BACHELOR OF TECHNOLOGY DESIGN LAB REPORT

Voice Controlled Home Automation with Arduino

Submitted as part of the coursework for the
Bachelor of Technology (B.Tech) program

Under the supervision of

Prof. Parijat Bhowmick

and guided by

Prof. Anirban Dasgupta

By:

ISHA ROY	**Roll No.: 220102042**
NITISH KUMAR SINGH	**Roll No.: 220102067**

Department of Electronics and Electrical Engineering
Indian Institute of Technology
Guwahati, Assam, India

ABSTRACT

This project demonstrates a voice-controlled home automation system developed using an Arduino Uno microcontroller and the VC-02 offline voice recognition module. The system provides an accessible interface for controlling home appliances through voice commands, particularly benefiting elderly and disabled individuals. Unlike cloud-based solutions, this offline approach ensures privacy, faster response times, and reliability in environments with limited internet connectivity. The VC-02 module processes voice inputs locally, converting recognized commands into hexadecimal codes transmitted via UART serial communication to the Arduino. The microcontroller interprets these codes to control connected devices such as lights and fans through relay circuits. Key hardware components include the Arduino Uno for processing, VC-02 for voice recognition, and relays for appliance control, while the SoftwareSerial library facilitates communication between modules. The system's offline capability eliminates dependency on internet services, making it cost-effective and suitable for diverse environments. Practical applications span smart homes, accessibility solutions, and industrial automation. Testing confirmed reliable command recognition and device control with response times under one second. Future enhancements may include expanded device support, mobile app integration, and energy monitoring features. This project highlights the potential of offline voice recognition technology in creating practical, user-friendly automation systems that balance functionality with privacy and accessibility.

Contents

INTRODUCTION

What is this project ?

- This project is all about controlling your home appliances and systems by using your voice as input, for the use of disabled and elderly.
- Automation has a major role in our day-to-day lives.
- Home automation allows us to control electrical appliances like ac, fan, etc.
- It can also provide home security and emergency systems too.
- It may also control home security systems such as access control and alarm systems.

Why we need to automate ?

- The main objective of **home automation** is to provide assistance to **handicapped** and **elderly individuals**, enabling them to **easily control household appliances** such as **fans** and **lights** using simple **voice commands**. This system can also play a vital role in **alerting them during critical or emergency situations**, ensuring enhanced **safety** and **convenience**.
- Other important objectives of home automation include:
  - Enhancing the **comfort of living** by automating routine tasks like switching devices on or off without manual effort.
  - Providing **better centralized control** over multiple **electronic devices** through a **single platform** such as a smartphone or a voice assistant.
  - Creating a **more organized**, **efficient**, and **intelligent environment** in homes by using technology to save **time** and **energy**, and improving the overall **quality of life**.

Pros of Having an Automated Home

- **Support for the older generation:** Home automation provides much-needed **assistance to elderly individuals** by simplifying control over daily tasks and devices, reducing the need for physical effort and ensuring safety through alerts and automatic responses.
- **Increased energy efficiency:** Smart systems can intelligently **monitor and reduce energy usage** by turning off devices when not in use, optimizing heating/-cooling schedules, and providing insights into energy consumption patterns.
- **Flexibility for new devices:** Automated homes are designed to be **easily adaptable to future technologies**, allowing users to integrate new appliances, sensors, and systems without significant reconfiguration.
- **Smart homes may be suitable for disabled persons:** Automation enables **people with disabilities** to gain independence by allowing them to control lights, fans, doors, and even emergency responses through voice or app-based interfaces.
- **Increasing luxury:** Automated homes add a layer of **comfort and convenience**, allowing users to personalize lighting, ambiance, entertainment systems, and more, creating a seamless and luxurious living environment.
- **Full control over all smart appliances with only one device:** Users can manage and monitor their entire home setup using a **single control interface**—be it a smartphone, tablet, or voice assistant—making the entire system user-friendly and centralized.

COMPONENTS

- **Arduino Uno**
- **VC-02 Voice Recognition Module**
- **LED** (connected to digital pin 5)
- **Fan or Motor** (connected to digital pin 6)
- **Jumper Wires**
- **Power Supply**

2.1 Arduino Uno

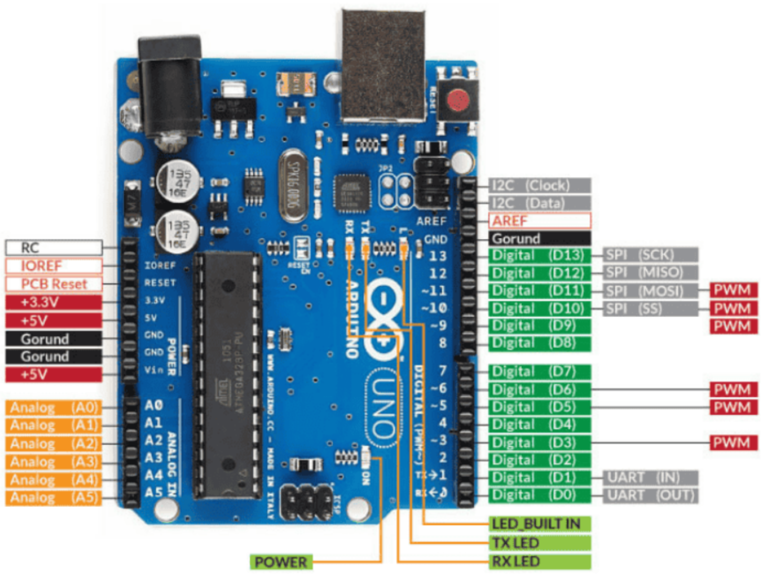


Figure 1: Arduino UNO R3 Pinout Diagram

Overview

The **Arduino UNO** pin layout is organized into distinct categories, including **Power Pins**, **Digital Pins**, **Analog Pins**, and **Special Function Pins**. Each category plays a specific role in enabling the functionality of the board. The various pinouts available under each category are discussed below.

Note: Refer to an Arduino UNO R3 board image with pin labels for better understanding.

Power Pins

Power pins are essential for operating the board and any connected devices. The main power pins include:

- **VIN:** Accepts external power input (typically 7–12V).
- **5V and 3.3V:** Provide regulated voltage output for external peripherals.
- **GND (Ground):** Common ground reference to complete the circuit.
- **IOREF:** Supplies a voltage reference for I/O pins.

Tip: Always verify voltage compatibility of connected components to avoid damage.

Digital Pins (0–13)

The Arduino UNO includes **14 digital pins**, which can act as either input or output pins. Functions such as `pinMode()`, `digitalWrite()`, and `digitalRead()` are used to interface with these pins.

- **Pins 0 (RX) and 1 (TX):** Reserved for UART serial communication.
- **Pins 2–13:** General-purpose digital I/O pins.
- **PWM Pins (3, 5, 6, 9, 10, 11):** Support Pulse Width Modulation for controlling brightness of LEDs, motor speed, etc.

Analog Pins (A0–A5)

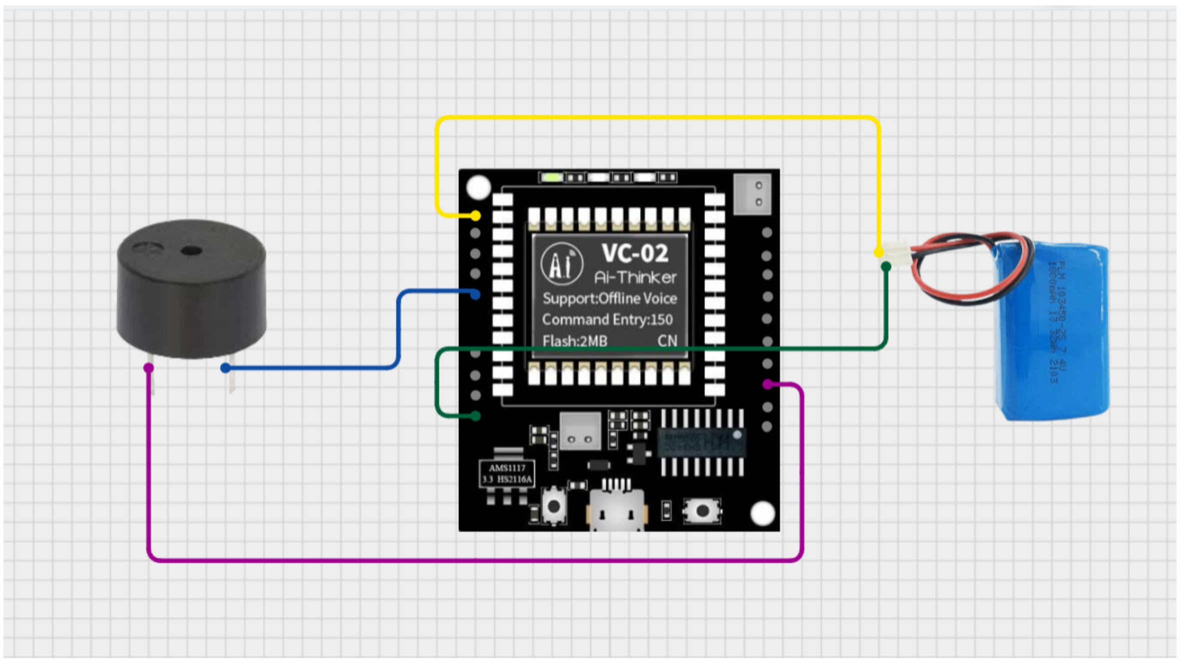
These pins allow for reading analog signals, often used with sensors.

- **Resolution:** 10-bit (values range from 0 to 1023).
- **Flexibility:** These pins can also function as digital I/O pins when necessary.

Special Function Pins

- **Reset:** Resets the microcontroller when triggered.
- **AREF:** Allows an external voltage reference for analog inputs.
- **Serial Pins (RX, TX):** Facilitate UART communication for serial data exchange.

2.2 Voice Recognition Module



Overview

- VC-02 is a compact offline voice recognition module suitable for embedded systems.
- It supports pre-trained English voice commands without requiring internet.
- Commonly used in DIY electronics, robotics, and automation systems.

Power and Voltage

- Operates between 3.3V and 5V, compatible with most microcontrollers.
- Low power consumption of less than 50mA.
- Suitable for battery-powered and portable projects.

Communication Interface

- Uses UART (Universal Asynchronous Receiver/Transmitter) for serial communication.
- Default baud rate is 9600 bps, compatible with standard microcontrollers.

Voice Command Recognition

- Supports up to 80 offline voice commands.
- Fast recognition speed with response time less than 1 second.
- Effective within a 2-meter distance.

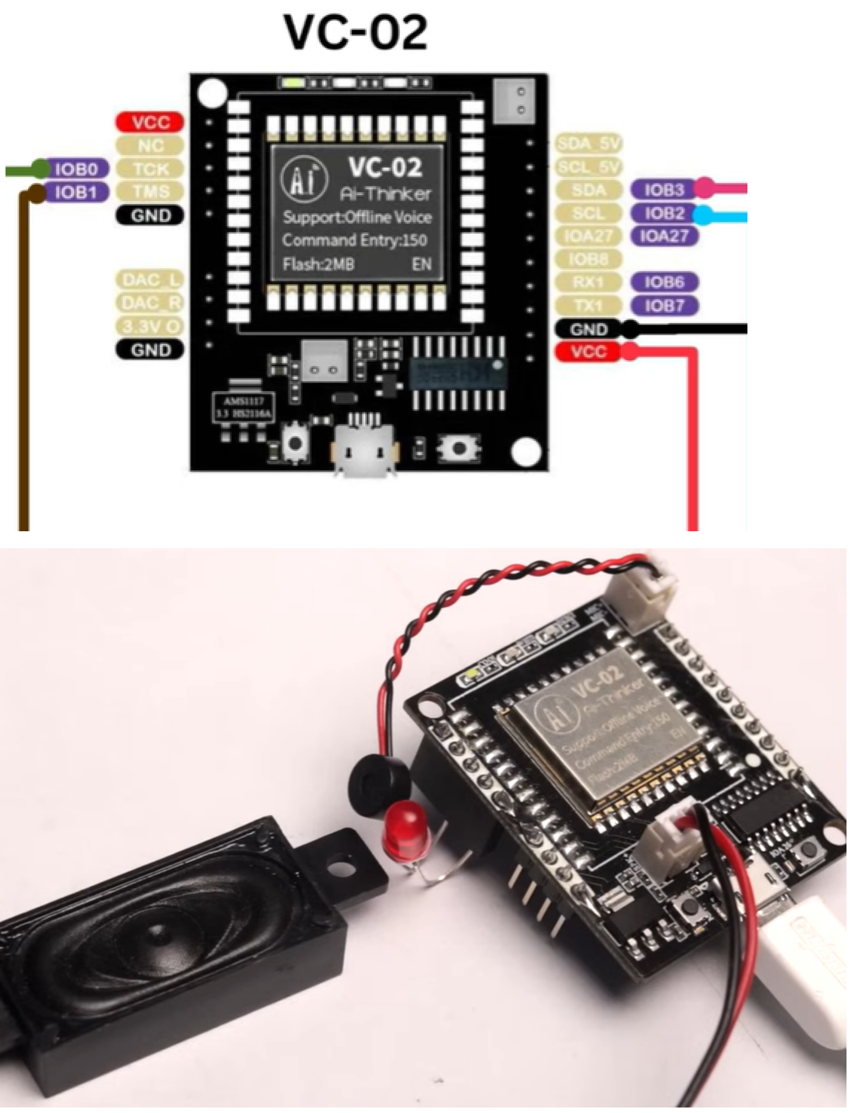


Figure 2: VC-02 module

Pin Configuration and Functions

- **VCC** – Power input (3.3V–5V)
- **GND** – Ground
- **TX** – Transmit data to microcontroller (connect to RX)
- **RX** – Receive data from microcontroller (connect to TX)
- **MIC+ / MIC-** – External microphone input terminals

Physical Features

- Compact dimensions: 30mm × 20mm × 5mm
- Lightweight and ideal for embedded use

Usage Guidelines

- Ensure correct wiring of TX/RX and power pins.
- Place microphone in a quiet environment for better recognition.
- Train voice commands clearly with proper spacing.
- Avoid exceeding voltage range to prevent damage.
- Set microcontroller's UART baud rate to 9600 bps.

Application Areas

- Voice-controlled home automation
- Human-machine interaction without touch
- DIY electronics and robotics
- Assistive technology for people with disabilities

2.3 Serial Communication in Arduino Using SoftwareSerial Library

Introduction

Serial communication is a method that allows the Arduino board to talk to other devices, like a computer or another Arduino. This is done by sending and receiving data one bit at a time through special pins. Most Arduino boards have two dedicated pins for serial communication:

- **Pin 0 (RX)** – used to receive data
- **Pin 1 (TX)** – used to send data

This built-in communication is known as UART or TTL Serial Communication.

Why Use the SoftwareSerial Library?

By default, Arduino only has one serial port. But sometimes, you may want to connect multiple serial devices at the same time (like a Bluetooth module and a GPS sensor). In this case, you can use the **SoftwareSerial** library.

The SoftwareSerial library allows you to use other digital pins as extra serial communication ports. This gives your Arduino the ability to send and receive serial data on pins other than 0 and 1.

Important Terms in SoftwareSerial Library

Here are some important terms and functions used in the SoftwareSerial library:

- **SoftwareSerial Object:** This is the virtual serial port you create by choosing which digital pins will be used for receiving and sending data.

- **Begin Communication:** You must start the communication by setting a *baud rate* (data speed) for the SoftwareSerial port.
- **Read:** This function allows the Arduino to read incoming data on the SoftwareSerial port.
- **Write:** This function is used to send raw data from the Arduino to another device through the SoftwareSerial port.
- **Print and Println:** These are used to send readable text messages through the SoftwareSerial port, similar to writing to the Serial Monitor.
- **Available:** This checks if any data has been received and is ready to be read.
- **Baud Rate:** This is the speed at which data is sent over a serial line. A common baud rate is 9600 bits per second.

Using SoftwareSerial in Projects

When building Arduino projects that need to talk to more than one serial device at a time, the SoftwareSerial library is very useful. For example, you can:

- Use the built-in serial (pins 0 and 1) to connect your Arduino to a computer.
- Use SoftwareSerial (like pins 10 and 11) to connect a Bluetooth module or other device.

Simulation and Testing

You can test your projects using simulation software like **Proteus**. Once you write and upload your code in the Arduino IDE, a HEX file is created automatically. This HEX file can be imported into Proteus to simulate your circuit without needing actual hardware.

Conclusion

The **SoftwareSerial** library is a powerful tool that allows your Arduino to communicate with multiple serial devices at once. It's simple to use and very helpful in expanding the communication abilities of your project.

WORKING PRINCIPLE

Introduction

This project focuses on creating a low-cost, efficient, and offline-capable voice-controlled home automation system. Using the VC-02 AI voice recognition module and Arduino Uno, users can operate electrical appliances like fans and lights through simple voice commands. The system operates entirely offline, removing the need for internet connectivity, ensuring faster response time and improved privacy.

The core objective of this project is to enable users to control home appliances using voice commands through offline processing. The working principle revolves around three main stages: voice command recognition, hexadecimal code mapping, and device control logic implemented using the Arduino platform.

3.1 Voice Command Recognition

The VC-02 AI voice recognition module is programmed to recognize specific spoken commands. When the user speaks a predefined command like "Turn on fan", the VC-02 processes the audio signal using its onboard AI engine. If the spoken phrase matches a trained voice sample, the module sends out a corresponding hexadecimal code via UART.

To interface the VC-02 module with the Arduino Uno, the `SoftwareSerial` library is used to emulate a serial port using digital pins 2 (RX) and 3 (TX). This is necessary as the Arduino Uno has only one hardware serial interface, typically reserved for USB communication.

3.2 Hexadecimal Code Mapping

Each voice command is mapped to a unique hexadecimal code. For instance, the VC-02 might send `A190` to indicate a "Fan ON" command. The Arduino reads these bytes using `mySerial.read()`, converting each byte to a two-character hexadecimal string.

To maintain a consistent format, values less than 0x10 are padded with a leading zero. These hexadecimal values are concatenated to form a command string (e.g., "0A90" becomes `0A90`) and converted to uppercase to avoid mismatches due to case differences.

- `AA11` – LED ON
- `AA00` – LED OFF
- `A190` – FAN ON
- `A145` – FAN OFF

3.3 Device Control Logic

Once the Arduino forms the complete hexadecimal string, it compares it against predefined values using `if-else` statements. If a match is found, the respective GPIO pin is toggled using `digitalWrite()` to control the connected relay module, which switches the appliance.

For example:

- If `hexCommand == "A190"`, the fan is turned ON by setting `fanPin` to `HIGH`.
- If `hexCommand == "AA11"`, the light is turned ON by setting `ledPin` to `HIGH`.

This loop runs continuously to detect and respond to new commands.

Code Logic

This section explains the detailed code logic used in the Arduino-based voice-controlled home automation project. The Arduino receives hexadecimal commands from the VC-02 voice recognition module via UART serial communication, interprets them, and triggers corresponding GPIO pins to control appliances like LEDs and fans.

Importing Libraries and Serial Setup

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX = pin 2, TX = pin 3
```

Pin 2 is configured as RX to receive data from the VC-02 module's TX.

0.0.1 Initializing Serial Communication and GPIOs

```
void setup() {
Serial.begin(9600);          // For debugging using Serial Monitor
mySerial.begin(9600);        // Communication with VC-02 module
pinMode(ledPin, OUTPUT);     // LED control pin
pinMode(fanPin, OUTPUT);     // Fan control pin
}
```

Reading and Converting Serial Data

```
if (mySerial.available()) {
String hexCommand = "";

while (mySerial.available()) {
int byteRead = mySerial.read();

if (byteRead < 0x10) hexCommand += "0";
hexCommand += String(byteRead, HEX);

}

hexCommand.toUpperCase();
Serial.println("Received Hex: " + hexCommand);
}
```

This code:

- Reads raw bytes using `mySerial.read()`

- Converts each byte to a two-character hexadecimal string
- Concatenates all bytes to form a full command
- Converts the string to uppercase for consistent matching

Command Matching and Appliance Control

```
if (hexCommand == "A190") {
digitalWrite(fanPin, HIGH);
}
else if (hexCommand == "A145") {
digitalWrite(fanPin, LOW);
}
else if (hexCommand == "AA11") {
digitalWrite(ledPin, HIGH);
}
else if (hexCommand == "AA00") {
digitalWrite(ledPin, LOW);
}
```

Each condition corresponds to a specific appliance action. This logic ensures the system reacts accurately to recognized voice commands.

Conclusion

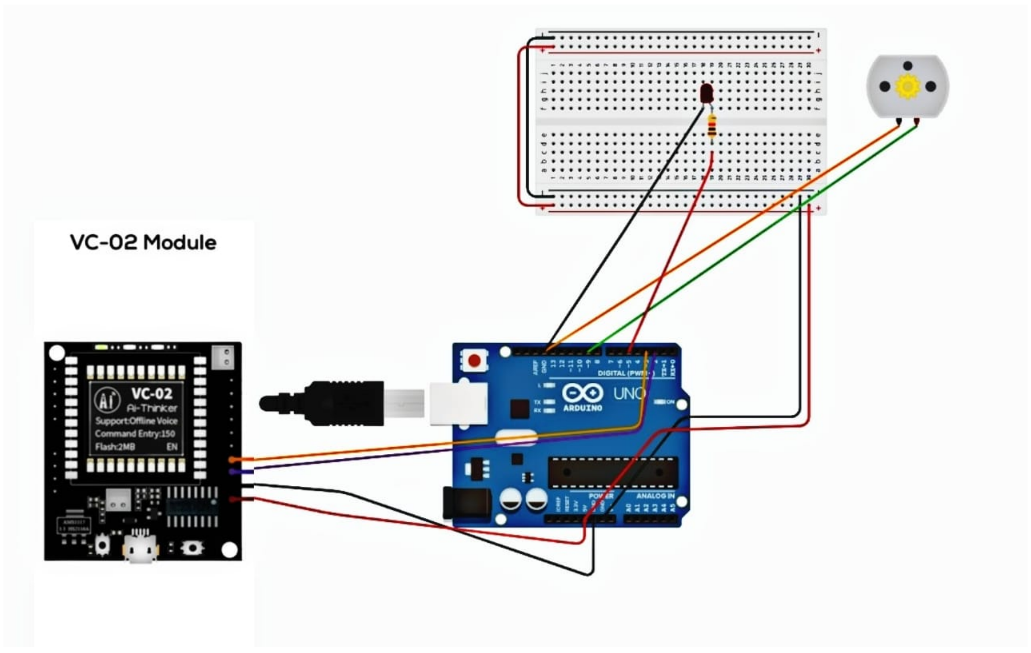
The code logic and working principle described in this document provide a clear overview of how voice recognition data is received, processed, and used to control home appliances in an offline environment. The use of the VC-02 module in conjunction with Arduino makes the system both cost-effective and efficient.

CIRCUIT AND CODE EXPLANATION

4.1 Hardware Connections

This section explains the hardware setup used in the voice-controlled home automation system. The system uses a **VC-02 voice recognition module**, **Arduino Uno**, an **LED**, a **relay module**, and an **AC output socket** (such as for a fan). The components are connected to allow the Arduino to control devices using voice commands processed offline by the VC-02 module.

IMAGE 1: Overall Circuit Diagram



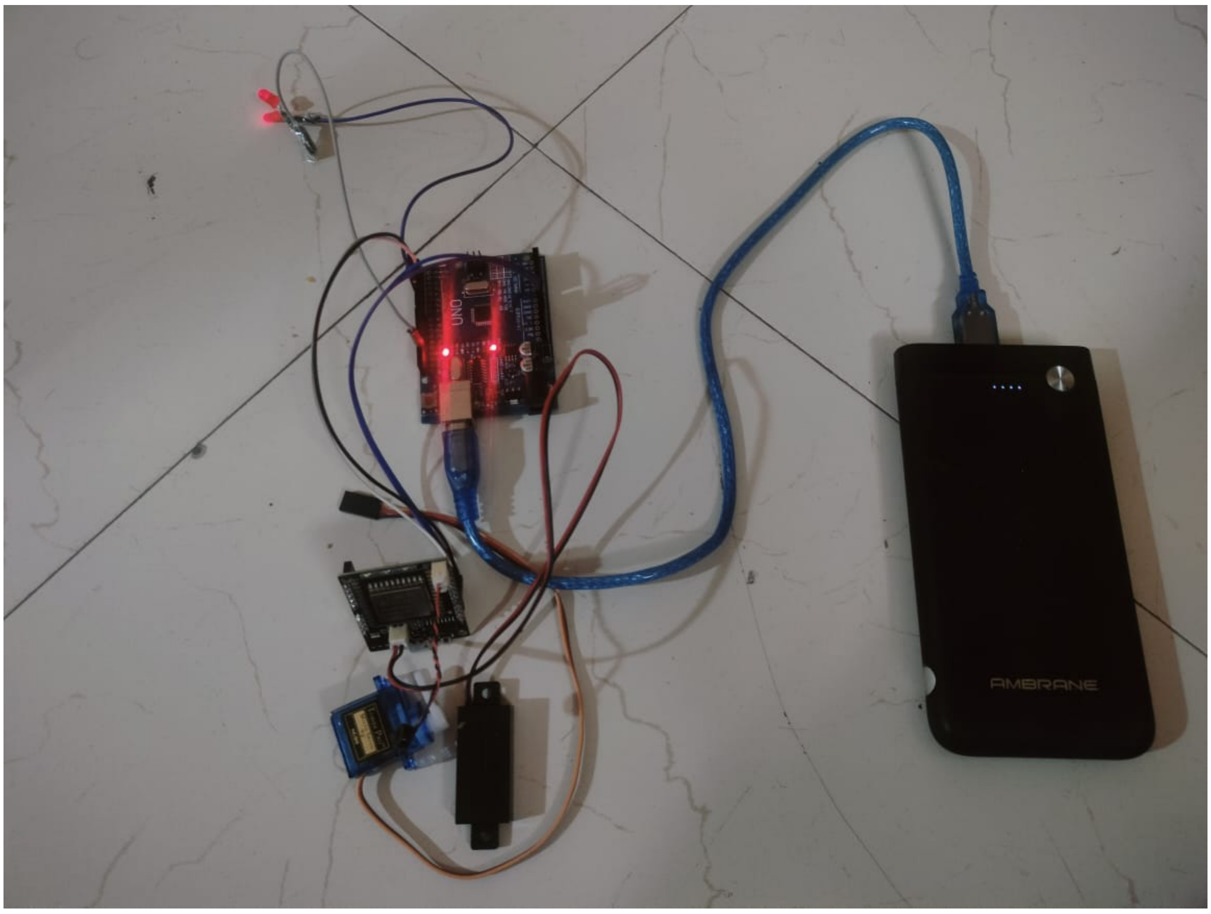
The full circuit connects the VC-02 voice module with the Arduino Uno through digital pins and includes a breadboard for connecting the LED and relay module.

Connections from VC-02 to Arduino Uno

- **TX (VC-02) to D2 (Arduino)** – Sends recognized command data to Arduino.
- **RX (VC-02) to D3 (Arduino)** – Receives control or debug data from Arduino.
- **VCC (VC-02) to 5V (Arduino)** – Supplies power to the module.
- **GND (VC-02) to GND (Arduino)** – Common ground for all components.

Once powered, the VC-02 listens for voice commands and sends unique hexadecimal codes to the Arduino when a match is detected.

IMAGE 2: LED Connection

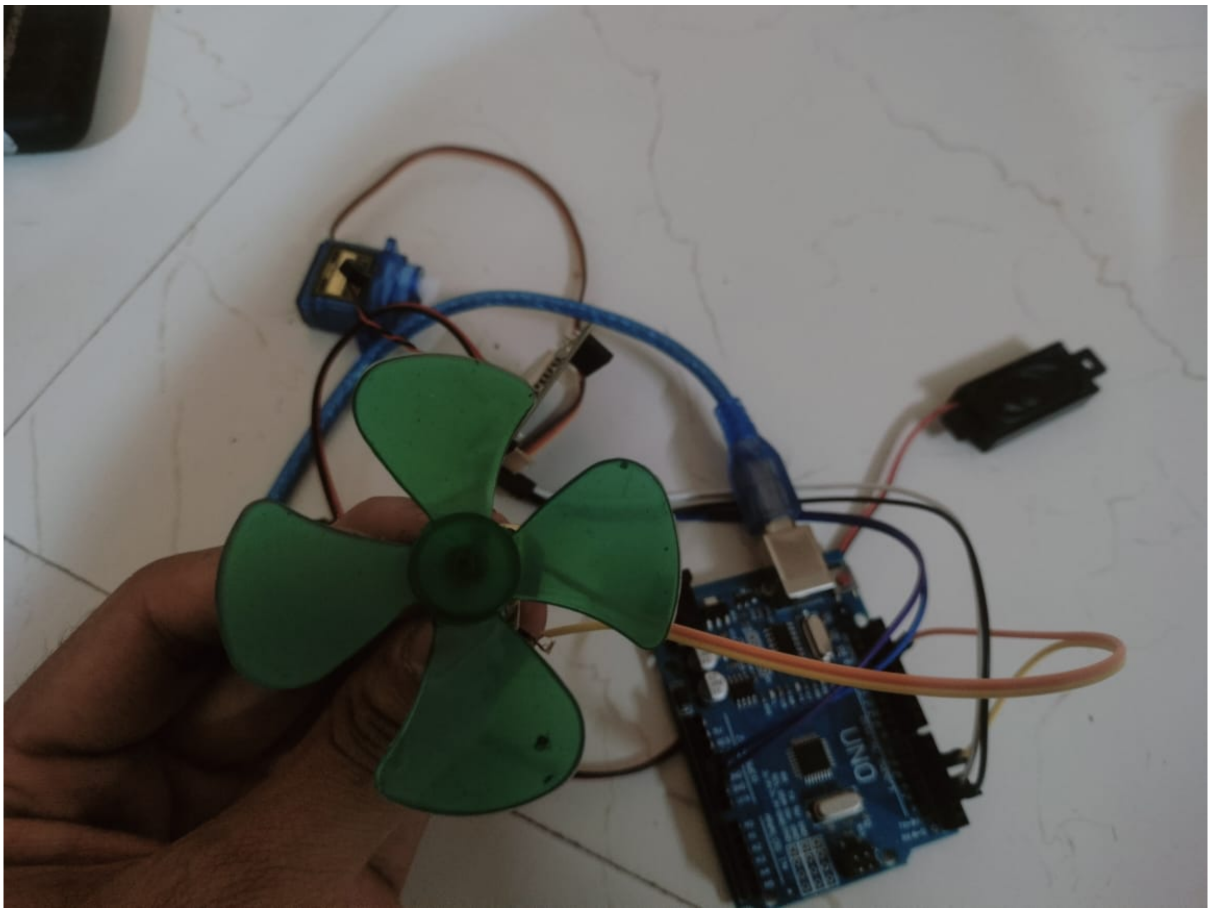


An LED is connected through a current-limiting resistor:

- One end of the LED is connected to pin D8 via a resistor.
- The other end is connected to GND.

When the Arduino receives a command like `"AA11"`, it turns the LED ON by setting pin D8 to `HIGH`.

IMAGE 3: Relay and Output Device (Fan) Connection



A relay is used to control an AC device (like a fan):

- Signal pin of the relay connects to Arduino pin D9.
- Relay VCC to Arduino 5V, GND to Arduino GND.
- Fan or AC socket is connected to the relay's output terminal.

When the Arduino receives a command such as `"A190"`, it activates the relay by sending a `HIGH` signal to pin D9. This closes the relay switch and powers the appliance.

Connection Summary Table

Component	Arduino Pin	Function
VC-02 TX	D2	Sends command data to Arduino
VC-02 RX	D3	Receives optional data from Arduino
VC-02 VCC	5V	Supplies power to VC-02 module
VC-02 GND	GND	Common ground
LED (via resistor)	D8	Controls LED ON/OFF
Relay Signal	D9	Controls fan/socket ON/OFF
Relay VCC/GND	5V / GND	Powers the relay module

This hardware setup allows the Arduino to control multiple devices using simple voice commands without requiring internet connectivity.

4.2 Code Walkthrough

The code logic behind this offline voice-controlled home automation system is centered around receiving, interpreting, and acting upon hexadecimal commands sent by the VC-02 voice recognition module. Initially, necessary libraries are imported, and `SoftwareSerial` is used to establish UART communication between the Arduino Uno and the VC-02 module using digital pins 2 (RX) and 3 (TX). This is essential as the Arduino Uno has only one hardware serial port, typically used for debugging. In the `setup()` function, serial communication is initialized at 9600 baud for both the module and the serial monitor. The GPIO pins connected to appliances like the fan and LED are configured as outputs. In the `loop()`, the Arduino continuously checks for incoming serial data from the VC-02 module. When data is available, it reads the bytes one by one, converts each to a hexadecimal string, pads values under 0x10 with a leading zero for consistency, and concatenates them to form a complete command. The string is converted to uppercase to ensure uniformity in matching, and then printed to the serial monitor for debugging. The final part of the logic matches the received hexadecimal string with predefined values using `if-else` conditions and triggers the appropriate GPIO pins using `digitalWrite()` to turn the appliances ON or OFF accordingly.

4.3 Serial Communication Handling

The VC-02 voice recognition module communicates with the Arduino via UART, and this serial interaction is managed using the `SoftwareSerial` library. Since the Arduino Uno lacks multiple hardware serial ports, `SoftwareSerial` provides a convenient way to emulate an additional serial port using digital pins. The VC-02 module sends hexadecimal codes when it detects a voice command that matches one of its trained samples. For example, a command like "Turn on fan" might be translated into a hexadecimal code like `A190`. The Arduino reads this data using `mySerial.read()`, which returns one byte at a time. Each byte is then converted into a two-character hexadecimal string. If the byte is less than 16 (0x10), a leading zero is added to maintain consistency. These hexadecimal pairs are appended into a full command string, which is then converted to uppercase to eliminate case sensitivity issues during comparison. While this method is straightforward and effective, it relies on the assumption that the input is always correctly structured and timely. To improve reliability, it would be beneficial to implement additional features such as timeout mechanisms to prevent partial reads, checksum validation to ensure data integrity, and pattern recognition to confirm that the command length matches expectations.

4.4 Output Device Logic

The final step in the system's operation is the execution of the output device logic, which is responsible for controlling the appliances based on the interpreted voice commands. Once the full hexadecimal string is received and formed, the Arduino compares it against a set of hardcoded values using `if-else` statements. Each command is linked to a specific device action. For instance, if the command matches `A190`, the fan relay is activated by setting the corresponding pin to `HIGH`, whereas `AA11` activates the LED. Similarly, `A145` and `AA00` are used to turn the fan and LED off, respectively. The logic assumes that commands will always be valid and does not maintain or check the current state of the appliances, which means it may redundantly issue ON commands even if the device is already ON. For future enhancements, implementing a system to track device states internally would reduce unnecessary toggles and conserve energy. Additionally, integrating feedback mechanisms such as status LEDs or serial prints confirming action execution would greatly improve user experience and debugging capabilities. As more appliances are added, the current `if-else` logic could become cumbersome, making a shift to a more scalable structure, such as using a dictionary or a switch-case system, a more efficient choice.

APPLICATIONS

The voice-controlled home automation system powered by the VC-02 module and Arduino Uno presents a multitude of real-world applications. With the rise of Internet of Things (IoT) devices and smart living solutions, systems like these are gaining immense popularity. This section details the major applications where this project can be implemented effectively.

Cons of Having Voice Automation

Voice automation, while offering convenience, also comes with several limitations and risks. The following are some of the main drawbacks:

- **Extreme installation cost when fully automated:** Setting up a fully voice-controlled home can be very expensive due to the cost of smart devices and professional setup.
- **Technological problems in connected homes:** Devices may malfunction, lose connection, or not respond properly, which can disrupt daily routines.
- **Some initial learning efforts are necessary:** Users may need time to understand how to use voice commands effectively and configure devices properly.
- **Helplessness if technology fails:** In the event of system or hardware failure, users may lose the ability to control devices or access their home efficiently.
- **Reliable internet connection is crucial:** Most voice-activated systems depend heavily on internet connectivity. Poor or no connection limits their functionality.
- **You may lock yourself out of your own house:** If the voice recognition fails or the system crashes, users might be denied entry to their homes or access to vital controls.
- **Maintenance and repair issues:** As with any technology, voice-controlled systems require maintenance and might need regular updates or repairs, which can be complex and costly.

5.1 Smart Home Automation

One of the primary applications of the system is in smart home automation. With the ability to control lights, fans, and potentially other electrical devices using simple voice commands, this technology enhances the convenience and comfort of home environments. Users no longer need to manually interact with switches, as their voice acts as the interface.

For instance, in the evening, a user can walk into their room and simply say, "Turn on the light," and the VC-02 module recognizes the command, sends a specific hexadecimal code to the Arduino, and switches the light on via a relay module. This makes everyday tasks easier and adds a futuristic touch to home settings. Moreover, by expanding the voice command database, additional appliances such as air conditioners, coffee machines, or televisions can be included in the system.

Another benefit is the energy efficiency it offers. Users can easily turn off all devices before leaving the house using a single command. This prevents wastage of electricity and promotes a greener environment.

5.2 Accessibility Solutions

This technology is especially beneficial for individuals with physical disabilities or limited mobility. Traditional methods of controlling electrical devices often require physical movement, which may not be feasible for everyone. Voice-controlled systems empower differently-abled individuals to gain autonomy over their living environment.

For example, someone with motor impairments can control lighting, fans, and other devices without needing to physically reach switches. This fosters a sense of independence and improves the quality of life.

Additionally, senior citizens who may have difficulty navigating through rooms or handling switches can benefit immensely from such systems. This also reduces the risk of accidents, such as tripping in the dark while looking for a switch.

5.3 Offline Control Systems

Unlike many modern voice assistants that rely on cloud-based services and internet connectivity, this project works completely offline. The VC-02 module processes voice commands locally, which means there is no delay or dependency on internet services.

Offline systems are advantageous in several scenarios:

- In areas with limited or no internet access, such as rural or remote locations.
- In environments where data privacy is a concern, since no information is sent to external servers.
- In industrial settings where robust and uninterrupted operation is critical.

Since the module supports up to 150 custom voice commands, it can be trained for a wide variety of actions, ensuring adaptability to different contexts.

CONCLUSION

6.1 Summary

This project successfully demonstrates an efficient and user-friendly voice-controlled home automation system using the VC-02 module and Arduino Uno. The implementation of the `SoftwareSerial` library enables smooth communication between the microcontroller and the voice module, allowing multiple serial devices to interact simultaneously.

The working principle involves three core stages: recognizing a voice command via the VC-02 module, converting that to a hexadecimal code, and executing the appropriate device control logic on the Arduino. By using offline voice processing, this setup eliminates the need for an internet connection, resulting in faster response times and improved data privacy.

The practical applications of this system are vast and impactful. It finds use in smart homes for enhancing convenience and efficiency, serves as an accessibility solution for the elderly and disabled, and proves valuable in environments with limited internet access.

6.2 Future Scope

There are several opportunities for enhancing this system in future versions:

- **Increased Device Control:** Future iterations can incorporate control of more devices, including fans with adjustable speeds, RGB LED strips, air conditioners, or smart locks.
- **Integration with Mobile Apps:** A Bluetooth or WiFi interface can be added to enable monitoring and control through a smartphone application.
- **Enhanced Security:** Integration with motion sensors, cameras, and alarm systems can transform the setup into a full-fledged smart security system.
- **Voice Feedback:** Adding a speaker system for voice confirmation of commands can provide feedback to users, ensuring a more interactive experience.
- **Energy Monitoring:** Sensors can be added to monitor power consumption of each appliance, helping users make informed decisions about energy usage.

With continuous innovation in electronics and voice recognition technology, such projects have a promising future and can greatly influence the evolution of human-technology interaction.

RESULTS AND OBSERVATIONS

During the testing and simulation phase of the project, the system exhibited reliable performance in recognizing and acting upon voice commands. Below are some key observations:

- The VC-02 module accurately identified trained voice commands in various environmental conditions.
- The Arduino successfully interpreted the hexadecimal codes and toggled GPIO pins accordingly.
- Simulations conducted using Proteus confirmed circuit functionality before hardware implementation.
- The offline nature of the system ensured quick response times without network lag.

The results validate the robustness of the proposed voice-controlled automation system. Moreover, the simulation environment allowed iterative testing, helping to fine-tune the command mapping and logic.

REFERENCES

1. Arduino UNO R3 Datasheet. Available at: `https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf`
2. Arduino Official Documentation. Available at: `https://www.arduino.cc/en/Guide/ArduinoUno`
3. VC-02 Voice Recognition Module Product Manual. AI Thinker. Available at: `https://www.ai-thinker.com/pro_view-61.html`
4. Introduction to UART Communication. SparkFun. Available at: `https://learn.sparkfun.com/tutorials/serial-communication`
5. Home Automation using Arduino UNO and Voice Module Projects. Instructables. Available at: `https://www.instructables.com`
6. Voice Recognition Module VC-02 Quick Start Guide – Makerfabs. Available at: `https://forum.arduino.cc/t/how-to-program-ai-thinker-vc-02/1132521`
7. Electronics Tutorials: Arduino Digital and Analog Pins. Available at: `https://www.electronics-tutorials.ws`
8. Datasheet and pinout diagrams sourced from component packaging and manufacturer websites.