

CRESTA FIRST GRADE COLLEGE

182/145/C, BANNUR MAIN ROAD, MYSORE-570028, KARNATAKA.



Project Report on

“VOICE - CONTROLLED SMART HOME AUTOMATION SYSTEM”

In the partial fulfillment of the requirement for the award of the degree
BACHELOR OF COMPUTER APPLICATION

Submitted by

VARSHINI (RBCI2108) - IOT

Under the guidance of

Ms. Vidyashree H. G

Assistant Professor,

**Department of Computer Science CRESTA First
Grade College, Mysore.**



DEPARTMENT OF COMPUTER APPLICATION

UNIVERSITY OF MYSORE

2023-2024

CRESTA FIRST GRADE COLLEGE

DEPARTMENT OF COMPUTER APPLICATION

Allanahalli, Mysuru-28

2023-2024



CERTIFICATE

This is to certify that the project work entitled “**Voice-Controlled Smart Home Automation System**” is a Bonafede work carried out by **Varshini (RBCI2108)-IoT** at the **Department of Computer Science, CRESTA First Grade College** in partial fulfillment for the award of Bachelor of Computer Applications, University of Mysore during the academic year 2023-2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the Bachelor of Computer Application.

Signature of the Guide

Ms. Vidyashree H. G
Asst. Professor, Dept of BCA,
CRESTA First Grade College,
Mysuru-28

Signature of the Principal

Dr. Rakesh H M
Assistant Principal,
CRESTA First Grad College,
Mysuru-28

Name of the Examiners

1.

2.

Signature with Date

DECLARATION

I, **Varshini (RBCI2108)-IoT**, student of final semester BCA of CRESTA First Grade College, Mysuru, hereby declare that the dissertation entitled “**Voice-Controlled Smart Home Automation System**” has been independently carried out by me at **CRESTA First Grade College, Mysuru** and submitted in partial fulfillment of the requirement for the award of **Bachelor of Computer Applications** affiliated to the **University of Mysore**, during the academic year 2023-2024. Further, the matter embodied in the report is an original and Bonafide work done by me.

To my knowledge, this dissertation has not been submitted to any college or university or published at any time prior to this.

Place: Mysuru

Date:

**Varshini
(RBCI2105) - IoT**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement ground my efforts with success.

At the outset, I would like to thank **ALMIGHTY** for The showers of blessings throughout my work, which enabled me to complete the work successfully.

i express my sincere thanks to **SAMBHAV SHAH, Secretary of Shah Knowledge Foundation**, for his kind help and constant encouragement and for providing me with the necessary facilities to carry out this work successfully.

i want to thank our principal, **Dr. RAKESH H. M**, for giving me all the guidance required apart from being a constant source of inspiration and motivation.

In particular, i would like to take this opportunity to express my Honora, Respect, Deepest Gratitude and Genuine Regards to my guide, **Ms. Vidyashree H.G**, Assistant Professor, Department of Computer Application, for giving me all the guidance required for my project apart from being a constant source of inspiration and motivation.

i owe my special thanks to **My Parents** for their moral support and warm wishes, and, finally, i would like to express appreciation to all **My Guiders** for their unconditional support which helped me to complete this work successfully.

Varshini (RBCI2108)-IoT

ABSTRACT

The advent of the Internet of Things (IoT) has revolutionized the way we interact with everyday devices, enabling smarter and more efficient control over home environments. This project explores the development and implementation of a voice-controlled home automation system utilizing an Arduino microcontroller, a Bluetooth module, and a relay. By integrating these components, users can seamlessly operate household appliances through voice commands issued via the "AMR Voice" Android application. The system's design focuses on simplicity, affordability, and ease of use, making advanced home automation accessible to a broader audience. This paper details the step-by-step construction process, including component selection, circuit design, software development, and testing. The result is a robust and user-friendly solution that enhances convenience, improves energy efficiency, and exemplifies the potential of IoT technologies in modern home automation.

In the era of smart technologies, the integration of voice control into smart home automation systems has

emerged as a transformative advancement. This project presents the development and implementation of a “voice controlled smart home automation system”, The system enables users to effortlessly manage lights, appliances, and security features through natural language voice commands, offering unprecedented convenience and accessibility in smart home environments. By combining state-of-the-art voice recognition technology with open-source platforms, the project aims to redefine the future of home automation, catering to the evolving needs and preferences of modern homeowners. Through a structured approach encompassing design, implementation, and evaluation, the project contributes to the ongoing discourse on smart home technologies and human-computer interaction, paving the way for innovative solutions that enhance the quality of life for individuals and families worldwide.

TABLE OF CONTENTS

Chapter no	Title	Page no
	Acknowledgement	i.
	Abstract	ii.
01	Introduction	01
	1.1 background and motivation	01
	1.2 objectives	01
	1.3 components and their features	01
02	Literature survey	02-03
	2.1 survey papers	
03	System analysis	04-06
	3.1 propose system	04
	3.1.1 overview	04
	3.1.2 key components	04
	3.1.3 system design and implementation	04
	3.1.4 advantages	04
	3.1.5 limitations	04
	3.2 difference between existing system and proposed system	05
	3.3 objectives	06
04	System requirements	07
	4.1 hardware requirements	07
	4.1.1 methodology	07
	4.2 software requirements	07
05	System design	08-14
	5.1 arduino UNO	08
	5.1.1 Features of Arduino	08

	5.1.2 arduino UNO board description	09-11
	5.2 arduino UNO	12
	5.3 HC-06	12-13
	5.4 5v 1 channel relay module	13-14
06	Software description	15-25
	6.1 procedure to install Arduino software(IDE)	15-18 18-22
	6.2 program structure	22-24
	6.3 arduino code libraries	24-25
	6.4 declaration	
07	Implementation	26-29
	7.1 circuit design and assembly	26
	7.2 software development	26-27
	7.3 optimization and documentation	27-28
	7.4 advantages	28
	7.5 applications	28-29
08	Testing and result	30
	8.1 testing and result	30
	8.2 testing and debugging	30
09	Conclusion and future enhancement	31-32
10	References	33

LIST OF FIGURES

Figures	Name	Page no
5.1	Arduino uno	08
3.1.4	Arduino uno board	09
5.2	Arduino board	12
5.3	HC-06	12
5.4	5v1 channel relay module	13
3.1.5.1(b)	A to mini-B cable	15
3.1.5.2	Downloading Arduino IDE	15
3.1.5.4	Launching Arduino IDE	16
3.1.5.5(a)	Creating a new project	17
3.1.5.5(b)	Opening an existing project	17
3.1.5.6	Selecting the Arduino board	18
3.1.4.7	Selecting the serial port	18
3.1.4.8	Arduino IDE tool bar	19
3.1.6	Structure of a sketch	20

3.2.2.1	Including libraries on IDE	23
---------	----------------------------	----

LIST OF TABLES

Table no	Name	Page no
3.1.4	Pin configuration	09-11
5.4	Relay module	13-14

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

In the contemporary world, technology continues to evolve at an unprecedented pace, significantly impacting various facets of daily life. Among these advancements, the Internet of Things (IoT) stands out as a transformative force, particularly in the realm of home automation. The concept of smart homes, where devices communicate and operate autonomously or under minimal human intervention, has shifted from a futuristic idea to a present-day reality. This evolution is driven by the increasing accessibility of microcontrollers, sensors, and wireless communication modules, coupled with the growing ubiquity of smartphones and smart assistants.

The motivation behind this project is to harness the capabilities of IoT to create a voice-controlled home automation system. Such a system not only adds convenience by allowing users to control appliances through simple voice commands but also enhances the overall efficiency and security of the home environment. Traditional home automation systems often involve complex setups and high costs, limiting their adoption. In contrast, this project aims to develop a cost-effective and user-friendly solution, leveraging the Arduino platform, which is known for its versatility and ease of use.

1.2 OBJECTIVES

The primary objectives of this project are:

1. To design and implement a voice-controlled home automation system using Arduino.
2. To integrate a Bluetooth module for wireless communication between the Arduino and an Android device.
3. To develop a functional interface using the "AMR Voice" Android application to receive and interpret voice commands.
4. To control a relay module for switching household appliances on and off based on the received voice commands.
5. To ensure the system is reliable, responsive, and easy to set up for users with minimal technical expertise.

1.3 COMPONENTS AND THEIR FUNCTIONS

1. Arduino Board (e.g., Arduino Uno): Acts as the central processing unit for the system, interpreting commands and controlling the relay.
2. Bluetooth Module (e.g., HC-05 or HC-06): Facilitates wireless communication between the Arduino and the Android device.
3. Relay Module: Operates as an electronic switch, allowing the Arduino to control high-voltage appliances safely.
4. Android Device with "AMR Voice" App: Serves as the user interface, capturing voice commands and transmitting them to the Arduino via Bluetooth.
5. Power Supply and Jumper Wires: Provide necessary power and connectivity to the devices.

CHAPTER 2

LITERATURE SURVEY

2.1 SURVEY

The literature surrounding smart home automation systems and voice control technology provides valuable insights into the evolution, challenges, and opportunities in this rapidly growing field. Several studies have explored various aspects related to smart homes, voice assistants, and human-computer interaction, shedding light on the following key themes:

1. Voice Control in Smart Homes:

Research on voice-controlled smart home systems has highlighted the potential benefits of using natural language voice commands to interact with smart devices. Studies have emphasized the convenience and accessibility offered by voice control interfaces, enabling users to perform tasks hands-free and with minimal effort

2. User Experience and Acceptance:

Understanding user perceptions and acceptance of voice-controlled smart home systems is crucial for their widespread adoption. Literature in this area has examined factors influencing user experience, including ease of use, reliability, privacy concerns, and satisfaction with voice recognition accuracy

3. Security and Privacy:

Security and privacy concerns are paramount in smart home automation systems, particularly concerning voice data collection and processing by voice assistants. Literature has addressed privacy implications of voice controlled systems, highlighting the need for transparent data handling practices and user consent mechanisms

4. Future Trends and Directions:

Anticipating future trends and advancements in voice-controlled smart home technology is essential for guiding research and development efforts. Literature has discussed emerging trends such as context-aware voice recognition, personalized user interactions, and integration with artificial intelligence and machine learning techniques

Paper :1

- **Reference on Voice-Controlled Home Automation using Arduino:**

- **Title:** "Voice Controlled Home Automation System Using Arduino"
- **Authors:** T. S. Somashekar, K. R. Shobha, and K. R. Venugopal
- **Published in:** International Journal of Computer Applications (IJCA)
- **Abstract:** This paper presents a voice-controlled home automation system using an Arduino microcontroller. The system integrates a Bluetooth module to receive voice commands and control various household devices. The authors discuss the hardware and software implementation, providing insights into the design and functionality of the system

Paper:2

- **Reference on Bluetooth-Based Home Automation:**

- **Title:** "Bluetooth Based Home Automation System Using Cell Phone"
- **Authors:** R.Piyare and M.Tazil

-
- **Published in:** IEEE 15th International Symposium on Consumer Electronics

CHAPTER 3

SYSTEM ANALYSIS

3.1 Proposed System

3.1.1 Overview

The proposed system aims to offer a simplified, cost-effective, and user-friendly solution for voice-controlled home automation using readily available components such as Arduino, a Bluetooth module, and a relay. The system allows users to control home appliances via voice commands issued through an Android app, specifically the "AMR Voice" app, which communicates directly with the Arduino via Bluetooth.

3.1.2 Key Components

1. Arduino Board (e.g., Arduino Uno): The central microcontroller that processes commands and controls the relay.
2. Bluetooth Module (e.g., HC-05 or HC-06): Facilitates wireless communication between the Arduino and the Android device.
3. Relay Module: Acts as an electronic switch to control high-voltage appliances.
4. Android Device with "AMR Voice" App: Captures voice commands and transmits them to the Arduino via Bluetooth.
5. Power Supply and Jumper Wires: Provide necessary power and connectivity to the components.

3.1.3 System Design and Implementation

1. Hardware Assembly:
 - Connect the Bluetooth module to the Arduino, ensuring correct pin connections (VCC, GND, TX, RX).
 - Connect the relay module to a digital pin on the Arduino, providing power and ground connections.
2. Software Development:
 - Write Arduino code to handle Bluetooth communication and control the relay based on received commands.
 - Configure the "AMR Voice" app to recognize voice commands and send corresponding text strings to the Bluetooth module.
3. Testing:
 - Conduct thorough testing in various scenarios to ensure reliability and responsiveness.
 - Optimize the system for performance and ease of use.

3.1.4 Advantages

1. Cost-Effective: Utilizes affordable components, making it accessible to a wider audience.
2. Simplicity: Easy to set up and use, suitable for users with minimal technical expertise.
3. Privacy: Local processing of commands ensures greater privacy compared to cloud-based systems.
4. Offline Capability: Operates independently of internet connectivity, relying solely on Bluetooth communication.

3.1.5 Limitations

1. Limited Range: Bluetooth communication has a limited range compared to Wi-Fi-based systems.
2. Basic Functionality: Currently supports only simple on/off commands, lacking advanced features of existing systems.
3. Single Device Control: Initially designed to control a single appliance, requiring further development for multi-device integration.

3.2 Difference Between Existing System and Proposed System

● Functional Comparison

- Voice Assistants vs. Bluetooth Control:

Existing systems use advanced voice assistants integrated with cloud services for complex commands and multi-device control.

The proposed system uses Bluetooth communication via an Android app for basic on/off commands.

- Smart Hubs vs. Arduino:

Existing systems often rely on smart hubs to connect and manage multiple devices.

The proposed system uses an Arduino microcontroller to directly control a relay, simplifying the setup.

- Cloud Services vs. Local Processing:

Existing systems leverage cloud services for command processing, enabling advanced functionalities but raising privacy concerns.

The proposed system processes commands locally on the Arduino, enhancing privacy and reliability in offline scenarios.

● Cost and Accessibility

- High Cost vs. Affordability:

Existing systems can be expensive due to the cost of smart assistants, hubs, and compatible devices.

The proposed system is designed to be affordable, utilizing low-cost components like Arduino and Bluetooth modules.

- Complex Setup vs. Simplicity:

Setting up existing systems can be complex, requiring technical knowledge and configuration of multiple devices.

The proposed system offers a straightforward setup process, suitable for beginners and non-technical users.

● Privacy and Security

- Cloud Dependency vs. Local Control:

Existing systems depend on cloud services, which can pose privacy risks due to continuous data transmission and storage.

The proposed system operates locally, mitigating these privacy concerns by avoiding cloud-based processing.

- Internet Requirement vs. Offline Capability:

Existing systems often require a stable internet connection for operation, limiting functionality in areas with poor connectivity.

The proposed system operates independently of the internet, using Bluetooth for direct communication.

● Customization and Scalability

- Advanced Features vs. Basic Functionality:

Existing systems offer advanced features like scheduling, automation, and multi-device scenarios.

The proposed system currently supports basic on/off control but provides a foundation for future enhancements.

- Multi-Device Integration vs. Single Device Control:

Existing systems can manage numerous devices through smart hubs and cloud platforms.

The proposed system is initially designed for single device control, with potential for expansion through additional relays and sensors.

3.3 Objectives

- 1. Design a Voice-Controlled Home Automation System:** Create a system that allows users to control home appliances using voice commands.
- 2. Integrate Bluetooth Communication:** Implement Bluetooth connectivity between the Arduino and an Android device for wireless control.
- 3. Develop an Android Interface:** Utilize the "AMR Voice" app to capture and send voice commands to the Arduino.
- 4. Control Relay Module:** Use the Arduino to control a relay module for switching appliances on and off.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

1. Arduino Board : (e.g., Arduino Uno)

- Acts as the main microcontroller to process commands and control the relay.

2. Bluetooth Module : (e.g., HC-05 or HC-06)

- Facilitates wireless communication between the Arduino and the Android device.

3. Relay Module:

- Allows the Arduino to switch high-voltage appliances on and off safely.

4. Power Supply:

- Provides necessary power for the Arduino and relay module (typically a 9V adapter or battery).

5. Jumper Wires:

- Connect the various components securely.

6. Android Device with "AMR Voice" App:

- Captures voice commands and transmits them to the Arduino via Bluetooth.

4.1.1 Methodology

Component Selection and Acquisition

The first step in developing the voice-controlled home automation system is selecting and acquiring the appropriate components. The following components were chosen for their affordability, availability, and compatibility:

1. Arduino Board (e.g., Arduino Uno): A versatile and widely-used microcontroller that serves as the system's brain.

2. Bluetooth Module (e.g., HC-05 or HC-06): Facilitates wireless communication between the Arduino and the Android device.

3. Relay Module: Acts as an electronic switch, allowing the Arduino to control high-voltage appliances safely.

4. Android Device with "AMR Voice" App: Captures voice commands and transmits them to the Arduino via Bluetooth.

5. Power Supply and Jumper Wires: Provide necessary power and connectivity to the components.

4.2 Software Requirements

1. Arduino IDE:

- Used for writing, compiling, and uploading code to the Arduino board.

2. "AMR Voice" Android App:

- Captures voice commands and sends them to the Arduino via Bluetooth.

3. Arduino Libraries:

- SoftwareSerial library for handling Bluetooth communication if using an Arduino Uno.

4. Bluetooth Pairing Software:

- Android settings for pairing the device with the Bluetooth module.

CHAPTER 5

SYSTEM DESIGN

5.1 ARDUINO UNO



The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

5.1.1 Features of Arduino

The key features of Arduino have been discussed below as follows:

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.

Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package

5.1.2 Arduino UNO board description

We chose the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduino have majority of these components in common.

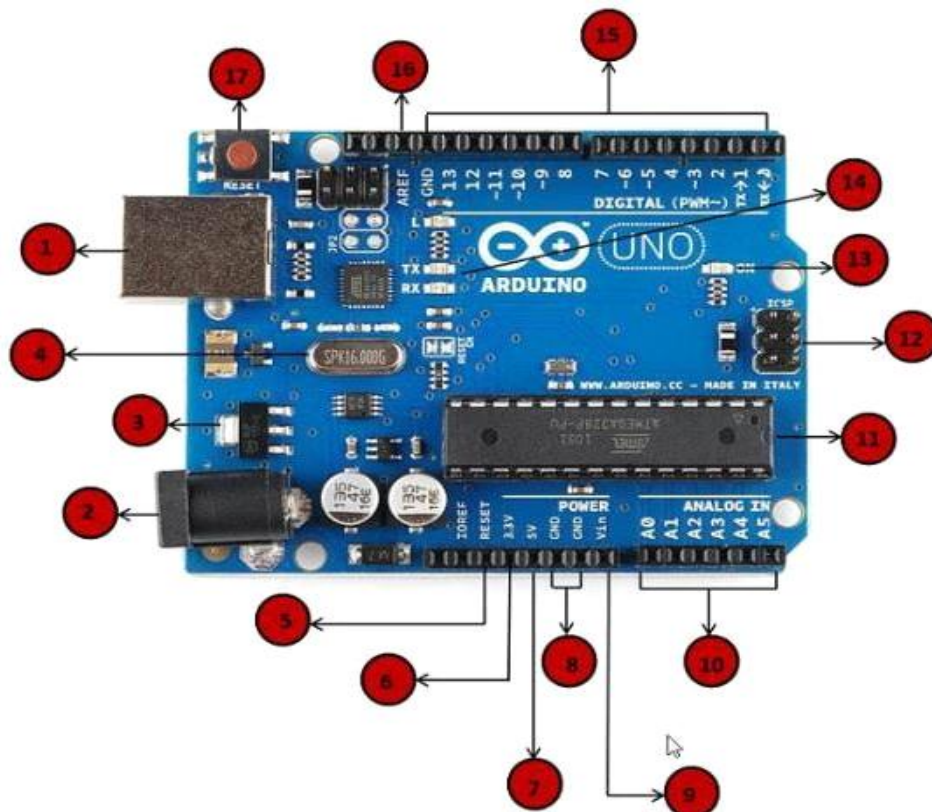







Fig. 3.1.4 Arduino UNO Board

The pin configuration of the Arduino UNO Board is discussed in the below Table 3.1.4:

Pin No.	Description
1	Power USB: Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
2	Power (Barrel Jack): Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

	<p>Voltage Regulator: The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.</p>
	<p>Crystal Oscillator: The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz</p>
	<p>Arduino Reset: You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).</p>
	<p>Pins (3.3, 5, GND, Vin)</p> <p>3.3V (6) : Supply 3.3 output volt</p> <p>5V (7) : Supply 5 output volt</p> <p>(Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.)</p> <p>GND (8- Ground): There are several GND pins on the Arduino, any of which can be used to ground your circuit.</p> <p>Vin (9) : Even this pin can be used to power the Arduino UNO board from an external power source, like AC mains power supply.</p>
	<p>Analog pins: This board has six analog input pins A0 to A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>







	<p>Main microcontroller: Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.</p>
	<p>ICSP pin: Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.</p>
	<p>Power LED indicator: This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
	<p>TX and RX LEDs: On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
	<p>Digital I/O: The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labelled “~” can be used to generate PWM.</p>
	<p>AREF: AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the Analog input pins.</p>

Table 3.1.4 Pin Configuration of Arduino UNO Board

5.2 ARDUINO UNO



The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

5.3 HC-06



HC-06 is the popular Bluetooth module. This HC06 module is slave mode only. It's very easy to add wireless serial connectivity for your device with this module. Examples for Arduino and other boards are available. Once you pair with other Bluetooth devices you work like with normal UART to exchange data.

This module has built-in 3.3V voltage regulator and helps to break out the important pins (Vcc, Gnd, Txd, Rxd). Based on CSR BC4 chip, Bluetooth V2.0 + EDR. You can set the baud rate, name and pair password by AT commands when there is no Bluetooth connection. This module is a slave- it can be paired with Computer- Bluetooth master- mobile phone- PDA- PSP and so on.

Features:

Bases at CSR BC04 Bluetooth technology.

with build-in 2.4GHz PCB antenna

It's at the Bluetooth class 2 power level.

Range test: 10 meters

Operating voltage: 3.3V to 6V DC

Operating current in pairing is in the range of 30~40mA.

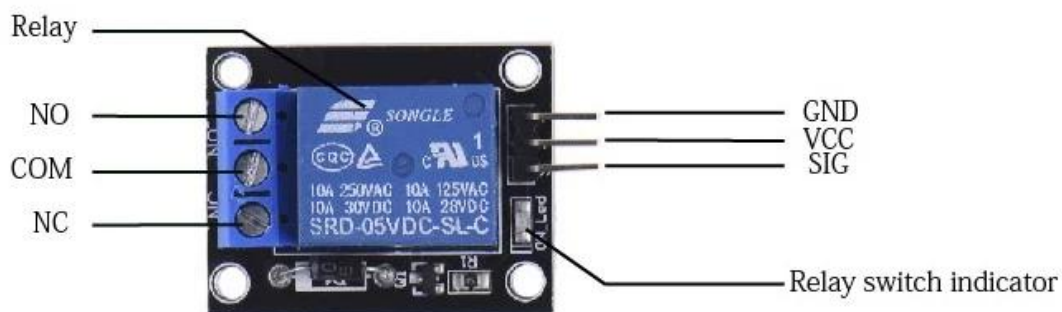
Operating current in communication is 8mA.

Interface via UART, default baud rate at 9600 bps

Operating temperature range: -25 °C – +75°C

Dimensions 27mm×13mm×2.2mm

5.4 5V 1 Channel Relay Module



This 1-channel 5V control Single-Pole Double-Throw (SPDT) High-level trigger AC power relay board can be controlled directly via a [microcontroller](#) and switch up to 10A at 250 VAC. The inputs of 1 Channel 5V Relay Module are isolated to protect any delicate control circuitry.

The default state of the [relay](#) when the power is off for COM (Power) to be connected to NC (Normally Closed). This is the equivalent of setting the relay board IN pin to HIGH (has +5V sent to it).

NO .of channels	1
Trigger voltage(VDC)	5
Trigger Current (mA)	20
Switching Voltage (VAC)	250@10A
Switching Voltage (VDC)	30@10A
Length (mm)	38
Width (mm)	26
Height (mm)	18

Weight (gm)	13
Shipment Weight	0.017 kg
Shipment Dimensions	$7 \times 5 \times 2$ cm

CHAPTER 6

SOFTWARE DISCRIPTION

Arduino ide is the software used to write-compile-upload program to arduino.Its a open source software

6.1 Procedure to Install Arduino Software (IDE)

Step 1 – First we must have an Arduino board and a USB cable. In case we use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, we will need a standard USB cable (A plug to B plug), the kind we would connect to a USB printer as shown in Fig. 3.1.5.1 (a). In case we use Arduino Nano, we will need an A to Mini-B cable instead as shown in Fig. 3.1.5.1 (b).



Fig 3.1.5.1 (b) A to Mini-B Cable

Step 2 – Download Arduino IDE Software.

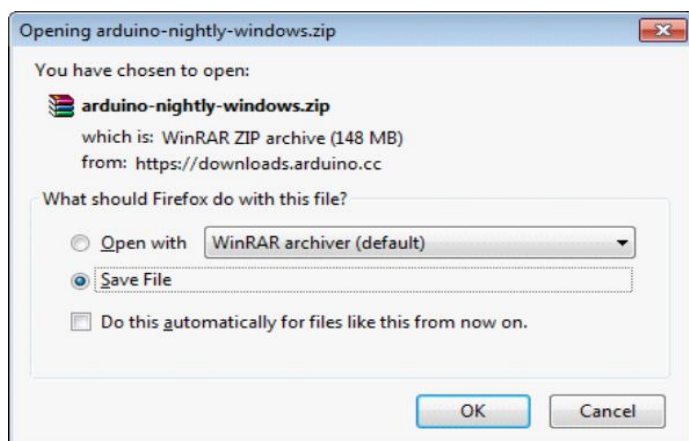


Fig. 3.1.5.2 Downloading Arduino IDE

One can get different versions of Arduino IDE from the [Download page](#) on the Arduino Official website. We must select our software, which is compatible with our operating system (Windows, IOS, or Linux). Unzip the file, after downloading it completely.

Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 – Launch Arduino IDE.

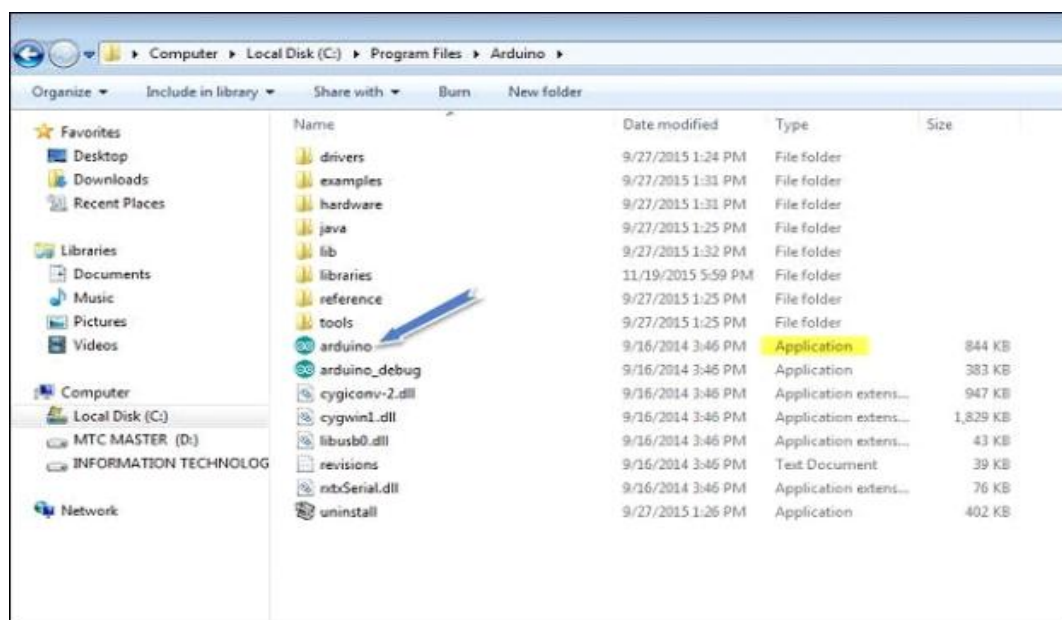


Fig. 3.1.5.4 Launching Arduino IDE

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

Step 5 – Open your first project.

Once the software starts, we have two options:

- Create a new project.

To create a new project, select File → **New**, as shown in Fig. 3.1.5.5 (a).

- Open an existing project example.

To open an existing project example, select File → Example → Basics → Blink, as shown in Fig. 3.1.5.5 (b).

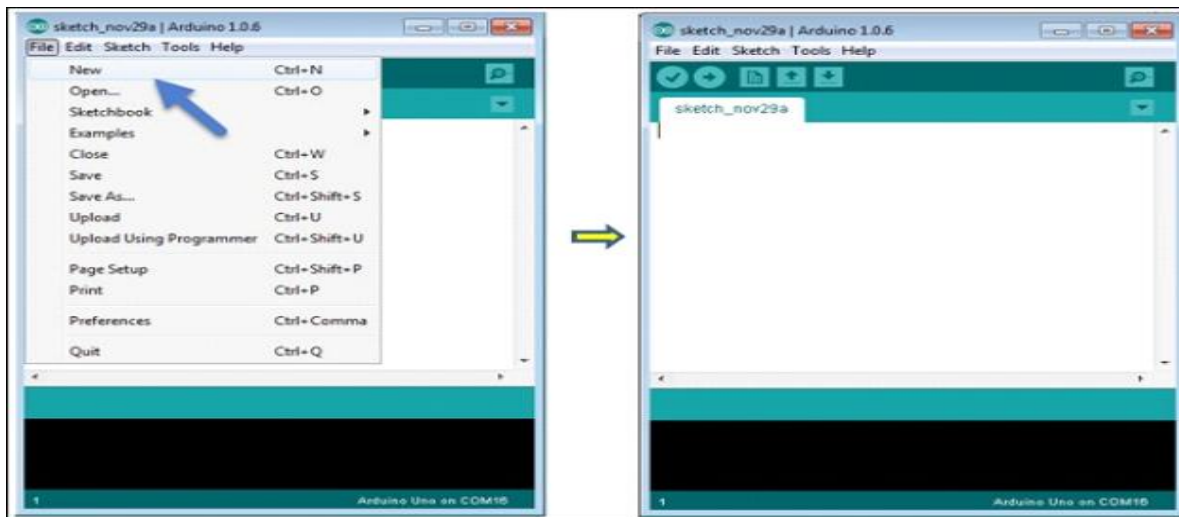


Fig. 3.1.5.5 (a) Creating a New Project

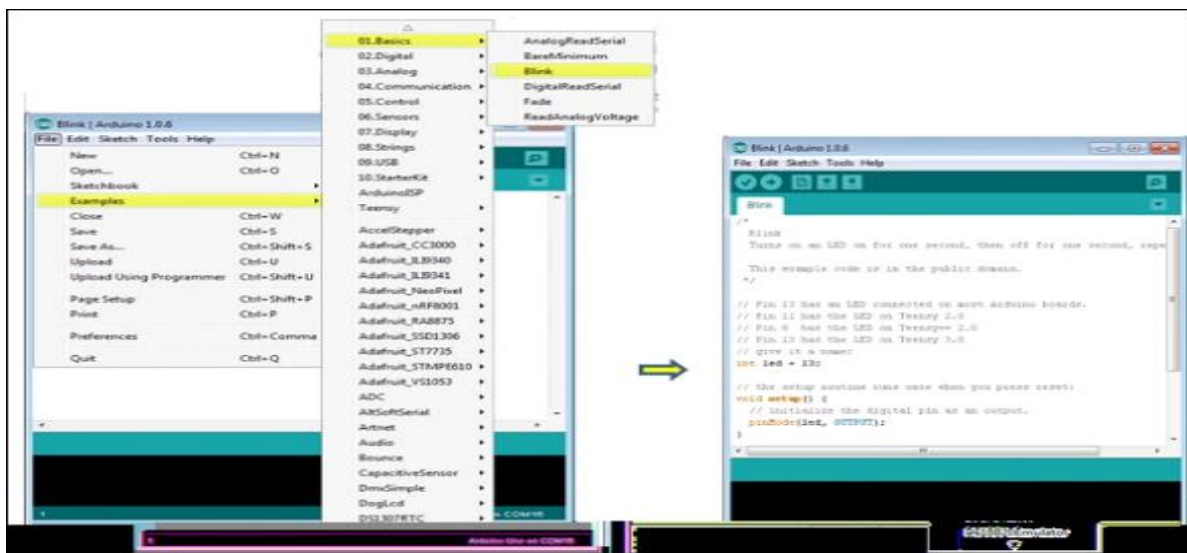


Fig. 3.1.5.5 (b) Opening an Existing Project

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. We can select any example from the list

Step 6 – Select the respective Arduino board.

To avoid any error while uploading our program to the board, we must select the correct Arduino board name, which matches with the board connected to our computer.

Go to Tools → Board and select the board.

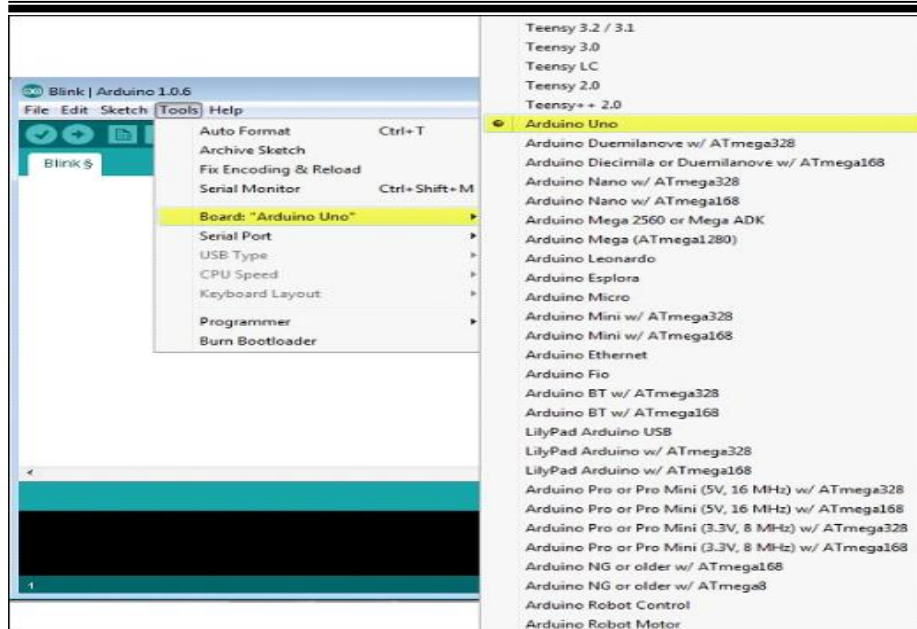


Fig. 3.1.5.6 Selecting the Arduino Board

Step 7 – Select the serial port.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu.

This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

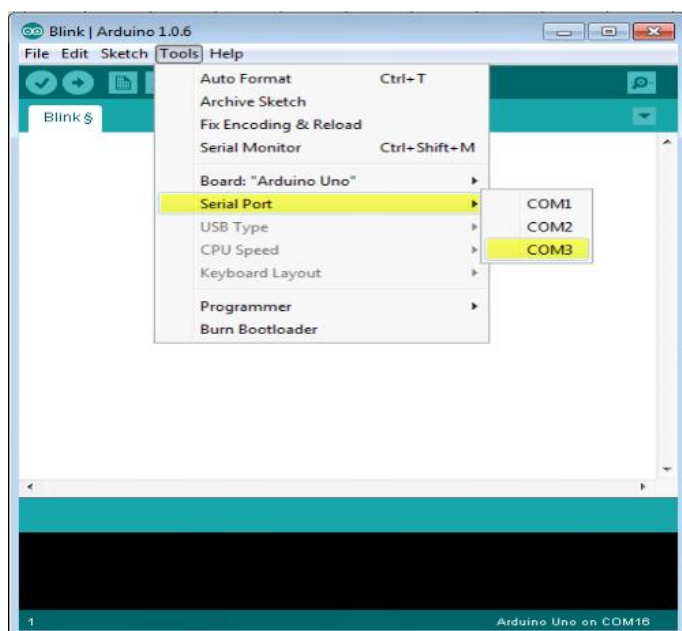


Fig. 3.1.4.7 Selecting the Serial Port

Step 8 – Upload the program to the board.

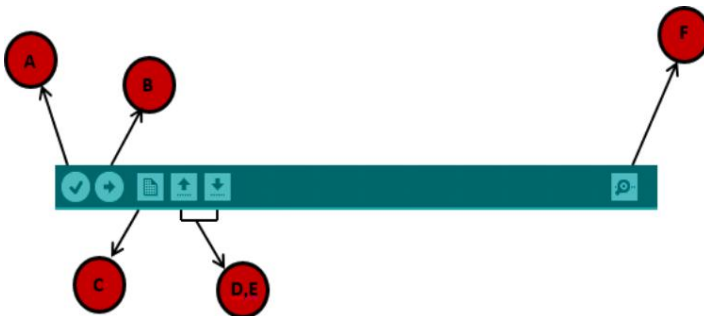


Fig. 3.1.4.8 Arduino IDE Toolbar

Before explaining how to upload our program to the board, we should know the function of each symbol appearing in the Arduino IDE toolbar.

A – Used to check if there is any compilation error.

B – Used to upload a program to the Arduino board.

C – Shortcut used to create a new sketch.

D – Used to directly open one of the example sketch.

E – Used to save your sketch.

F – Serial monitor used to send and receive the serial data from the board.

Now, simply click the "Upload" button in the environment. Wait few seconds; we will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

6.2 Program Structure

In this section, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL. Arduino programs are called sketch. Arduino programs can be divided in three main parts: **Structure**, **Values** (variables and constants), and **Functions**. Let us learn about the Arduino software program, step by step, and how to write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions:

- Setup() function
- Loop() function

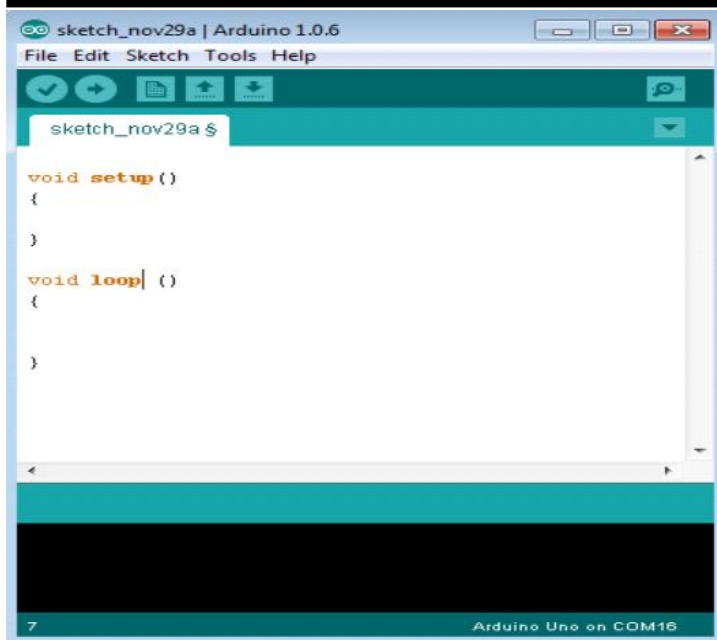


Fig. 3.1.6 Structure of a Sketch

The **setup ()** function is called when a sketch starts. It is used to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

After creating a **setup ()** function, which initializes and sets the initial values, the **loop ()** function does precisely what its name suggests, and loops consecutively, allowing our program to change and respond. It actively controls the Arduino board.

Programming using Embedded C

C is a high-level programming language intended for system programming. Embedded C is an extension that provides support for developing efficient programs for embedded devices. Yet, it is not a part of the C language. In our Internship program, we employed Embedded C programs to write sketches to be dumped on Arduino Uno.

Introduction to Embedded C

Embedded C programming language is an extension to the traditional C programming language that is used in embedded systems. The embedded C programming language uses the same syntax and semantics as the C programming language.

The only extension in the Embedded C language from normal C Programming Language is the I/O Hardware Addressing, fixed-point arithmetic operations, accessing address spaces, etc.

Basic Structure of Embedded C Program:

The embedded C program has a [structure similar to C programming](#). The five layers of Embedded C programming structure are:

- Comments
- Pre-processor directives
- Global declaration
- Local declaration
- Main function()

The whole code follows the below outline. This is the basic structure of the embedded c program. Each code has a similar outline. Now let us learn about each of this layer in detail.

Outline of an Embedded C code is as shown below:

- Multiline Comments Denoted using `/*.....*/`
- Single Line Comments Denoted using `//`
- Pre-processor Directives `#include<...>` or `#define`
- Global Variables Accessible anywhere in the program
- Function Declarations Declaring Function
- Main Function Main Function, execution begins here
- 7. {
- Local Variables Variables confined to main function
- Function Calls Calling other Functions
- Infinite Loop Like while (1) or for (;;)
 - Statements
 - 12.
 - 13.
 - 14. }
- 15. Function Definitions Defining the Functions
- 16. {
- Local Variables Local Variables confined to this Function
- Statements
- 19.
- 20.

```
21.    }
```

Comment Section: Comments are simple readable text, written in code to make it more understandable to the reader. Usually comments are written in `//` or `/* */`.

Example: `//Test program`

Pre-processor Directives Section: The Pre-Processor directives tell the compiler which files to look in to find the symbols that are not present in the program.

For Example, in 8051 Keil compiler we use,

- `#include<reg51.h>`

Global Declaration Section: The global variables and the user-defined functions are declared in this part of the code. They can be accessed from anywhere.

- ```
1. void delay (int);
```

**Local Declaration Section:** These variables are declared in the respective functions and cannot be used outside the main function.

**Main Function Section:** Every C programs need to have the main function. So does an embedded C program. Each main function contains 2 parts. A declaration part and an Execution part. The declaration part is the part where all the variables are declared. The execution part begins with the curly brackets and ends with the curly close bracket. Both the declaration and execution part are inside the curly braces.

**Example:**

- ```
1.    void main(void) // Main Function 2.  {
3.    P1 = 0x00;
4.    while(1)
5.    {
6.    P1 = 0xFF;
7.    delay(1000);
8.    P1 = 0x00;
9.    delay(1000);
10.   }
11.   }
```

Function Definition Section: The function is defined in this section.

6.3 Arduino Code libraries

Library Structure

- A library is a folder comprised with C++ (.cpp) code files and C++ (.h) header files.
- The .h file describes the structure of the library and declares all its variables and functions.
- The .cpp file holds the function implementation.

Importing Libraries

The first thing to do is to find the library we want to use out of the many libraries available online. After downloading it to our computer, we just need to open Arduino IDE and click on Sketch > Include Library > Manage Libraries. We can then select the library we want to import. Once the process is complete the library will be available in the sketch menu.

Arduino Code Explanation

Arduino code is written in C++ with an addition of special methods and functions. C++ is a human-readable programming language. When we create a sketch (the name given to Arduino code files), it is processed and compiled to machine language.

Code Structure

The basic concepts which one should know to write a program on Arduino IDE are discussed below:

Libraries

In Arduino, much like other leading programming platforms, there are built-in libraries that provide basic functionality. In addition, its possible to import other libraries and expand the Arduino board capabilities and features. These libraries are roughly divided into libraries that interact with a specific component or those that implement new functions.

To import a new library, we need to go to Sketch > Import Library

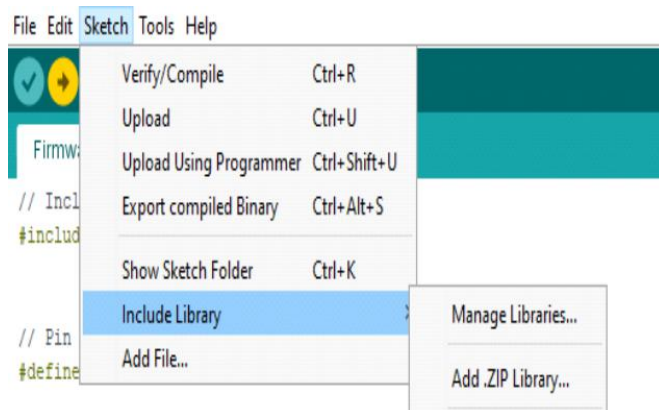


Fig. 3.2.2.1 Including Libraries on IDE

In addition, at the top of our file, we have to use `#include` to include external libraries. We can also create custom libraries to use in isolated sketches.

Pin Definitions

To use the Arduino pins, we need to define which pin is being used and its functionality. A convenient way to define the used pins is by using:

#define pinName pinNumber.

The functionality is either input or output and is defined using the `pinMode ()` method in the setup section.

6.4 Declarations

Variables

Whenever we are using Arduino, we need to declare global variables and instances to be used later on. In a nutshell, a variable allows us to name and store a value to be used in the future. For example, we would store data acquired from a sensor in order to use it later. To declare a variable we simply define its type, name and initial value. Its worth mentioning that declaring global variables isn't an absolute necessity. However, it's advisable to declare variables to make it easy to utilize our values further down the line.

Instances

In software programming, a class is a collection of functions and variables that are kept together in one place. Each class has a special function known as a constructor, which is used to create an instance of the class. In order to use the functions of the class, we need to declare an instance for it.

● Setup()

Every Arduino sketch must have a setup function. This function defines the initial state of the Arduino upon boot and runs only once.

Here we'll define the following:

- Pin functionality using the `pinMode` function
- Initial state of pins.
- Initialize classes.
- Initialize variables.
- Code logic

● Loop()

The loop function is also a must for every Arduino sketch and executes once setup () is complete. It is the main function and as its name hints, it runs in a loop over and over again. The loop describes the main logic of our circuit.

● Code Logic

The basic Arduino code logic is an if-then structure and can be divided into 4 blocks:

- **Setup** - will usually be written in the setup section of the Arduino code, and performs things that need to be done only once, such as sensor calibration.

- **Input** - at the beginning of the loop, read the inputs. These values will be used as conditions (if) such as the ambient light reading from an LDR using `analogRead ()`.
- **Manipulate Data** - this section is used to transform the data into a more convenient form or perform calculations. For instance, the `AnalogRead ()` gives a reading of 0-1023 which can be mapped to a range of 0-255 to be used for PWM. (See `analogWrite ()`)
- **Output** - this section defines the final outcome of the logic (then) according to the data calculated in the previous step. Looking at an example of the LDR and PWM, turns on an LED only when the ambient light level goes below a certain threshold.
- **From Software to Hardware**

There is a lot to be said of Arduinos software capabilities, but its important to remember that the platform is comprised of both software and hardware. The two work in tandem to run a complex operating system.

Code → Compile → Upload → Run

At the core of Arduino, is the ability to compile and run the code.

After writing the code in the IDE we need to upload it to the Arduino. Clicking the Upload button (the right-facing arrow icon), will compile the code and upload it if it passed compilation. Once the upload is complete, the program will start running automatically.

CHAPTER 7

IMPLEMENTATION

The implementation phase begins with assembling the hardware. The Bluetooth module is connected to the Arduino, ensuring the correct alignment of RX and TX pins to facilitate seamless data transmission. The relay module is connected to a designated digital pin on the Arduino, with the relay's power supply secured to prevent electrical hazards.

The Arduino code is developed to handle serial communication via the Bluetooth module. It listens for specific strings (e.g., "turn on" and "turn off") and triggers the relay accordingly. This code includes safety checks to handle unexpected inputs and ensure reliable operation.

On the software side, the "AMR Voice" app is configured to recognize voice commands and send corresponding text strings to the Bluetooth module. The app's user-friendly interface allows easy connection to the Bluetooth module, enabling real-time command transmission.

7.1 Circuit Design and Assembly

Creating a clear and functional wiring diagram is essential to ensure all components are correctly connected and the system operates as intended. The following steps outline the process of assembling the hardware:

1. Connect the Bluetooth Module to the Arduino:

- VCC to 5V: Provide power to the Bluetooth module.
- GND to GND: Connect the ground to complete the circuit.
- TX to RX (pin 0): Transmit data from the Bluetooth module to the Arduino.
- RX to TX (pin 1): Receive data from the Arduino to the Bluetooth module.

2. Connect the Relay Module to the Arduino:

- VCC to 5V: Provide power to the relay module.
- GND to GND: Connect the ground to complete the circuit.
- IN1 to Digital Pin 2: Use a digital pin on the Arduino to control the relay.

3. Ensure Stable Power Supply: Verify that the power supply is stable and can provide sufficient power for both the Arduino and the relay module.

7.2 Software Development

The software component of the project involves programming the Arduino to handle Bluetooth communication and control the relay based on received commands. Additionally, the "AMR Voice" app must be configured to recognize and transmit voice commands. The following steps detail the software development process:

The software development for the voice-controlled home automation system involves several key steps, including the setup and configuration of the Android application, the programming of the Arduino microcontroller, and the integration of the Bluetooth communication module. Below are the detailed steps and components involved in the software development process:

1. Android Application Setup

- **Application Used:** AMR Voice App
- **Purpose:** To capture voice commands and send them to the Arduino via Bluetooth.
- **Configuration:**
 - Install the AMR Voice application on the Android device.
 - Pair the Android device with the Bluetooth module connected to the Arduino.
 - Configure the AMR Voice app to recognize specific voice commands that will control the home automation devices.

2. Arduino Programming

- **Programming Language:** Arduino IDE (based on C/C++)
- **Code Overview:**
 - **Initialization:** Set up the serial communication and initialize the pins connected to the relay module.
 - **Voice Command Processing:** Receive and interpret the voice commands sent from the AMR Voice app via Bluetooth.
 - **Control Logic:** Execute specific actions based on the voice commands, such as turning on/off lights, appliances, or other connected devices.

3. Bluetooth Communication

- **Module Used:** HC-05 or HC-06 Bluetooth module
- **Configuration:**
 - Connect the Bluetooth module to the Arduino using the appropriate pins (TX, RX, VCC, GND).
 - Ensure that the Bluetooth module is configured to communicate at the same baud rate as set in the Arduino code.

4. Integration and Testing

- **Integration:** Combine the hardware and software components, ensuring that the Android device, Bluetooth module, and Arduino are correctly communicating.
- **Testing:** Conduct various tests to ensure that voice commands are accurately recognized and executed by the system.

5. Code Documentation

- **Comments:** Ensure the code is well-commented to explain the purpose of each function and key lines of code.
- **User Manual:** Provide a manual for users detailing how to set up and use the system, including pairing the Bluetooth module and configuring the AMR Voice app.

The software development section integrates the voice recognition capabilities of the AMR Voice app with the hardware control provided by the Arduino microcontroller and Bluetooth module. Through careful programming and thorough testing, the system is designed to offer a reliable and user-friendly home automation solution.

7.3 Optimization and Documentation

Optimizing the system for performance and ease of use involves refining both the hardware and software components. Proper documentation ensures the system can be easily replicated and understood by others. The following steps outline the optimization and documentation process:

1. Hardware Optimization:

- Stable Connections: Ensure all connections are secure and components are properly soldered or connected with reliable jumper wires.
- Power Management: Verify that the power supply is sufficient and stable for all components, especially when controlling high-power appliances.

2. Software Optimization:

- Code Efficiency: Optimize the Arduino code for better performance, reducing any unnecessary delays or processing overhead.
- Error Handling: Implement robust error handling to manage unexpected inputs or communication failures gracefully.

3. User Manual: Create a detailed user manual that includes:

- Component List: A list of all required components with specifications.
- Wiring Diagram: A clear and easy-to-follow wiring diagram.
- Step-by-Step Instructions: Detailed instructions for assembling the hardware, uploading the Arduino code, and configuring the "AMR Voice" app.
- Troubleshooting Guide: Common issues and solutions to help users resolve potential problems.

4. Code Documentation: Comment the Arduino code thoroughly to explain the functionality of each section and the purpose of key lines of code.

7.4 Advantages

1. Cost-Effective:

- Utilizes affordable and widely available components, making it accessible for a broad audience.

2. Simplicity:

- Easy to set up and use, even for individuals with minimal technical expertise.

3. Privacy:

- Local processing of voice commands ensures greater privacy compared to cloud-based systems.

4. Offline Functionality:

- Operates independently of internet connectivity, relying solely on Bluetooth communication.

5. Customizability:

- The system can be easily modified and expanded to include additional features and devices.

6. User-Friendly Interface:

- The "AMR Voice" Android app provides an intuitive interface for issuing voice commands.

7. Energy Efficiency:

- Allows users to control appliances remotely, potentially reducing unnecessary power consumption.

8. Reliability:

- Local processing and direct control via Bluetooth reduce latency and dependency on external servers.

9. Safety:

- The relay module safely handles the switching of high-voltage appliances, ensuring user safety.

7.5 Applications

1. Lighting Control:

- Turn lights on or off using voice commands for convenience and energy savings.

2. Appliance Control:

- Control household appliances like fans, heaters, or air conditioners through voice commands.

3. Home Security:

- Integrate with security systems to activate alarms or lock doors via voice commands.

4. Elderly and Disabled Assistance:

- Provide a convenient way for elderly or disabled individuals to control home devices without physical effort.

5. Entertainment Systems:

- Control TV, speakers, or other entertainment devices with voice commands for a seamless experience.

6. Smart Kitchens:

- Voice-activate kitchen appliances like coffee makers or microwaves to streamline cooking and preparation.

7. HVAC Control:

- Adjust heating, ventilation, and air conditioning systems through voice commands for optimal comfort.

8. Garden and Outdoor Lighting:

- Control outdoor lighting systems for security and ambiance through voice commands.

9. Energy Management:

- Schedule and control energy-intensive devices to operate during off-peak hours, reducing electricity bills.

10. Home Automation Hubs:

- Use the system as a central hub to control multiple smart devices and sensors throughout the home.

11. Office Automation:

- Implement in office environments to control lighting, projectors, and other office equipment via voice commands.

12. Hospital and Healthcare:

- Integrate into healthcare facilities for better control of medical equipment and lighting in patient rooms.

13. Hotel Room Control:

- Provide guests with a high-tech experience by allowing them to control room amenities through voice commands.

14. Retail and Commercial Spaces:

- Manage lighting, security, and HVAC systems in retail stores or commercial buildings for improved efficiency.

15. Educational Institutions:

- Use in classrooms or laboratories to control equipment and lighting, enhancing the learning environment

CHAPTER 8

TESTING AND RESULTS

8.1 Testing and Results

Testing involves various scenarios to ensure the system's robustness. Commands are issued in different conditions (e.g., varying distances and background noise levels) to evaluate the Bluetooth module's range and the voice recognition accuracy of the "AMR Voice" app. The relay's response time and reliability are also assessed to ensure timely and correct operation of connected appliances.

The results demonstrate the system's capability to accurately interpret and execute voice commands, providing a seamless user experience. The integration of Arduino with the Bluetooth module and relay proves to be effective, showcasing the potential of affordable IoT solutions in home automation.

8.2 Testing and Debugging

Testing the system in various scenarios is crucial to ensure its reliability and responsiveness. The following steps outline the testing and debugging process:

- 1. Initial Test:** Verify basic functionality by issuing voice commands through the "AMR Voice" app and observing the relay's response.
- 2. Range Testing:** Test the Bluetooth communication range by issuing commands at different distances from the Arduino.
- 3. Environment Testing:** Test the system in various environments with different levels of background noise to ensure the voice recognition accuracy.
- 4. Stress Testing:** Continuously issue commands to assess the system's performance and identify any potential issues with prolonged use.
- 5. Debugging:** Use the Serial Monitor to identify and fix any issues with command recognition and relay control.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

The voice-controlled home automation system developed in this project offers a practical and cost-effective solution for integrating smart home technology into everyday life. By leveraging Arduino and Bluetooth communication, this system allows users to control home appliances via simple voice commands issued through an Android device.

Advantages:

1. **Affordability:** The use of inexpensive and readily available components makes the system accessible to a wide audience.
2. **Simplicity:** Easy setup and user-friendly operation cater to individuals with minimal technical expertise.
3. **Privacy and Offline Capability:** Local processing ensures greater privacy and functionality without relying on internet connectivity.
4. **Customizability:** The system can be expanded and tailored to meet specific user needs and preferences.

Applications:

The system can be applied in various scenarios, including residential, commercial, and healthcare settings, to control lighting, appliances, security systems, and more.

Hardware and Software Requirements:

The system requires basic hardware components like an Arduino board, Bluetooth module, relay module, and an Android device with the "AMR Voice" app. The software setup involves programming the Arduino using the Arduino IDE and configuring the Android app for voice command transmission.

This project successfully illustrates the development of a voice-controlled home automation system using Arduino. It emphasizes the practicality of leveraging readily available components to create innovative and accessible technology solutions. While the current system is limited to basic on/off commands, future enhancements could include integrating additional sensors for more complex automation tasks, expanding the range of voice commands, and incorporating machine learning algorithms for improved voice recognition and user customization.

Future Enhancements:

While the current system provides essential functionality, future improvements could include multi-device control, advanced voice commands, sensor integration, custom mobile app development, and the incorporation of machine learning for improved performance.

While the current system provides basic functionality, several enhancements can be made to improve its capabilities and user experience:

1. **Multi-Device Control:** Expand the system to control multiple appliances by adding more relay modules and configuring additional voice commands.
2. **Advanced Voice Commands:** Implement more complex voice commands for advanced control, such as dimming lights or setting specific temperatures.
3. **Sensor Integration:** Incorporate sensors (e.g., motion sensors, temperature sensors) to enable automated responses based on environmental conditions.
4. **Mobile App Development:** Develop a custom mobile app to provide a more tailored user interface and additional features such as scheduling and remote access.

5. Machine Learning: Integrate machine learning algorithms to improve voice recognition accuracy and personalize the system based on user preferences.

Summary:

This project demonstrates the feasibility of creating a user-friendly, affordable, and efficient voice-controlled home automation system. It highlights the potential for future innovations and serves as a foundational step towards more advanced and personalized smart home solutions, bringing the benefits of IoT technology to a broader audience.

CHAPTER 10

REFERENCES

1] Reference on Voice-Controlled Home Automation using Arduino:

- **Title:** "Voice Controlled Home Automation System Using Arduino"
- **Authors:** T. S. Somashekar, K. R. Shobha, and K. R. Venugopal
- **Published in:** International Journal of Computer Applications (IJCA)
- **Link:** Voice Controlled Home Automation System Using Arduino (IJCA)

2] Reference on Bluetooth-Based Home Automation:

- **Title:** "Bluetooth Based Home Automation System Using Cell Phone"
- **Authors:** R.Piyare and M.Tazil
- **Published in:** IEEE 15th International Symposium on Consumer Electronics
- **Link:** [Bluetooth Based Home Automation System Using Cell Phone \(IEEE Xplore\)](#)

3] Reference on Arduino-Based Home Automation with Voice Control:

- **Title:** "Home Automation using Arduino and Bluetooth"
- **Authors:** Jayashri Bangali, Arvind Shaligram
- **Published in:** International Journal of Computer Applications (IJCA)
- **Link:** Home Automation using Arduino and Bluetooth (IJCA)