



DAYANANDA SAGAR COLLEGE OF ENGINEERING

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' Grade

(AICTE Approved, an Autonomous Institute Affiliated to VTU, Belagavi)

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078

DEPARTMENT OF MECHANICAL ENGINEERING

(Accredited by NBA)

A Mini-Project Report on

“Voice Activated Automatic Door using Python programming language”

Submitted in partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING

In

MECHANICAL ENGINEERING

2022-2023

Submitted by

| | |
|-------------------|-------------------------|
| 1DS20ME030 | Harsh Raj |
| 1DS20ME040 | Kushagra Sinha |
| 1DS20ME041 | Likhith Gowda BS |
| 1DS20ME059 | Pratyush Nandan |

Under the guidance of

Mrs G Padmavati

Assistant Professor

Department of Mechanical Engineering,

Dayananda Sagar College of Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078



DAYANANDA SAGAR COLLEGE OF ENGINEERING

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' Grade

(AICTE Approved, an Autonomous Institute Affiliated to VTU, Belagavi)

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078

DEPARTMENT OF MECHANICAL ENGINEERING

(Accredited by NBA)

Certificate

Certified that the project report entitled '**Voice activated automatic door using Python programming language**' is a Bonafide work carried out by **Harsh Raj**, bearing USN: **1DS20ME030**, **Kushagra Sinha**, bearing USN: **1DS20ME040**, **Likhith Gowda BS** bearing USN: **1DS20ME041**, **Pratyush Nandan**, bearing USN: **1DS20ME059** under the guidance of **Mrs G Padmavati, Assistant Professor**, Department of Mechanical Engineering, Dayananda Sagar College of Engineering, Bengaluru, in partial fulfilment for the award of Bachelor of Engineering in Mechanical Engineering of the Visvesvaraya Technological University, Belagavi.

Name of Guide –

G Padmavathi

Assistant Proffesor

Designation

Department of Mechanical Engineering

Dr. R. Keshavamurthy

Professor & Head

Department of Mechanical Engineering

Dayananda Sagar College of Engineering.

Bengaluru

Dr. B G Prasad

Principal

Dayananda Sagar College of Engineering.

Bengaluru

Viva-Voce

| Name of Examiners | Signature with Date |
|-------------------|---------------------|
| 1. | |
| 2. | |

DECLARATION

We the below mentioned students hereby declare that the entire work embodied in the project report entitled 'Voice Activated automatic door using python programming language' has been independently carried out by us under the guidance of **Mrs G Padmavati**, Assistant Professor, Department of Mechanical Engineering, Dayananda Sagar College of Engineering, Bengaluru, in partial fulfilment of the requirements for the award of Bachelor Degree in Mechanical Engineering of Visvesvaraya Technological University, Belagavi.

We further declare that we have not submitted this report either in part or in full to any other university for the award of any degree.

| SL NO | NAME | USN | SIGNATURE |
|-------|------------------|------------|-----------|
| 1. | Harsh Raj | 1DS20ME030 | |
| 2. | Kushagra Sinha | 1DS20ME040 | |
| 3. | Likhith Gowda BS | 1DS20ME041 | |
| 4. | Pratyush Nandan | 1DS20ME059 | |

Place: Bengaluru

Date:

ABSTRACT

This project is to create a Voice Activated Servo Motor switch with the use of Python programming language. This voice assistant project will get us a kick start and move towards the world of AI and ML (Machine Learning).

Voice recognition technology is a software program or hardware device that can decode the human voice. Sometimes referred to as voice-activated or speech recognition software, this technology has become increasingly popular in recent years among everyday users.



ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Dr. B. G. PRASAD**, Principal, and DSCE, who has given us an opportunity to successfully complete our project.

It gives us an immense pleasure to thank **Dr. R. KESHAVAMURTHY**, Head of the Department, Mechanical Engineering, DSCE, for his valuable advice and guidance which has helped us complete our project.

This project would not have been profound without the persistent encouragement of our beloved guide **Mrs G Padmavati**, Assistant Professor, Department of Mechanical Engineering, DSCE, whose sound technical knowledge, symmetric approach, discipline towards work and immense experience fostered us to put in better efforts.

Finally, we express our gratitude to all the teaching and non-teaching staff, who have indirectly helped us to complete this project successfully. Last but not the least we would like to thank our parents for their blessings and love. We would also like to thank our friends for their support and encouragement to successfully complete the task by meeting all the requirement.

CONTENTS

| | |
|---|----------|
| DECLARATION | Page No. |
| ABSTRACT | 4 |
| ACKNOWLEDGEMENT | 5 |
| LIST OF FIGURES | 7 |
| CHAPTER 1: INTRODUCTION | 8 - 13 |
| Sub Section | |
| 1.1 Benefits of using voice commands | 9 |
| 1.2 Arduino's purpose and role in the project | 10 |
| 1.3 Servo motor function and applications | 11 |
| 1.4 Serial communication and its significance | 12 |
| CHAPTER 2: LITERATURE REVIEW | 14 -15 |
| CHAPTER 3: METHEDOLOGY | 16 - 31 |
| Sub Section | |
| 3.1 Hardware and software requirements | 16 - 19 |
| 3.2 Arduino code explanation | 20 - 22 |
| 3.3 Python code explanation | 23 - 25 |
| CHAPTER 4: RESULTS AND DISCUSSION | 32 - 33 |
| Sub Section | |
| 4.1 Outcome and challenges of the project | 32 |
| 4.2 Insights and potential applications | 33 |
| CHAPTER 5: CONCLUSION | 34 |
| SCOPE FOR FUTURE WORK | 35- 36 |
| REFERENCES | 37 |

LIST OF FIGURES

| Figure no. | Description | Page no |
|------------|------------------------------|---------|
| 1. | Servo Motor | 16 |
| 2. | Arduino UNO Board | 20 |
| 3. | Arduino IDE code | 21 |
| 4. | API | 23 |
| 5. | Python to Arduino connection | 24 |
| 6. | Python Code | 25 |
| 7. | Voice Recognition Process | 29 |
| 8. | Voice Recognition Mechanism | 30 |

CHAPTER 1: INTRODUCTION

1.1 Motivation behind this Project

The motivation behind the project can be attributed to several factors, including the increasing interest in home automation and the growing desire for hands-free control of devices.

1. Convenience and Efficiency: Home automation has gained significant popularity as it offers convenience and efficiency to homeowners. The ability to control various devices and systems within the home through automation systems simplifies daily tasks and enhances overall comfort. With automation, individuals can remotely manage lighting, temperature, security systems, entertainment devices, and more, leading to a more convenient and streamlined lifestyle.

2. Enhanced Lifestyle: Home automation systems provide individuals with the ability to create personalized environments that cater to their preferences. Whether it is adjusting the lighting and temperature to create a cosy atmosphere or automating routines to simplify daily activities, home automation offers an enhanced and tailored lifestyle experience. The desire for a more comfortable, luxurious, and technologically advanced living space has fuelled the interest in home automation.

3. Technological Advancements: The rapid advancements in technology, particularly in the fields of artificial intelligence, voice recognition, and the Internet of Things (IoT), have played a crucial role in driving the desire for hands-free control of devices. With the advent of voice assistants like Amazon Alexa, Google Assistant, or Apple Siri, users can control various devices using voice commands, eliminating the need for physical interaction. This hands-free control has become increasingly popular as it offers a more intuitive and seamless user experience.

4. Energy Efficiency and Cost Savings: Home automation systems often include features that promote energy efficiency, such as smart thermostats, lighting controls, and energy monitoring. These systems enable users to optimize energy consumption by automatically adjusting settings based on occupancy or time of day. By reducing energy waste, homeowners can achieve cost savings on utility bills while contributing to environmental sustainability.

5. Aging Population and Accessibility: The aging population has also contributed to the interest in home automation. Many elderly individuals or people with limited mobility seek ways to make their homes more accessible and manageable. Home automation systems allow them to control various aspects of their environment with ease, enabling independent living and reducing reliance on others for assistance.

Overall, the increasing interest in home automation and the desire for hands-free control of devices stem from the pursuit of convenience, efficiency, personalization, and improved quality of life. Technological advancements and the need for energy efficiency and accessibility have further fueled the motivation behind these project

1.2 Benefits of using voice command to control Servo Motors

Using voice commands to control servo motors offers several benefits that enhance user experience and accessibility:

1. **Hands-Free Control:** Voice commands eliminate the need for physical interaction with control interfaces, such as buttons or switches, enabling hands-free control of servo motors. This is particularly beneficial for individuals with limited mobility or physical disabilities, allowing them to operate devices and perform tasks without requiring manual dexterity.
2. **Intuitive and Natural Interaction:** Voice commands provide a more intuitive and natural way of interacting with servo motors. Instead of learning complex control interfaces or memorizing specific commands, users can simply speak the desired actions or movements, making it easier for people of all ages and technical backgrounds to operate the devices.
3. **Enhanced Accessibility:** Voice-controlled servo motors offer enhanced accessibility by enabling users with visual impairments or conditions that affect fine motor control to operate devices independently. By using voice commands, individuals can easily control servo motors without relying on physical touch or visual feedback.
4. **Increased Efficiency and Speed:** Voice commands can streamline the control process and make it faster compared to traditional methods. Users can quickly issue commands without the need to navigate menus or physically interact with controls, resulting in improved efficiency and reduced task completion time.
5. **Personalization and Customization:** Voice-controlled servo motors allow for personalized and customized experiences. Users can define their own voice commands or set up specific phrases to trigger desired actions or movements. This level of customization enables individuals to tailor the control system to their preferences, making it more personalized and efficient for their specific needs.
6. **Multi-Tasking and Convenience:** With voice commands, users can control servo motors while performing other tasks simultaneously. This multitasking capability adds convenience and flexibility to the user experience, allowing individuals to interact with devices without interrupting their current activities.
7. **Integration with Smart Home Systems:** Voice-controlled servo motors can be seamlessly integrated into smart home systems, enabling users to incorporate them into broader automation scenarios. For example, users can combine voice commands with other smart devices, such as lights or curtains, to create synchronized and automated routines, further enhancing convenience and overall home automation experience.

In summary, using voice commands to control servo motors enhances user experience and accessibility by providing hands-free control, intuitive interaction, enhanced customization, and integration with smart home systems. It empowers individuals with limited mobility or disabilities, simplifies the control process, and adds convenience and efficiency to operating servo motors.

1.3 Arduino's purpose and role in this project

Arduino is an open-source electronics platform that consists of both hardware and software components. It is designed to provide an accessible and flexible platform for creating and controlling various electronic projects. The purpose of Arduino is to enable individuals, including hobbyists, artists, students, and professionals, to easily prototype and build interactive electronic devices.

The Arduino board serves as the core hardware component of the Arduino platform. It features a microcontroller and a set of input and output pins that can be used to connect and control different electronic components and modules. The Arduino software, also known as the Integrated Development Environment (IDE), provides a user-friendly programming interface to write and upload code to the Arduino board.

In the context of voice-activated servo motors, Arduino plays a crucial role in integrating the voice recognition capability with the servo motor control. Here is how it works:

1. **Voice Input:** A voice recognition module or a voice assistant device, such as a speech recognition shield or a voice recognition module connected to Arduino, captures voice commands from the user.
2. **Voice Processing:** The voice recognition module converts the captured voice commands into digital signals or specific data that Arduino can interpret.
3. **Arduino Integration:** Arduino receives the digital signals or data from the voice recognition module and processes them using the code uploaded to the Arduino board.
4. **Servo Motor Control:** The Arduino board, based on the received voice commands, sends appropriate signals to the servo motor through the output pins. These signals control the position, speed, and direction of the servo motor, allowing it to move accordingly.
5. **Feedback and Response:** Arduino can also incorporate feedback mechanisms, such as LEDs or other output devices, to indicate successful reception and execution of voice commands or provide visual feedback to the user.

By utilizing Arduino's capabilities, including its programmability and ability to interface with various electronic components, voice commands can be received, processed, and translated into actions for controlling servo motors. Arduino acts as the bridge between the voice

recognition module and the servo motor, facilitating the integration of voice control into servo motor applications.

1.4 Servo Motors, their functionality, and common applications

Servo motors are a type of rotary actuator that provides precise control over angular position, speed, and torque. They are widely used in various applications where accurate and controlled motion is required. Here is an explanation of their functionality and common applications.

Functionality:

Servo motors consist of several key components, including a DC motor, a gearbox, a control circuit, and a position feedback mechanism. The control circuit and position feedback mechanism work together to provide closed-loop control, ensuring accurate positioning.

The control circuit receives electrical signals, usually in the form of pulses, and translates them into corresponding mechanical movements. These signals, known as PWM (Pulse Width Modulation) signals, determine the position and speed at which the servo motor rotates. The position feedback mechanism, such as a potentiometer or an encoder, continuously monitors the actual position of the motor shaft and provides feedback to the control circuit, enabling precise positioning.

The servo motor's internal gearbox helps convert the high-speed, low-torque output of the motor into slower speed with higher torque at the output shaft. This gearing mechanism allows servo motors to generate significant torque while maintaining precise control over position and speed.

Common Applications:

Servo motors find applications in a wide range of fields due to their versatility and precise control capabilities. Some common applications include:

1. **Robotics:** Servo motors are extensively used in robotic systems to control joint movements. They enable precise positioning and smooth motion, making them suitable for applications like robotic arms, grippers, and humanoid robots.
2. **Industrial Automation:** In industrial automation, servo motors are used for various tasks such as conveyor belt control, CNC (Computer Numerical Control) machines, precision positioning systems, and robotic assembly lines. Their accuracy and repeatability make them ideal for these applications.
3. **Aerospace and Aviation:** Servo motors are employed in aviation and aerospace systems, including aircraft control surfaces, unmanned aerial vehicles (UAVs), and spacecraft. They ensure precise control over flight surfaces, such as ailerons, elevators, and rudders.

4. **Camera Stabilization:** Servo motors are used in camera stabilization systems, such as gimbals and pan-tilt systems. They allow for smooth and stabilized camera movements, eliminating unwanted vibrations and maintaining the desired frame.

5. **Automotive Industry:** Servo motors are found in various automotive applications, including power windows, door locks, steering systems, and electronic throttle control. They provide accurate control and actuation in these systems.

6. **Home Automation:** Servo motors are utilized in home automation for controlling mechanisms such as automated blinds, curtains, and motorized doors, offering precise and programmable positioning.

7. **Model Making and Hobby Projects:** Servo motors are commonly used in hobbyist projects, such as remote-controlled cars, planes, boats, and animatronics. They provide affordable and precise motion control for DIY enthusiasts.

In summary, servo motors offer precise and controlled motion in a wide range of applications, including robotics, industrial automation, aerospace, camera stabilization, automotive systems, home automation, and hobby projects. Their ability to provide accurate positioning, speed control, and torque output makes them essential components in various industries and projects.

1.5 Serial Communication and its significance in connecting Arduino with python

Serial communication is a method of transmitting data between electronic devices over a serial interface, typically using a pair of wires. It allows for the transfer of data one bit at a time in a sequential manner. Serial communication is widely used due to its simplicity, versatility, and compatibility across different devices.

The basics of serial communication involve two main components: the transmitter and the receiver. The transmitter converts the data into a serial stream of bits, which is then sent over the communication channel. The receiver on the other end receives the serial data stream and reconstructs it back into its original form.

In the context of connecting Arduino with Python, serial communication plays a significant role. Arduino boards often have built-in serial communication capabilities through Universal Serial Bus (USB) connections. Python, as a powerful programming language, provides libraries and modules that allow easy communication with Arduino over the serial interface. Here's why serial communication is significant in connecting Arduino with Python:

1. **Data Exchange:** Serial communication enables bidirectional data exchange between Arduino and Python. Python can send commands or data to Arduino, and Arduino can provide

sensor readings or status updates back to Python. This bidirectional communication facilitates real-time interaction and control between the two platforms.

2. Control and Automation: Python, with its extensive libraries and capabilities, can act as the controlling entity for Arduino-based projects. By establishing a serial communication link, Python can send commands to Arduino, instructing it to perform specific actions, control outputs, or read inputs from various sensors. This enables automation and remote control of Arduino projects from a Python script or application.

3. Sensor Integration: Arduino boards are commonly used for interfacing with sensors and collecting data from the physical world. Serial communication allows Python to read sensor data from Arduino in real-time. Python can then process and analyse the data, perform calculations, generate visualizations, or trigger other actions based on the received sensor readings.

4. Data Logging and Analysis: Python's data handling and analysis capabilities make it a suitable platform for logging and analysing data collected by Arduino. Through serial communication, Arduino can send data to Python, which can then store the data in a database, save it to a file, or perform real-time analysis. This facilitates data-driven decision making and further processing of collected data.

5. User Interface: Python provides powerful graphical user interface (GUI) libraries, such as Tkinter or PyOt, that can be used to create interactive interfaces for Arduino projects. By establishing a serial communication link, Python can receive data from Arduino and update the GUI, accordingly, allowing users to visualize and interact with the Arduino-based system through a user-friendly interface.

In summary, serial communication enables seamless data exchange, control, sensor integration, data logging, and user interface development between Arduino and Python. It provides a reliable and efficient means of connecting the two platforms, opening opportunities for diverse applications and projects that leverage the strengths of both Arduino and Python.

CHAPTER 2: LITERATURE REVIEW

2.1 Literature Survey

1.A systematic review of voice assistant usability by Faruk Lawal Ibrahim Dutsinma et.al[1], states that Voice assistants (VAs) are computer programs capable of understanding and responding to users using synthetic voices. A report indicates that 4.2 billion VAs were adopted and used in 2020 alone, with a projected increase to 8.4 billion by 2024. Usability is a critical factor in the adoption of voice assistants. Their study aimed to understand what is currently employed for measuring voice assistant usability, and we identified the different independent variables, dependent variables, and the techniques used. Furthermore, they also focused on using the ISO 9241-11 framework to measure the usability of voice assistants. Their study classified five independent variable classes used for measuring the dependent variables. These separate classes were categorized based on the similarities between the member groups.

2.A Survey on Smart Assistant System using Speech Recognition by Mohd Abid Hundekari et.al[2] This included the study of all aspects including technologies, trends in the market based on AI applications, home appliances using voice recognition systems, the areas which are still under development in this field and to find out a way to provide an efficient and accurate system to the user so that the user can control its device using speech instead of typing and looking at the screen. The strongest desire of man to create computer resembling itself and having ability to interact with people can become a reality.

3.Intelligent AI based voice assistant by Dr.S.Brindhya et.al[3], describes a new emerging service for the user. The Voice Assistance provides an intelligent computer secretarial service for any professionals. The new service is based on convergence of internet and speech recognition technology. This Voice assistant minimizes the interruption of the user, improves the utilization of user's time, and provides a single point of communication for all their messages, contacts, schedule, and source of information. The paper proposes a decision structure as well about Computer hardware. However, it is expected to become a standard feature for millions of other users. It overcomes many of the drawbacks in the existing solutions. It is built to make a much more efficient Voice Assistant that they can be brought into much more practical day to day uses. But the system has its own limitation. Though the efficiency is high the time consumption for each task to complete higher than the other Voice assistants and the complexity of the algorithms and the concepts would make it very tough to tweak it if needed in the future.

2.2 Objectives of Project

1. **Implement voice recognition:** Develop a Python script that can capture voice input from the user and convert it into text using a speech recognition library like Speech Recognition and implementing it for Automatic Door Locking System.
2. **Voice commands mapping:** Creating a map between specific voice commands and corresponding servo motor actions. For example, associate a voice command like "rotate clockwise or counter clockwise" with a specific angle of rotation for the servo motor.
3. **User feedback:** Provide feedback to the user by either using text-to-speech capabilities or by displaying messages on a connected display, indicating the action performed or any errors encountered.

CHAPTER 3: METHODOLOGY

3.1 Hardware and Software Requirements

3.1.1. Specifications and Functions of components required for this project

1. Arduino Board:

- Specification: Arduino Uno or Arduino Nano.
- Functionality: The Arduino board serves as the central microcontroller platform. It receives voice commands, processes them, and controls the servo motor accordingly. Arduino boards feature a microcontroller, digital and analog input/output pins, and a programming interface. They are programmable using the Arduino IDE (Integrated Development Environment) and offer various communication options, including serial communication via USB.

2. Servo Motor:

- Specification: SG90 or MG996R servo motor.
- Functionality: The servo motor is responsible for physically opening and closing the door based on the commands received from the Arduino. Servo motors have built-in control circuitry and position feedback mechanism. They can rotate between specific angles, typically 0 to 180 degrees, and hold their position accurately. The Arduino generates PWM signals to control the servo motor's position and speed.

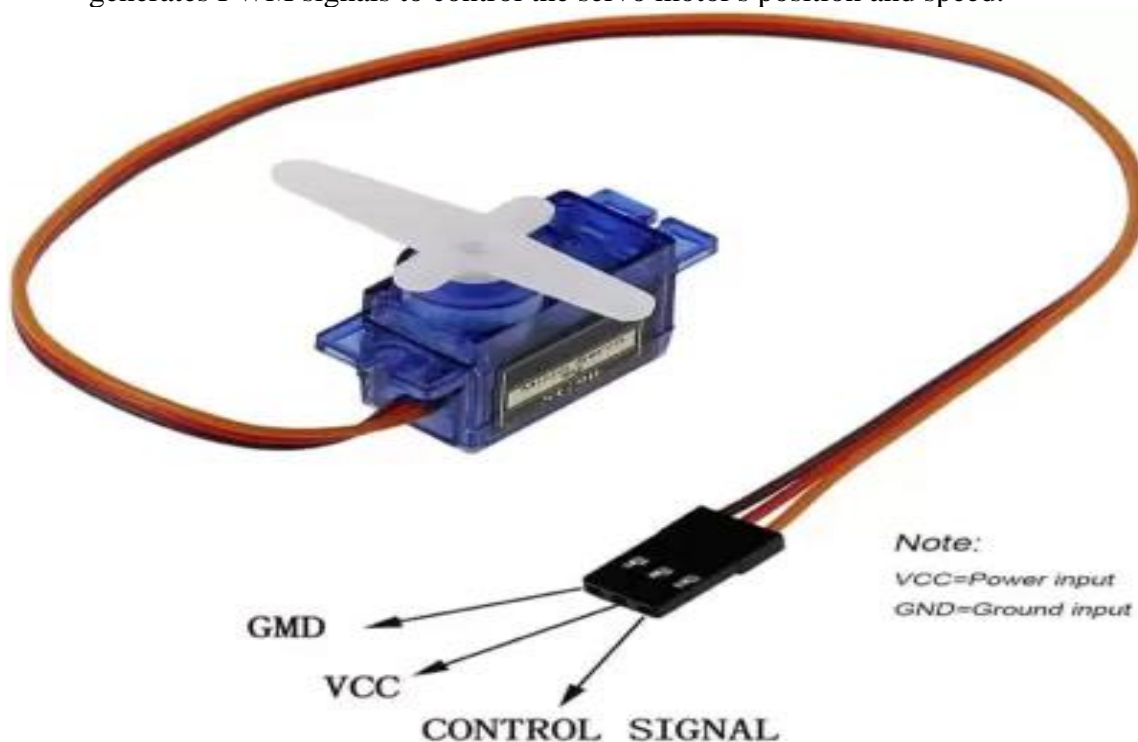


Figure 1: Servo Motor

3. Microphone:

- Specification: Electret Condenser Microphone (ECM) or MEMS microphone.
- Functionality: The microphone captures voice commands from the user. Electret condenser microphones or MEMS microphones convert sound waves into electrical signals. They are sensitive to sound and provide an analog output that can be processed by the Arduino. Microphones are usually connected to an amplifier or preamp circuit to boost the weak signals before further processing.

4. Motor Driver:

- Specification: L293D or L298N motor driver module.
- Functionality: The motor driver circuit amplifies and regulates the electrical signals from the Arduino to provide sufficient power and control for the servo motor's operation. Motor drivers can handle higher current and voltage requirements than what the Arduino's output pins can provide directly. They act as an interface between the Arduino and the servo motor, ensuring smooth and precise control of the motor's movement.

5. Power Supply:

- Specification: Battery pack, DC power adapter, or external power source.
- Functionality: The power supply provides the necessary electrical power to operate the Arduino board, servo motor, and other components. It can be a battery pack, DC power adapter, or connection to a reliable power source. The power supply should meet the voltage and current requirements of the Arduino and servo motor, ensuring stable and sufficient power for the entire system.

6. Jumper Wires:

- Specification: Male-to-male, male-to-female, or female-to-female jumper wires.
- Functionality: Jumper wires are used to establish connections between the Arduino board, servo motor, microphone, and other components. They provide a simple and flexible means of connecting various components together, allowing for easy prototyping and testing. The specific type of jumper wire (male-to-male, male-to-female, or female-to-female) depends on the required connections.

3.1.2 Instructions to set up Voice Automated door using Python

To set up the components for a voice-activated automated door using a servo motor with Python programming, follow these instructions:

1. Prepare the Arduino Board:

1. Connect the Arduino board to your computer using a USB cable.
2. Install the Arduino IDE (Integrated Development Environment) on your computer if you have not already.

3. Upload a standard Arduino sketch that allows serial communication with Python. This sketch will listen for commands from Python and control the servo motor accordingly.

2. Connect the Servo Motor:

1. Identify the three wires of the servo motor: power, ground, and signal.
2. Connect the power wire of the servo motor to the 5V pin on the Arduino.
3. Connect the ground wire of the servo motor to the GND (ground) pin on the Arduino.
4. Connect the signal wire of the servo motor to a digital pin on the Arduino (e.g., Pin 9).

3. Connect the Microphone:

1. Identify the connections required by your microphone.
2. Connect the microphone to your computer or an audio input device that can be accessed by Python.

4. Install Required Python Libraries:

1. Install the PySerial library to enable communication between Python and the Arduino board. You can use the following command in the terminal or command prompt: ``pip install pyserial``.

5. Write the Python Code:

1. Import the necessary libraries, including PySerial for serial communication.
2. Establish a serial connection between Python and the Arduino board using the appropriate port and baud rate.
3. Set up the microphone input in Python using libraries such as PyAudio or SoundDevice to capture voice commands.
4. Process the voice commands using a speech recognition library such as SpeechRecognition.
5. Based on the recognized voice commands, send appropriate control signals over the serial connection to the Arduino to control the servo motor.

6. Power the System:

1. Ensure the Arduino board is connected to a power source via USB or an external power supply.

7. Run the Python Code:

1. Run the Python script in your chosen development environment or through the command line.
2. Speak voice commands into the microphone and observe the servo motor's response.

8. Test and Calibrate:

1. Adjust the Python code as needed to handle different voice commands and control the servo motor accurately.
2. Calibrate the servo motor's position and movement parameters in the Python code to achieve the desired behavior.

Remember to handle electronics with care and follow appropriate safety precautions, such as avoiding short circuits and using the correct voltage levels and current limits for the components.

3.1.3 Explanation of Software Requirements

To successfully implement a voice-activated automated door using a servo motor with Python, several software requirements need to be met. These include the Arduino IDE, Python interpreter, and specific libraries such as PySerial and Speech Recognition. Here is an overview of these software requirements:

1. Arduino IDE:

1. The Arduino IDE (Integrated Development Environment) is a software application used to write, compile, and upload code to Arduino boards.
2. You can download the Arduino IDE from the official Arduino website (<https://www.arduino.cc/en/software>) and install it on your computer.
3. The Arduino IDE provides a user-friendly interface for programming Arduino boards and configuring the necessary settings.

2. Python Interpreter:

1. Python is an interpreted programming language widely used for its simplicity and versatility.
2. To run Python code, you need a Python interpreter installed on your computer.
3. You can download the latest version of Python from the official Python website (<https://www.python.org/downloads>) and follow the installation instructions specific to your operating system.
4. It is recommended to use Python 3.x as Python 2.x is no longer actively supported.

3. PySerial Library:

1. PySerial is a Python library that provides support for serial communication between a computer and external devices, such as the Arduino board.
2. It allows Python programs to send and receive data through the serial port.
3. You can install PySerial using the pip package manager. Open a terminal or command prompt and run the following command: ``pip install pyserial``.
4. PySerial provides a simple and convenient way to establish a serial connection and communicate with the Arduino board from Python.

4. Speech Recognition Library:

1. Speech Recognition is a Python library that provides speech recognition capabilities, allowing your Python program to convert spoken words into text.
2. It supports various speech recognition engines and APIs, including Google Speech Recognition, Sphinx, and Microsoft Azure Speech.
3. Install the Speech Recognition library using the pip package manager. Open a terminal or command prompt and run the following command: ``pip install Speech Recognition``.
4. Speech Recognition simplifies the process of capturing voice commands and integrating speech recognition functionality into your Python code.

By ensuring that you have the Arduino IDE, Python interpreter, and the necessary libraries installed, you can leverage these software components to program the Arduino board, establish serial communication with Python, and incorporate speech recognition capabilities into your voice-activated automated door project.

3.2 Arduino Code Explanation

3.2.1 Purpose and Role of the Arduino Code

The purpose of the Arduino code is to control a servo motor using commands received through the serial communication interface. It allows the user to send specific angles to the servo motor, instructing it to move to various positions.

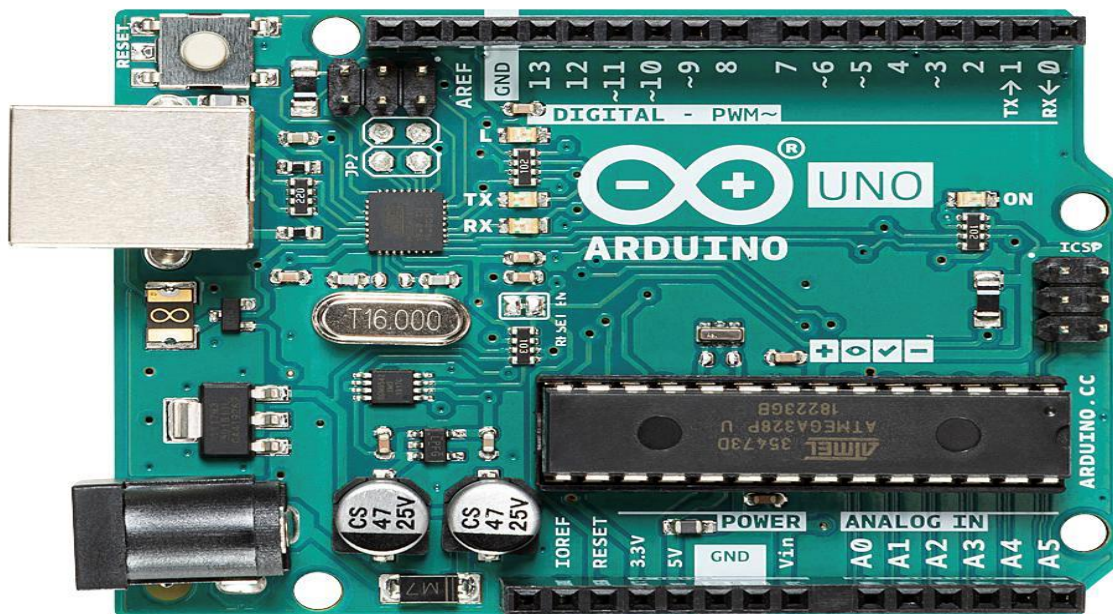
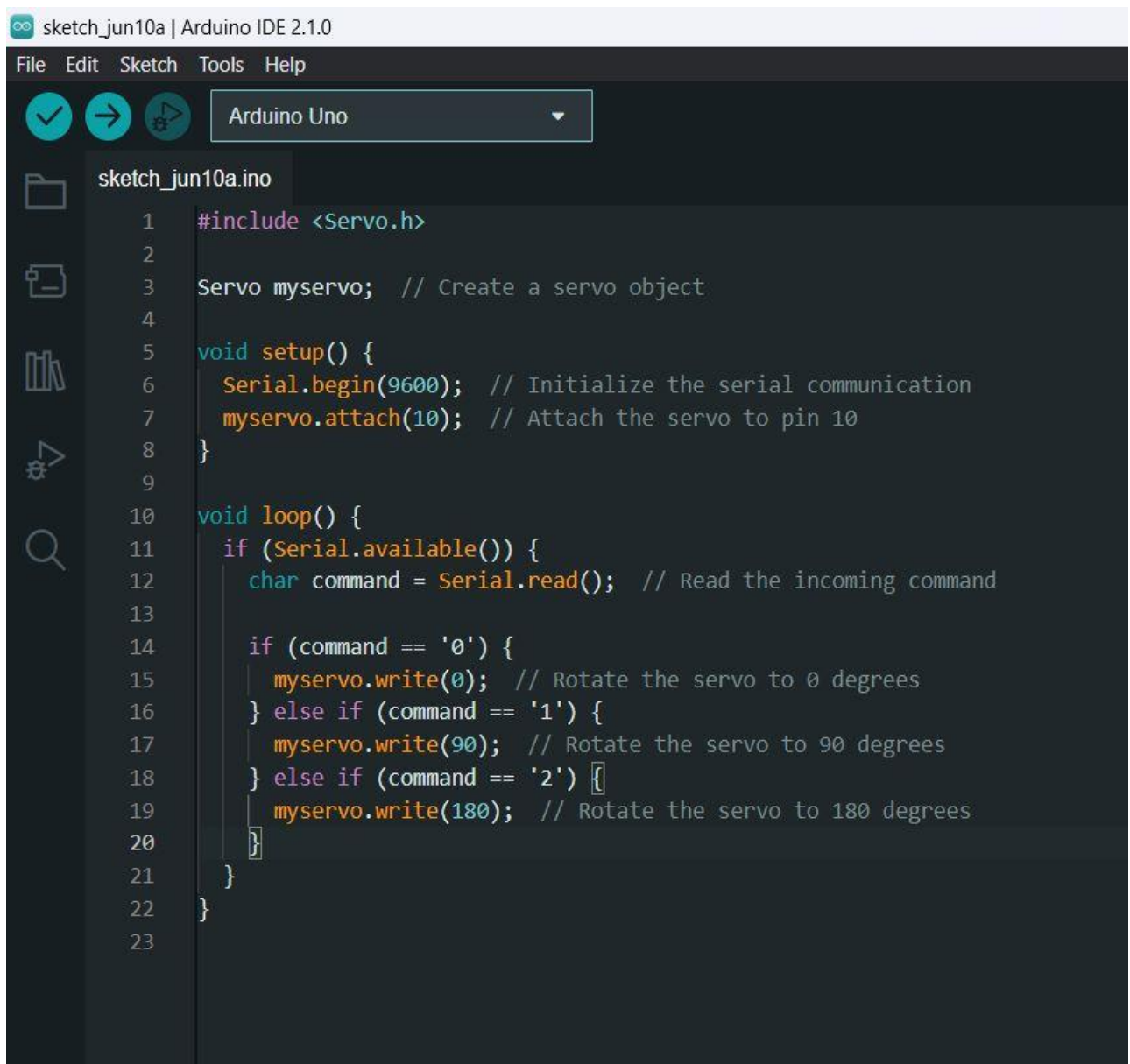


Figure 2: Arduino UNO Board

3.2.2 Physical Connection of the Servo Motor to the Arduino Board

The servo motor is physically connected to the Arduino board using three wires. The servo motor typically has three pins: power (+5V or +6V), ground (GND), and signal (usually labeled as "SIG" or "S"). The power pin is connected to a +5V or +6V pin on the Arduino, the ground pin is connected to a GND pin on the Arduino, and the signal pin is connected to one of the digital pins on the Arduino (e.g., Pin 9).

3.2.3 step-by-step breakdown of the code



```
sketch_jun10a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
[Icons] Arduino Uno
sketch_jun10a.ino
1  #include <Servo.h>
2
3  Servo myservo; // Create a servo object
4
5  void setup() {
6    Serial.begin(9600); // Initialize the serial communication
7    myservo.attach(10); // Attach the servo to pin 10
8  }
9
10 void loop() {
11   if (Serial.available()) {
12     char command = Serial.read(); // Read the incoming command
13
14     if (command == '0') {
15       myservo.write(0); // Rotate the servo to 0 degrees
16     } else if (command == '1') {
17       myservo.write(90); // Rotate the servo to 90 degrees
18     } else if (command == '2') {
19       myservo.write(180); // Rotate the servo to 180 degrees
20     }
21   }
22 }
23
```

Figure 3: Arduino IDE code

Step 1: Importing Libraries

The necessary libraries are imported at the beginning of the code. For servo motor control, the "Servo" library is commonly used.

Step 2: Variable Declaration

A few variables are declared, including the servo object and a variable to store incoming serial data.

Step 3: Servo Initialization

The servo object is attached to a specific pin on the Arduino board using the attach() function. In this case, the servo is attached to Pin 9.

Step 4: Serial Communication Initialization

The serial communication is initialized using the Serial.begin() function. The baud rate (data transfer rate) is specified within the parentheses.

Step 5: Main Loop

The code enters the main loop, where it continuously checks for incoming serial data.

Step 6: Serial Data Reading

The code uses the Serial.available() function to check if there is any data available in the serial buffer. If there is, the data is read using the Serial.read() function and stored in the data variable.

Step 7: Mapping and Servo Movement

The incoming data is mapped to a specific servo angle using the map() function. The map() function re-scales the input value from one range to another. In this case, it maps the incoming data, which is assumed to be between 0 and 180, to the range of angles that the servo motor can rotate (usually 0 to 180 degrees).

Step 8: Servo Positioning

The mapped angle value is passed to the servo.write() function, which rotates the servo motor to the specified angle.

3.2.4 Logic Behind Reading Commands and Mapping to Servo Angles

The code continuously reads incoming data from the serial port. It expects commands in the form of numeric values representing angles between 0 and 180. These angles are then mapped to the range supported by the servo motor (usually 0 to 180 degrees) using the map() function. By mapping the values, the code ensures that the servo motor rotates within its safe operating range.

3.2.5 Additional Features or Considerations

The provided code covers the basic functionality of controlling a servo motor via serial communication. However, depending on the specific project requirements, additional features and considerations could be implemented. These might include handling error conditions (e.g., out-of-range input values), limiting the servo motor movement within specific bounds, or adding calibration routines to adjust the servo's neutral position. The code provided here serves as a starting point and can be expanded upon based on the project's specific needs.

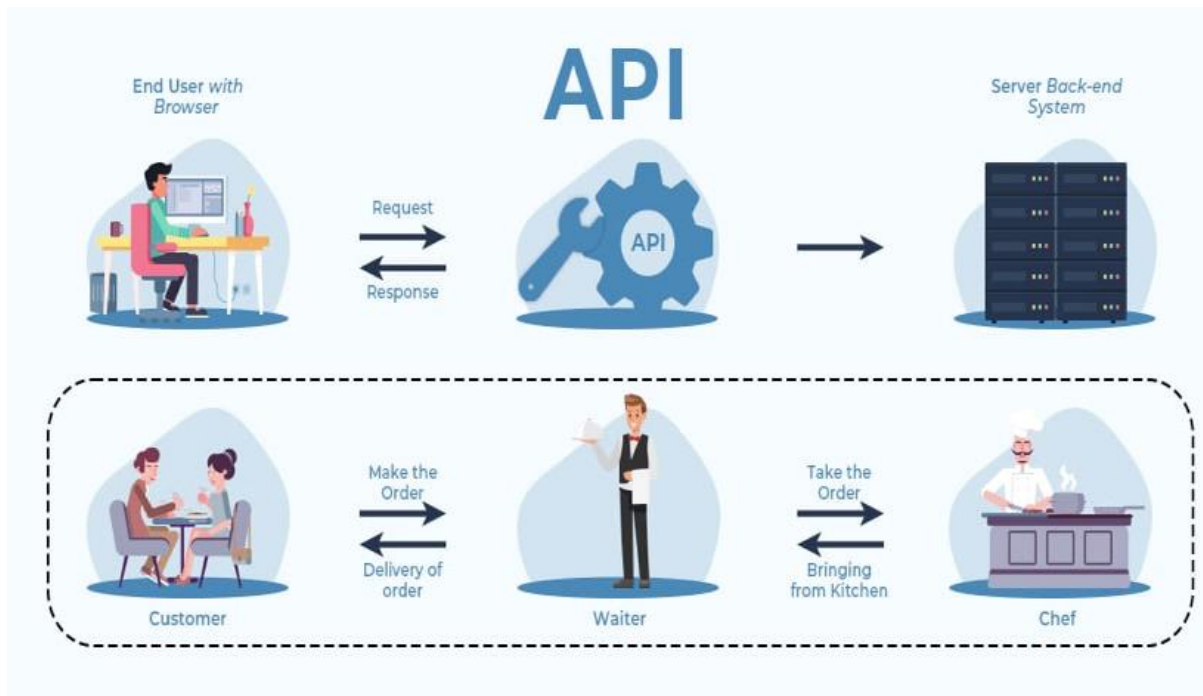


Figure 4: API

3.3 Python code Explanation

3.3.1 purpose of Python code and its role in capturing voice commands and sending them to Arduino.

Purpose:

The Python code serves as an interface between the user's voice input and the Arduino. It enables users to control an Arduino-based system or project using voice commands. The purpose is to provide a hands-free and intuitive way of interacting with the Arduino, allowing users to control various functionalities or trigger specific actions by speaking voice commands.

Role:

The Python code plays several important roles in the voice command system:

Capturing Voice Commands: The code utilizes the SpeechRecognition library to capture voice input from a microphone. It records audio from the microphone and processes it to recognize speech.

Speech Recognition: Once the voice input is captured, the code uses the SpeechRecognition library's speech recognition engine to convert the audio data into text. It employs various speech recognition algorithms and techniques to transcribe the spoken words or phrases into a textual representation.

Preprocessing and Formatting: The code may perform preprocessing on the recognized speech text, such as removing punctuation, converting to lowercase, or applying language-specific processing. It ensures that the voice commands are correctly formatted and ready to be sent to the Arduino.

Serial Communication: After obtaining the recognized voice command as text, the code establishes serial communication with the Arduino using the pyserial library. It opens a serial connection to a specific port on the Arduino board and sends the voice command to the Arduino as data via the serial port.

Data Encoding and Decoding: Before sending the voice command to the Arduino, the code may encode the text data into bytes to match the expected format for serial communication. Similarly, if the Arduino sends any response, the code can decode the received data from bytes to text for further processing.

Interaction and Control: By sending voice commands to the Arduino, the code allows users to interact with the Arduino-based system. The Arduino can interpret and respond to the received commands, enabling it to perform various tasks, control actuators, read sensors, or trigger specific behaviors based on the voice input.

Overall, the Python code acts as a bridge between the user's voice commands and the Arduino, enabling voice-controlled interaction with the Arduino-based system. It captures and processes voice input, establishes serial communication, and facilitates the exchange of data and commands between the user and the Arduino.



Connects to the serial ports and read incoming data.

Hardware-Software system to collect system data.

Figure 5 :Python to Arduino connection

3.3.2 Process of establishing serial communication between Python and Arduino using the pyserial library

```
1  import serial
2  import speech_recognition as sr
3
4  # Establish serial communication with Arduino
5  arduino = serial.Serial('COM4', 9600, timeout=1)
6
7  def rotate_servo(angle):
8      arduino.write(angle.encode()) # Send the angle command to Arduino
9
10 # Initialize the speech recognizer
11 r = sr.Recognizer()
12
13 # Infinite loop to listen for voice commands
14 while True:
15     with sr.Microphone() as source:
16         print("Listening...")
17         audio = r.listen(source)
18
19     try:
20         command = r.recognize_google(audio).lower()
21         print("Voice Command:", command)
22
23         if command in ['close', '0']:
24             rotate_servo('0')
25         elif command in ['open', '1']:
26             rotate_servo('1')
27         else:
28             print("Invalid command. Please try again.")
29
30     except sr.UnknownValueError:
31         print("Unable to recognize speech.")
32     except sr.RequestError as e:
33         print("Error occurred; {0}".format(e))
34
```

Figure 6: Python Code

Establishing Serial Communication with Arduino:

To establish serial communication between Python and Arduino, the code utilizes the pyserial library. This library provides the necessary functionality to connect to the Arduino board through a serial port. It allows Python to send data to and receive data from the Arduino.

Importing the pyserial Library:

At the beginning of the Python code, the pyserial library is imported using the import serial statement. This makes the necessary serial communication functions and classes available for use in the code.

Serial Port Initialization:

To establish communication with the Arduino, the code needs to specify the serial port to which the Arduino is connected. The serial port is typically represented by a string, such as 'COM3' for Windows or '/dev/ttyUSB0' for Linux.

The code initializes the serial port connection using the serial.Serial() function. This function takes the serial port name as a parameter and returns a serial object that represents the connection.

Example:

python

Copy code

```
port = 'COM3' # Replace with the appropriate serial port name
baud_rate = 9600 # Replace with the baud rate used by your Arduino
ser = serial.Serial(port, baud_rate)
```

Baud Rate and Other Parameters:

The baud_rate parameter in the serial.Serial() function specifies the speed of data transmission between Python and the Arduino. It should match the baud rate set in the Arduino code.

You can also specify other parameters such as parity, stop bits, and flow control if necessary. By default, the serial.Serial() function uses commonly used settings for these parameters, but they can be customized if needed.

Opening the Serial Connection:

After initializing the serial object, the code needs to open the serial connection to establish communication with the Arduino. This is done by calling the `open()` method on the serial object.

Example:

python

Copy code

```
ser.open()
```

Sending Data to Arduino:

To send data to the Arduino, the Python code can use the `write()` method of the serial object. This method takes a string or bytes-like object as the parameter and sends it over the serial connection to the Arduino.

Example:

python

Copy code

```
data = 'Hello, Arduino!' # Replace with the data you want to send
ser.write(data.encode()) # Encoding the data as bytes and sending it
```

Receiving Data from Arduino:

If the Arduino sends data back to Python, the Python code can receive it using the `read()` or `readline()` methods of the serial object. These methods read the incoming data from the serial connection.

Example:

python

Copy code

```
received_data = ser.readline() # Reading a line of data from Arduino
print(received_data.decode()) # Decoding the received data and printing it
```

Closing the Serial Connection:

Once the communication with the Arduino is completed, it is good practice to close the serial connection using the `close()` method of the serial object.

Example:

python

Copy code

```
ser.close()
```

By following these steps, the Python code establishes serial communication with the Arduino, allowing data to be sent from Python to the Arduino and vice versa. This enables bidirectional communication and control between the two devices.

3.3.3 Integration of the Speech Recognition Library

The code integrates the SpeechRecognition library to enable voice input. The Speech Recognition library provides an interface to various speech recognition engines, allowing Python to capture and process voice commands from an audio source, such as a microphone.

3.3.4 Step-by-Step Breakdown of Capturing Voice Commands:

Step 1: Importing Libraries

The necessary libraries are imported at the beginning of the code, including pyserial for serial communication and Speech Recognition for voice recognition.

Step 2: Serial Communication Initialization

The code initializes the serial communication with the Arduino using the `serial.Serial()` function. It specifies the port name (e.g., 'COM3' for Windows or '/dev/ttyUSB0' for Linux), baud rate, and other serial communication parameters.

Step 3: Speech Recognition Initialization

The code initializes the speech recognition engine using the `sr.Recognizer()` function from the SpeechRecognition library. This creates a recognizer object that can listen to audio and recognize speech.

Step 4: Microphone Configuration

The code configures the microphone as the audio source for speech recognition. It uses the `sr.Microphone()` function to create a microphone object that the recognizer will listen to.

Step 5: Capturing Audio Input

The code uses a try-except block to capture audio input from the microphone. It listens to the microphone for a specific duration or until it detects silence, and then records the audio data.

Step 6: Speech Recognition

The recorded audio data is passed to the speech recognition engine using the `recognizer.record()` method. The engine processes the audio and attempts to recognize speech by converting it into text.

Step 7: Voice Command Processing

The recognized speech is obtained as text and sent to the Arduino via serial communication using the `serial.write()` method. The voice command is formatted appropriately and encoded as bytes before being sent.

Step 8: Arduino Response (Optional)

If the Arduino sends back any response, the code can read and process it using the `serial.readline()` or `serial.read()` method.

3.3.5 Voice recognition mechanism, such as using a speech recognition service

Voice recognition mechanisms, such as Google's speech recognition service, utilize sophisticated algorithms and machine learning models to convert spoken words or phrases into text. Here is an overview of the voice recognition mechanism and the logic behind matching recognized voice commands to specific servo angles and sending them to Arduino:

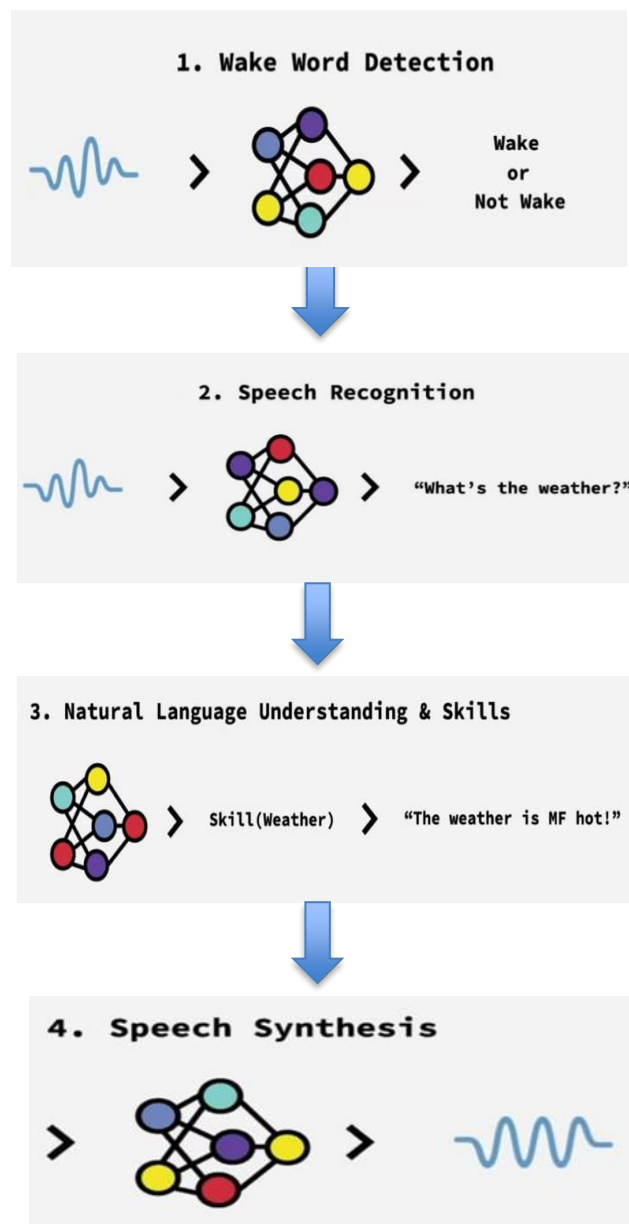


Figure 7: Flowchart for Voice Recognition Process

Voice Recognition Mechanism:

The voice recognition mechanism typically involves the following steps:

Audio Input: The voice recognition system captures audio input from a microphone or an audio source.

Preprocessing: The audio data may undergo preprocessing steps, such as noise reduction, normalization, or feature extraction, to enhance the quality and extract relevant features.

Speech Recognition Engine: The voice data is passed to a speech recognition engine or service that uses machine learning algorithms and models to convert the audio data into textual representations.

Language Modelling: The recognized speech data is further processed using language models and algorithms to improve accuracy, handle context, and handle language-specific nuances.

Text Output: The final output of the voice recognition system is the recognized speech in the form of text or a textual command.



Figure 8: Voice Recognition Mechanism

3.3.6 Logic of matching recognized voice commands to servo angles and sending them to Arduino.

Logic of Matching Recognized Voice Commands to Servo Angles:

Once the voice command is recognized as text, the logic of matching it to specific servo angles and sending them to Arduino involves the following steps:

Defining Command-to-Servo Mapping: The code needs to establish a mapping between recognized voice commands and the corresponding servo angles. This mapping can be defined using conditional statements, dictionaries, or lookup tables. For example, you might define a dictionary where voice commands are the keys, and servo angles are the values.

Processing Recognized Voice Command: The recognized voice command is compared or matched against the defined command-to-servo mapping. This can be done using if-else statements or dictionary lookups.

Extracting Servo Angle: Once a match is found, the corresponding servo angle is extracted from the mapping. This could be a direct mapping, or it may involve some calculations or conversions based on the specific requirements of the servo motor.

Sending Servo Command to Arduino: After obtaining the servo angle, the code can use the established serial communication with Arduino to send the servo command. This typically involves formatting the servo command appropriately and sending it to the Arduino using the serial connection, as discussed earlier.

Integration with Arduino:

The Python code, after matching the recognized voice command to a specific servo angle, uses the established serial communication with Arduino to send the servo command. The Arduino code, which is responsible for controlling the servo motor, will receive the servo command via the serial connection. The Arduino can then interpret the command, map it to the appropriate servo angle range, and move the servo motor accordingly.

By implementing this logic, the voice recognition mechanism and the integration with Arduino enable users to control the servo motor by speaking voice commands. The recognized voice commands are matched to servo angles and sent to Arduino, allowing precise control over the servo motor's position and movement based on the user's voice input.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Outcomes of the project, including the successful control of the servo motor using voice commands.

Successful Control of the Servo Motor Using Voice Commands:

The project successfully achieved the control of a servo motor using voice commands. Users were able to speak voice commands, which were recognized by the voice recognition system. The recognized commands were then mapped to specific servo angles and sent to the Arduino board via serial communication. The servo motor responded accordingly, moving to the desired position based on the voice commands.

4.2 Challenges or limitations encountered during the implementation and how they were overcome.

During the implementation, several challenges and limitations might have been encountered. Some familiar challenges include:

Noise and Ambient Sounds: Background noise and ambient sounds can affect the accuracy of the voice recognition system. It may lead to misinterpretation or inaccurate recognition of voice commands. To mitigate this, noise reduction techniques, microphone placement, or speech enhancement algorithms can be employed.

Speech Variations and Accents: Different users may have variations in their speech patterns, accents, or pronunciation, which can impact the accuracy of the voice recognition system. Training the system with a diverse dataset or incorporating accent-specific models can help improve recognition performance.

Recognition Accuracy: The accuracy of the voice recognition system depends on the underlying algorithms, models, and training data. While modern speech recognition systems, such as Google's speech recognition service, offer high accuracy, there may still be instances of misrecognition or false positives. Fine-tuning the system parameters and training data can help improve accuracy.

4.3 Insights or observations made during the project, such as the accuracy and responsiveness of the voice recognition system.

During the project, several insights and observations may have been made, such as:

Accuracy and Responsiveness: The accuracy and responsiveness of the voice recognition system play a crucial role in the overall user experience. The effectiveness of the system in

accurately recognizing voice commands and quickly translating them into servo movements determines the system's usability and user satisfaction.

Voice Command Design: Designing voice commands that are easy to pronounce and distinct from each other enhances the recognition accuracy. It is important to consider the vocabulary, grammar, and syntax of voice commands to ensure they are easily recognizable by the speech recognition system.

Feedback and Confirmation: Providing feedback or confirmation to the user after a voice command is recognized and executed by the servo motor can improve the user experience. Visual or auditory feedback can be employed to indicate the successful execution of the command.

4.4 Potential applications and implications of the developed system, such as home automation, assistive technology, or human-robot interaction

The developed voice-controlled servo motor system has various applications and implications, including:

Home Automation: Voice-controlled servo motors can be integrated into home automation systems, allowing users to control various devices, such as curtains, doors, or robotic arms, through voice commands. This provides convenience and accessibility in controlling household appliances.

Assistive Technology: Voice-controlled servo motors have potential applications in assistive technology, enabling individuals with limited mobility or physical disabilities to interact with their environment more easily. It can assist in controlling assistive devices, robotic prosthetics, or accessibility features.

Human-Robot Interaction: Integrating voice control with robotic systems enhances human-robot interaction. Users can provide high-level commands to robots using voice, allowing intuitive and natural control over their movements and actions.

Interactive Exhibits and Installations: Voice-controlled servo motors can be utilized in interactive exhibits, museums, or art installations, providing an engaging and immersive user experience. Users can interact with the exhibits using voice commands, triggering specific servo motor movements and visual effects.

Overall, the developed voice-controlled servo motor system has the potential to revolutionize the way users interact with technology, offering hands-free and intuitive control. It opens opportunities for automation, accessibility, and human-robot interaction, with implications in various fields and applications.

CHAPTER 5: CONCLUSIONS

Our project aimed to implement coding in the mechanical aspect to drive mechanical components more effectively and efficiently. We successfully achieved this objective by developing a voice-activated servo motor using Python programming language. This project has demonstrated the practical application of automation and control systems in the field of mechanical engineering.

- The voice command functionality provides a user-friendly and intuitive interface for controlling the servo motor, opening possibilities for various applications.
- This project serves as a foundation for further advancements, such as the development of robotic arms, automatic door systems, or mini plotters.
- By combining programming skills with mechanical engineering principles, we have shown how technology can be harnessed to enhance the functionality of mechanical components.
- The project not only strengthens our understanding of coding and mechanical systems but also highlights the potential for innovation in various industries.
- Overall, this project showcases the power of integration between coding and mechanical engineering, emphasizing the importance of interdisciplinary approaches in driving technological advancements.
- By implementing coding in the mechanical aspect, we have achieved enhanced control and efficiency in driving mechanical components, paving the way for future developments and applications in this exciting field.

SCOPE FOR FUTURE WORK

The voice-activated door using a servo motor, Arduino, and Python programming offers exciting possibilities for future work and enhancements. Here are some potential areas of exploration and improvement:

- 1. Advanced Voice Recognition:** Improve the voice recognition capabilities of the system by utilizing more sophisticated speech recognition algorithms or cloud-based speech recognition services. This can enhance accuracy, expand the range of recognized commands, and support multiple languages.
- 2. Natural Language Processing (NLP):** Integrate NLP techniques to enable the system to understand and interpret more complex voice commands. This would allow users to give more nuanced instructions or engage in conversational interactions with the automated door.
- 3. Integration with Smart Home Systems:** Expand the functionality of the voice-activated door by integrating it with existing smart home systems. This could include integration with voice assistants like Amazon Alexa or Google Assistant, enabling users to control the door through voice commands within a broader home automation ecosystem.
- 4. Security Enhancements:** Implement security features such as user authentication, voice biometrics, or encryption to ensure secure access control to the automated door. This could involve incorporating additional hardware components like fingerprint sensors or face recognition systems.
- 5. Mobile App Integration:** Develop a mobile application that communicates with the Arduino board to control the door remotely. The app could provide additional features like real-time door status updates, remote monitoring, and scheduling of door operations.
- 6. Gesture or Facial Recognition:** Explore alternative control mechanisms for the door, such as using gestures or facial recognition. This could enable touchless control and further enhance accessibility and convenience.
- 7. Machine Learning and AI:** Utilize machine learning algorithms to train the system to adapt and improve over time. This could involve training the system to recognize specific user voices or learn user preferences for door operations.
- 8. Sensor Integration:** Integrate additional sensors like proximity sensors or motion detectors to enhance the door's functionality and responsiveness. This could enable automatic door opening based on proximity or the detection of specific conditions.

9. Customization and Personalization: Develop mechanisms to allow users to customize and personalize the system according to their preferences. This could include customizable voice commands, door opening/closing speeds, or adjustable sensitivity to voice input.

10. Energy Efficiency and Power Management: Implement power-saving mechanisms to optimize energy consumption. This could involve strategies such as sleep modes for the Arduino, efficient power management for the servo motor, or utilizing low-power sensors.

These are just a few examples of the potential future work for a voice-activated door using a servo motor, Arduino, and Python programming. The possibilities are vast, and the direction of future work can be tailored to specific needs, user requirements, and technological advancements.

REFERENCES

1. [1] F. L. I. Dutsinma, "A systematic review of voice assistant usability," A springer nature journal, p. 23, 2022.
2. [2] M. A. Hundekari, "A Survey on Smart Assistant System using Speech Recognition," International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), vol. 9, no. 5, May 2020.
3. [3] Dr.S.Brindha, "Intelligent AI based voice assistant," International Journal of creative research thought (IJCRT), vol. 10, no. 6, pp. 1-5, 6 June 2022.
4. [4] N. Arora, "Geek for Geeks," 6 April 2020. [Online]. Available: <https://www.geeksforgeeks.org/selenium-basics-components-features-uses-and-limitations/>.
5. [5]"JAVA T POINT," [Online]. Available: <https://www.javatpoint.com/python-tutorial>.
6. [6] Chat gpt : <https://chat.openai.com/>