Voting Machine Project

Submitted By: -

KEERTHAN GOWDA M [ENG22MCA041]
ELAKKIYA M[ENG22MCA036]
NEMALAPURI ANANYA [ENG22MCA052]
PRITHVI RAJ K [ENG22MCA054]

In partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

DAYANANDA SAGAR

UNIVERSITY SCHOOL OF

ENGINEERING

KUDLUGATE, HOSUR ROAD, BANGALORE -560068

March 2024

ACKNOWLEDGEMENT

I wish to express my salutations to the beloved and highly esteemed institute **DAYANANDA SAGAR UNIVERSITY.** I thank **Dr. UDAY KUMAR REDDY**, Dean, School of Engineering, Dayananda Sagar University, Bangalore, for his valuable suggestions and expert advice. I would like to thank **Dr. Vasanthi Kumari P**, Chairman, Department of Computer Applications, Dayananda Sagar University for her valuable guidelines and constant support throughout my project.

I am grateful to our guide and would like to express special thanks to **Ms.Padmageetha BG**, Associate Professor, Department of Computer Applications, DSU for her constant support, necessary advice, and valuable guidance that helped me in completing the project/report work successfully.

I would like to thank all the Teaching and non-teaching staff of the Department of Computer Applications, DSU for their help. I would like to thank my parents and friends who have always supported me in every path of the Report work.

I perceive this opportunity as a milestone in my career development. I will strive to use the gained skills and knowledge in the best possible way and I will continue to work on my improvement in order to achieve the desired career objectives. I hope to continue the cooperation with all of you in the future.

KEERTHAN GOWDA M (ENG22MCA041)
ELAKKIYA M(ENG22MCA036)
NEMALAPURI ANANYA (ENG22MCA052)
PRITHVI RAJ K (ENG22MCA054)

DAYANANDA SAGAR UNIVERSITY

Kudlu Gate, Hosur Road, Bangalore-560068

DEPARTMENT OF COMPUTER APPLICATIONS

Bonafide Certificate

This is to certify that the project titled "**Voting Machine Project**" is a Bonified record of the work done by **KEERTHAN GOWDA M (ENG22MCA041), ELAKKIYA M(ENG22MCA036), NEMALAPURI ANANYA (ENG22MCA052)** and **PRITHVI RAJ K (ENG22MCA054)** In partial fulfillment of the requirements for the award of the degree of **Master of Computer Applications** in **DAYANANDA SAGAR UNIVERSITY, BANGALORE**, during the year 2022-2023

Guide	**Chairman**
Ms. Padmageetha BG	**Dr. Vasanthi Kumari.P**
Associate Professor	Chairperson
Dept. of Computer Applications, Applications,	Dept. of Computer
Dayananda Sagar University University	Dayananda Sagar

Project Viva Held on: -

Internal Examiner	**External Examiner**

Abstract:

The Voting Machine Project Documentation presents the design and implementation of a simple electronic voting system using Arduino microcontroller technology. This project encompasses the hardware requirements, circuit diagram, code explanation, and potential expansions for the voting machine. The hardware components include an Arduino Uno, 16x2 LCD display, push buttons, resistors, and jumper wires, arranged on a breadboard. The Arduino code manages the functionality of the voting machine, including vote casting, result display, and system reset. Through a straightforward setup and loop structure, the project provides a foundation for electronic voting systems, with potential for expansion into more advanced features such as voter authentication and support for multiple candidates.

INDEX

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION OF THE PROJECT

The introduction for this topic could focus on the increasing digitalization of electoral processes and the importance of developing secure and efficient electronic voting systems. It could highlight the need for accessible, user-friendly voting solutions that ensure transparency, accuracy, and fairness in democratic processes. Introducing the Arduino-based electronic voting machine serves as a practical example of how technology can be leveraged to streamline voting procedures, especially in smaller-scale or experimental settings. Furthermore, the introduction might discuss the significance of open-source hardware and software solutions in democratizing access to voting technology and fostering innovation in electoral systems. By exploring the functionalities and components of the Arduino-based voting machine, the introduction sets the stage for a deeper examination of its implementation, challenges, and potential impact on modern democracy.

The global landscape of electoral systems is undergoing a transformation propelled by technological advancements. Traditional paper-based voting methods are increasingly being augmented or replaced by electronic systems, driven by the aim of enhancing efficiency, accessibility, and accuracy in the democratic process. In this context, the development of electronic voting machines (EVMs) represents a pivotal innovation, promising to revolutionize how elections are conducted and votes are counted.The advent of EVMs opens up a realm of possibilities for modernizing electoral procedures, catering to the diverse needs and expectations of voters in the digital age. By harnessing the power of embedded systems and microcontrollers like Arduino, it becomes feasible to design and implement cost-effective, customizable voting solutions tailored to specific contexts and requirements. These systems offer numerous advantages, including rapid tabulation of results, reduced chances of human error, and the potential for remote or mobile voting, thereby enhancing voter convenience and participation.However, alongside the promise of electronic voting lies a host of challenges and considerations. Security concerns loom large, as the integrity of electoral processes must be safeguarded against potential tampering, hacking, or manipulation. Ensuring the confidentiality, anonymity, and verifiability of votes is paramount to maintaining trust and credibility in the democratic process. Moreover, issues related to accessibility, inclusivity, and digital literacy must be addressed to prevent disenfranchisement and ensure equitable access to voting technology for all segments of society.Against this backdrop, the Arduino-based electronic voting machine presented here offers a compelling case study in the development and deployment of EVMs. Its open-source nature facilitates transparency and scrutiny, allowing for community-driven enhancements and auditability of the voting system.

1.2 OBJECTIVES

. The objective of this project is to design and implement an Arduino-based electronic voting machine (EVM) capable of efficiently and accurately tallying votes cast for multiple candidates or options. The primary goals include:

Development of a user-friendly interface: Create a voting interface that is intuitive and accessible to voters, ensuring ease of use and minimizing the potential for errors during the voting process.

Implementation of vote counting logic: Design algorithms to accurately count and store votes for each candidate or option, ensuring the integrity and reliability of the voting results.

Integration of feedback mechanisms: Incorporate visual and auditory feedback mechanisms, such as LED indicators and buzzer alerts, to provide real-time confirmation to voters and election officials when a vote is successfully cast.

Enhancement of security measures: Implement measures to enhance the security and integrity of the voting system, including encryption techniques, tamper detection mechanisms, and safeguards against unauthorized access or manipulation of voting data.

Validation and testing: Conduct rigorous testing and validation procedures to verify the functionality, accuracy, and reliability of the electronic voting machine under various scenarios and conditions.

Documentation and dissemination: Document the design, implementation, and operation of the Arduino-based EVM, providing comprehensive instructions and guidelines for replication and deployment in other electoral contexts. Disseminate findings and insights to the broader community to foster knowledge-sharing and facilitate improvements in electronic voting technology.

By achieving these objectives, the project aims to contribute to the advancement of electronic voting systems, promoting transparency, efficiency, and inclusivity in democratic processes while addressing key challenges and concerns associated with the adoption of technology in elections.

CHAPTER 2

Literature Review

Advancements in Arduino-Based EVMs:

Arduino microcontrollers have revolutionized the development of EVMs by providing a versatile platform for building custom electronic voting systems. Researchers have leveraged Arduino's open-source hardware and software architecture to create EVM prototypes tailored to specific electoral requirements and constraints. For instance, studies have demonstrated the feasibility of implementing Arduino-based EVMs with features such as voter verification, ballot encryption, and tamper detection mechanisms. Additionally, Arduino's compatibility with various sensors, displays, and communication modules enables the integration of advanced functionalities, such as biometric authentication, real-time data transmission, and accessibility features for voters with disabilities.

Security Considerations:

Security remains a paramount concern in Arduino-based EVMs, given the potential risks of tampering, manipulation, and unauthorized access. Researchers have proposed several strategies to enhance the security of these systems, including cryptographic protocols for secure communication, hardware-based encryption techniques to protect voting data, and physical security measures to prevent tampering with EVM components. Furthermore, studies have emphasized the importance of rigorous testing, code review, and independent auditing to identify and mitigate vulnerabilities in Arduino-based EVM implementations.

Usability and Accessibility:

Ensuring the usability and accessibility of Arduino-based EVMs is essential to promote voter confidence and participation. Researchers have explored various design considerations to enhance the user experience, such as intuitive interfaces, clear instructions, and feedback mechanisms to confirm vote casting. Moreover, efforts have been made to accommodate diverse voter needs, including multilingual interfaces, audio feedback for visually impaired voters, and ergonomic designs for ease of use. By prioritizing usability and accessibility, Arduino-based EVMs can cater to a broader range of voters and promote inclusivity in the

electoral process.

Deployment and Adoption Challenges:

Despite the advancements in Arduino-based EVM technology, several challenges hinder widespread deployment and adoption. Limited resources, technical expertise, and regulatory barriers pose obstacles to the development and implementation of Arduino-based EVMs in electoral contexts. Moreover, concerns regarding the reliability, integrity, and auditability of electronic voting systems have led to skepticism and resistance from stakeholders, including election officials, policymakers, and the general public. Addressing these challenges requires concerted efforts to build trust, foster collaboration, and establish robust governance frameworks for electronic voting technology.

CHAPTER 3

SYSTEM CONFIGURATION

3.1 Hardware Configuration

1.Arduino board

2.lcd display

3.jumper wiers

4.buzzer

5.led

6.Bread board

3.2 Software Configuration

1. OS : Windows X
2. Arduino ide
3. Thinker cad

3.3 Designing language

Arduino c

Arduino C, also known as Arduino programming language or Arduino sketch language, is a simplified variant of C and C++ programming languages tailored specifically for programming Arduino microcontroller boards. It provides a user-friendly interface and abstracts away many of the complexities associated with traditional C programming, making it accessible to beginners and hobbyists without extensive programming experience.

Key features of Arduino C include:

1. **Setup and Loop Functions**: Arduino programs are structured around two main functions: `setup()` and `loop()`. The `setup()` function is executed once when the Arduino board starts up or is reset, and it is typically used to initialize variables, pin modes, and peripherals. The `loop()` function, on the other hand, runs continuously after `setup()` completes, allowing users to define the main logic of their program.

2. **Built-in Functions and Libraries**: Arduino C comes with a rich set of built-in functions and

libraries that simplify common tasks such as reading from sensors, controlling actuators, and interfacing with external devices. These libraries abstract low-level hardware operations, enabling users to focus on application logic rather than hardware details.

3. **Simple Syntax**: Arduino C features a simple syntax that resembles C and C++, making it easy for programmers familiar with these languages to transition to Arduino development. However, Arduino C omits some advanced language features and libraries found in standard C/C++, such as dynamic memory allocation and advanced data structures, to optimize for simplicity and ease of use.

4. **Hardware Abstraction**: Arduino C abstracts away hardware-specific details, allowing users to write code that is portable across different Arduino boards and microcontrollers. This abstraction layer simplifies the development process and facilitates code reuse, as programmers can write code that works across a range of Arduino-compatible hardware platforms.

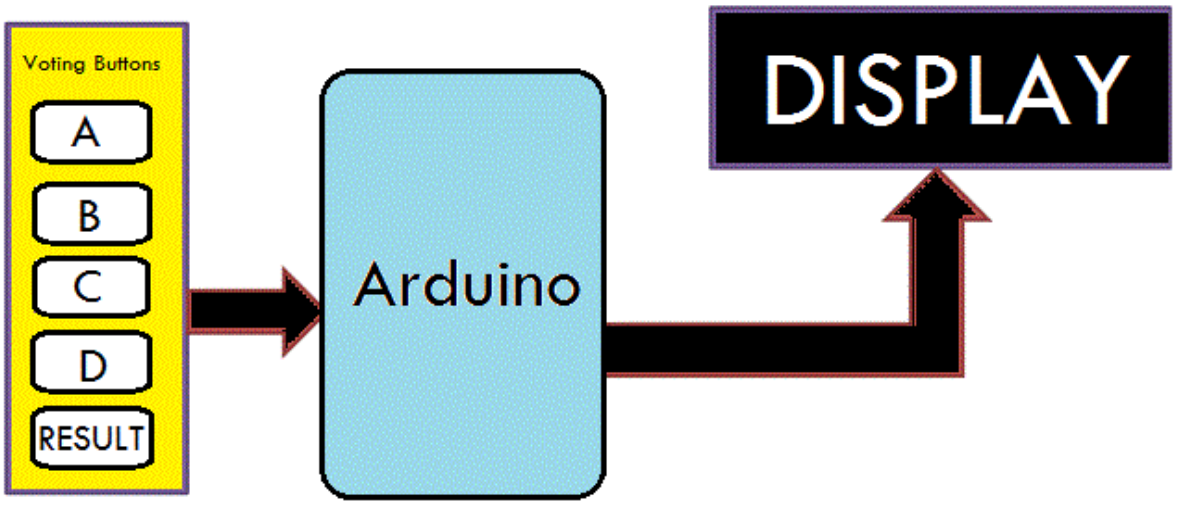
5. **IDE Integration**: Arduino C is integrated into the Arduino Integrated Development Environment (IDE), a user-friendly development environment with features such as syntax highlighting, code completion, and built-in serial monitoring. The IDE provides a seamless development experience, allowing users to write, compile, and upload Arduino sketches to their boards with ease.

Overall, Arduino C strikes a balance between simplicity and functionality, making it an ideal choice for prototyping, experimentation, and educational purposes. While it may lack some of the advanced features of standard C/C++, its ease of use and extensive community support make it a popular choice for hobbyists, educators, and professionals alike.

CHAPTER 4

SOLVING METHODOLOGY

4.1.BLOCK DIAGRAM



4.2 BLOCK DIAGRAM CONNECTIONS:

The block diagram you sent depicts a simplified voting system design with voting buttons and a display. Here's a breakdown of the components:

Voting Buttons: These buttons likely correspond to different candidates or options in a vote. When a user presses a button, a signal is sent to the Arduino.

Arduino: This is a microcontroller board that reads the input from the voting buttons and controls the display. It likely tallies the votes cast for each option.

Display: This shows the results of the vote. It could be a simple LCD display showing the total number of votes cast for each option, or it could be a more sophisticated display that shows additional information, such as the percentage of votes cast for each option.

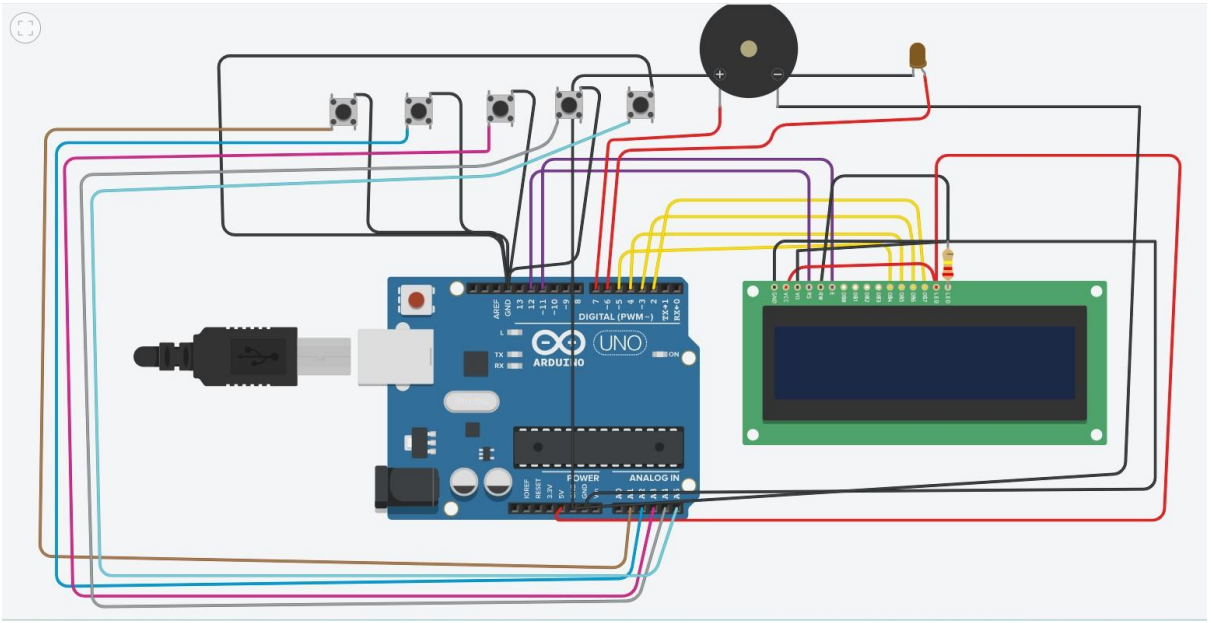
The block diagram doesn't show how votes are cast securely or how they are prevented from being tampered with. In a real-world voting system, security measures would be essential to ensure the integrity of the election.

Overall, this block diagram represents a basic design for a simple voting system. Arduino-based voting systems are sometimes used in educational settings to teach students about the voting process. However, they would not be secure enough to be used in a real election.

CHAPTER 5

CIRCUIT DIAGRAM

CIRCUIT



5.1 PIN CONFIGURATION:

1. LCD Display Pins:

- Pins 12, 11, 5, 4, 3, and 2 of the Arduino are used to interface with the LCD display.
- These pins are connected to specific pins on the LCD module for communication and control.
- The pins are defined in the LiquidCrystal library and are initialized as follows:

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

- The pin configuration for the LCD display might vary depending on the specific module used and how it's connected to the Arduino.

2. Push Button Pins:

- Five push buttons are used for voting, and each button is connected to a separate digital input pin on the Arduino.

- The pins are defined as sw1, sw2, sw3, sw4, and sw5, with corresponding pin numbers 15, 16, 17, 18, and 19.
- These pins are configured as input pins with pull-up resistors enabled:

COMPONENTS USED FOR THIS PROJECT

- ARDUINO UNO
- BUZZER
- LCD
- LED
- BUTTON
- JUMPER WIERS
- RESISTOR
- BREAD BOARD

CHAPTER 6

IMPLEMENTATION

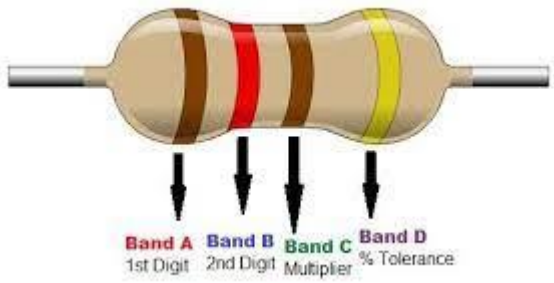
6.1 HARDWARE IMPLEMENTATION

1Arduino Uno: The Arduino Uno board acts as the central processing unit of the voting machine. It communicates with the LCD display and reads the status of the push buttons to manage the voting process.

16x2 LCD Display: The LCD display is used to provide visual feedback to voters and display the voting results. It is connected to the Arduino via

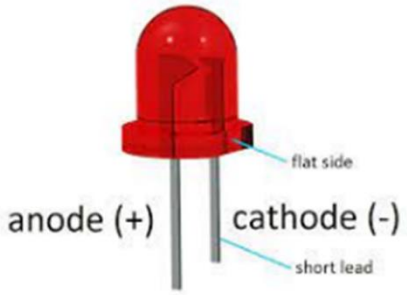
Push Buttons: Five push buttons are connected to digital input pins on the Arduino. These buttons allow voters to cast their votes for different candidates or options.

Resistors: Pull-up resistors (typically 10kΩ) are connected between the digital input pins and the positive voltage supply (5V). They ensure that the input pins read a stable logic HIGH state when the buttons are not pressed.

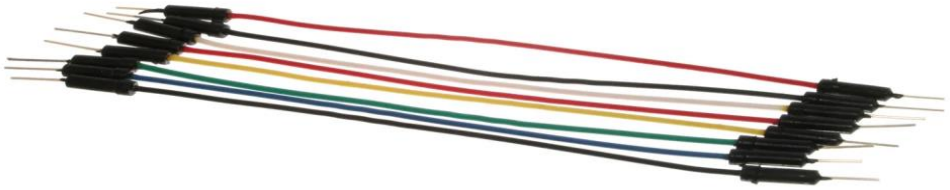


Buzzer: The buzzer is connected to a digital output pin on the Arduino. It produces sound feedback when displaying the voting results. The frequency and duration of the tone can be controlled programmatically.

LED: The LED is connected to a digital output pin on the Arduino. It provides visual feedback when a button is pressed during the voting process, indicating that the vote has been recorded.



JUMPER WIERS: Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.



BREAD BOARD: A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike a perfboard or stripboard, breadboards do not require soldering or destruction of tracks and are hence reusable.

CHAPTER 7

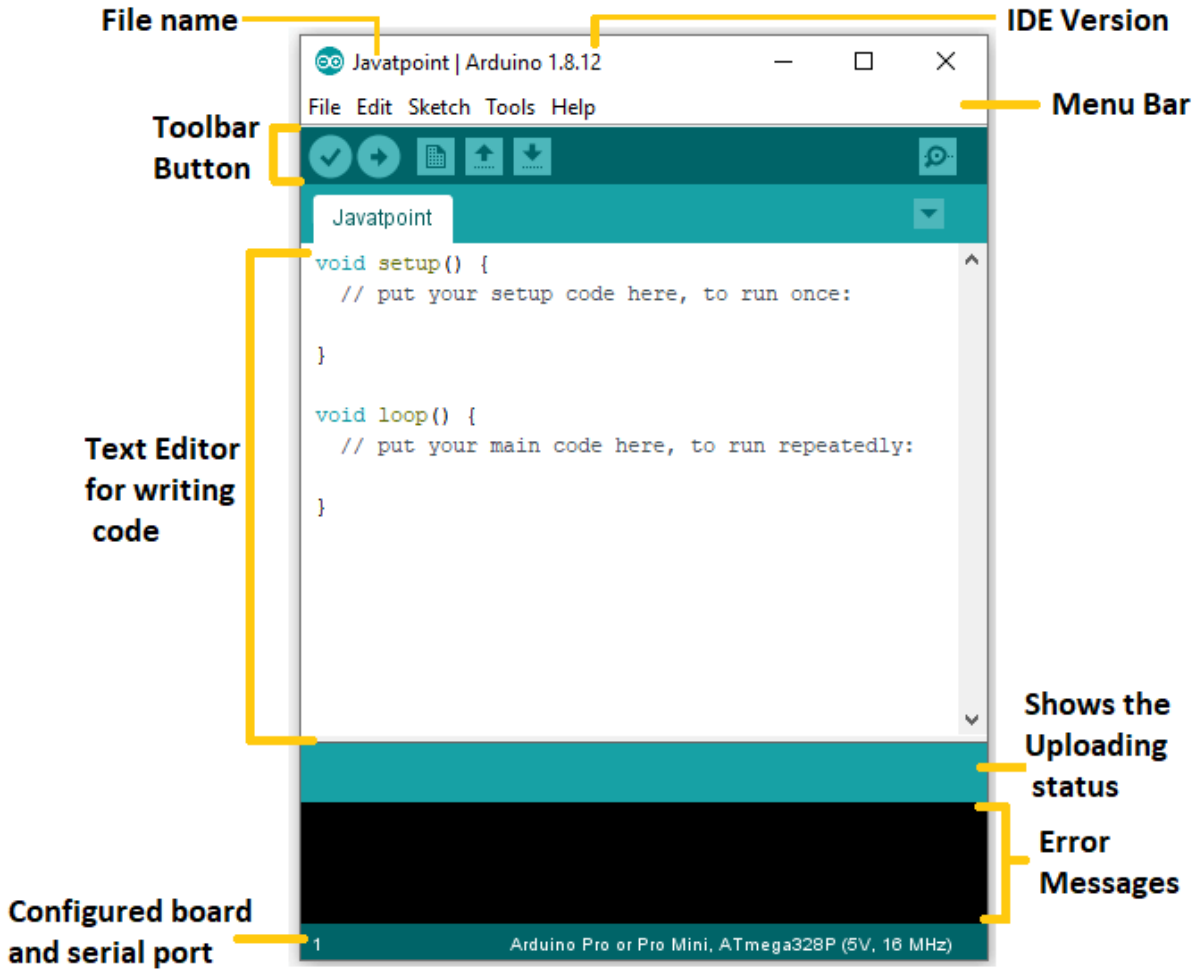
Software

7.1 Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Arduino IDE will appear as:

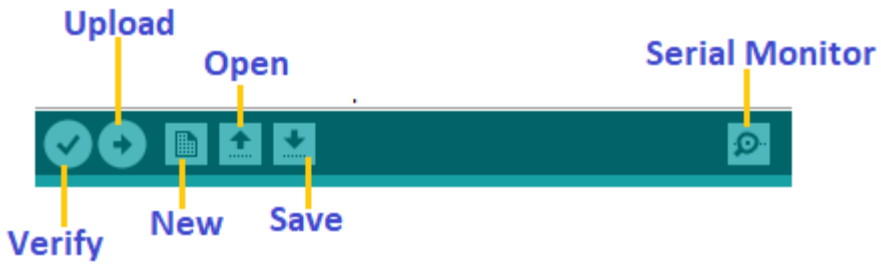


Let's discuss each section of the Arduino IDE display in detail.

Toolbar Button

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**.

It is shown below:



Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.

If the uploading is failed, it will display the message in the error window.

We do not require any additional hardware to upload our sketch using the Arduino Bootloader. A **Bootloader** is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open

The Open button is used to open the already created file. The selected file will be opened in the current window.

Save

The save button is used to save the current sketch or code.

New

It is used to create a new sketch or opens a new window.

Verify

The Verify button is used to check the compilation error of the sketch or the written code.

Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

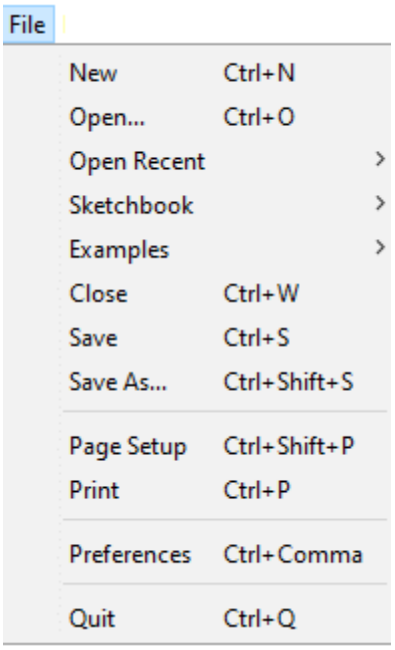
It is shown below:

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

Menu Bar

File

When we click on the File button on the Menu bar, a drop-down list will appear. It is shown below:



Let's discuss each option in detail.

New

The New button opens the new window. It does not remove the sketch which is already

present.

Open

It allows opening the sketch, which can be browsed from the folders and computer drivers.

Open Recent

The Open Recent button contains the list of the recent sketches.

Sketchbook

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

Examples

It shows the different examples of small projects for a better understanding of the IDE and the board. The IDE provides examples of self-practice.

Close

The Close button closes the window from which the button is clicked.

Save

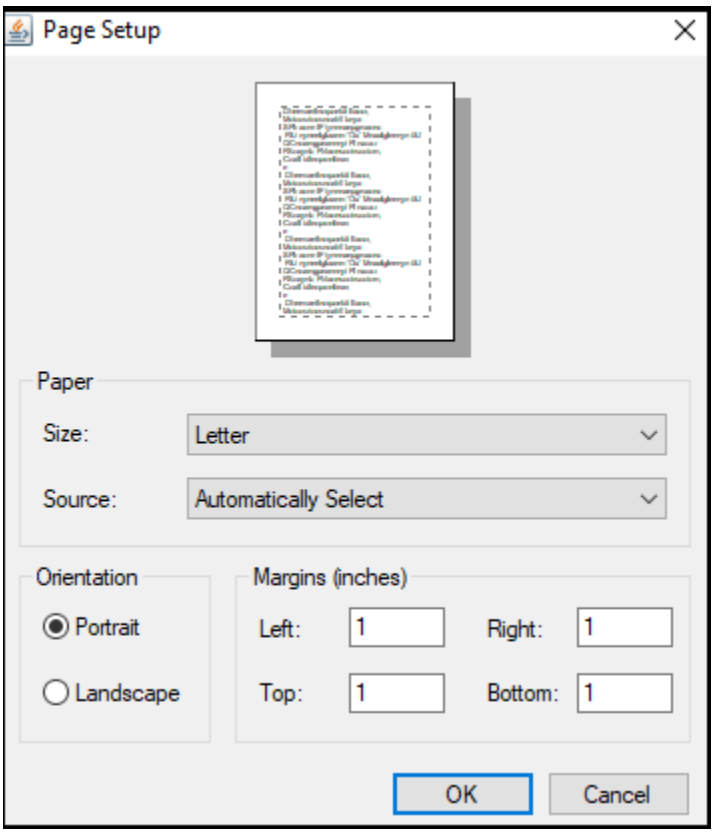
The save button is used to save the current sketch. It also saves the changes made to the current sketch. If we have not specified the name of the file, it will open the '**Save As...**' window.

Save As...

We can save the sketch with a different name using the '**Save As...**' button. We can also change the name accordingly.

Page Setup

It allows setting the page margins, orientation, and size for printing. The '**Page Setup**' window will appear as:



Print

According to the settings specified in the 'Page Setup', it prepares the current sketch for printing.

Preferences

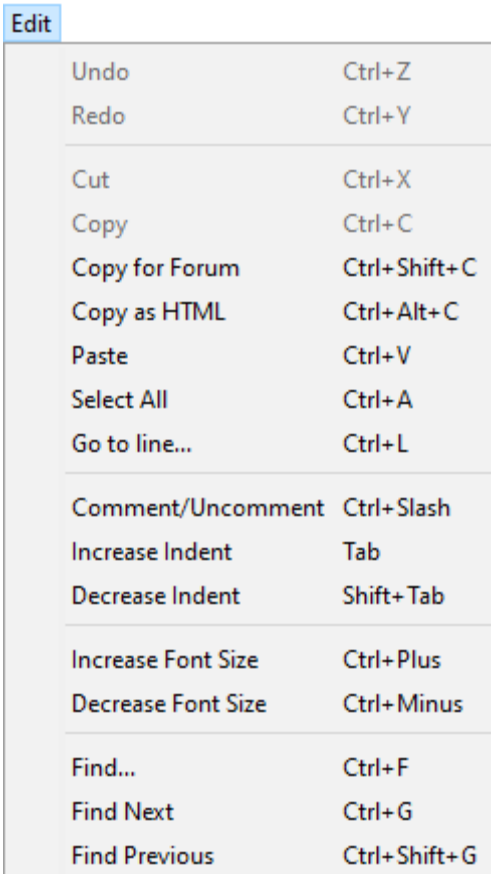
It allows the customization settings of the Arduino IDE.

Quit

The Quit button is used to close all the IDE windows. The same closed sketch will be reopened when we will open the Arduino IDE.

- **Edit**

When we click on the Edit button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

Undo

The Undo button is used to reverse the last modification done to the sketch while editing.

Redo

The Redo button is used to repeat the last modification done to the sketch while editing.

Cut

It allows us to remove the selected text from the written code. The text is further placed to the clipboard. We can also paste that text anywhere in our sketch.

Copy

It creates a duplicate copy of the selected text. The text is further placed on the clipboard.

Copy for Forum

The 'Copy for Forum' button is used to copy the selected text to the clipboard, which is also suitable for posting to the forum.

Copy as HTML

The 'Copy for Forum' button is used to copy the selected text as HTML to the clipboard. It is desirable for embedding in web pages.

Paste

The Paste button is used to paste the selected text of the clipboard to the specified position of the cursor.

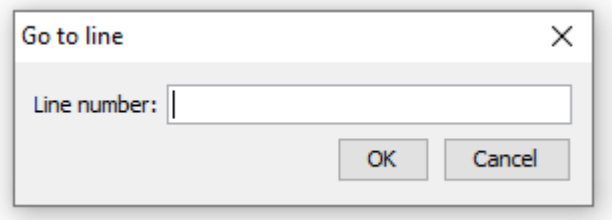
Select All

It selects all the text of the sketch.

Go to line...

It moves the cursor to the specified line number.

The window will appear as:



Comment/Decomment

The Comment/ Decomment button is used to put or remove the comment mark (**//**) at the beginning of the specified line.

Increase Indent

It is used to add the space at the starting of the specified line. The spacing moves the text towards the right.

Decrease Indent

It is used to subtract or remove the space at the starting of the specified line. The spacing moves the text towards the left.

Increase Font Size

It increases the font size of the written text.

Decrease Font Size

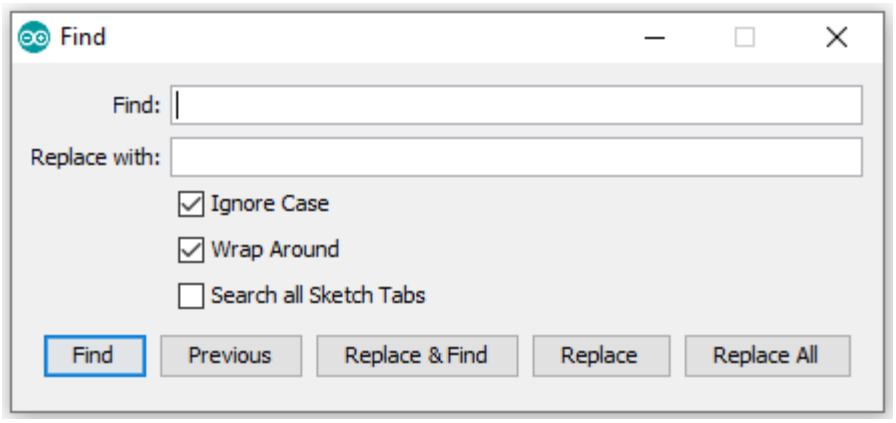
It decreases the font size of the written text.

Find...

It is used to find the specified text. We can also replace the text. It highlights the text in

the sketch.

The window will appear as:



Find Next

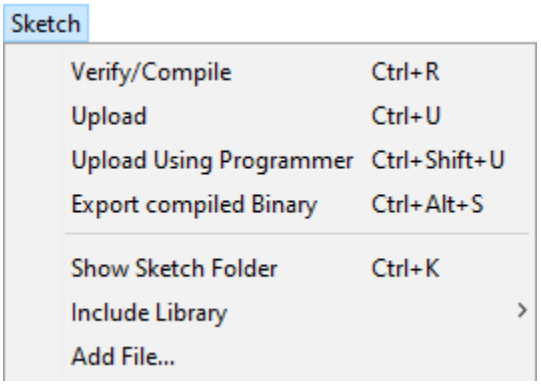
It highlights the next word, which has specified in the '**Find...**' window. If there is no such word, it will not show any highlighted text.

Find Previous

It highlights the previous word, which has specified in the '**Find...**' window. If there is no such word, it will not show any highlighted text.

- **Sketch**

When we click on the Sketch button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

Verify/Compile

It will check for the errors in the code while compiling. The memory in the console area is also reported by the IDE.

Upload

The Upload button is used to configure the code to the specified board through the port.

Upload Using Programmer

It is used to override the Bootloader that is present on the board. We can utilize the full capacity of the Flash memory using the '**Upload Using Programmer**' option. To implement this, we need to restore the Bootloader using the **Tools**-> **Burn Bootloader** option to upload it to the USB serial port.

Export compiled Binary

It allows saving a .**hex** file and can be kept archived. Using other tools, .hex file can also be sent to the board.

Show Sketch Folder

It opens the folder of the current code written or sketch.

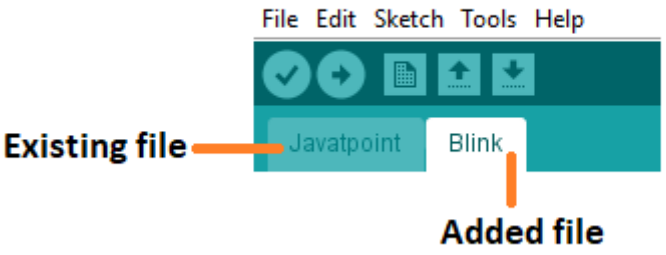
Include Library

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

Add File...

The Add File... button is used to add the created file in a new tab on the existing file.

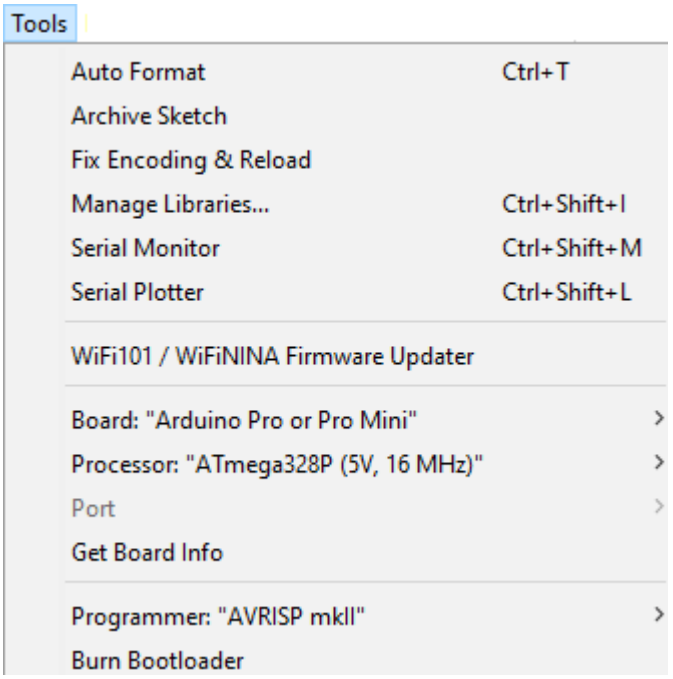
For example, let's add '**Blink**' file to the '**Javatpoint**' file. The tab will now appear as:



We can also delete the corresponding file from the tab by clicking on the **small triangle** -> **Delete** option.

Tools

When we click on the Tools button on the Menu bar, a drop-down list appears. It is shown below:



Let's discuss each option in detail.

Auto Format

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

Archive Sketch

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

Fix Encoding and Reload

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

Manage Libraries...

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

Serial Monitor

It allows the exchange of data with the connected board on the port.

Serial Plotter

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

WiFi101 Firmware Updater

It is used to check and update the Wi-Fi Firmware of the connected board.

Board

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

Processor

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

Port

It consists of the virtual and real serial devices present on our machine.

Get Board Info

It gives the information about the selected board. We need to select the appropriate port before getting information about the board.

Programmer

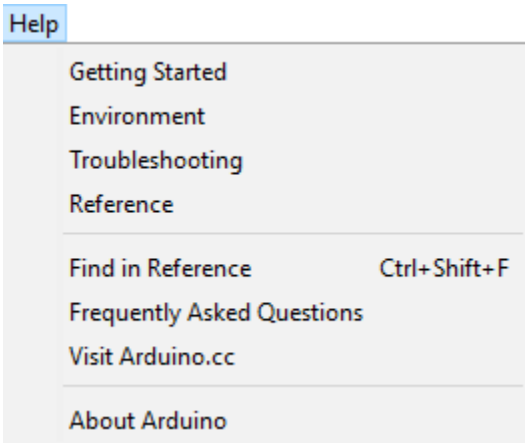
We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.

Burn Bootloader

The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader. Before burning the bootloader, we need to make sure about the correct selected board and port.

Help

When we click on the Help button on the Menu bar, a drop-down list will appear. It is shown below:



The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. We can also consider the image shown above, which includes all the options under the Help section.

Some documents like Getting started, Reference, etc., can be accessed without the internet connection as well. It will directly link us to the official website of Arduino.

7.2Thinker cad

Tinkercad is a really easy 3D CAD program to learn to use, the software has a simple interface and is based around building up a design by using 3D shapes from the library and combining them (by adding or subtracting shapes) to build your model.

Tinkercad is web based and teachers can set up an account and invite students via a join code. This allows students to access their designs out of school and also to share their designs with the global Tinkercad community.

We recommend Tinkercad for primary schools and initial secondary school projects as it is very easy to get started with, we have even had children in a nursery school in our community using Tinkercad with support. However although simple to learn, this does not limit the scope of the software to produce complex designs, so it can be used for all ages and ability levels. However as part of the Autodesk CAD software suite, it is relatively easy for students to progress from Tinkercad

to Autodesk Fusion 360 (or Autodesk Inventor) as their skills and the complexity of their designs develop.

Another benefit of Tinkercad is that it has now evolved to incorporate “circuits” functionality. This allows students to design circuits, program micro-controllers and incorporate the electronics directly into their 3D designs.

Tinkercad Circuits is the easiest way to get your students started with learning electronics. Using our interactive circuit editor, students can explore, connect, and code virtual projects with a bottomless toolbox of simulated components.

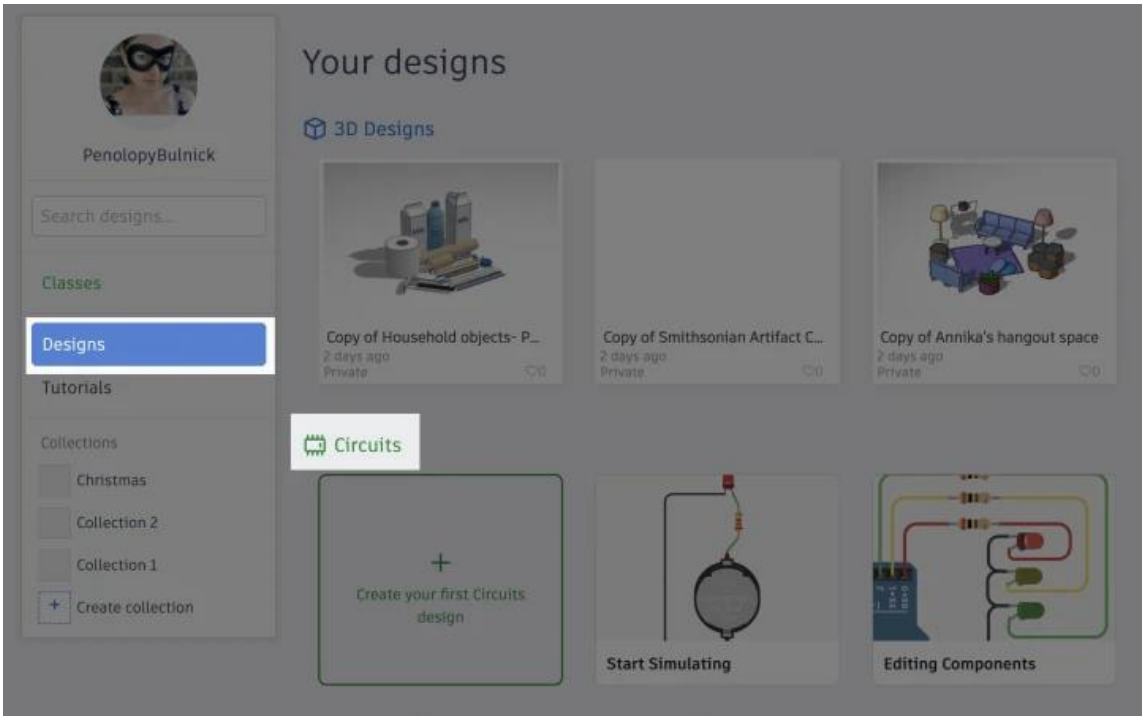
Available in 16 languages, on any computer with an internet connection, Tinkercad Circuits is an unmatched resource for electronics education.

1. Exploring Circuits
2. Learning Basic Circuits
3. Micro:bit in Tinkercad
4. Arduino in Tinkercad
5. Bridging 3D Design and Circuits

6.Teaching Circuits with Tinkercad

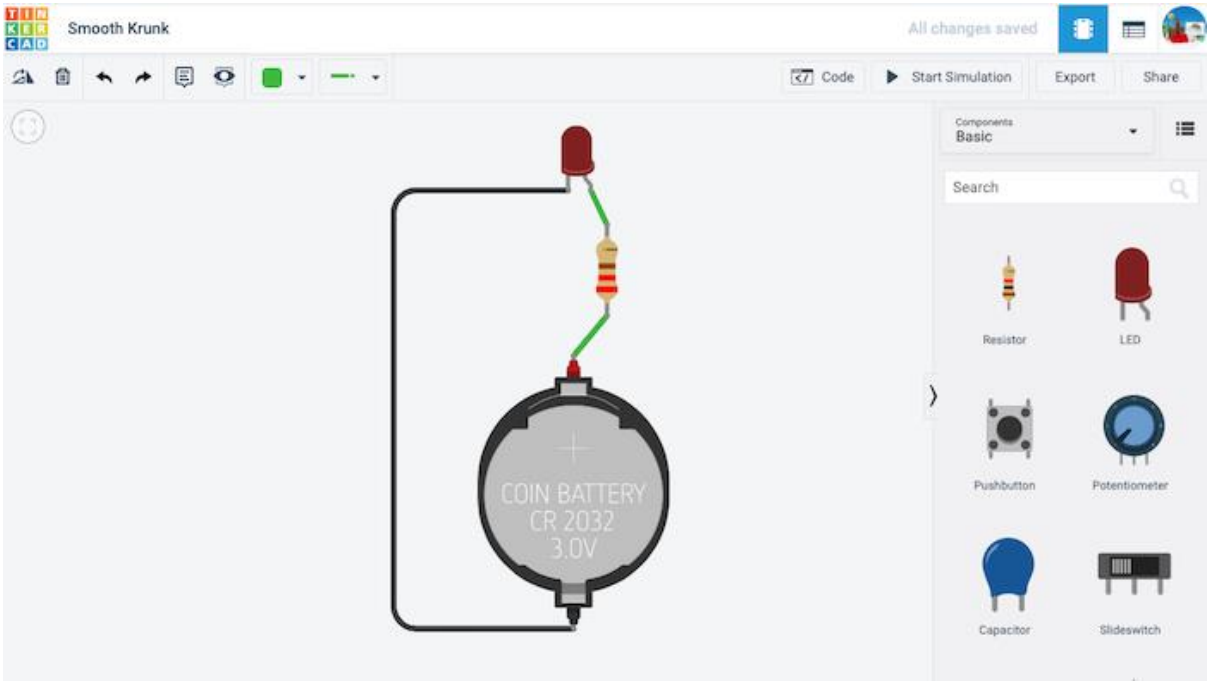
Exploring Circuits

After signing in to Tinkercad, you’ll find a dashboard of your recent designs. You'll find the Circuits section under 3D Designs. If you are a teacher, your dashboard will default to Classes. You can find Circuits, by clicking Designs in the left side.



While on your dashboard, you can scroll through your existing 3D, Codeblocks or Circuits designs. You can also create a new design by clicking the blue + New button in the upper right hand corner of the dashboard and selecting the editor you'd like to open.

Tinkercad's Circuits editor has a similar layout to its 3D design editor. You'll find a large window on the left for creating your design. On the right side you'll see a panel filled with components you can drag and drop into the workspace to create your circuit.



Unlike Tinkercad's 3D design editor, the workspace in Circuits is two-dimensional. You can move your components around by selecting and dragging them, or pan the view around your design by clicking and dragging the empty space around it.

You can also zoom in and out of your design by using the scroll wheel on your mouse, a two-finger gesture on your trackpad, or a key combination of Command + and Command -.

A "Zoom to fit" button is located in the top left corner of the workspace, which will center and zoom your design to fill the window. Pressing the letter F on your keyboard works as a handy shortcut for this same command.

If this is your first time using the Circuits editor, we encourage you to explore the different buttons and options available to you in the menu bar across the top. Hovering your mouse over any of the buttons should reveal an explanation for what it does, as well as any keyboard shortcuts that accomplish the same command.

Likewise, spend a minute browsing some of the other options and menus available in the component panel. By default, the Circuits editor presents you with a selection of the most popular basic components for learning electronics. To access more components, use the dropdown menu to select the All Components view, or search for specific components using the search bar beneath the menu.

You'll also notice that we have more than just components in this menu. Further down, you'll find a selection of Starters. These are pre-made circuit examples that students can drag into the workspace, simulate, edit, and remix.

These Starters fall into four main categories: Basic, Arduino, Micro:bit, and Circuit Assemblies. Every one of our Starters comes to life in some way when the Start Simulation button is pressed.

Basic Starters are made from the kind of common electronic components typically used to introduce students to electronics (LEDs, batteries, hobby motors, resistors, and switches). These examples use no microcontrollers, and no code.

With Arduino Starters, students can see the kind of advanced interactions that are possible with programmable microcontrollers. Each of the Arduino Starters include a code view, which students can directly edit using either a built-in blocks-style interface, a text-based editor, or a combined view.

Our most recent addition are the micro:bit Starters. Similar to the Arduino starters, these are microcontroller projects that use the popular micro:bit educational electronics board. They also include an editable code view.

Finally, we have the Circuit Assemblies Starters. There are only a handful of these and they're directly linked to projects that tie-in both 3D design and basic electronics. These projects include the Glow Circuit, Move Circuit, and Spin Circuit.

If you've exhausted our Starters and you're still hungry for more our Gallery page includes a selection of community contributed designs to spark your inspiration. Alternatively, you can use the Circuits view of our search tool to locate specific designs that may be useful to you.

Learning Basic Circuits

Tinkercad is more than just a platform for exploring electronics. We include a selection of interactive lessons that students can use to learn electronics at their own pace.

The hub for all Tinkercad Circuits instructional content can be found on our Tinker > Circuits page, accessible from the top navigation bar of your design dashboard.

Scrolling down on the Circuits page allows you to dive into lessons with varying degrees of difficulty.

A student's journey into learning electronics with Tinkercad will typically begin with our starter lessons. Here, they'll learn how to use the Circuits editor to simulate projects, edit aspects of individual components, wire elements together, and add new components to their designs.

Then they'll move into our electronics lessons. These focus on fundamentals, such as breadboard wiring, Ohm's law, and series and parallel circuits.

Because many students jump right in to working with micro:bit early-on, we've also included our selection of interactive micro:bit lessons within the Skills section. You can learn more about these

lessons in the next section.

Micro:bit in Tinkercad

The BBC micro:bit is a popular and inexpensive circuit board designed for students to learn electronics and coding. As the newest addition to Tinkercad circuits, it benefits from having our most recent starter examples and lessons.

Micro:bit Starters

The quickest way to start exploring micro:bit in Tinkercad is to drag out one of the example designs from the Starters menu. Each of these designs can be brought to life in some way using the simulation mode, and include code that students can copy, modify, and build on.

Coding micro:bit

To view or edit code on any of our programmable circuits (including Arduino), simply press the Code button (shown below) to toggle the coding tools in and out of view.

Students can code their micro:bit creations using a Scratch-based system of code blocks, similar to the system used on Microsoft's MakeCode website. Coding with blocks is a fantastic way for beginners to visualize the logic and patterns of code.

As you can see from the Starters examples, in spite of its approachability, blocks programming is capable of some relatively sophisticated interactions. We like to call it a "low floor, high ceiling" programming language.

That said, your more advanced or adventurous students have the option of exploring micro:bit coding using the popular scripting language of Python. This option can be found using the dropdown menu above the blocks code tools (shown below).

Students can change this code view from the default Blocks option, to a Blocks + Text (Python) view, or a purely Text view. The Blocks + Text view is perfect for beginners, allowing them to explore coding possibilities using familiar blocks while seeing the same concepts represented in the written language of Python.

Please note: Moving your blocks code to Python is a one-way street. When changing any blocks-based code to a fully text-based option you'll be presented with a warning that the conversion cannot be undone.

Any attempt to revert a design back to a Blocks view of the code will erase the code. We've done our best to make this as clear as possible within the editor to prevent students from accidentally deleting their code. As their instructor, you may want call special attention to this and have students duplicate a backup of their designs before exploring the different code environments.

For more information on our Python integration for micro:bit (including our Python debugging tool) check out our separate blog post titled Python Coding with micro:bit in Tinkercad Circuits.

Debugging micro:bit Code

If your code isn't quite up to snuff, our built-in debugger will let you know. The debugging interface will kick-in automatically if it detects an error in your code.

Another advantage of debugging is that you can add breakpoints in your code. By selecting a line number in your code, the highlighted line marks a point where the code will pause during simulation.

When debugging with break points, two buttons will appear above the code window during simulation. You can use these buttons to either resume the execution of the code after the breakpoint, or step to the next line of code.

Adding breakpoints is also a useful way to reveal variable values while simulating your project.

The example above shows the Compass project located in our Starters menu. When the simulation is paused at a breakpoint anywhere in the code you can hover over a variable (in this case "angle") and see its value at that moment in time. For this example, we can play around with the board's orientation and check to see if the math accurately indicates the direction.

Micro:bit Lessons

For a more structured approach to learning the fundamentals of micro:bit, students can work through our self-paced, interactive lessons.

There are five distinct lessons within this project. Similar to our interactive lessons for basic circuits and Arduino, students are provided with a sidebar outlining the instructions they need to follow to complete each task.

If you're using Tinkercad Classrooms to onboard and manage your student experience with Tinkercad, you'll be able to see a student's completed lessons in the Lessons section of their design gallery (see below), or within the Designs section of your class dashboard.

Arduino in Tinkercad

For over 15 years, the Arduino ecosystem of microcontroller boards and code libraries have become an indispensable part of electronics education. But if you've ever worked with a student to connect an Arduino board to their computer, download Arduino's IDE software, update the libraries, and configure the ports, you know that it can be a bumpy road.

The Arduino simulation within Tinkercad simplifies the learning experience. It's free, works on any computer with an Internet connection, and scales to any class size. Best of all, Tinkercad Circuits provides a bottomless supply of virtual components that students can use to build and simulate their projects.

When they're ready to physically prototype their projects, Tinkercad Circuits makes it easy to export their code as a native Arduino (.ino) file that they can upload to their board.

Arduino Starters

One of the fastest ways to explore the possibilities of Arduino in Tinkercad is to simply drag one of our sample Starter circuits into your workspace.

We have nearly two dozen Arduino Starter circuits to choose from. Each example includes sample code that you can view, simulate, and modify.

Coding Arduino in Tinkercad

Tinkercad allows you to code your Arduino using two different approaches.

Our Blocks code editor offers beginners a visual system of functions that they can drag and rearrange. All of our Arduino Starters, and most of our interactive Arduino lessons will include or refer to Blocks code.

Part of the magic of learning to code Arduino in Tinkercad is that our editor will automatically generate text-based code (C++) from students' blocks code. By switching the code view to Blocks + Text, students can see the logic of their blocks code translated to C++ code.

Modifications made to their blocks code will instantly update in the text view, providing insight into the logic and syntax of C++.

Of course, once students are ready to create their code directly in the text editor, they can switch to a pure text view. This view offers an experience similar to programming with Arduino's IDE editor.

Using Arduino Code Libraries

Tinkercad's Arduino text editor includes eleven built-in libraries. You can view and add these libraries to your project by selecting the file box icon above your code (show below).

The included libraries represent some of the most popular and common libraries used in Arduino. It is possible, though, to run code that requires Arduino libraries beyond the included examples.

When you open an Arduino library source file (.c or .cpp) you'll find that it is simply a clipping of C++ Arduino code. By copying and pasting this library clipping into the appropriate sections of your Arduino code, you may be able to effectively make it work.

Your mileage may vary. Libraries are oftentimes made to adapt Arduino to specialized hardware or shields. If these hardware components aren't included in Tinkercad, no amount of code editing will make the project work.

Debugging Arduino Code

One of the biggest challenges in learning to code is learning how to troubleshoot problems when the code isn't working as planned. When a student's code doesn't work, Tinkercad's error console will automatically pop into view. Similar to how errors are reported in the Arduino IDE, this error console will report and highlight the suspected issues that need fixing.

Tinkercad's Arduino debugger also includes a unique and useful code break feature. By selecting

lines of code, you can define moments within your code where you'd like the simulation to pause. These pauses will happen automatically during simulation, allowing students to read values and troubleshoot problems. A button above their code (shown below) resumes the simulation or advances to the next break.

Arduino Lessons

Our selection of Arduino lessons (found in the Circuits Learning Center) provide students with an interactive, self-paced system for learning how to work with electronics using Arduino.

As students open these lessons, they'll find a sidebar on the left with instructions on how to complete their design.

There are 28 lessons available at this time. The last 15 are taken from the Arduino Projects Book by Scott Fitzgerald and Michael Shiloh. It is the book that Arduino includes in their official Starter Kit. Companion videos for these lessons can be found on Arduino's Remote Learning Playlist.

If you're using Tinkercad Classrooms to onboard and manage your student experience with Tinkercad, you'll be able to see a student's completed lessons in the Lessons section of their design gallery, or within the Designs section of your class dashboard.

Arduino Lesson Plan

Program an LED Light Show is free, standards-aligned lesson plan developed by Dr. Ben Finio of Cornell University's Sibley School of Mechanical and Aerospace Engineering. In it, students learn the basics of building a circuit and programming an Arduino to control it.

The lesson plan comes complete with video resources and PowerPoint slides and is geared towards students in grades 6-12 (US).

Bridging 3D Design and Circuits

Because Tinkercad works as both an electronics editor, and a 3D design tool, naturally there are opportunities to bridge these two capabilities.

On Instructables, you can find several classroom-ready activities that weave together basic

electronics, 3D design and 3D printing. These activities include the Glow Circuit, Move Circuit, and Spin Circuit.

For an Arduino-based activity that combines 3D design and electronics, check out this guide for creating a LED Ring Butterfly.

In each of these examples, students will find pre-built 3D components in Tinkercad's 3D editor, under the Circuits Assemblies menu. The Circuits Components menu compliments these designs with accurate 3D representations of popular components.

General Tips

Keyboard Shortcuts

Within the Circuits workspace there are some handy keyboard shortcuts worth knowing.

F - Zoom to fit

N - Create note

Shift + N - Hide/reveal notes

R - Rotate

Additionally, you can quickly change the color of a wire by selecting it and pressing any number key (0-9).

By holding shift you can select multiple items in the workspace to move or delete.

Annotation

Both educators and students can take advantage of the Notes tool for explaining designs or provisioning feedback. The Notes feature can be found in the toolbar above the Circuits workspace (or activated using the N key on your keyboard).

Once a note has been placed on a design, its placement can be further adjusted to prevent it from obscuring components. If too many notes are cluttering a design, their visibility can be toggled on

and off using the button with the eye-shaped icon (shown above).

Sharing Designs

By default, anything you or your students design in Tinkercad is considered private. Private designs can't be shared between users and they will not appear in Tinkercad Gallery search results.

To share a Tinkercad Circuits design with your students you must first make it public. It's an easy change to make using the design's Properties window (shown below). We also have a step-by-step guide on making and sharing public designs in Tinkercad.

For a teacher to view a student's design, a student must have an individual account (in which case they can follow the steps outlined in the link above), or be a part of a Tinkercad Classrooms account that you manage.

Public designs also include HTML code that you can use to embed your designs on a website (just like the above example). This can come in handy for including designs within a course overview page for your class, or as part of an Instructables guide.

CONCLUTION AND FUTURE ENHANCEMENT

FUTURE ENHANCEMENT

Future Enhancements for Arduino-Based Electronic Voting Machines

1. **Enhanced Security Features**: Implementing advanced cryptographic protocols and secure communication channels can bolster the security of Arduino-based electronic voting machines (EVMs). Future enhancements could focus on incorporating features such as end-to-end encryption, digital signatures, and blockchain technology to ensure the integrity and confidentiality of voting data.

2. **Biometric Authentication**: Integrating biometric authentication mechanisms, such as

fingerprint or iris scanning, can enhance voter verification and prevent fraudulent voting in Arduino-based EVMs. Biometric data could be securely stored and matched against voter records to ensure that only eligible individuals are allowed to cast their votes.

3. **Accessibility Improvements**: Further enhancing the accessibility features of Arduino-based EVMs can ensure inclusivity for voters with disabilities. Future enhancements could include audio feedback for visually impaired voters, tactile interfaces for users with mobility impairments, and multi-language support to accommodate diverse linguistic preferences.

4. **Remote and Mobile Voting**: Exploring the feasibility of remote and mobile voting capabilities in Arduino-based EVMs can expand access to the electoral process and promote voter participation. Future enhancements could involve the development of secure mobile applications or web-based interfaces that allow voters to cast their ballots from anywhere using their smartphones or computers.

5. **Real-Time Monitoring and Reporting**: Implementing real-time monitoring and reporting functionalities in Arduino-based EVMs can enable election officials to track voting activity, detect anomalies, and generate comprehensive reports instantaneously. Future enhancements could involve integrating network connectivity and data analytics capabilities to facilitate real-time data collection and analysis during elections.

6. **Redundancy and Failover Mechanisms**: Implementing redundancy and failover mechanisms can ensure the reliability and fault tolerance of Arduino-based EVMs, particularly in the event of hardware or software failures. Future enhancements could include redundant power supplies, backup storage solutions, and automatic failover procedures to minimize disruptions and safeguard against data loss.

7. **Usability and User Experience Refinements**: Continuously improving the usability and user experience of Arduino-based EVMs can enhance voter satisfaction and confidence in the electoral

process. Future enhancements could focus on refining the graphical user interface, streamlining the voting workflow, and incorporating user feedback to iteratively optimize the design of EVMs.

8. **Open-Source Collaboration and Community Engagement**: Encouraging open-source collaboration and community engagement can foster innovation and drive continuous improvement in Arduino-based EVMs. Future enhancements could involve establishing collaborative platforms, organizing hackathons or competitions, and providing resources and support for developers to contribute to the development of EVM software and hardware.

By focusing on these future enhancements, Arduino-based electronic voting machines can evolve into more secure, accessible, and reliable tools for facilitating democratic elections, ultimately strengthening trust and confidence in the electoral process.

CONCLUSION

The Voting Machine project demonstrates a basic implementation of an electronic voting system using Arduino. It can be further expanded with features such as voter authentication, multiple candidate support, and result storage for more sophisticated applications.

REFERENCES

1.Arduino Documentation:

- The official Arduino website (https://www.arduino.cc/) provides extensive documentation, tutorials, and reference materials for learning about Arduino boards, programming, and interfacing with various components.

2.LiquidCrystal Library Documentation:

- The Arduino LiquidCrystal library reference (https://www.arduino.cc/en/Reference/LiquidCrystal) offers detailed information on how to use the library for interfacing with LCD displays.

3.Electronics Tutorials:

- Websites like SparkFun (https://learn.sparkfun.com/tutorials), Adafruit (https://learn.adafruit.com/), and Electronics Hub (https://www.electronicshub.org/) offer tutorials and guides on electronics components, circuits, and projects.

4.Books on Arduino and Electronics:

- Books such as "Arduino Cookbook" by Michael Margolis and "Getting Started with Arduino" by Massimo Banzi and Michael Shiloh cover Arduino basics, programming, and various projects.

5.Online Forums and Communities:

- Forums like the Arduino Forum (https://forum.arduino.cc/) and subreddits like r/arduino on Reddit are great places to ask questions, share projects, and learn from the community.

6.Datasheets and Manufacturer Websites:

- Refer to datasheets for specific components used in the project, such as the LCD display, push buttons, buzzer, and LED. Manufacturer websites often provide detailed specifications and application notes for their products.When citing these sources or any other relevant resources, make sure to follow the citation style guidelines appropriate for your context, such as APA or MLA. If you require a formal reference list, you can list the sources mentioned above along with any additional resources you consulted during the development of your project.

APPENDIX

CODE

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define sw1 15

#define sw2 16

#define sw3 17
```

```
#define sw4 18
#define sw5 19

#define buzzerPin 7  // Define the pin for the buzzer
#define ledPin 6     // Define the pin for the LED

int vote1 = 0;
int vote2 = 0;
int vote3 = 0;
int vote4 = 0;

void setup() {
  pinMode(sw1, INPUT_PULLUP);
  pinMode(sw2, INPUT_PULLUP);
  pinMode(sw3, INPUT_PULLUP);
  pinMode(sw4, INPUT_PULLUP);
  pinMode(sw5, INPUT_PULLUP);

  pinMode(buzzerPin, OUTPUT); // Set the buzzer pin as output
  pinMode(ledPin, OUTPUT);    // Set the LED pin as output

  lcd.begin(16, 2);
  lcd.print("Voting Machine");
  lcd.setCursor(0, 1);
```

```
  lcd.print("Circuit Digest");
  delay(3000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("BJP");
  lcd.setCursor(4, 0);
  lcd.print("INC");
  lcd.setCursor(8, 0);
  lcd.print("AAP");
  lcd.setCursor(12, 0);
  lcd.print("OTH");
}

void loop() {
  lcd.setCursor(1, 1);
  lcd.print(vote1);
  lcd.setCursor(5, 1);
  lcd.print(vote2);
  lcd.setCursor(9, 1);
  lcd.print(vote3);
  lcd.setCursor(13, 1);
  lcd.print(vote4);
```

```
 if (digitalRead(sw1) == LOW) {
   vote1++;
   digitalWrite(ledPin, HIGH); // Turn on LED when button is pressed
   delay(200); // Debounce delay
   digitalWrite(ledPin, LOW); // Turn off LED after debounce
 }

 if (digitalRead(sw2) == LOW) {
   vote2++;
   digitalWrite(ledPin, HIGH); // Turn on LED when button is pressed
   delay(200); // Debounce delay
   digitalWrite(ledPin, LOW); // Turn off LED after debounce
 }

 if (digitalRead(sw3) == LOW) {
   vote3++;
   digitalWrite(ledPin, HIGH); // Turn on LED when button is pressed
   delay(200); // Debounce delay
   digitalWrite(ledPin, LOW); // Turn off LED after debounce
 }

 if (digitalRead(sw4) == LOW) {
   vote4++;
   digitalWrite(ledPin, HIGH); // Turn on LED when button is pressed
```

```
  delay(200); // Debounce delay
  digitalWrite(ledPin, LOW); // Turn off LED after debounce
 }

 if (digitalRead(sw5) == LOW) {
  int totalVotes = vote1 + vote2 + vote3 + vote4;
  if (totalVotes > 0) {
   lcd.clear();
   if (vote1 > vote2 && vote1 > vote3 && vote1 > vote4) {
    lcd.print("BJP Wins");
    tone(buzzerPin, 1000, 1000); // Play a 1-second tone with a frequency of 1000 Hz
   } else if (vote2 > vote1 && vote2 > vote3 && vote2 > vote4) {
    lcd.print("INC Wins");
    tone(buzzerPin, 1000, 1000); // Play a 1-second tone with a frequency of 1000 Hz
   } else if (vote3 > vote1 && vote3 > vote2 && vote3 > vote4) {
    lcd.print("AAP Wins");
    tone(buzzerPin, 1000, 1000); // Play a 1-second tone with a frequency of 1000 Hz
   } else if (vote4 > vote1 && vote4 > vote2 && vote4 > vote3) {
    lcd.print("OTH Wins");
    tone(buzzerPin, 1000, 1000); // Play a 1-second tone with a frequency of 1000 Hz
   } else {
    lcd.print("Tie Up Or No Result");
    tone(buzzerPin, 500, 1000); // Play a 1-second tone with a frequency of 500 Hz
   }
```

```
    } else {
      lcd.clear();
      lcd.print("No Voting....");
    }
    delay(2000); // Display the result for 2 seconds
    noTone(buzzerPin); // Stop the buzzer
    vote1 = 0;
    vote2 = 0;
    vote3 = 0;
    vote4 = 0;
    lcd.clear();
  }
}
```

Code Explanation

The Arduino code handles the functionality of the voting machine. Here's a brief explanation of the main sections:

Setup

- Initializes the LCD display and sets up the push button pins as inputs with pull-up resistors.
- Displays an introductory message on the LCD.

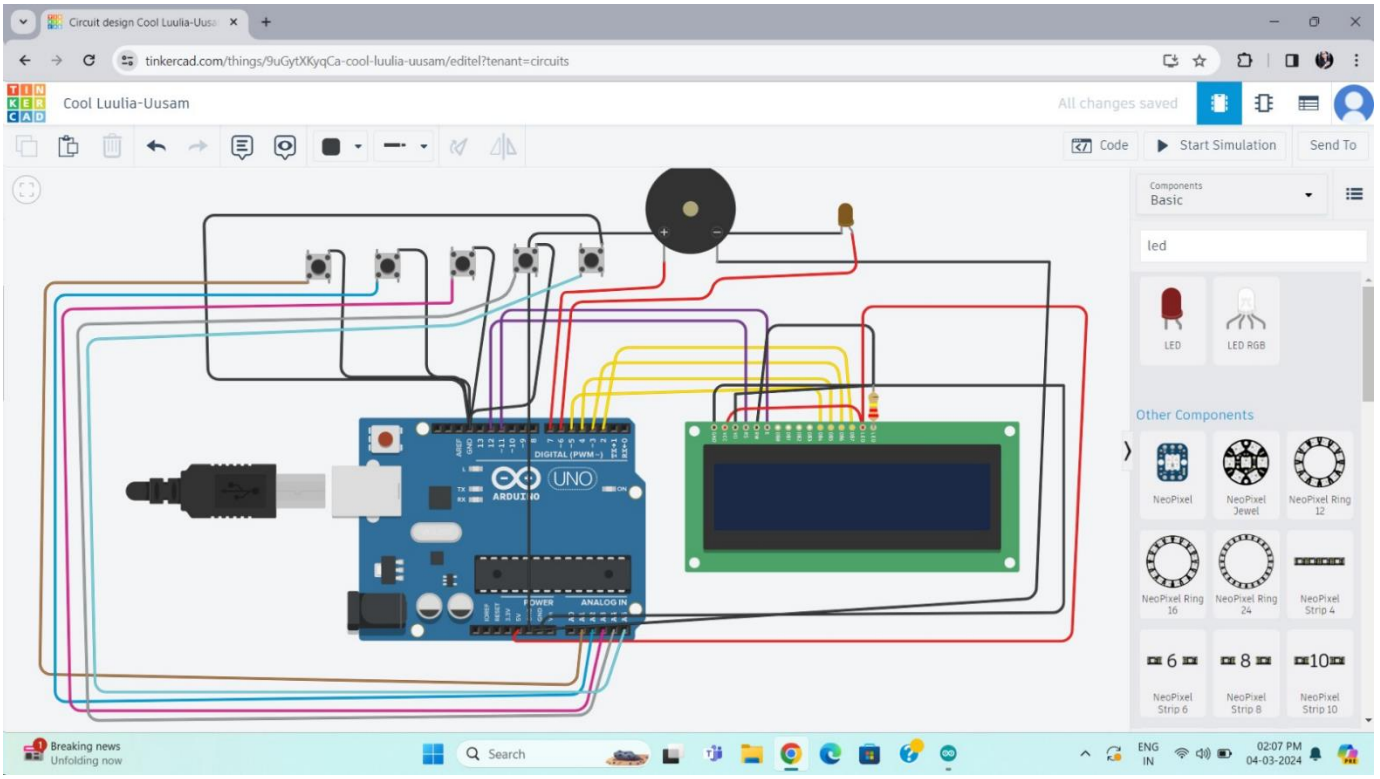
Loop

- Continuously checks the state of the push buttons.

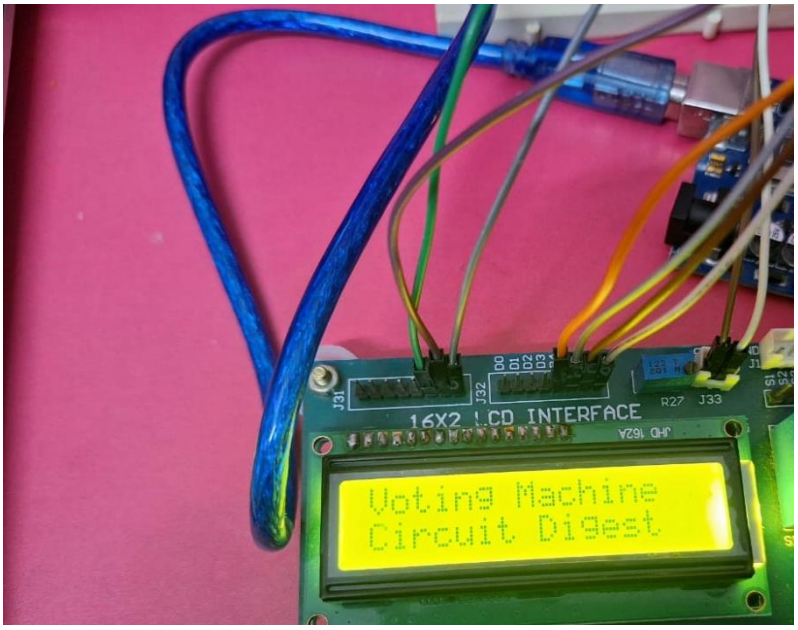
- If a button is pressed, the corresponding vote count is incremented, and the LED lights up briefly for visual feedback.
- When the fifth button is pressed (indicating the end of voting), the total votes are calculated, and the winner is displayed on the LCD with sound feedback from the buzzer.
- The LCD is then cleared, and the vote counts are reset for the next round of voting.

OUTPUT

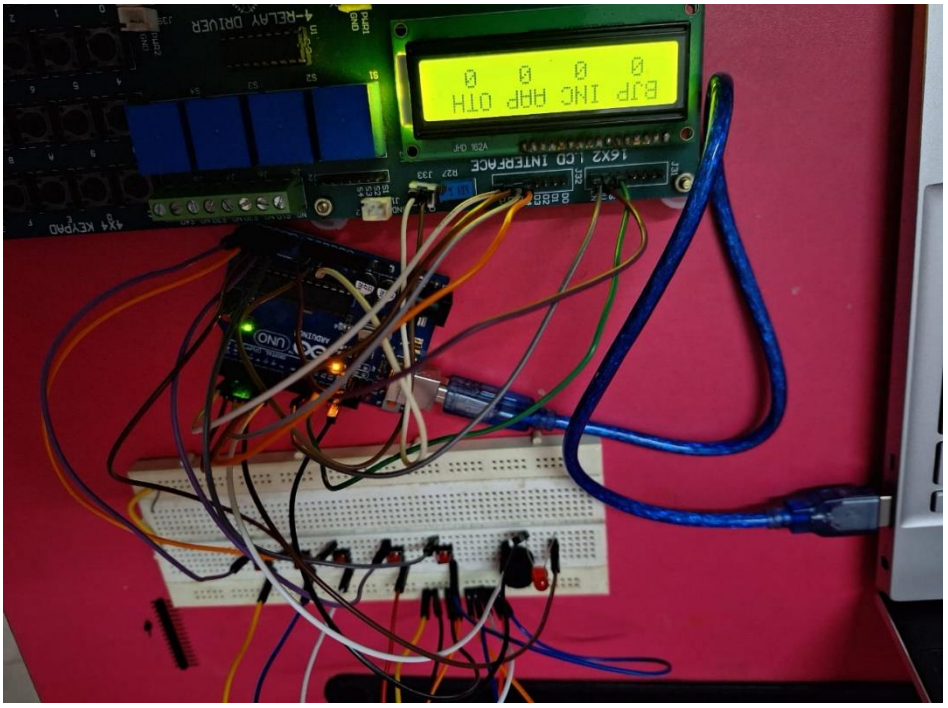
CIRCUIT CONNECTIONS


Circuit design Cool Luulia-Uusa
tinkercad.com/things/9uGytXKyqCa-cool-luulia-uusam/editel?tenant=circuits
Cool Luulia-Uusam
All changes saved
Code
Start Simulation
Send To
Components
Basic
led
LED
LED RGB
Other Components
NeoPixel
NeoPixel Jewel
NeoPixel Ring 12
NeoPixel Ring 16
NeoPixel Ring 24
NeoPixel Strip 4
NeoPixel Strip 6
NeoPixel Strip 8
NeoPixel Strip 10
Breaking news
Unfolding now
Search
ENG
IN
02:07 PM
04-03-2024

RESULTS



Before Voting



Casting the vote

Result announcement