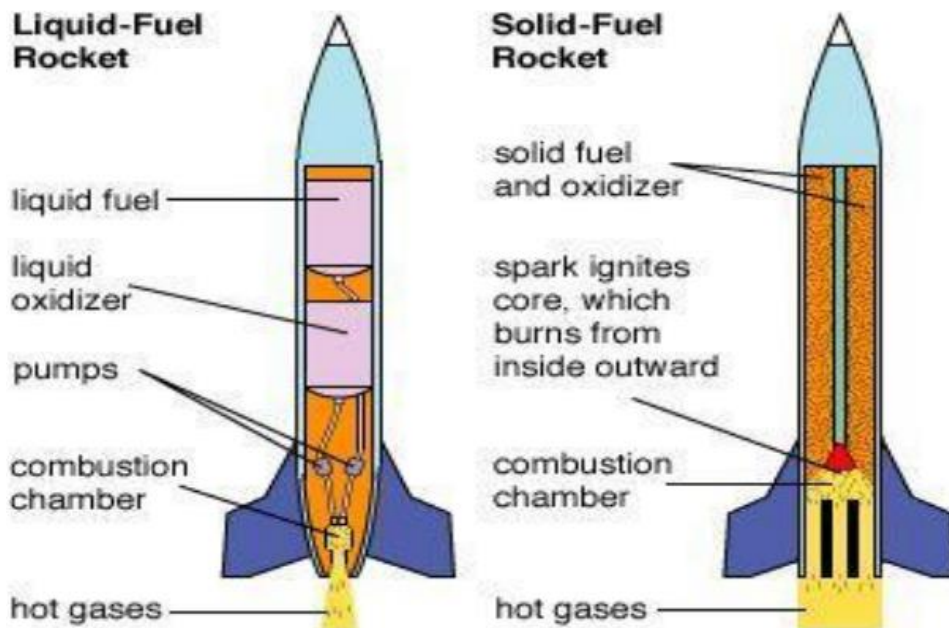
Components required:

1. Laptop with Internet Connection
2. 3D Printer
3. Tinkercad software

Theory:

Rockets are devices that produce the force, or push, needed to move an object forward. Rockets are used to launch spacecraft. They are also used to shoot missiles and fireworks.

How Rockets Work? Rockets carry fuel that is burned inside a chamber. The fuel burns when it is mixed with oxygen gas and ignited, or set on fire. As the fuel burns, it gives off hot gas that shoots out from an opening at the back of the chamber. The force of the gas moving backward pushes the rocket forward. This action is called jet propulsion. The engines of a jet airplane also use jet propulsion. But unlike a jet engine, rockets carry their own oxygen supply. This makes rockets valuable in outer space, where there is no oxygen. Rocket fuel can be liquid or solid. Two solid-fuel booster rockets launched the shuttle into space. Three liquid-fuel rocket engines allowed the shuttle to move in and out of orbit.



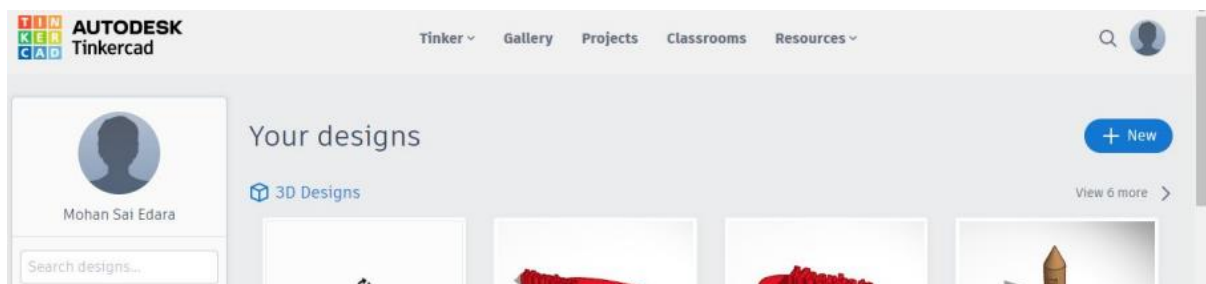
Source: kids.britannica.com

Precautions:

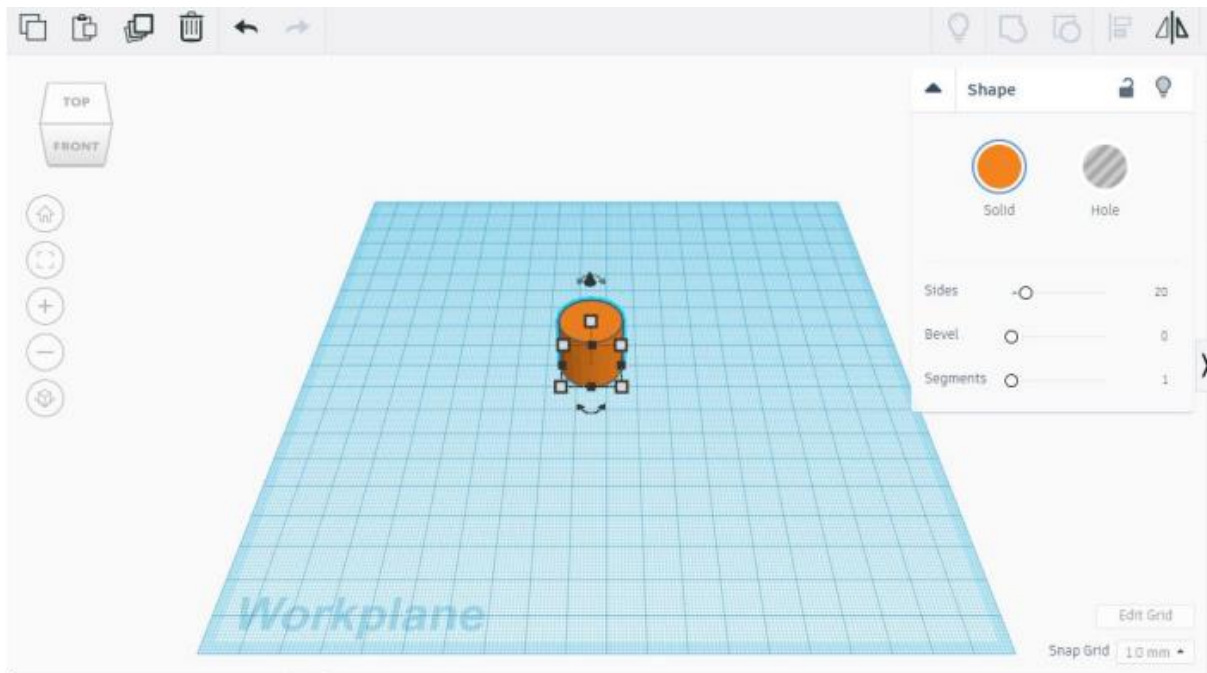
1. Put your 3D printer in an area which isn't easily accessible
2. Wear gloves when handling your 3D printer.
3. Keep a mental note in your head that your 3D printer gets very hot.
4. Only reach for your printer when you are certain it's off

Procedure:

1. Open a browser and type tinkercad in the url and press enter. Click on the first result www.tinkercad.com and you will be redirected to a tinkercad website. Click on sign in and use your google account or autodesk account to sign in to the tinkercad website.
2. After logging in, click on "+ New" to start a new design.

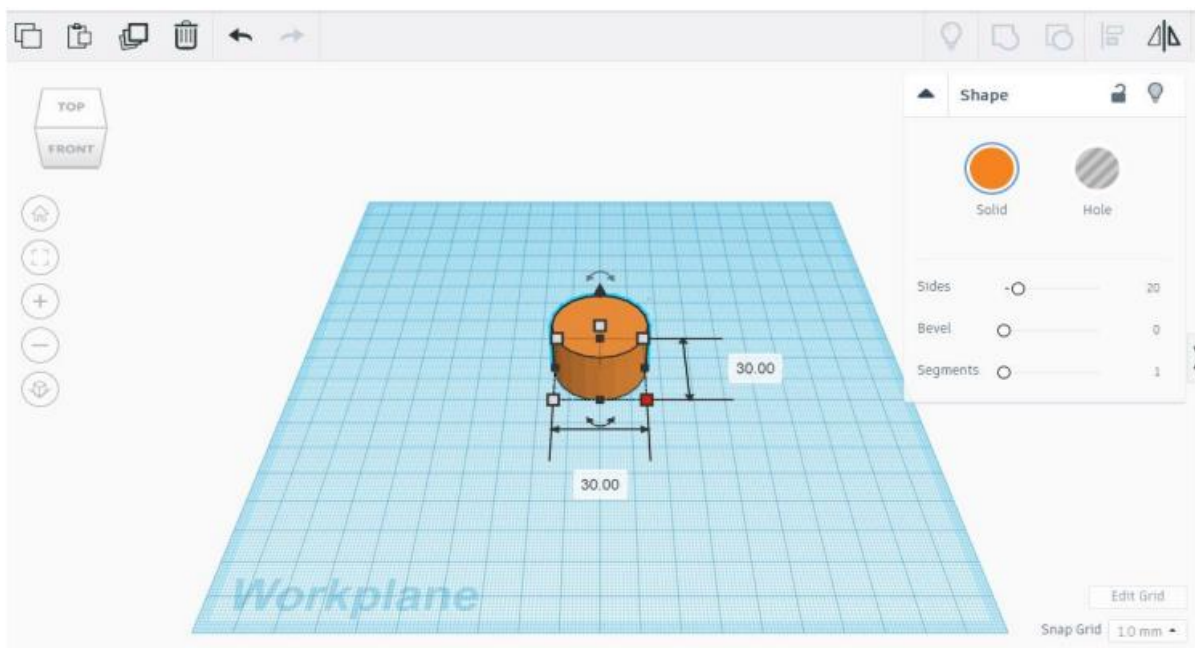


3. Drag and drop a Cylinder from the basic shapes menu on to the work plane.

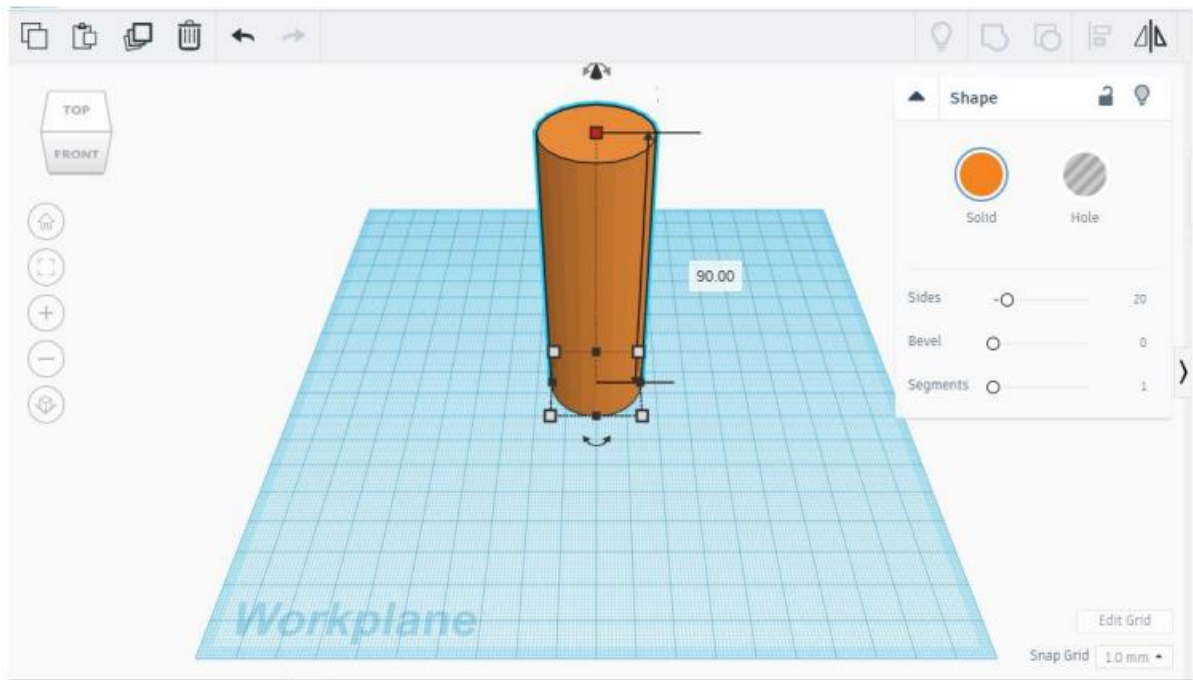


Source: tinkercad.com

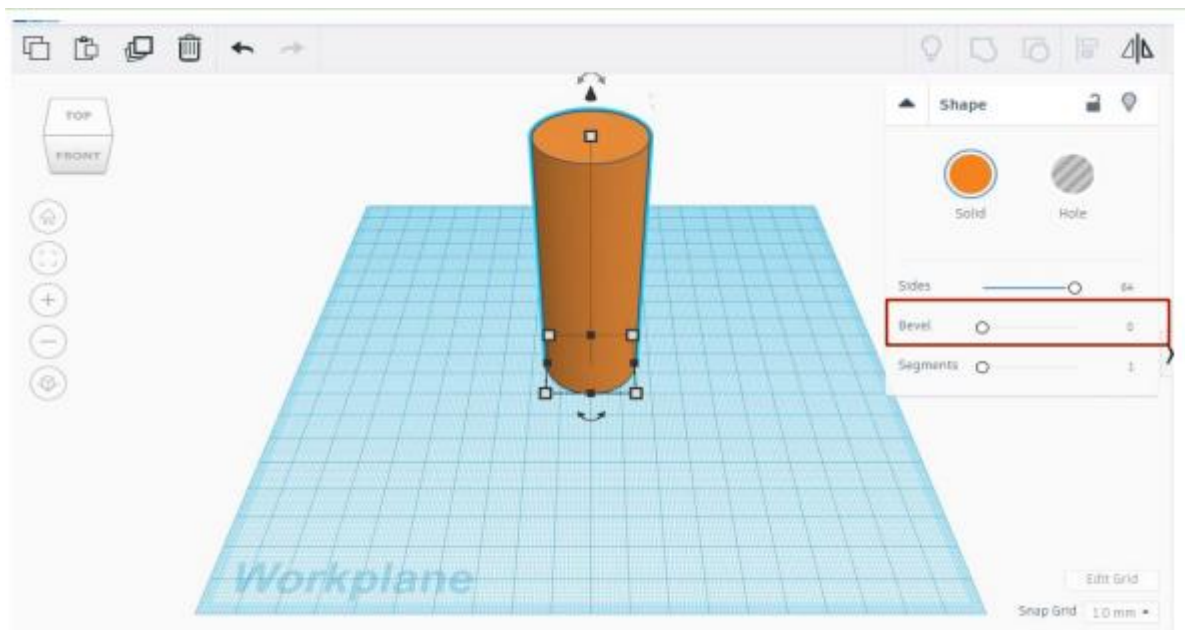
4. Now let's increase the size of the cylinder to 30 mm. Hold the shift key and drag the corner of the cylinder to 30 mm.



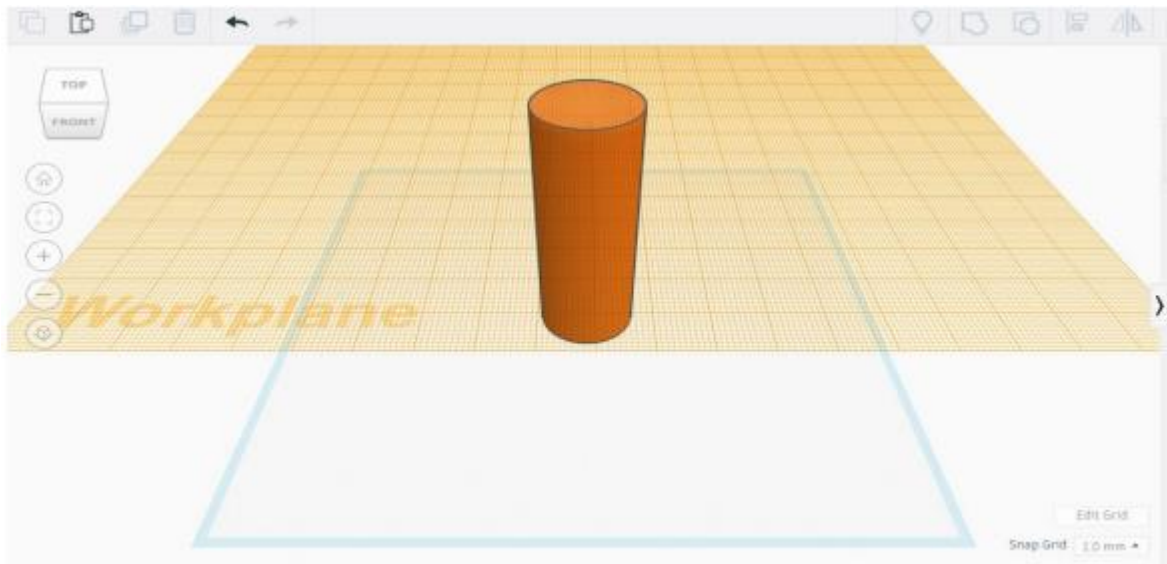
5. Let's increase the height of the Cylinder to 90mm. Click on the middle dot on the top and enter 90.



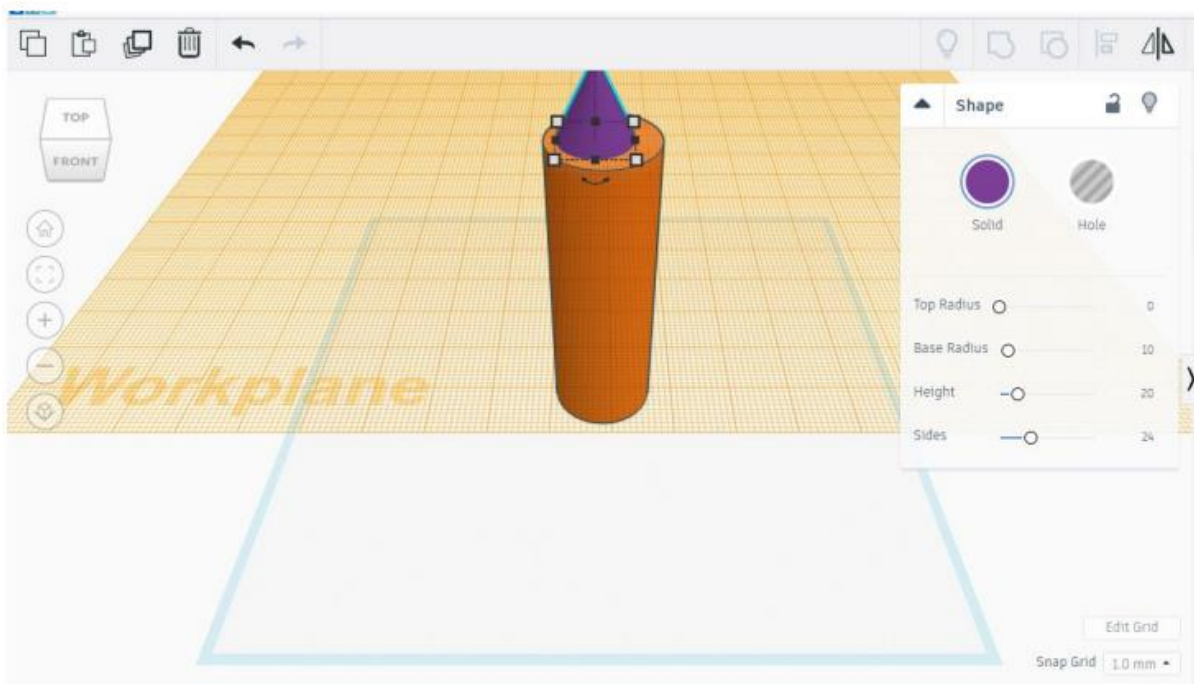
6. Now let's make the surface of the rocket body smooth. In the shape toolbar increase the sides slide bar to 64.



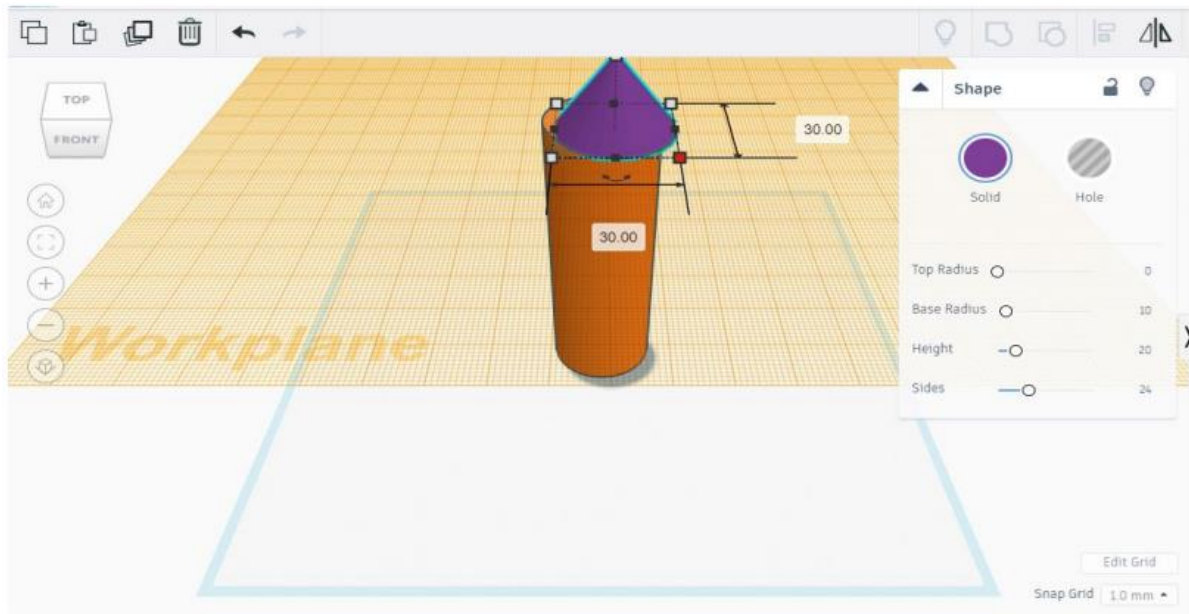
7. Drag and drop a work plane and place it on the cylinder as shown in the image.



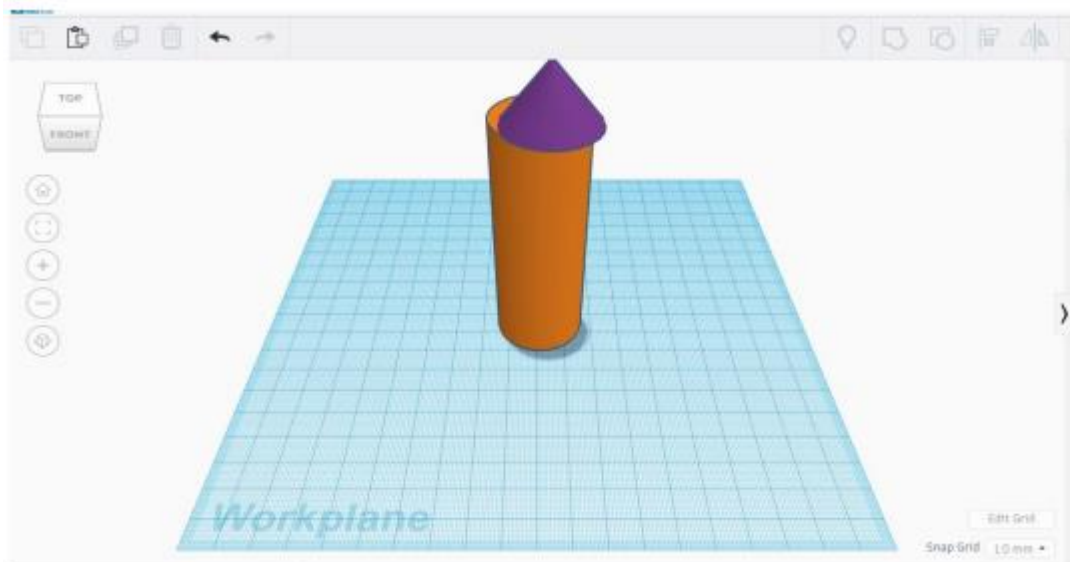
8. Now let's make the rocket head. Drag and drop a cone from the basic shapes toolbar on to the cylinder top.



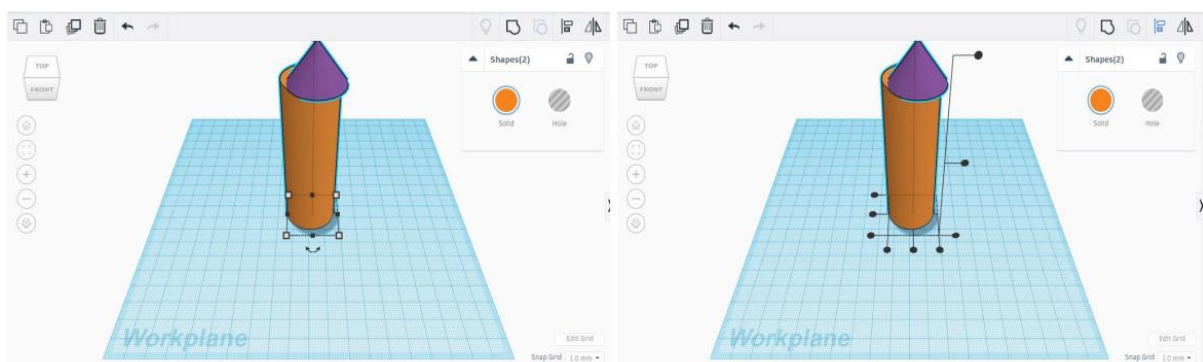
9. Click on the corner of the shape and enter 30 on both sides to change the dimensions to 30 mm.

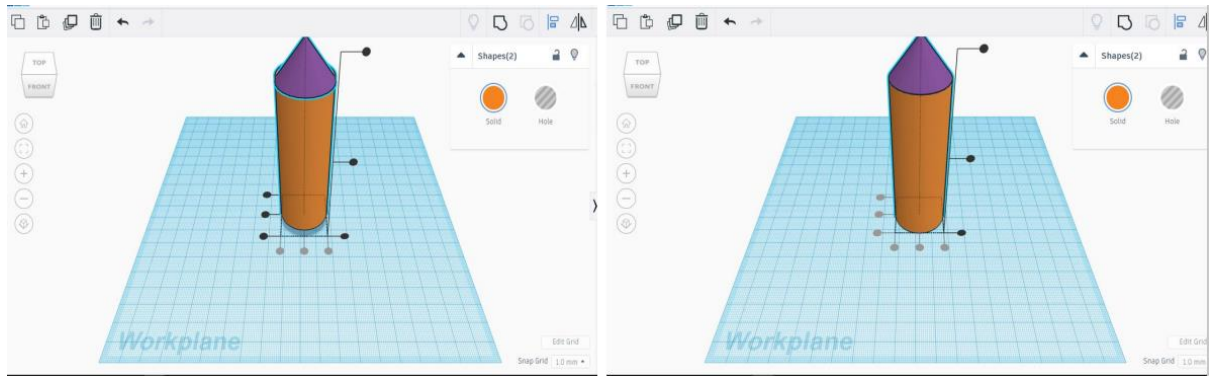


10. Drag and drop a work plane on the basic work plane. This will remove the existing workplane we have added earlier.

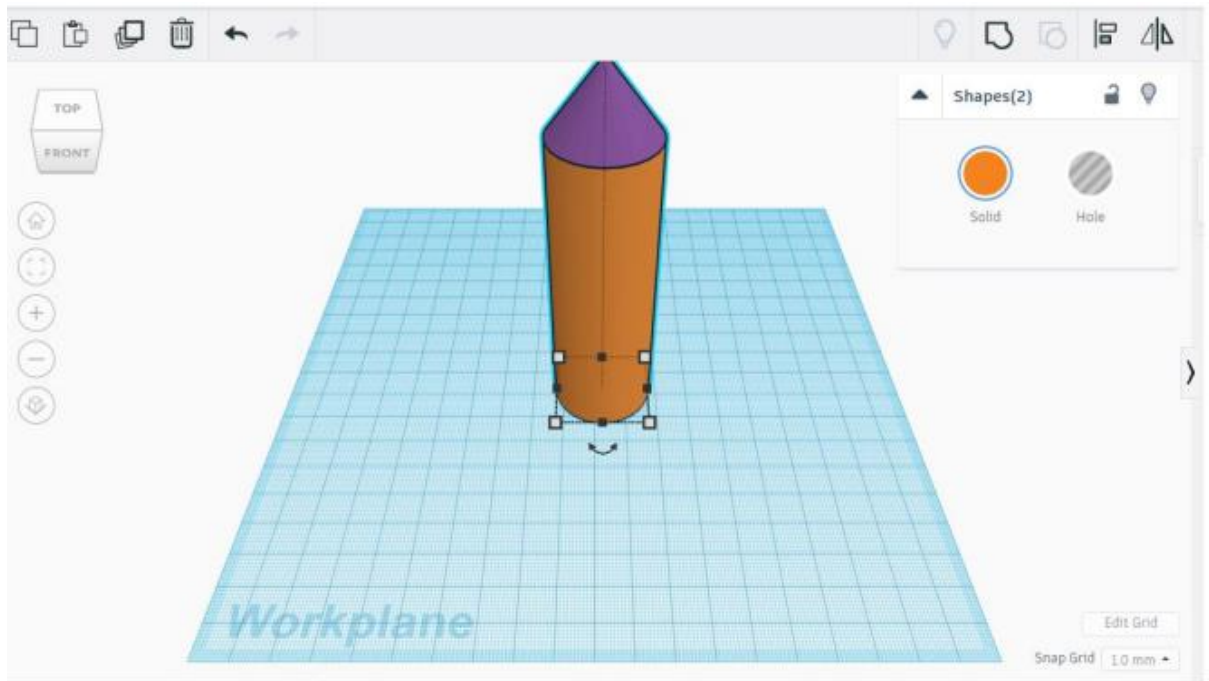


11. Now let's Align the shapes. Select all the shapes and click on Align icon or Press L. Then center align the shapes as shown in the images.

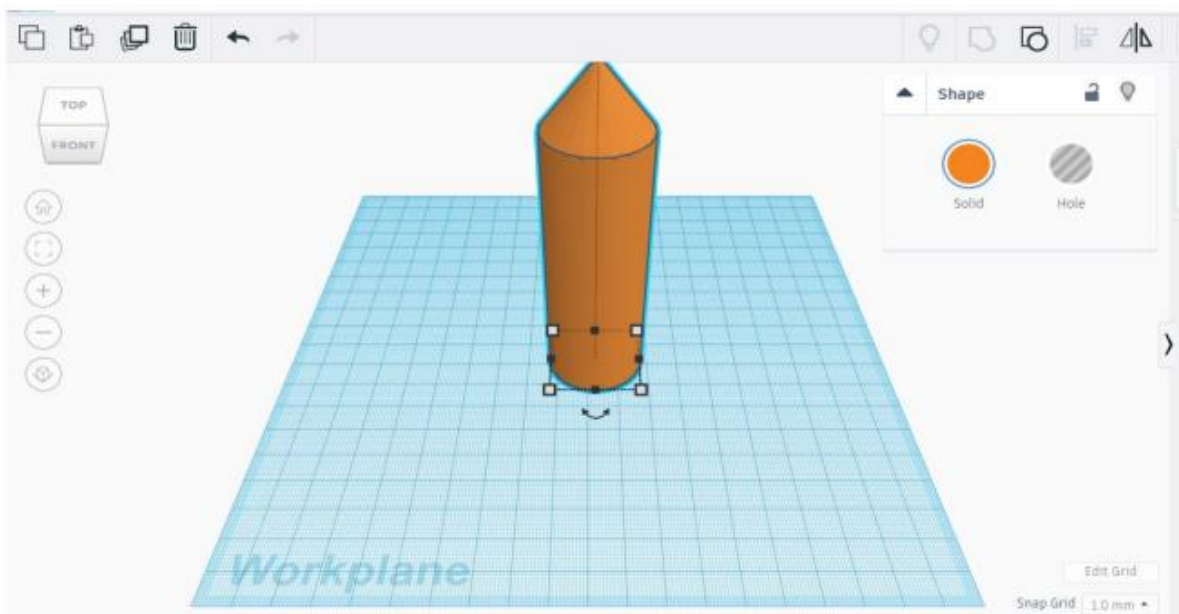




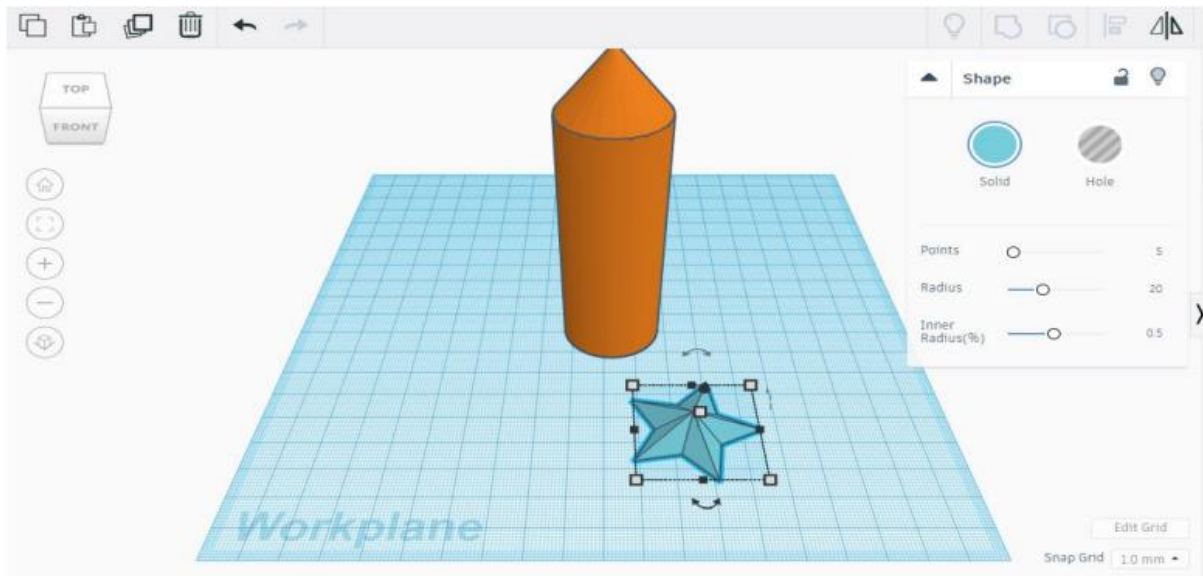
12. Now click on the align icon again to deactivate the command.



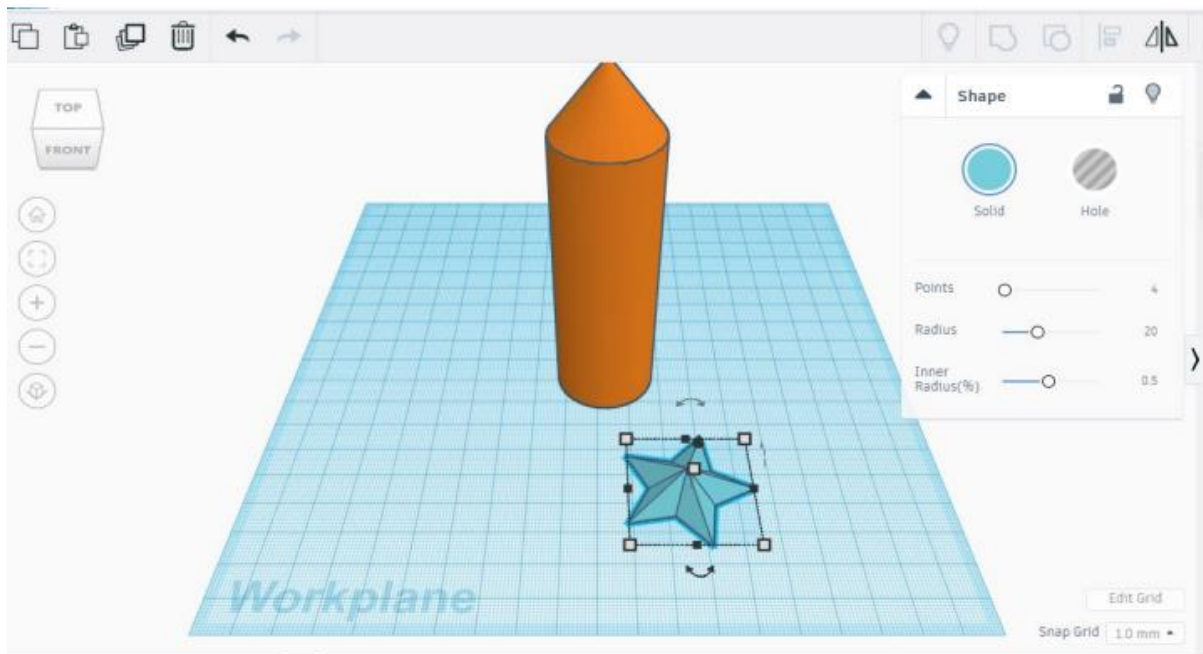
13. Click on the Group icon or press Ctrl+G to group the shapes.

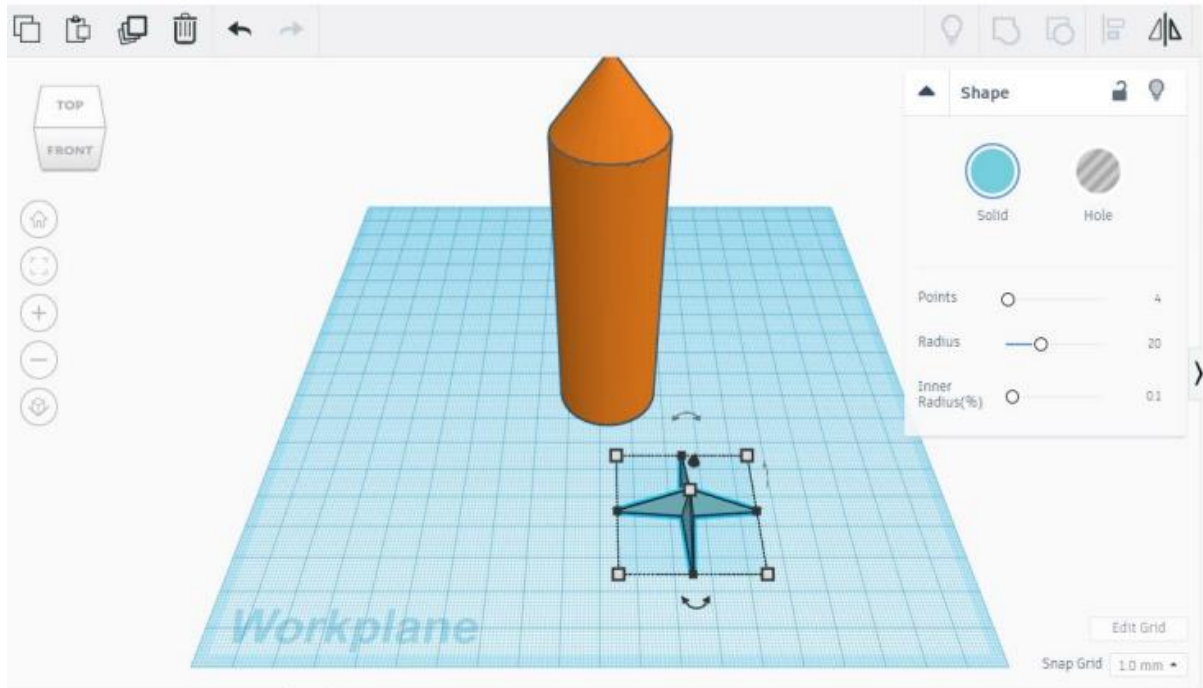


14. Drag and drop a star from the shapes menu as shown in the image.

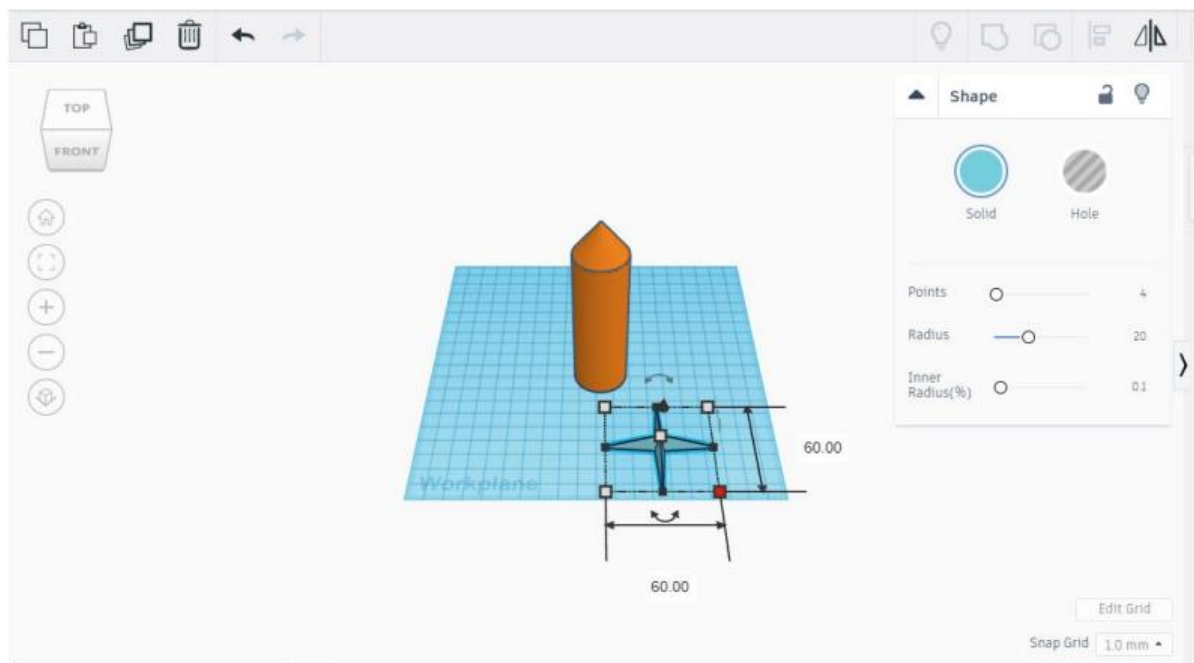


15. Let's add 4 fins to the rocket. We need only 4 sides. Click on the points and enter 4 in place of 5 and then decrease the inner radius. Enter 0.1 in place of inner Radius%

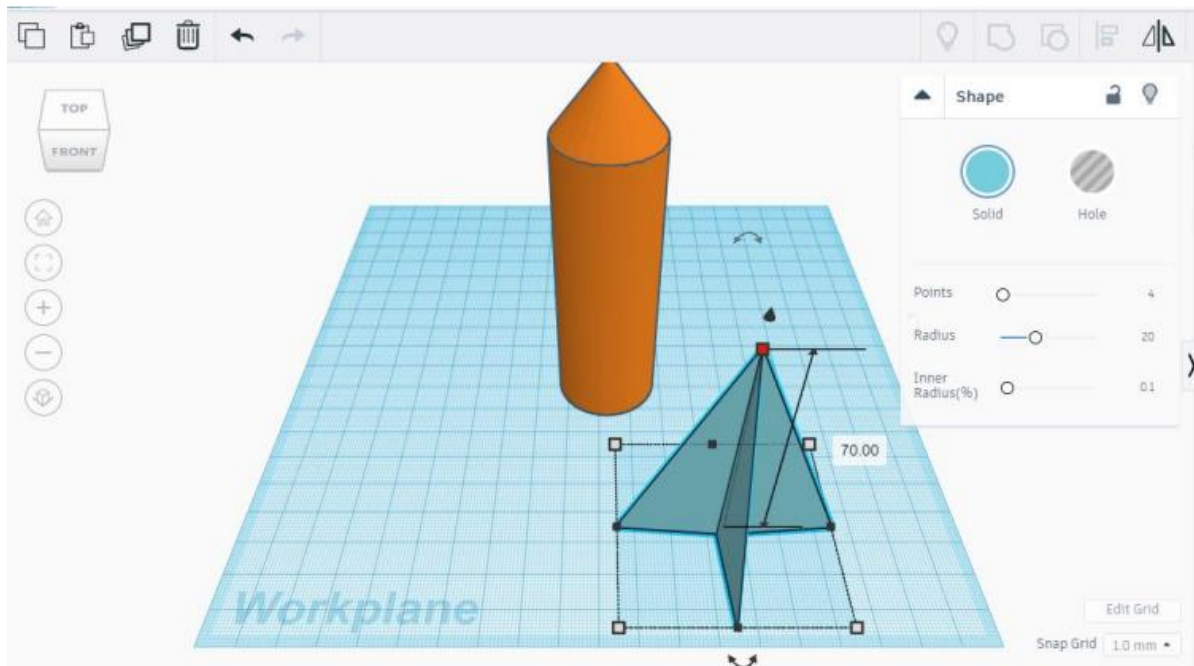




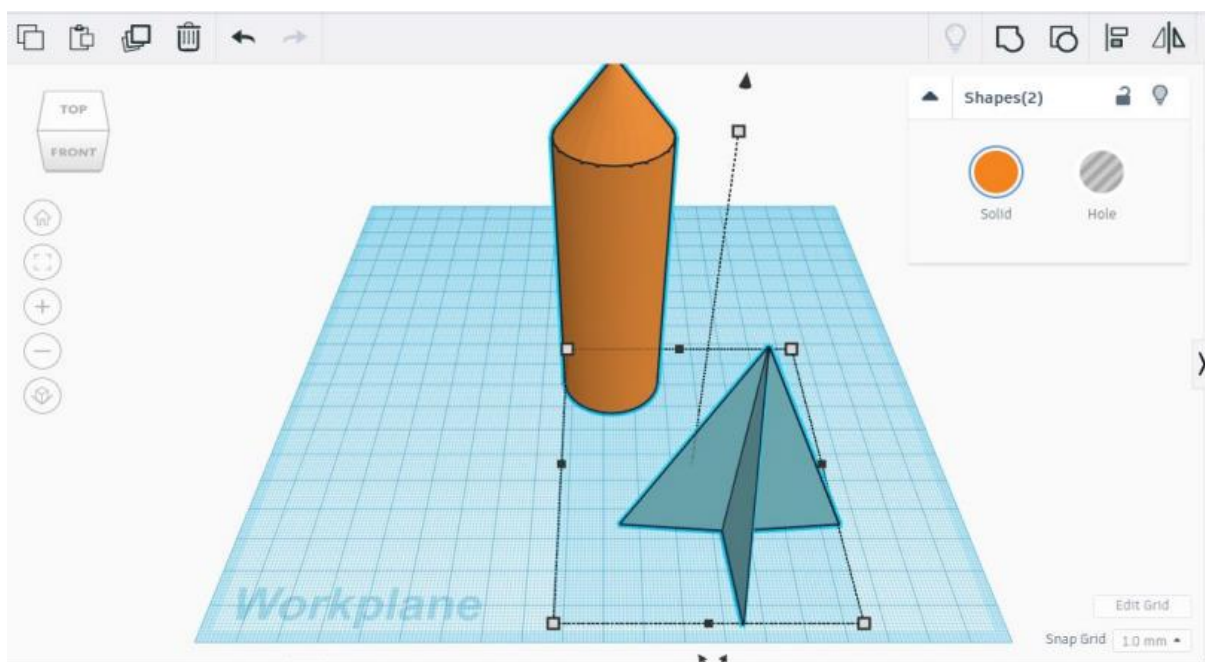
16. Click on the corner and change the dimensions to 60mm on both sides as shown in the image.

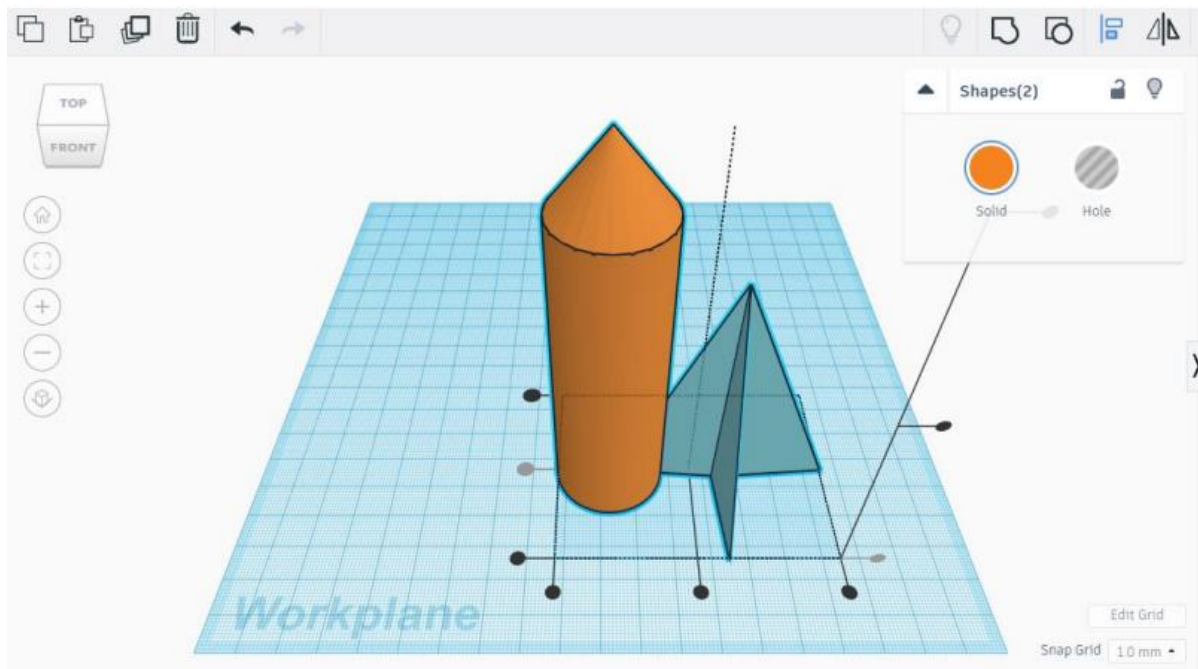
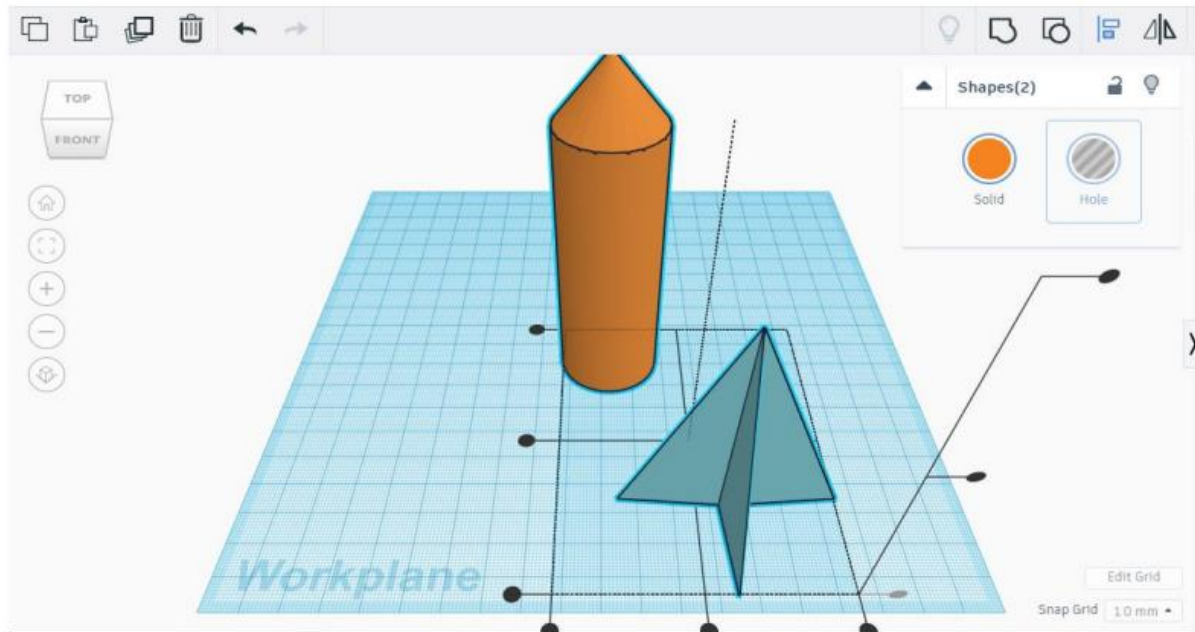


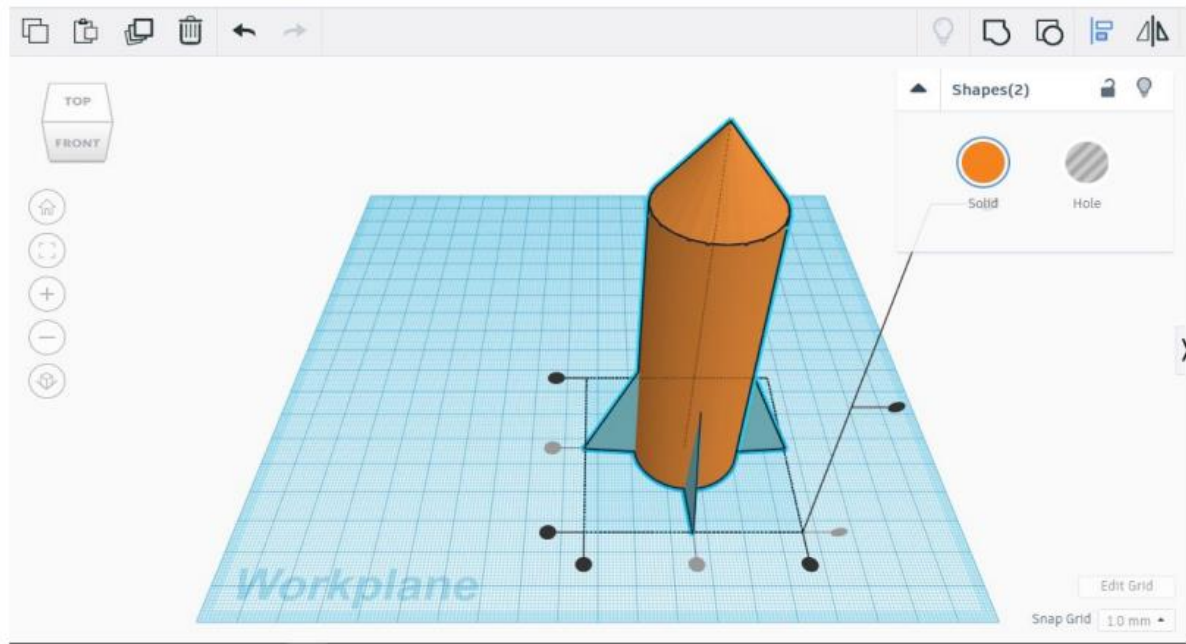
17. Change the height of the fins to 70mm.



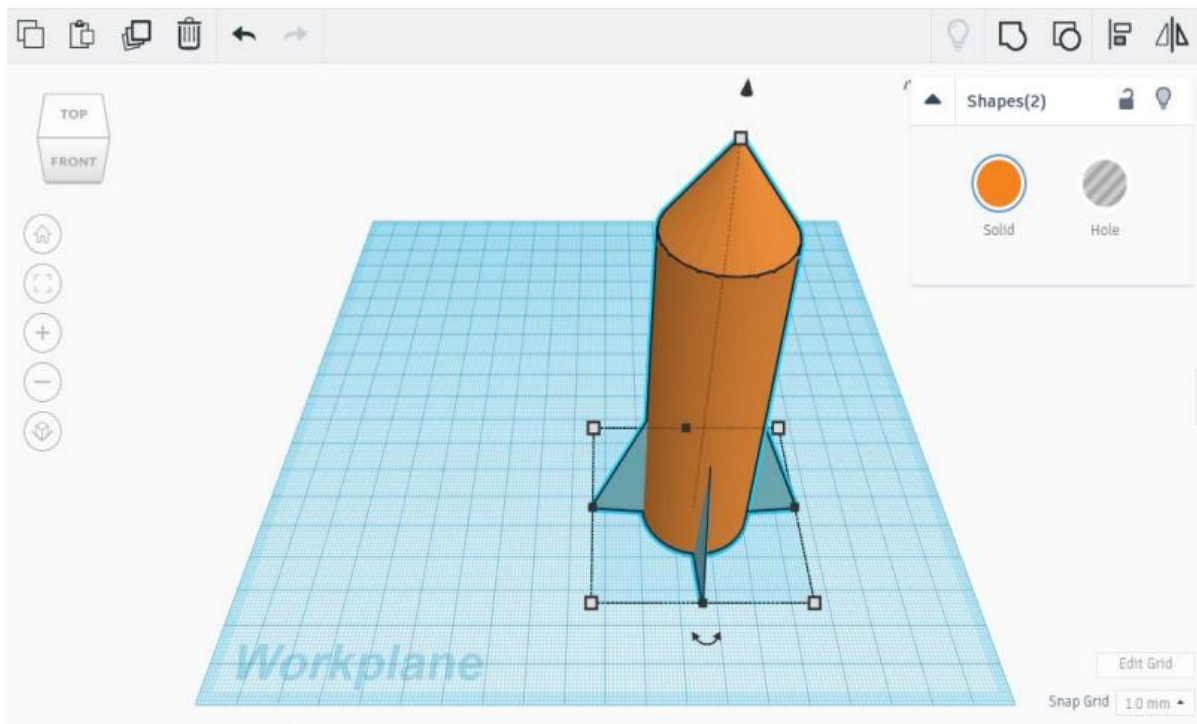
18. Now, select the shapes and center align the rocket fins with the rocket body as shown in the images given below.

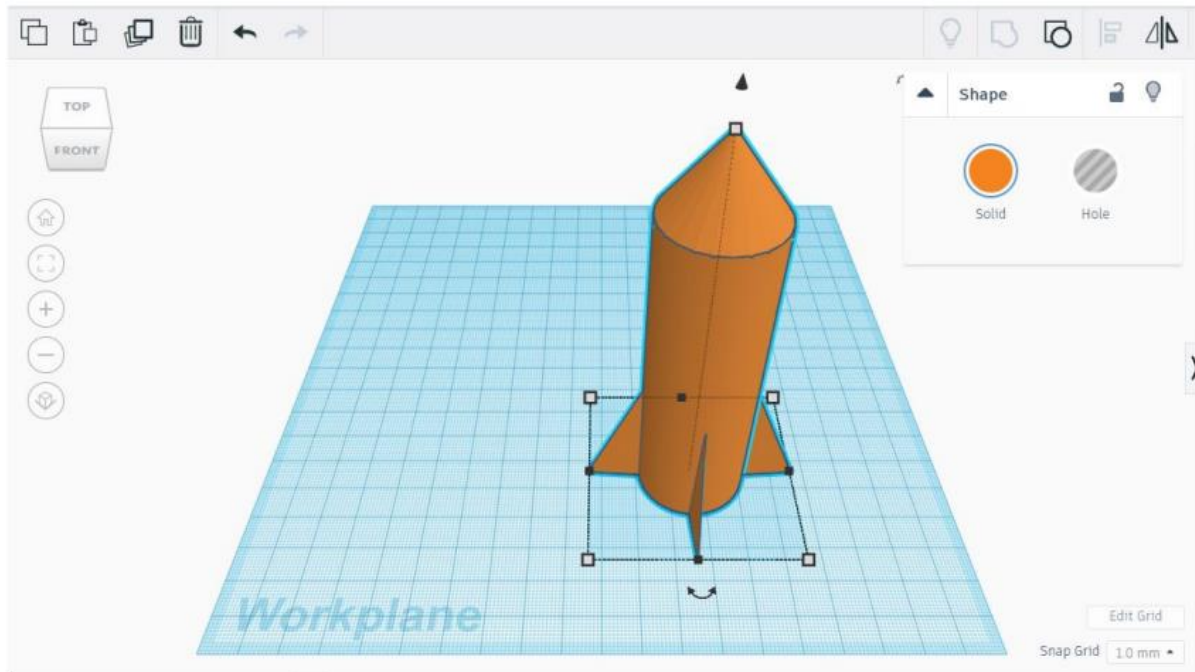






19. Select both the shapes and group them.





Try it yourself:

Now that you have built the rocket, think of a creative name for your rocket and add it on the cylinder using the concepts you have learnt so far. An image has been added for your reference.

Reflection and Learnings:

1. What do you think is the difference between a Rocket and a Satellite?
2. What will happen when you use the “Workplane” option in Tinkercad?
3. Imagine yourself to be a rocket scientist, you along with your team got a chance to redesign the shape of the rocket. Think creatively and draw your redesigned shape in the space given below.
4. What did you or your team struggle with while doing this activity and why? How can you improve next time?

12. Build a live soil moisture monitoring project, and monitor soil moisture levels of a remote plan in your computer dashboard

Aim:

To build a live soil moisture monitoring project and display the soil moisture levels of a remote plan in computer.

Components required:

1. ESP 32
2. Soil Moisture Sensor(Capacitive)
3. F-M Jumper Wires - 3
4. Plant Pot with Soil

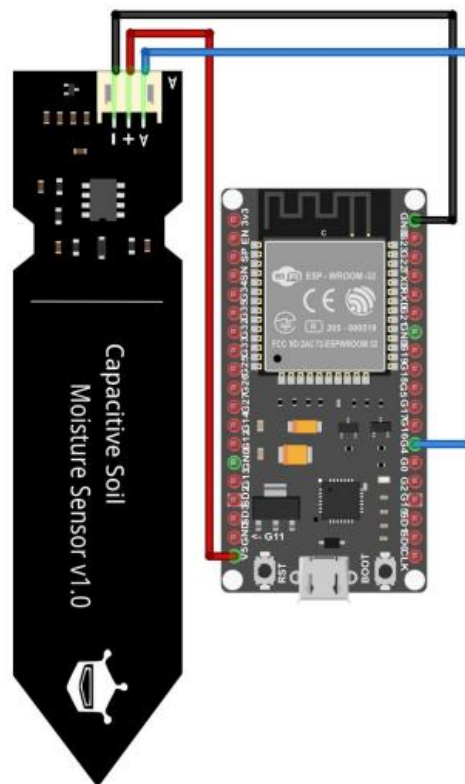
Theory:

IoT soil moisture sensors have become very popular nowadays as we constantly looking for unique solutions to solve global warming and food shortages. integrating IoT sensors in agriculture for monitoring is crucial for our future. Soil moisture sensors are sensors that are able to sense the amount of moisture in the soil. By knowing the amount of moisture, we can translate the digital and analog data into a human language, to know whether the plants need more water or not. By incorporating such sensors you can make better decisions for your smart garden, irrigation system, or it-based solution. traditionally, soil moisture sensors used to be analog. Over time, IoT-based soil moisture sensors became more popular with the ability to integrate cloud and database.

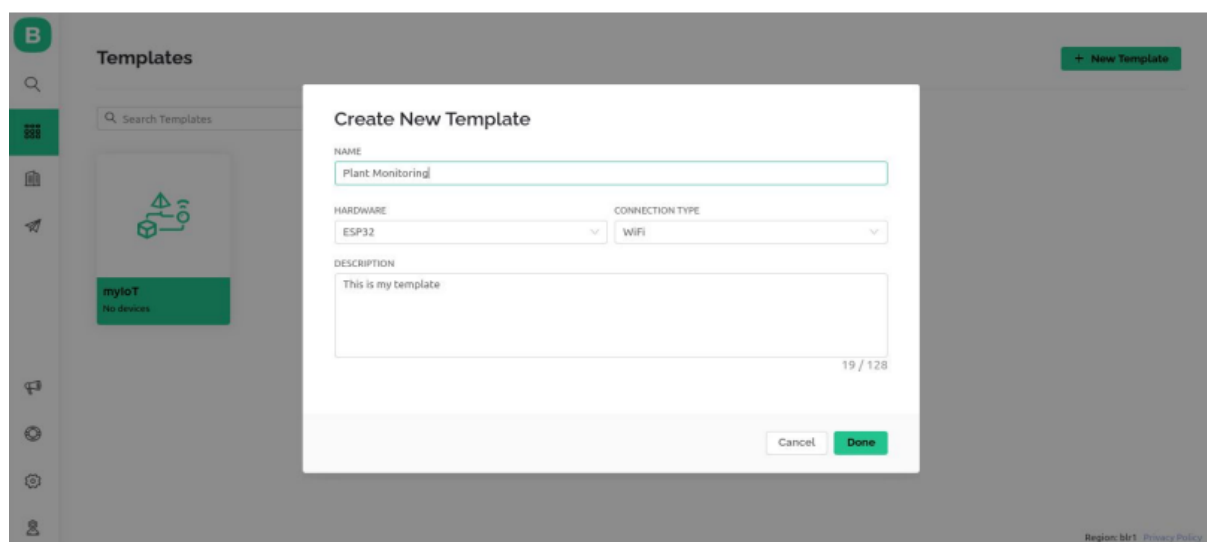


Procedure:

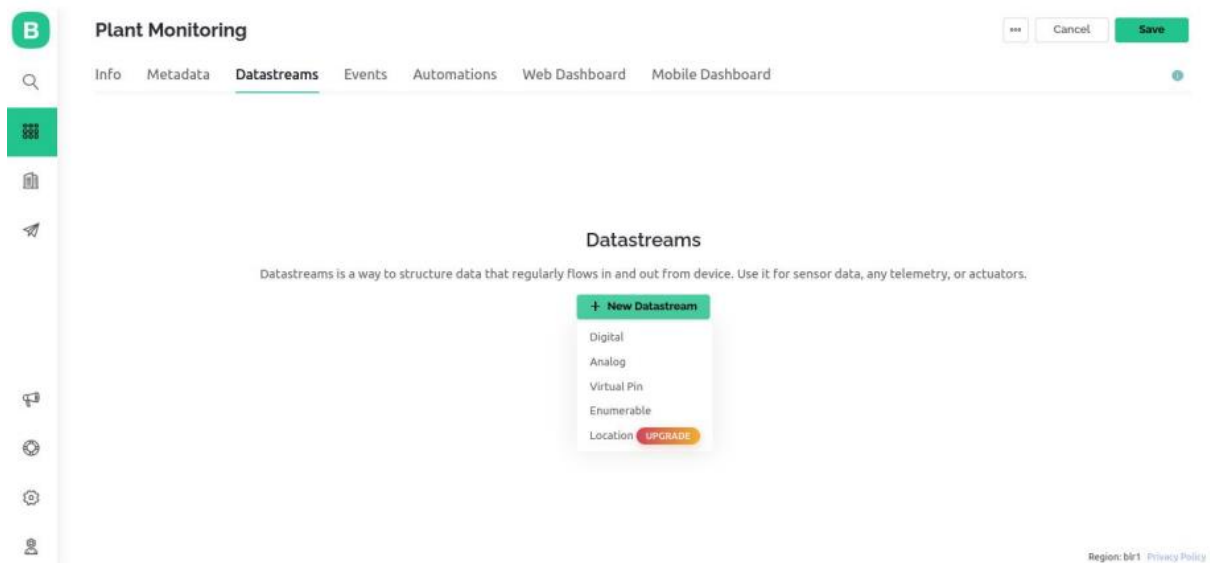
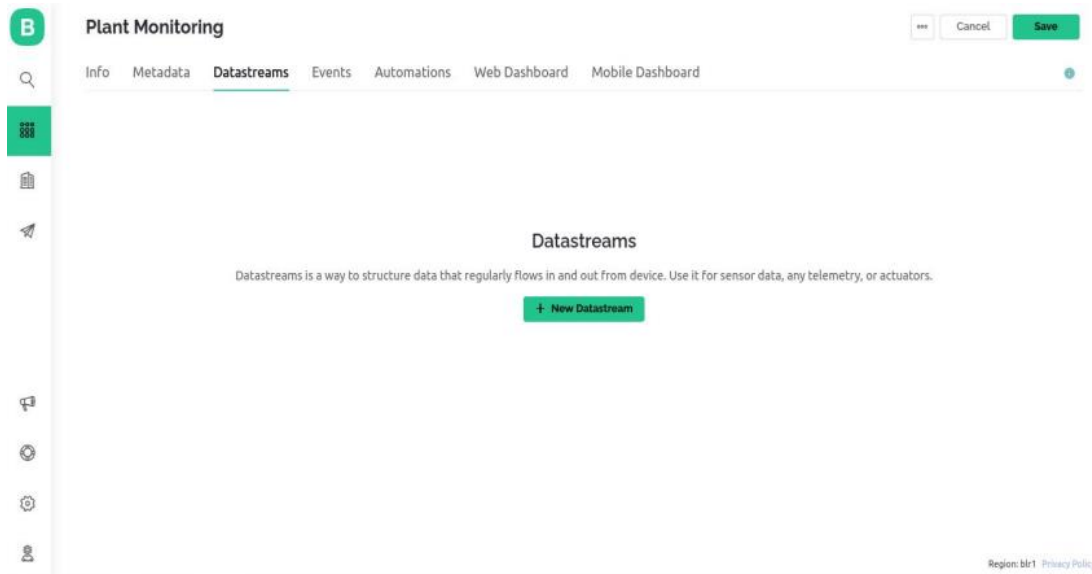
1. Connect soil moisture sensor and ESP32 as per circuit diagram.



3. Login to <https://blynk.cloud/dashboard/login> using the account we already created for previous activity (Control LED using Mobile App)
4. Go to the templates tab and click New Template and give a name for template (Plant Monitoring), Hardware(ESP32), Connection Type (WiFi).



5. Go to Data Streams, Click New Data Stream and Create Virtual Pin.



6. Create a Virtual Pin with following details

Name: soilmoisture (we can give any name)

Pin:V0

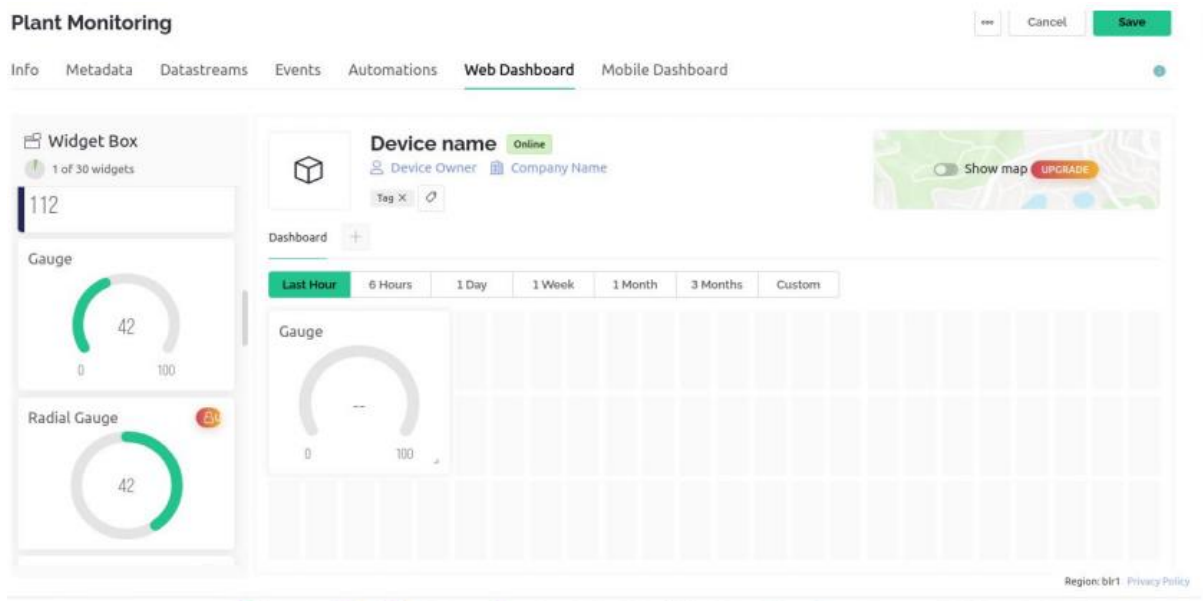
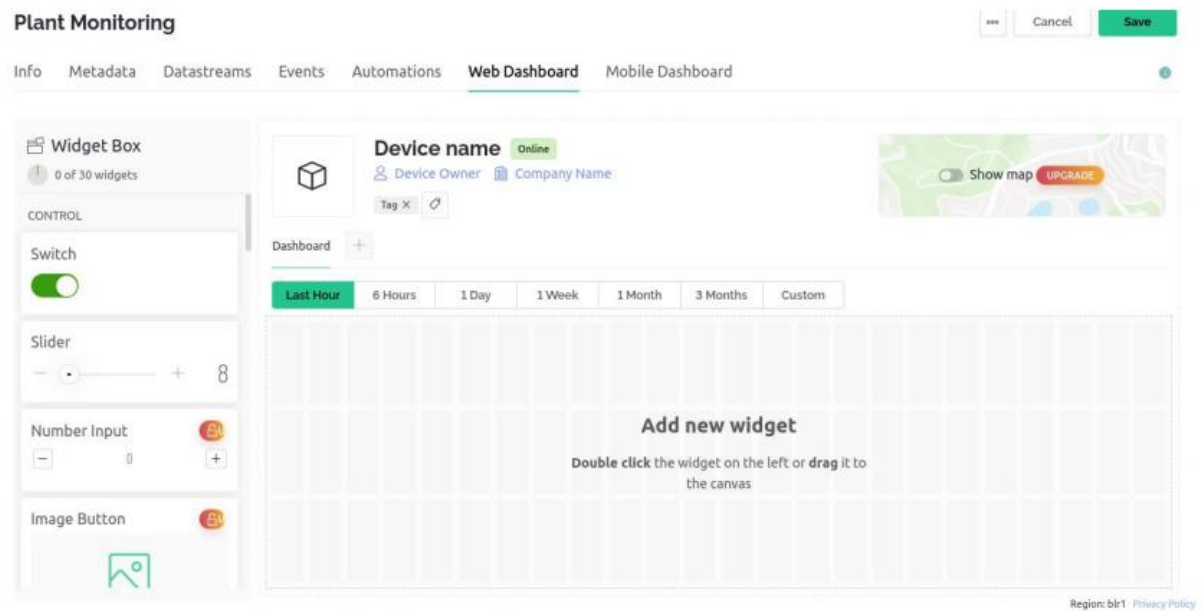
Data Type : Double

Min: 0

Max :255

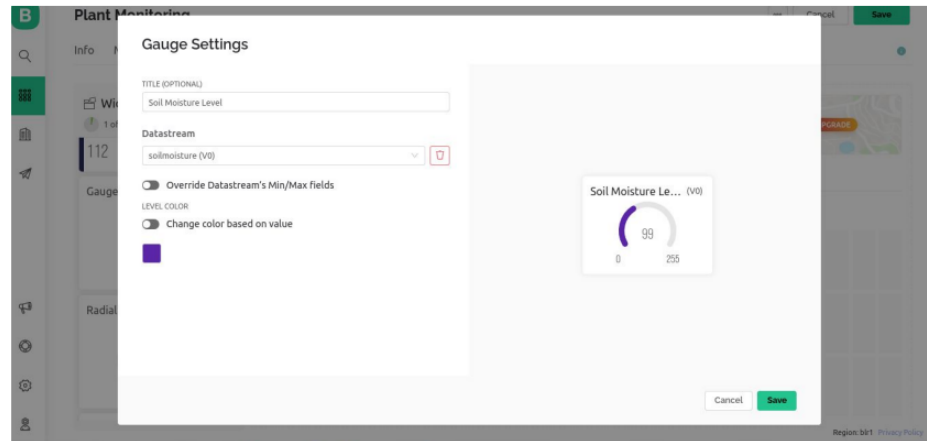


7. Go to the Web Dashboard and insert Gauge Widget

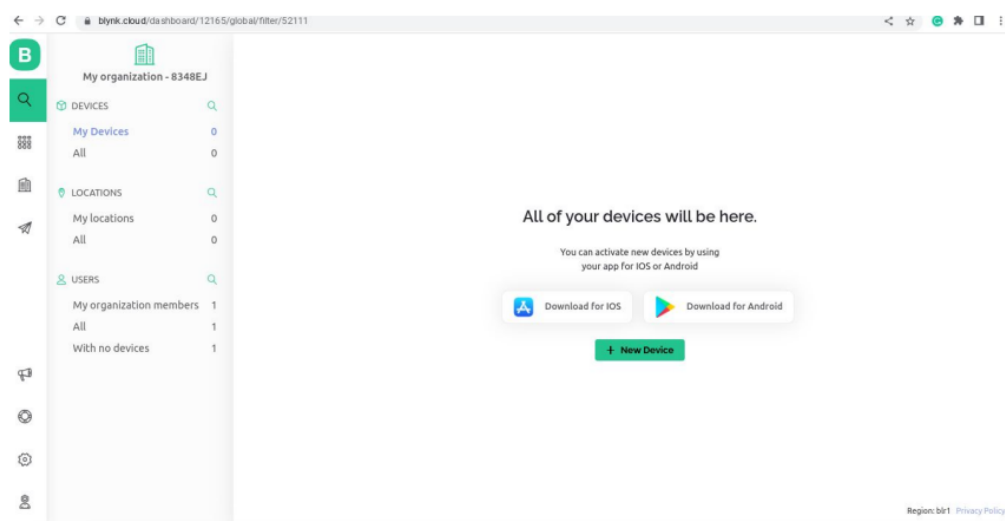


8. Go to Gauge Settings and change name, then select virtual pin V0

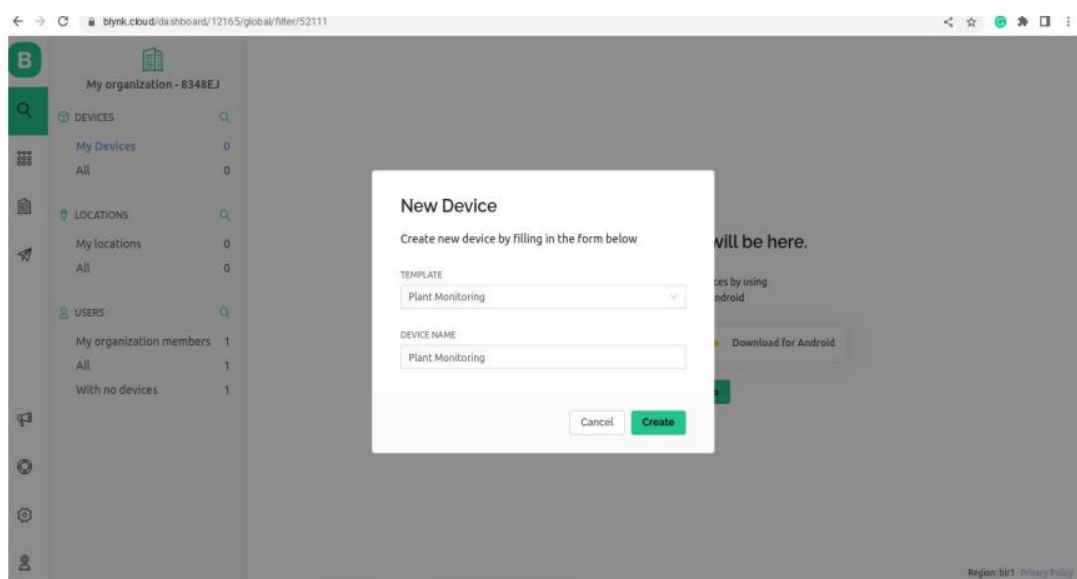
And Click on "Save"



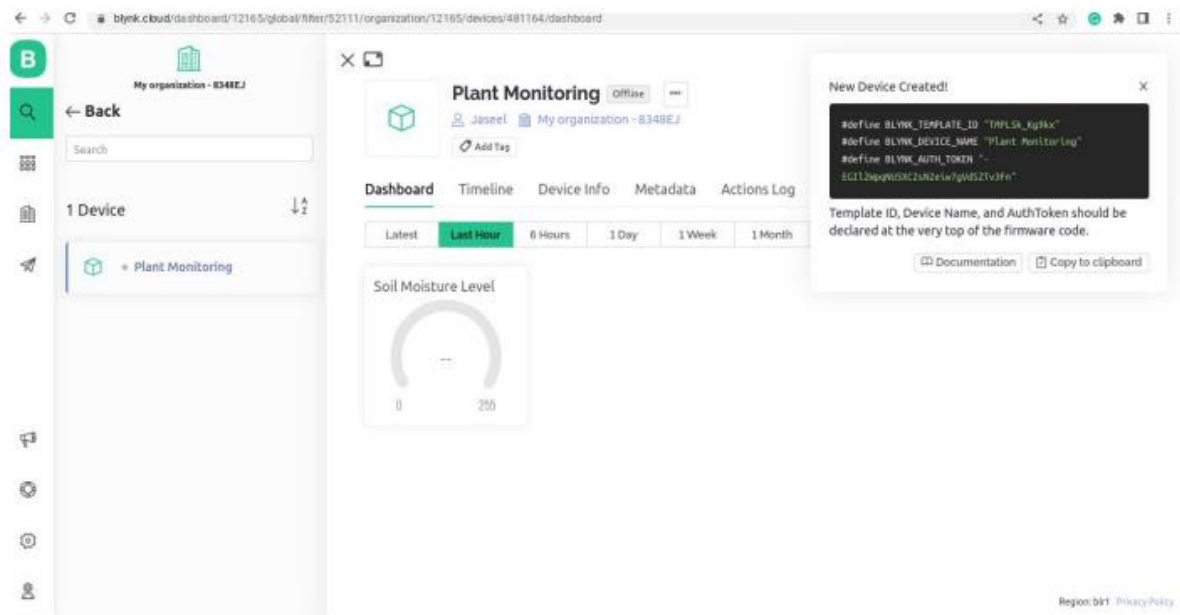
9. Go to the search option and click on the New Device.



Select the template we have already created (Plant Monitoring) and click on create



2. Copy the Template ID, Device Name and Auth Token.



3. Open Arduino IDE and Enter the code given below and upload.

Note : Change Template ID, Device Name and Auth Token
Change SSID and Password of your Wifi hotspot

4. Open Blynk web dashboard then you can see the real time value of soil moisture.

Code:

```
//IoT Plant Monitoring
//Copy paste your Template
ID,Device Name and Auth Token
#define BLYNK_TEMPLATE_ID ""
#define BLYNK_DEVICE_NAME ""
#define BLYNK_AUTH_TOKEN ""
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = ""; // type your wifi name
char pass[] = ""; // type your wifi password
BlynkTimer timer;
void sendSensor()
{
float soilmoisture = analogRead(34);
Blynk.virtualWrite(V0, soilmoisture);
}
```

```

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(100L, sendSensor);
}

void loop()
{
  Blynk.run();
  timer.run();
}

```

Safety Measures:

1. Never touch any electrical equipment wet hands.
2. Your work area must always be kept dry.
3. If you smell anything burning, immediately disconnect the power and examine your power supply to find out what went wrong.

Reflection and Learnings:

1. Share at least 3 skills or lessons that you learned today.
2. How will you apply these skills outside class?
3. What do you think is the difference between a Moisture Sensor and a UV Sensor?
4. How did you contribute to the team and how do you think it helped? How else could you have contributed?
5. What communication challenges did you and your team face? How can your team communicate better next time?

