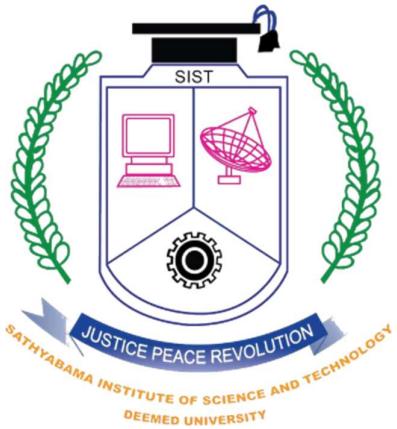
NON-AMBULATORY INDIVIDUAL LOCOMOTION WITH VOICE CONTROL SYSTEM

Submitted in partial fulfilment of requirements for the award of
Bachelor of Engineering in Electronics and Communication Engineering

By

Ashwin Balaji V (40130018)

Fredric S (40130056)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2024

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Ashwin Balaji V (40130018)Fredric S(40130056) who carried out the project entitled "NON-AMBULATORY INDIVIDUAL LOCOMOTION WITH VOICE CONTROL SYSTEM" under my /our supervision from November 2023 to April 2024.

INTERNAL GUIDE
DR. F V JAYASUDHA, M.E., Ph.D.

HEAD OF THE DEPARTMENT
Dr. T. RAVI, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

We **Ashwin Balaji V**(40130018)**Fredric S**(40130056)hereby declare that the Project Report entitled "**NON-AMBULATORY INDIVIDUAL LOCOMOTION WITH VOICE CONTROL SYSTEM**" done by us under the guidance of **Dr. JAYASUDHA, M.E., Ph.D.**, Assistant Professor,Department of Electronics and Communication Engineering, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

Date: **SIGNATURE OF THE CANDIDATES**

Place:

1.

2.

ABSTRACT

Currently, a lot of nations are putting great emphasis on smart technology. In order to compete with other nations around the world, our Prime Minister of India has also launched the missions of Smart India and Digital India. Physically challenged people and the elderly are currently experiencing challenges, not people who are able to. So, we came up with the concept of a smart wheelchair for physically impaired people in order to make their lives easier and improve them. A smart wheelchair is one that can move on its own at the user's command, reducing the need for the wheelchair user to exert physical force to propel the wheels. Also, it gives physically or visually handicapped people the chance to go from one place to another place.

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **The Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr.N.M.NANDHITHA,M.E.,Ph.D.,** Dean, School of Electrical and Electronics Engineering and **Dr.T.RAVI,M.E.,Ph.D.,** Head of the Department, Dept. Of Electronics and Communication Engineering for providing us necessary support during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our Project Guide **Dr. JAYASUDHA, M.E., Ph.D.** Assistant Professor, Department of Electronics and Communication Engineering for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.2 GENERAL

The "Revolutionizing Care: Smart Health Solutions for Wheelchair Patients" project is a groundbreaking healthcare system designed specifically for individuals reliant on wheelchairs for mobility. It leverages cutting-edge technology to cater to the unique healthcare needs of this demographic, ultimately enhancing their overall well-being and quality of life. At the heart of this innovation lies a micro-controller, acting as the system's central control unit, orchestrating all processes. Integrated smart sensors, including MEMs for wheelchair movement detection, and heart rate sensor. By revolutionizing care through advanced technology and personalized monitoring, this system represents a significant step forward in improving the lives of wheelchair-bound individuals.

1.2 SCOPE OF THE PROJECT:

Enhancing healthcare for wheelchair-bound individuals through smart sensors, wearable devices, and real-time monitoring technologies for improved well-being.

1.3 EXISTING SYSTEM

- Helpers or attendees need to handle handicapped person in wheelchair.
- Sensors not used in wheelchair.
- Basic wheelchair with no health monitoring.
- Lack of real-time health data.
- Limited emergency notification capabilities.
- There is no voice control system.

1.3.1 EXISTING SYSTEM DISADVANTAGES

- Handicapped person suffered more in wheelchair.
- Labor charge is applicable to take care of the person.

CHAPTER 2
LITERATURE SURVEY

Title 1: Voice Recognition based Intelligent Wheelchair and GPS Tracking System.
Author: Nasrin Aktar1 , Israt Jahan2 , Bijoya Lala3.
Year : 2019
Description:
Development of a voice recognition based intelligent wheelchair system for physically handicapped people who are unable to drive the wheelchair by hand is represented in this paper where the patient can operate the wheelchair using voice commands and the location of patient can be traced using GPS module in the wheelchair that tracks and sends the information to smartphone application (app) via Firebase. Voice module V3 is used to record patient's voice and recognize that voice to follow the instructions of the patient. This kit converts the voice commands to hexadecimal numbers and then the data is fed to the Wi-Fi module to control the wheelchair. Wi-Fi module directs the motor driver IC to move the wheels in desired direction. Motor speed can also be controlled in three stages-low, medium, high. This system also offers obstacle detection automatically using IR sensor and a smartphone app has been developed for the family members of patient to know about the location of the patient. Use of firebase makes the system fast and android app offers low cost and user friendly environment than the conventional GSM based navigation systems. As this system simultaneously offers voice operated wheelchair, motor speed control, obstacle detection and GPS tracking of patient using android app, hopefully it will be a fruitful system for the handicapped people worldwide.

Title 2: Smart Wheelchair- An Implementation of Voice and Android Controlled System.
Author: Muhammad Saad Amin, Syed Tahir Hussain Rizvi, Sameer Malik.
Year : 2021
Description:
The inspiration driving android and voice automated smart wheelchair venture is to construct a smart wheelchair that helps physically impaired people to locomote from

one spot then onto the next. To overcome this disability, a smart voice-controlled fully automated wheelchair is designed for physically disabled, patients, or pregnant women. This smart wheelchair will help them move from one place to another without any problem. Numerous wheelchairs are accessible with various running advancements, yet the expense is high and it isn't much successful. For the most part, designing voice and android control wheelchairs is to conquer a few burdens of the current frameworks. The customer needs to interact with the wheelchair with the help of the application. This framework enables the client to vigorously communicate with the wheelchair at various dimensions of the control (turn left, turn right, proceed, return and stop). This task utilizes a microcontroller circuit and motor drivers to make the development of the wheelchair.

Title 3 Design and Development of Voice Controllable Wheelchair.
Author: Polash Pratim Dutta *, Abhishek Kumar, Aditi Singh, Kartik Saha.
Year : 2020
Description:
In this work a voice controlled wheelchair is made for physically challenged peoples or patients whose hind limbs are not in working condition. Hence, he or she can control the wheelchair by own voice commands or of the family members. To accomplish the task Arduino microcontroller board is used which receives the voice commands either from Bluetooth module via smartphone or from voice recognition module from microphone attached to it. Further matching it with preloaded voice commands and develop movement accordingly. In the subsequent sections of the paper give more concentration on further objectives to achieve reduction in motor speed smooth movement of vehicle by two motors and the wheels as per the command. The main objective of current work is to make it as simple as possible. Certain theoretical results are also obtained by formulation and design calculation.

Title 4: Voice Controlled Automatic Wheelchair.
Author Sumet Umchid, Pitchaya Limhaprasert, Sitthichai Chumsoongnern, Tanun Petthong and Theera Leeudomwong.
Year : 2018
Description:
In general, the people with physical disability come from many reasons such as

injury from accident, age and health problems. Therefore, wheelchair is needed to use when handicapped people would like to travel to any places by themselves. However, hands and arms must be used to operate the wheelchair. Consequently, people with hands and arms impairment finds difficult to use a typical wheelchair. These people need to get help from other people around to control the wheelchair and it will create a big problem when these people would like to travel alone. Therefore, the objective of this work is to design, develop and construct a voice controlled automatic wheelchair. The developed wheelchair is able to operate by using the voice commands through the given input. The principle of the developed wheelchair consists of motor system, voice recognition module that would be controlled by the microcontroller. The automatic obstacle detection system is included to the developed wheelchair by using ultrasonic sensors in order to brake the developed wheelchair immediately when any obstacles suddenly come in the way of the developed wheelchair. Therefore, the developed voice controlled wheelchair can provide easy access for people with physical disability and also offer automatic protection from obstacle collision if the mistake of any voice commands occurs.

Title 5: Voice and Gesture Controlled Wheelchair
Author: Ms. Cynthia Joseph, Aswin S, Sanjeev Prasad J,.
Year : 2020
Description:
A sector of physically challenged people finds it very difficult to use traditional wheelchairs. Researchers have been working on computer-controlled chairs which utilize sensors and quick control algorithms to minimize the level of human intervention. This paper is based on a design that aids the voice activation system for physically disabled people by incorporating manual operation. Arduino microcontroller and voice recognition have been used to support the movement of the wheelchair. The wheelchair does not respond to an incorrect speech command. Depending on the direction given through voice and gesture, the Arduino controls the wheelchair directions. Ultrasonic sensors are used to detect obstacles. The prototype is designed in such a way that it can be used independently and efficiently with less effort. It saves time, reduces cost and energy of the users.

CHAPTER 3
PROJECT DESCRIPTION

3.1 GENERAL

The "Revolutionizing Care: Smart Health Solutions for Wheelchair Patients" project is a groundbreaking healthcare system designed specifically for individuals reliant on wheelchairs for mobility. It leverages cutting-edge technology to cater to the unique healthcare needs of this demographic, ultimately enhancing their overall well-being and quality of life. At the heart of this innovation lies a micro-controller, acting as the system's central control unit, orchestrating all processes. Integrated smart sensors, including MEMs for wheelchair movement detection, and heart rate sensor will attached in this wheel chair for monitoring purpose. a buzzer provides audible alerts. This comprehensive approach not only ensures timely medical attention but also promotes patient autonomy and independence. By revolutionizing care through advanced technology and personalized monitoring, this system represents a significant step forward in improving the lives of wheelchair-bound individuals.

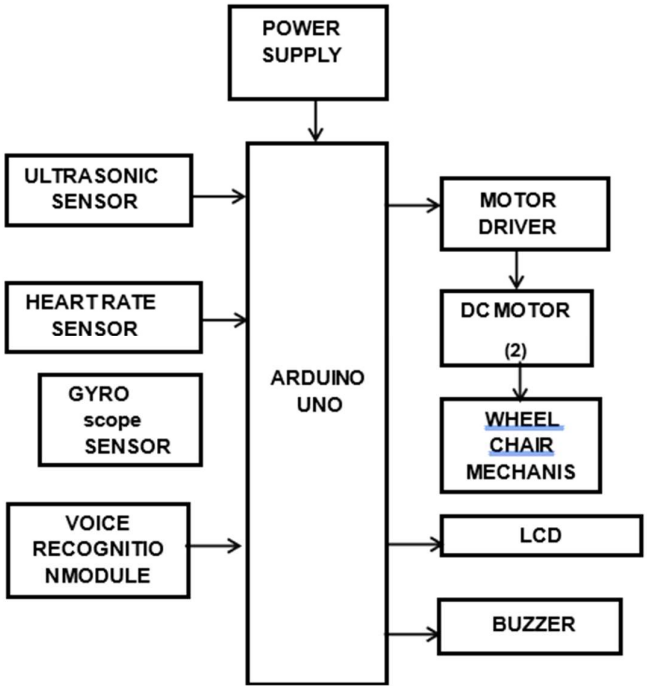


Fig:3.1 BLOCK DIAGRAM

3.2 WORKING PRINCIPLE

In this system, the entire process is operated and controlled by the micro-controller which acts as a brain of the system. All the instructions of the process are programmed into the controller. Voice recognition module is used to control the movement of the wheelchair. The ultrasonic sensor is used for auto braking system of the wheelchair to avoid obstacle collision. The gyro scope sensor for additional controlling of the wheelchair. Heart rate sensor is used to measure the heart rate of the patient. Buzzer is widely used to notify various events such as end of any process or as an alarming on emergency situation. LCD is used to show the current execution of the project.

3.3 MODULE NAME

- VOICE CONTROLLING WHEEL CHAIR
- ANGLE OF THE WHEELCHAIR
- ANALYZING THE DATA
- ABNORMAL ALERT

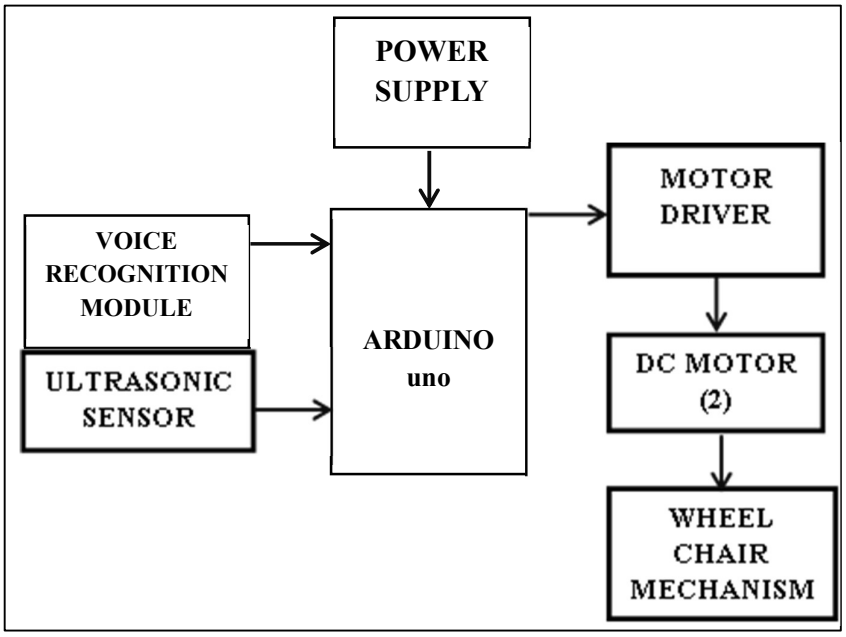


Fig:3.2 VOICE CONTROLLING WHEEL CHAIR

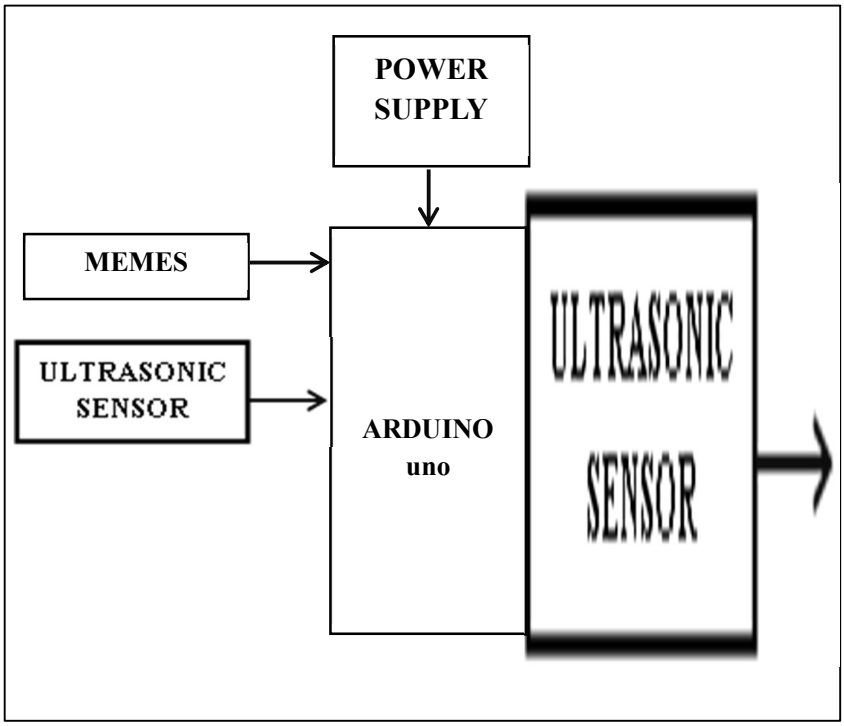


Fig:3.3 ANGLE OF THE WHEELCHAIR

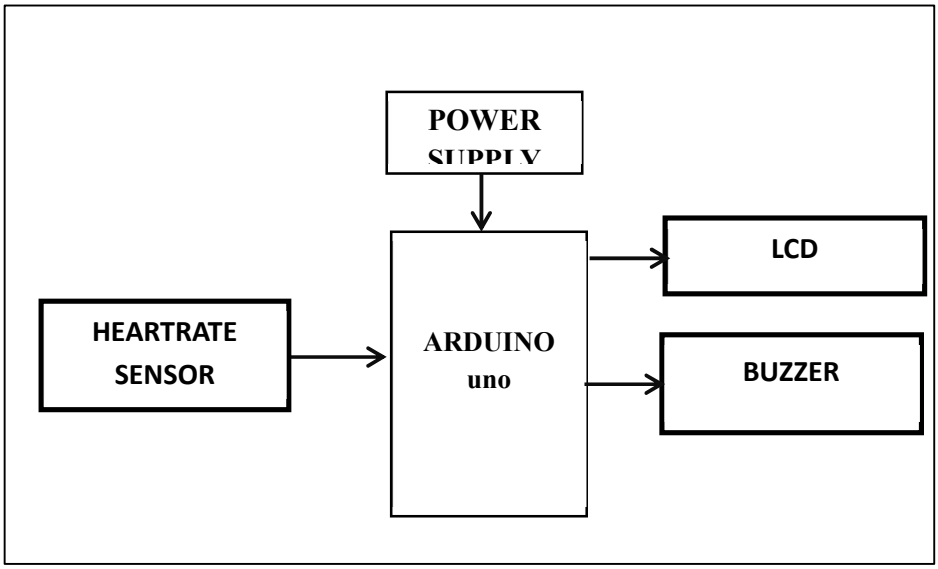


Fig:3.4 ANALYZING THE DATA

3.4 STANDARDS:

1. *Safety*: The smart wheelchair must adhere to safety standards to ensure the well-being of users during operation.
2. *Accessibility*: The design should conform to accessibility standards to accommodate users with various physical impairments.
3. *Reliability*: The smart wheelchair should meet reliability standards to ensure consistent performance and minimize the risk of malfunctions.
4. *Interoperability*: It should be compatible with existing assistive technologies and infrastructure to facilitate seamless integration and operation.

3.5 CONSTRAINTS:

1. Cost: The development and production costs must be kept within reasonable limits to ensure affordability for users and widespread adoption.

2. Technical limitations: The smart wheelchair's functionality may be constrained by technical limitations such as battery life, range, and terrain adaptability.

3. Regulatory compliance: Compliance with regulatory requirements and certifications is necessary to ensure legal conformity and market acceptance.

4. User acceptance: The design should consider user preferences, comfort, and ease of use to promote user acceptance and satisfaction.

3.6 TRADE-OFFS:

1. Complexity vs. simplicity: Balancing advanced features with user-friendly design to ensure usability without overwhelming users with complexity.

2. Autonomy vs. control: Providing autonomous navigation capabilities while allowing users to maintain control and autonomy over their movements.

3. Performance vs. cost: Achieving optimal performance while managing costs to make the smart wheelchair accessible to a wider user base.

4. Customization vs. standardization: Offering customizable features to accommodate individual needs while maintaining standardization for mass production and support.

CHAPTER 4
HARDWARE AND SOFTWARE DESCRIPTION

4.1 HARDWARE DESCRIPTION

4.1.1 ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IOT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

WHY ARDUINO?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low-cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes,

musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open-source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit

designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

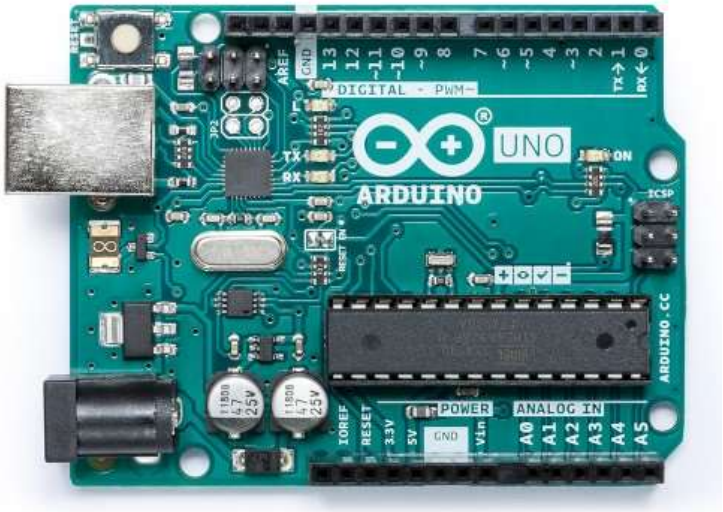


Fig:4.1 Arduino Uno

4.1.2 POWER SUPPLY

This section describes how to generate +5V DC power supply

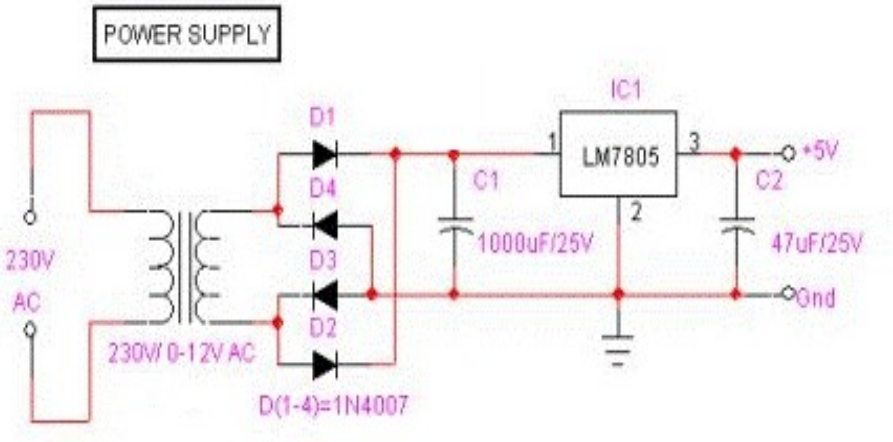


Fig:4.2 Power Supply

The power supply section is the important one. It should deliver constant output regulated power supply for successful working of the project. A 0-12V/1 mA transformer is used for this purpose. The primary of this transformer is connected in to main supply through on/off switch& fuse for protecting from overload and short circuit protection. The secondary is connected to the diodes to convert 12V AC to 12V DC voltage. And filtered by the capacitors, which is further regulated to +5v, by using IC 7805.

4.1.3 BUZZER

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

Fig:4.3 Buzzer

4.1.4 MEMS SENSOR:

Microelectromechanical systems, popularly known as MEMS, is the technology of very small electromechanical and mechanical devices. Advance in MEMS technology has helped us to develop versatile products. Many of the mechanical devices such as Accelerometer, Gyroscope, etc… can now be used with consumer electronics. This was possible with MEMS technology. These sensors are packaged similarly to other IC's. Accelerometers and Gyroscopes compliment each other so, they are usually used together. An accelerometer measures the linear acceleration or directional movement of an object, whereas Gyroscope Sensor measures the angular velocity or tilt or lateral orientation of the object. Gyroscope sensors for multiple axes are also available.

4.1.5 Gyroscope Sensor

Gyroscope sensor is a device that can measure and maintain the orientation and angular velocity of an object. These are more advanced than accelerometers. These can measure the tilt and lateral orientation of the object whereas accelerometer can only measure the linear motion.Gyroscope sensors are also called as Angular Rate Sensor or Angular Velocity Sensors. These sensors are installed in the applications where the orientation of the object is difficult to sense by humans.Measured in degrees per second, angular velocity is the change in the rotational angle of the object per unit of time.

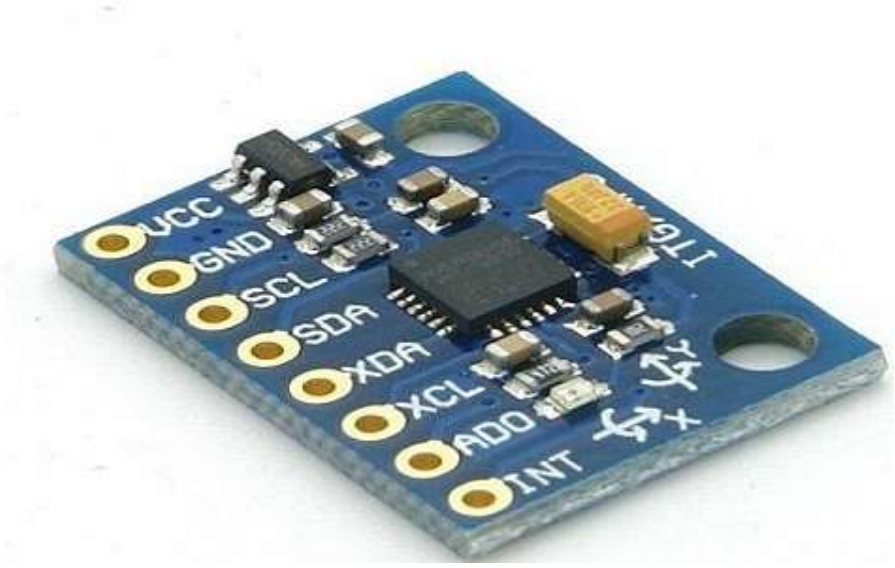


Fig:4.4 Gyroscope sensor

4.1.6 HEART BEAT SENSOR

This project uses bright infrared (IR) LED and a phototransistor to detect the pulse of the finger, a red LED flashes with each pulse. Pulse monitor works as follows: The LED is the light side of the finger, and phototransistor on the other side of the finger, phototransistor used to obtain the flux emitted, when the blood pressure pulse by the finger when the resistance of the photo transistor will be slightly changed. The project's schematic circuit as shown, We chose a very high resistance resistor R1, because most of the light through the finger is absorbed, it is desirable that the phototransistor is sensitive enough. Resistance can be selected by experiment to get the best results. The most important is to keep the shield stray light into the phototransistor. For home lighting that is particularly important because the lights at home mostly based 50HZ or 60HZ fluctuate, so faint heartbeat will add considerable noise.When running the program the measured values are printed. To get a real heartbeat from this could be challenging. The KY-039 Heartbeat Detector uses a very high resistance resistor R1, because most of the light through the finger is absorbed, it is desirable that the phototransistor is sensitive enough. The most important is to keep the shield stray light into the phototransistor. For home lighting that is particularly important because the lights at home mostly based 50HZ or 60HZ fluctuate, so faint heartbeat will add considerable noise.

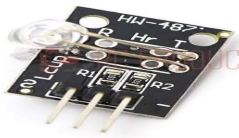


Fig:4.5 Heartbeat Sensor

4.1.7 ESP 32 CAM

The ESP32-CAM is a compact microcontroller module based on the ESP32 system-on-chip (SoC) with an integrated camera. It is widely used for projects that require wireless connectivity and image capture capabilities in a small form factor. The ESP32-CAM is a versatile and cost-effective solution for projects that require both wireless connectivity and image capture capabilities. Its small form factor and

robust feature set make it a popular choice among hobbyists, makers, and IoT enthusiasts for a wide range of applications. With a strong developer community and extensive documentation, it is relatively easy to get started with and offers a wide range of possibilities for creative projects.

Fig:4.6 ESP32-CAM

4.1.8 MOTOR DRIVER IC

Common DC gear head motors need current above 250mA. There are many integrated circuits like ATmega16 Microcontroller, 555 timers IC. But, IC 74 series cannot supply this amount of current. When the motor is directly connected to the o/p of the above ICs then, they might damage. To overcome this problem, a motor control circuit is required, which can act as a bridge between the above motors and ICs (integrated circuits). There are various ways of making H-bridge motor control circuit such as using transistor, relays and using L293D/L298.

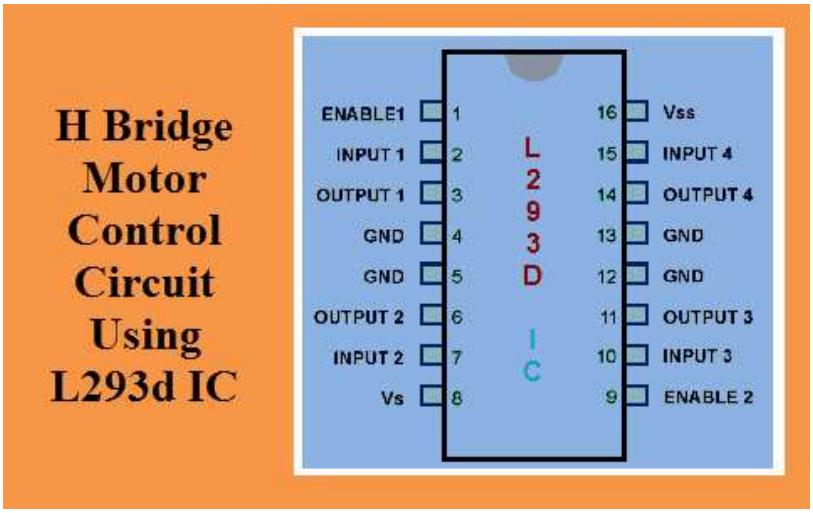


Fig:4.7 H Bridge Motor Control Circuit Using L293d IC

4.1.9 H-BRIDGE CIRCUIT:

An H bridge is an electronic circuit that allows a voltage to be applied across a load in any direction. H-bridge circuits are frequently used in robotics and many other applications to allow DC motors to run forward & backward. These motor control circuits are mostly used in different converters like DC-DC, DC-AC, AC-AC converters and many other types of power electronic converters. In specific, a bipolar stepper motor is always driven by a motor controller having two H-bridges.

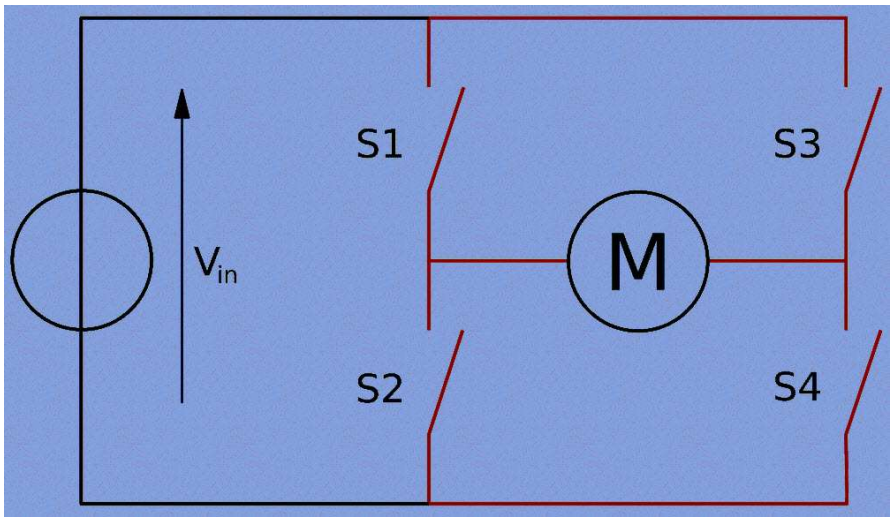


Fig 4.8 H-BRIDGE CIRCUIT

An H-bridge is fabricated with four switches like S1, S2, S3 and S4. When the S1 and S4 switches are closed, then a positive voltage will be applied across the motor. By opening the switches S1 and S4 and closing the switches S2 and S3, this voltage is inverted, allowing invert operation of the motor.

Generally, the H-bridge motor driver circuit is used to reverse the direction of the motor and also to break the motor. When the motor comes to a sudden stop, as the terminals of the motor are shorted. Or let the motor run free to a stop, when the motor is detached from the circuit. The table below gives the different operations with the four switches corresponding to the above circuit.

4.1.10 L293D MOTOR DRIVER IC:

L293D IC is a typical Motor Driver IC which allows the DC motor to drive on any direction. This IC consists of 16-pins which are used to control a set of two DC motors instantaneously in any direction. It means, by using a L293D IC we can control two DC motors. As well, this IC can drive small and quiet big motors.

This L293D IC works on the basic principle of H-bridge, this motor control circuit allows the voltage to be flowing in any direction. As we know that the voltage must be change the direction of being able to rotate the DC motor in both the directions. Hence, H-bridge circuit using L293D ICs are perfect for driving a motor. Single L293D IC consists of two H-bridge circuits inside which can rotate two DC motors separately. Generally, these circuits are used in robotics due to its size for controlling DC motors.

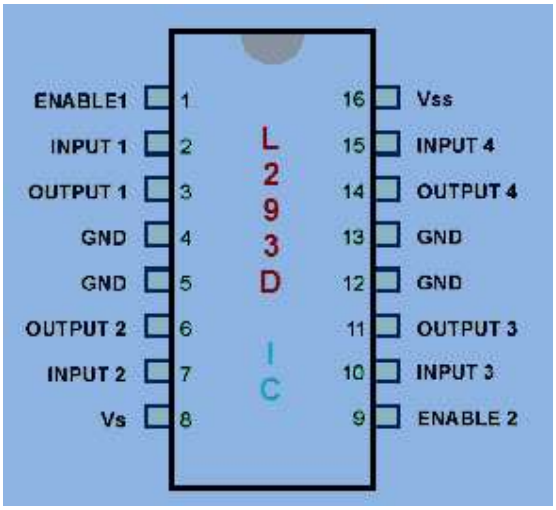


Fig 4.9 L293D IC Pin Configuration

4.1.11 DIRECT CURRENT (DC) MOTOR:

Almost every mechanical development that we see around us is accomplished by an electric motor. Electric machines are a method of converting energy. Motors take electrical energy and produce mechanical energy. Electric motors are utilized to power hundreds of devices we use in everyday life.

Electric motors are broadly classified into two different categories: Direct Current (DC) motor and Alternating Current (AC) motor. In this article we are going to discuss about the DC motor and its working. And also how a gear DC motors works.

A DC motor is an electric motor that runs on direct current power. In any electric motor, operation is dependent upon simple electromagnetism. A current carrying conductor generates a magnetic field, when this is then placed in an external magnetic field, it will encounter a force proportional to the current in the conductor

and to the strength of the external magnetic field. It is a device which converts electrical energy to mechanical energy. It works on the fact that a current carrying conductor placed in a magnetic field experiences a force which causes it to rotate with respect to its original position.

Practical DC Motor consists of field windings to provide the magnetic flux and armature which acts as the conductor.

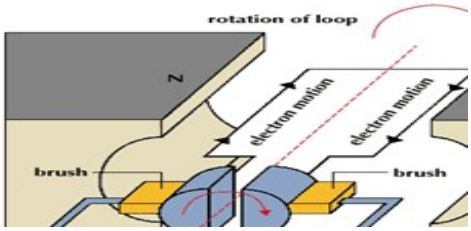


Fig:4.10 Brushless Dc Motors Work

The input of a brushless DC motor is current/voltage and its output is torque. Understanding the operation of DC motor is very simple from a basic diagram is shown in below. DC motor basically consist two main parts. The rotating part is called the rotor and the stationary part is also called the stator. The rotor rotates with respect to the stator.

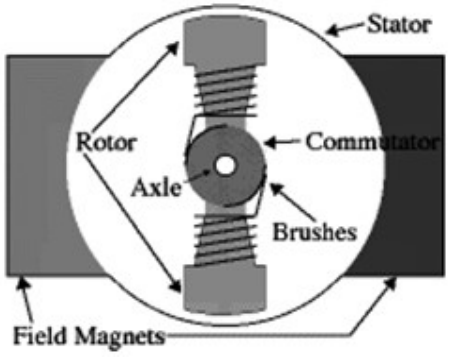


Fig:4.11 Dc Motor

The rotor consists of windings, the windings being electrically associated with the commutator. The geometry of the brushes, commutator contacts and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnets are misaligned and the rotor will turn until it is very nearly straightened with the stator's field magnets.

As the rotor reaches alignment, the brushes move to the next commutator contacts and energize the next winding. The rotation reverses the direction of current through

the rotor winding, prompting a flip of the rotor's magnetic field, driving it to keep rotating.

4.1.11.1 CONNECTING DC MOTOR WITH MICROCONTROLLER:

Microcontrollers can't drive the motors directly. So we need some kind of drivers to control the speed and direction of motors. The motor drivers will acts as interfacing devices between microcontrollers and motors. Motor drivers will act as current amplifiers since they take a low current control signal and provide a high current signal. This high current signal is used to drive the motors. Using L293D chip is the easy way for controlling the motor using microcontroller. It contains two H-bridge driver circuits internally.

This chip is designed to control two motors. L293D has two sets of arrangements where 1 set has input 1, input 2, output1,output 2, with enable pin while other set has input 3, input 4, output 3, output 4 with other enable pin.

Here is an example of DC motor which is interfaced with L293D microcontroller.

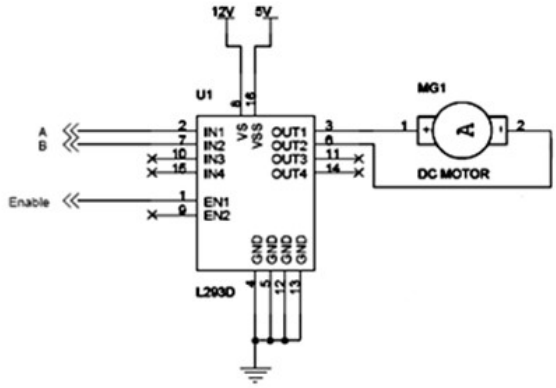


Fig:4.12 Dc Motor Interfaced With L293D Microcontroller

4.2 SOFTWARE REQUIREMENTS

4.2.1 EMBEDDED C

Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software.

Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all device working is based on microcontroller that are programmed by embedded C.

Let's see the block diagram representation of embedded system programming:

The Embedded C code written in above block diagram is used for blinking the LED connected with Port0 of microcontroller.

In embedded system programming C code is preferred over other language. Due to the following reasons:

- Easy to understand.
- High Reliability
- Portability
- Scalability

4.2.1.1 EMBEDDED SYSTEM PROGRAMMING:

Function is a collection of statements that is used for performing a specific task and a collection of one or more functions is called a programming language. Every language is consisting of basic elements and grammatical rules. The C language programming is designed for function with variables, character set, data types, keywords, expression and so on are used for writing a C program.

The extension in C language is known as embedded C programming language. As compared to above the embedded programming in C is also have some additional features like data types, keywords and header file etc is represented by #include<microcontroller name.h>.

4.2.1.2 BASIC EMBEDDED C PROGRAMMING STEPS:

Let's see the block diagram representation of Embedded C Programming Steps: The microcontroller programming is different for each type of operating system. Even though there are many operating system are exist such as Windows, Linux, RTOS, etc but RTOS has several advantage for embedded system development.

4.2.1.3 EMBEDDED SYSTEMS

Embedded System is a system composed of hardware, application software and real time operating system. It can be small independent system or large combinational system.

Our Embedded System tutorial includes all topics of Embedded System such as characteristics, designing, processors, microcontrollers, tools, addressing modes, assembly language, interrupts, embedded c programming, led blinking, serial communication, lcd programming, keyboard programming, project implementation etc.

SYSTEM:

System is a way of working, organizing or performing one or many tasks according to a fixed set of rules, program or plan.

It is an arrangement in which all the unit combined to perform a work together by following certain set of rules in real time computation. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan.

An Embedded System is a system that has software embedded into computer-hardware, which makes a system dedicated for a variety of application or specific part of an application or product or part of a larger system.

An embedded system can be a small independent system or a large combinational system. It is a microcontroller-based control system used to perform a specific task of operation.

An embedded system is a combination of three major components:

- Hardware: Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- Application software: Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
- Real Time Operating system (RTOS): RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and

provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.

4.2.2 ARDUINO SOFTWARE IDE

message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

4.2.2.1 *FILE*:

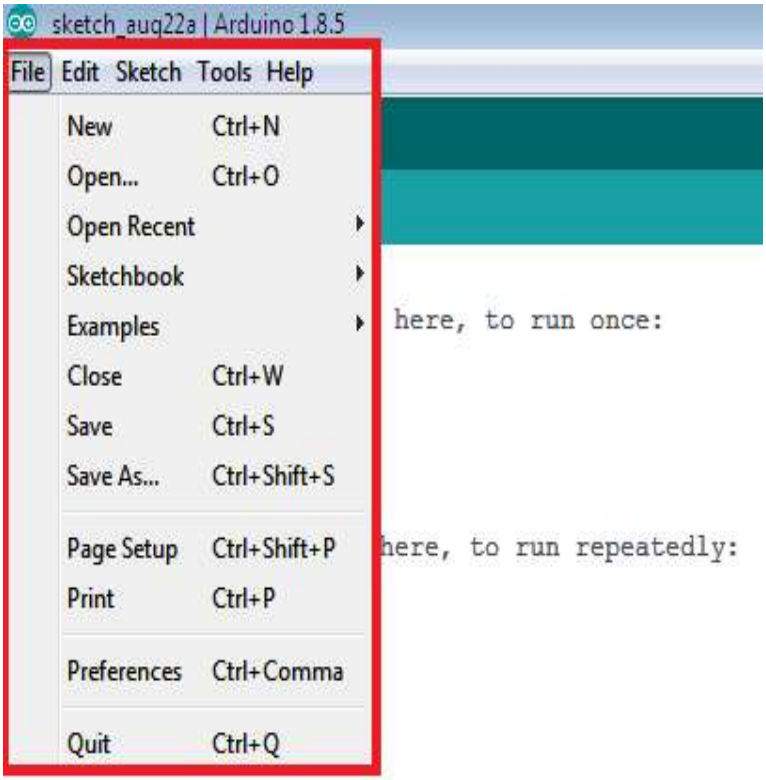


Fig:4.13 Arduino Ide File Columns

- *New*

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

- *Open*

Allows loading a sketch file browsing through the computer drives and folders.

- *OpenRecent*

Provides a short list of the most recent sketches, ready to be opened.

- *Sketchbook*

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

- *Examples*

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

- *Close*

Closes the instance of the Arduino Software from which it is clicked.

- *Save*

Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.

- *Saveas...*

Allows saving the current sketch with a different name.

- *PageSetup*

It shows the Page Setup window for printing.

- *Print*

Sends the current sketch to the printer according to the settings defined in Page Setup.

- *Preferences*

Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

- *Quit*

Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

4.2.2.2 *EDIT*:

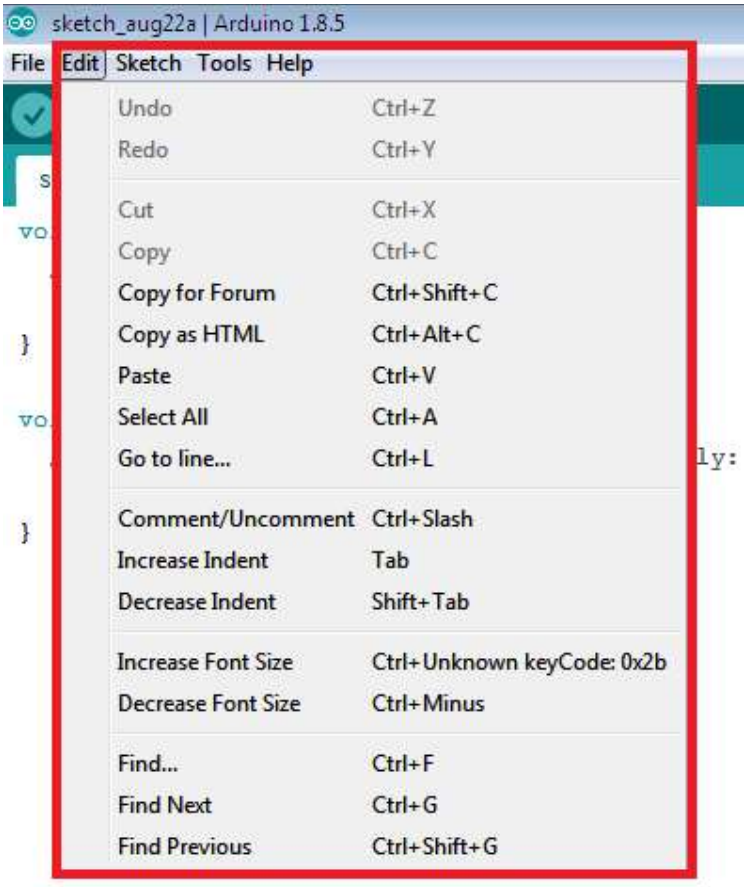


Fig:4.14 Arduino Ide Edit Columns

- *Undo/Redo*

Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

- *Cut*

Removes the selected text from the editor and places it into the clipboard.

- *Copy*

Duplicates the selected text in the editor and places it into the clipboard.

- *CopyforForum*

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

- *CopyasHTML*

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

- *Paste*

Puts the contents of the clipboard at the cursor position, in the editor.

- *SelectAll*

Selects and highlights the whole content of the editor.

- *Comment/Uncomment*

Puts or removes the // comment marker at the beginning of each selected line.

- *Increase/DecreaseIndent*

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

- *Find*

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

- *FindNext*

Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

- *FindPrevious*

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

4.2.2.3 SKETCH:

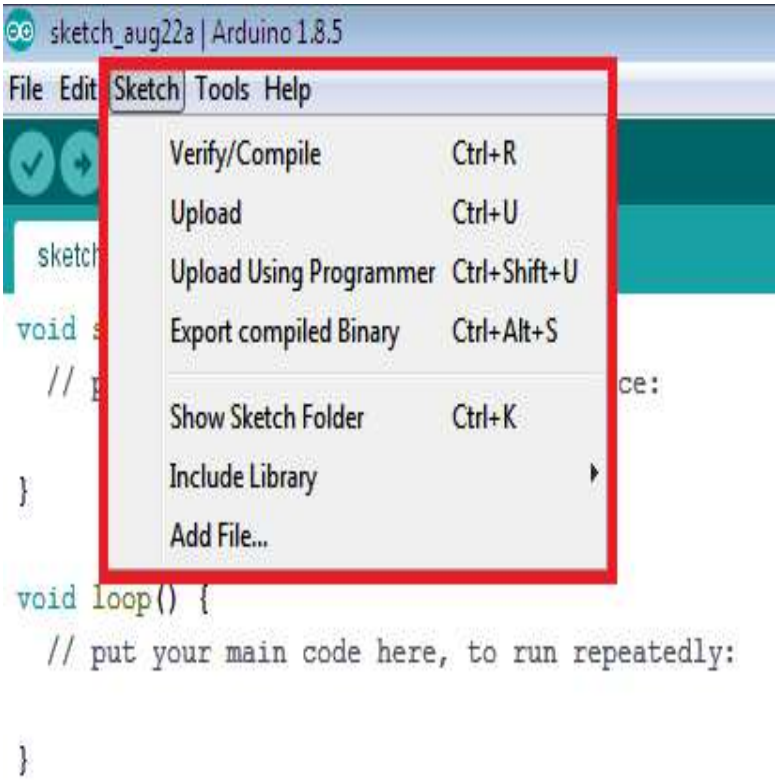


Fig:4.15 Arduino Ide Sketch Columns

- *Verify/Compile* :Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

- *Upload* :Compiles and loads the binary file onto the configured board through the configured Port.

- *Upload Using Programmer* :This will overwrite the boot loader on the board; you will need to use Tools > Burn Boot loader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.

- *Export Compiled Binary* :Saves a .hex file that may be kept as archive or sent to the board using other tools.

- *Show Sketch Folder* :Opens the current sketch folder.

- *Include Library* :Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

- *Add File...* :Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

4.2.2.4 *TOOLS*:

- *Auto Format* :This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

- *Archive Sketch* :Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

- *Fix Encoding & Reload* :Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

- *Serial Monitor* :Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- *Board* :Select the board that you're using. See below for descriptions of the various boards.

- *Port* :This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

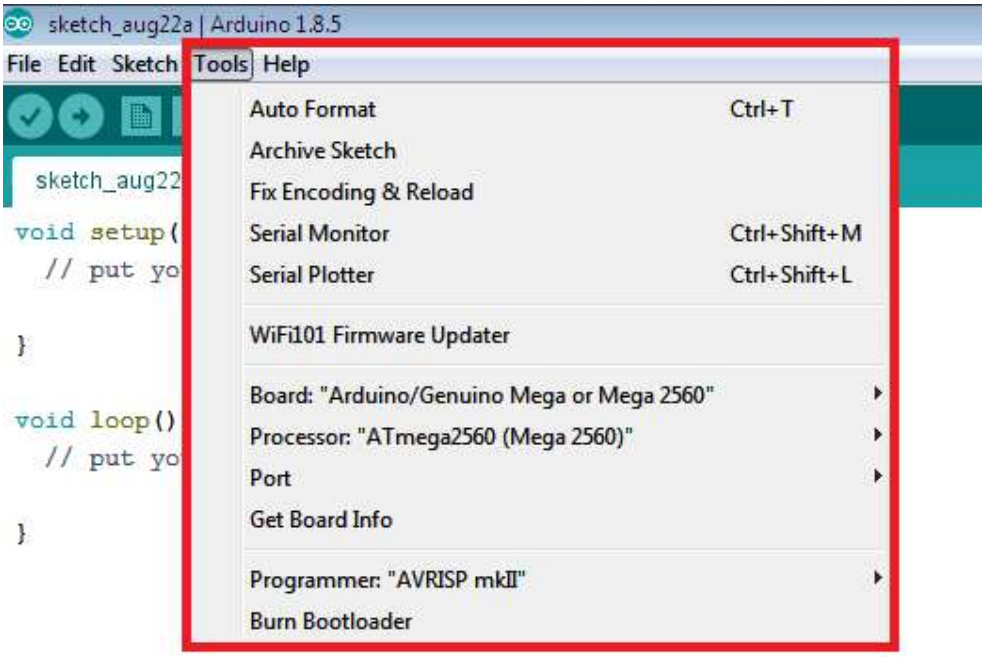


Fig:4.16 Arduino Ide Tools Columns

- *Programmer* :For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a boot loader to a new microcontroller, you will use this.

- *Burn Boot loader* :The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally comes without a boot loader). Ensure that you've selected the correct board from the Boards menu before burning the boot loader on the target board. This command also set the right fuses.

4.2.2.5 *Help*:

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The

documents are a local copy of the online ones and may link back to our online website.

Fig:4.17 arduino Help Tools columns

- *FindinReference* :This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

4.2.2.6 SKETCHBOOK:

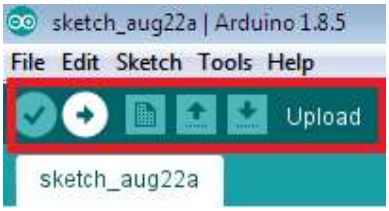
The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

4.2.2.7 TABS, MULTIPLE FILES, AND COMPILATION:

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

4.2.2.8 UPLOADING:



Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll

see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The boot loader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

4.2.2.9 LIBRARIES:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

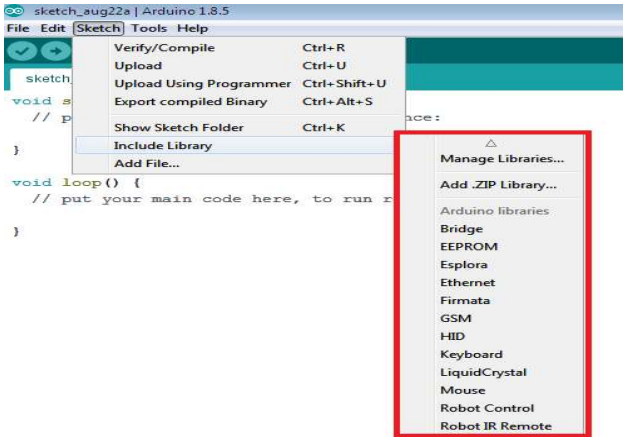


Fig:4.18 Arduino Sketch Tools Columns

4.2.2.10 THIRD-PARTY HARDWARE:

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

- SERIAL MONITOR:

  This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

  You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

- PREFERENCES:

  Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

- LANGUAGE SUPPORT:

  Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is

determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 RESULTS

The project's result demonstrates an effective hardware system facilitating mobility and communication for non-ambulatory individuals through voice control. Discussion highlights its positive impact on user independence, safety, and social inclusion. Challenges such as technical glitches or battery depletion underscore the need for ongoing maintenance and improvement. Despite occasional interruptions, the system's overall success in enhancing users' quality of life remains evident, reaffirming its significance in empowering individuals with limited mobility to navigate their environment and interact with others more freely. Continued research and development promise further advancements in assistive technology for this community.

5.1.1 FORWARD

When the output of the project indicates "forward," it signifies successful translation of the user's voice command into motion, propelling the wheelchair or mobility device in a forward direction. This outcome demonstrates effective integration of the voice recognition system with the motorized controls, enabling seamless execution of user instructions. The hardware's sensors ensure safe navigation, detecting obstacles and adjusting the trajectory accordingly. Overall, the "forward" output represents a significant milestone in empowering non-ambulatory individuals, granting them greater autonomy and freedom of movement through intuitive voice-controlled technology.

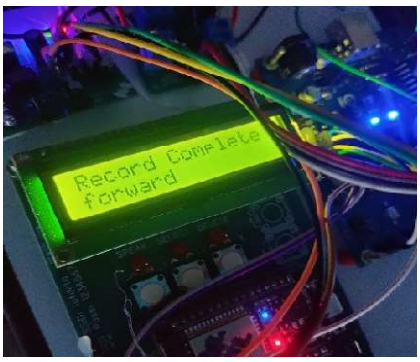


Fig:5.1 Forward (Voice Recognition)

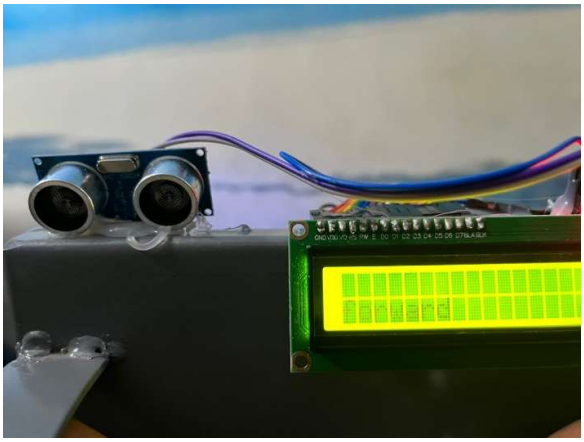


Fig:5.2 Forward (Gyroscope)

5.1.2 REVERSE

The hardware system could exhibit deficiencies like unreliable voice recognition or erratic mobility control, hindering its assistance for non-ambulatory individuals. Complex interfaces or frequent breakdowns may frustrate users, reducing their confidence in the technology and diminishing their quality of life. Inadequate obstacle detection could lead to accidents, posing safety hazards for users. The system may not cater to diverse user needs, worsening their sense of exclusion and dependence.

Connectivity issues might limit data sharing or remote monitoring opportunities, hampering the system's effectiveness.

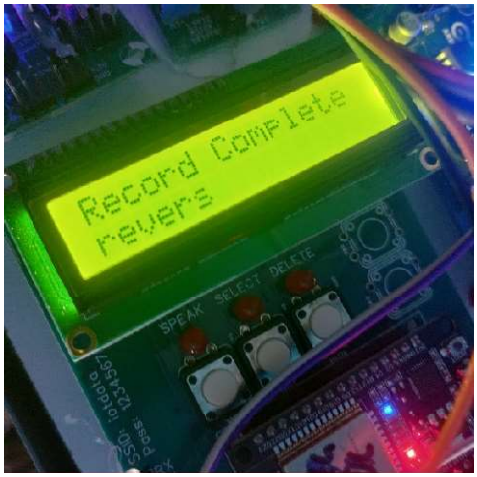


Fig:5.3 Revers (Voice Recognition)

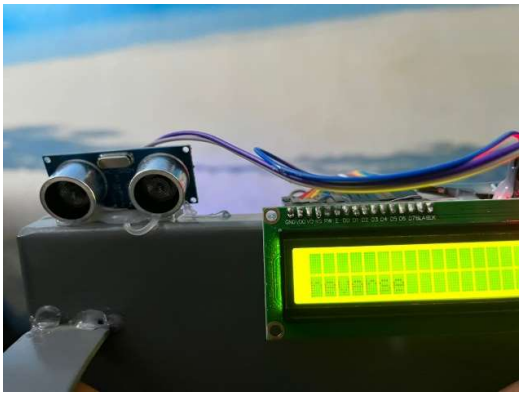


Fig:5.4 Reverse (Gyroscope)

5.1.3 RIGHT

The non-ambulatory individual locomotion with voice control system aims to provide mobility assistance to those with limited mobility. Utilizing voice commands, users can navigate their environment efficiently and independently. The system integrates advanced technology to interpret voice instructions and translate them into precise movements, facilitating seamless interaction with the environment. By prioritizing user autonomy and accessibility, it enhances the quality of life for individuals with mobility challenges, empowering them to navigate their surroundings with ease and confidence.

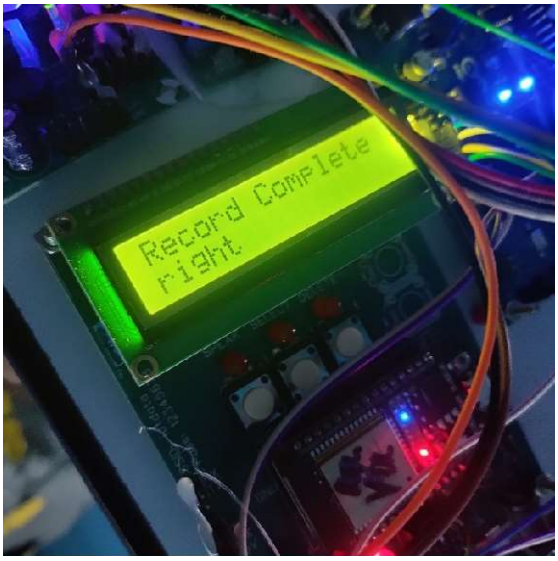


Fig:5.5 Right (Voice Recognition)

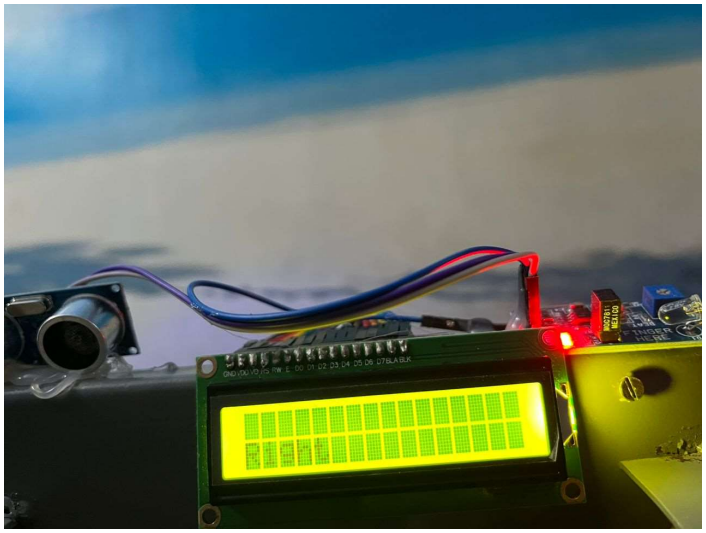


Fig:5.6 Right (Gyroscope)

5.1.4 LEFT

The output would entail a system designed for non-ambulatory individuals to navigate their environment using voice commands. It integrates voice recognition technology to interpret commands for movement, enabling users to control their locomotion effortlessly. This system aims to enhance autonomy and mobility for those with mobility impairments, providing them with greater independence in navigating their surroundings. Through intuitive voice control, users can initiate movements and navigate spaces with ease, empowering them to engage more fully in daily activities and interactions.

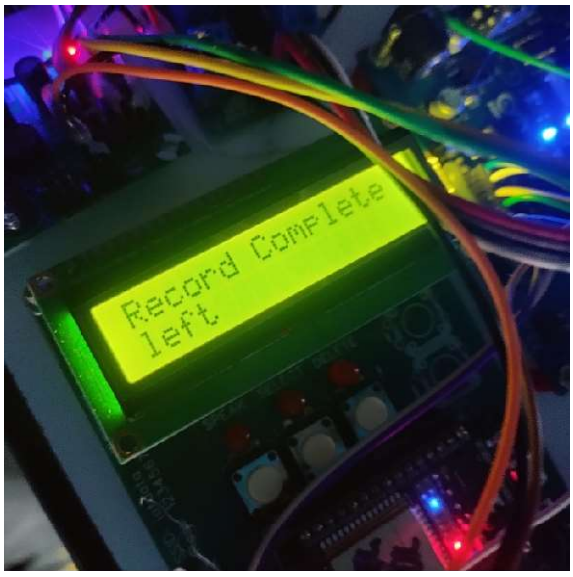


Fig:5.7 Left (Voice Recognition)

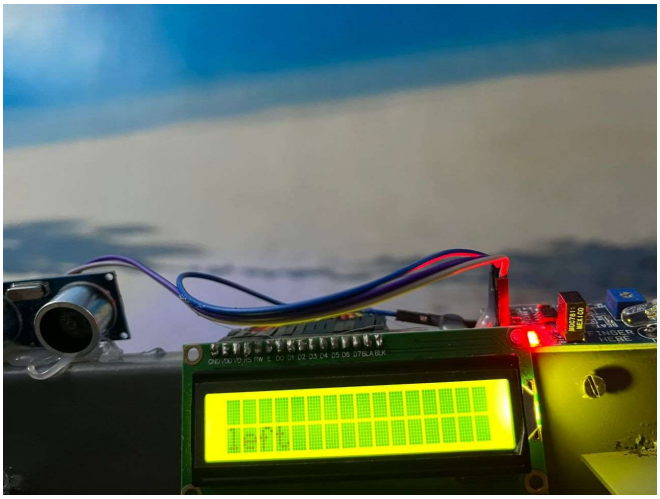


Fig:5.8 Left (Gyroscope)

5.1.5 STOP

When the output displays "stop" in this project, it signifies a halt in the system's operation due to technical issues, safety concerns, user commands, battery depletion, or scheduled maintenance. This pause prompts investigation, troubleshooting, and potential corrective action to resolve the underlying problem. Whether caused by a malfunction, safety hazard, or routine maintenance, addressing the issue ensures the continued functionality and reliability of the hardware system designed to aid non-ambulatory individuals with voice control, reaffirming its commitment to enhancing mobility, communication, and independence for users.

Fig:5.9 Stop (Voice Recognition)

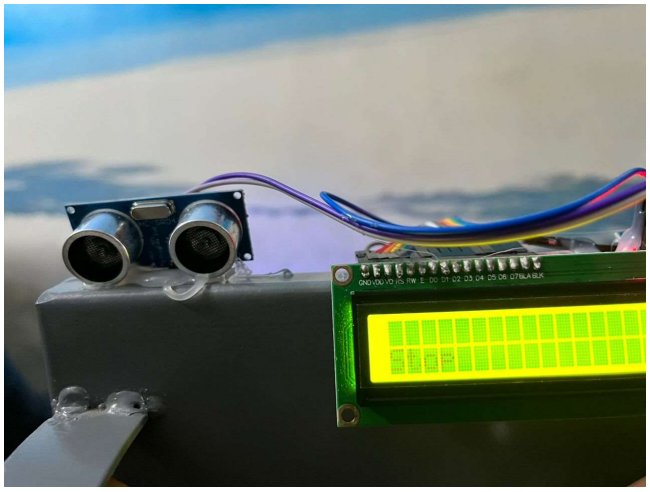


Fig:5.10 Stop (Gyroscope)

5.2 APPLICATIONS

- It is used for physically handicapped persons transportation without any others help.
- It is Effortless transport
- A Safe option for physically handicapped persons.

5.3 ADVANTAGES:

- Physically handicapped person feels good in wheelchair.
- No need for labour to take care of the person.
- Assuring collision-free travel.
- Autonomously transporting the user between locations.

CHAPTER 6
SUMMARY AND CONCLUSION

6.1 SUMMARY

Currently, a lot of nations are putting great emphasis on smart technology. In order to compete with other nations around the world, our Prime Minister of India has also launched the missions of Smart India and Digital India. Physically challenged people and the elderly are currently experiencing challenges, not people who are able to. So, we came up with the concept of a smart wheelchair for physically impaired people in order to make their lives easier and improve them. A smart wheelchair is one that can move on its own at the user's command, reducing the need for the wheelchair user to exert physical force to propel the wheels. Also, it gives physically or visually handicapped people the chance to go from one place to another place.

6.2 CONCLUSION

The "Revolutionizing Care: Smart Health Solutions for Wheelchair Patients" project has demonstrated a remarkable advancement in healthcare for individuals reliant on wheelchairs. The integrated system, utilizing smart sensors, wearable devices, and real-time monitoring technologies, has shown great promise in addressing their unique healthcare needs. The micro-controller serves as the central control unit, orchestrating the seamless coordination of various sensors to continuously track vital health metrics. The inclusion of MEMs for wheelchair movement detection, heart rate sensor for cardiac monitoring, The intelligent alert system has proven invaluable in promptly notifying caregivers and professionals of any deviations from baseline health parameters, reducing the risk of complications. This system not only represents a significant technological advancement but also fosters patient autonomy and independence. By revolutionizing care with tailored monitoring and advanced technology, it holds immense potential in improving the overall well-being and quality of life for wheelchair-bound individuals. The successful implementation and promising outcomes of this project underscore its potential for broader adoption in healthcare settings.

REFERENCES

[1] ‘WHO | World report on disability’, 2019. [Online]. Available: https://www.who.int/disabilities/world_report/2011/report/en/. [Accessed: 13-Jan-19].

[2] A. Sasou and H. Kojima, “Noise robust speech recognition applied to voice-driven wheelchair”, EURASIP Journal on Advances in Signal Processing, vol. 2009, p. 41, 2009.

[3] I. Klabi and M.S. Masmoudi, "Advanced user interfaces for intelligent wheelchair system", 1st IEEE Conference on Advanced Technologies for Signal and Image Processing, pp.130-136, Tunisia, 2014.

[4] M. Carmel, V. Brindha, and A. Abudhahir. "Facial expression recognition using PCA based interface for wheelchair." International Conference on IEEE Electronics and Communication Systems (ICECS), France, 2014.

[5] S. D. Suryawanshi, J. S. Chitode, and S. S. Pethakar, “Voice operated intelligent wheelchair,” International Journal of Advanced Research in Computer Science and Software Engineering, vol-3, issue-5, May 2013.

[6] M. H. A. Sibai, and S. A. Manap, “A study on smart wheelchair systems,” International Journal of Engineering Technology and Sciences (IJETS), vol-4, issue-1, December 2015.

[7] K.-H. Kim, H. K. Kim, J.-S. Kim, W. Son and S.- Y. Lee, “A biosignal-based human interface controlling a power-wheelchair for people with motor disabilities,” ETRI Journal, vol.28, no.1, 2006, pp.111–114.

[8] K. Choi, M. Sato and Y. Koike, “Consideration of the embodiment of a new, human-centered interface,” IEICE Trans. Inf. & Syst., vol.E89-D, no.6, 2006, pp.1826–1833.

[9] Madarasz R.L, Heiny L.C, Cromp R.F, Mazur N.M (1986), “The design of an autonomous vehicle for the disabled”, IEEE Robotics and Automation, Vol 2 No: 3 pg.117-126

[10] Pires G, Nunes U, De Almeida A.T (1998), “Robchair-A Semi-Autonomous Wheelchair for Disabled People”, Proc. 3rd IFAC Symposium on Intelligent Autonomous Vehicles (IAV’98) pg. 648- 652

APPENDIX

SOURCE CODE

```
#include <ultrasonic.h>

#include"lcd.h"

#include<Wire.h>

const int MPU1 = 0x68;          //address for SENSOR 1  (sensor that is placed on the shank)

float GX, GY, GZ;               //global variables for SENSOR 1 raw data  (sensor that is placed on the
shank)

float GYX, GYY, GYZ;

int minVal = 265;               //control the sensor angular velocity

int maxVal = 402;

ULTRASONIC U1;

char inchar;

String d = "";

char data1 = 'x';

int d1 = 1;

int d2 = 0;

#define mode_pin 2

int G_i;

int hr;

int count = 0;

int cm;

void setup()

{

  Serial.begin(115200);

  lcd.begin(16, 2);

  lcd.clear();

  lcd.setCursor(2, 0);
```

```
  lcd.print("INTELLIGENT");
  lcd.setCursor(3, 1);
  lcd.print("WHEELCHAIR");
  U1.begin(A0, A1);
  pinMode(BUZZER, OUTPUT);
  digitalWrite(BUZZER, LOW);
  pinMode(motor1_pin1, OUTPUT);  /* Motor control pin 1 */
  pinMode(motor1_pin2, OUTPUT);  /* Motor control pin 2 */
  pinMode(motor2_pin1, OUTPUT);  /* Motor2 control pin 1 */
  pinMode(motor2_pin2, OUTPUT);  /* Motor2 control pin 2 */
  digitalWrite(motor1_pin1, LOW);
  digitalWrite(motor1_pin2, LOW);
  digitalWrite(motor2_pin1, LOW);
  digitalWrite(motor2_pin2, LOW);
  pinMode(HR, INPUT);
  pinMode(mode_pin, INPUT_PULLUP);
  Wire.begin();                    //SENSOR 1  (sensor that is placed on the shank)
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:

  if (digitalRead(mode_pin) == 1)
```

```
{
  cm = U1.ultra();
  if (cm < 30)
  {
    lcd.setCursor(0, 1);
    lcd.print("Stop   ");
    digitalWrite(motor1_pin1, LOW);
    digitalWrite(motor1_pin2, LOW);
    digitalWrite(motor2_pin1, LOW);
    digitalWrite(motor2_pin2, LOW);
    digitalWrite(BUZZER, HIGH);
    delay(2000);
    digitalWrite(BUZZER, LOW);
  }
  gyro();
  heart();
  if (G_i != 120 && G_i != 0)
  {
    hr = G_i;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("HR:" + String(hr));
  }
}
if (digitalRead(mode_pin) == 0)
{
  cm = U1.ultra();
  // Serial.println(cm);
```

```
    if (cm < 30)
    {
      lcd.setCursor(0, 1);
      lcd.print("Stop   ");
      digitalWrite(motor1_pin1, LOW);
      digitalWrite(motor1_pin2, LOW);
      digitalWrite(motor2_pin1, LOW);
      digitalWrite(motor2_pin2, LOW);
      digitalWrite(BUZZER, HIGH);
      delay(2000);
      digitalWrite(BUZZER, LOW);
    }
    heart();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("HR:" + String(hr));
    if (G_i != 120 && G_i != 0)
    {
      hr = G_i;
    }
    serialEvent();
  }
}
void serialEvent()
{
  while (Serial.available() > 0)
  {
    inchar = Serial.read();
```

```
  if (inchar == '*')
  {
    while (Serial.available() > 0)
    {
      char t = Serial.read();
      if (count == 7)
      {
        break;
      }
      if (t == '#')
      {
        break;
      }
      d += t;
    }
  }
}
if (d == "forward")
{
  data1 = 'F';
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("FORWARD");
  Motor(data1);
}
if (d == "reverse")
{
  lcd.clear();
```

```
  lcd.setCursor(2, 0);
  lcd.print("REVERSE");
  data1 = 'B';
  Motor(data1);
}
if (d == "right")
{
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("RIGHT");
  data1 = 'R';
  Motor(data1);
}
if (d == "left")
{
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("LEFT");
  data1 = 'L';
  Motor(data1);
}
if (d == "stop")
{
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("STOP");
  data1 = 'S';
  Motor(data1);
```

```
  }
}
void Motor(char c)
{
  lcd.clear();
  switch (c)
  {
    case 'F':
      lcd.setCursor(0, 1);
      lcd.print("forward");
      digitalWrite(motor1_pin1, d2);
      digitalWrite(motor1_pin2, d2);
      digitalWrite(motor2_pin1, d2);
      digitalWrite(motor2_pin2, d2);
      delay(1000);
      digitalWrite(motor1_pin1, d1);
      digitalWrite(motor1_pin2, d2);
      digitalWrite(motor2_pin1, d1);
      digitalWrite(motor2_pin2, d2);
      break;
    case 'E':
      lcd.setCursor(0, 1);
      lcd.print("reverse");
      digitalWrite(motor1_pin1, d2);
      digitalWrite(motor1_pin2, d2);
      digitalWrite(motor2_pin1, d2);
      digitalWrite(motor2_pin2, d2);
      delay(1000);
```

```
  digitalWrite(motor1_pin1, d2);
  digitalWrite(motor1_pin2, d1);
  digitalWrite(motor2_pin1, d2);
  digitalWrite(motor2_pin2, d1);
  break;
 case 'R':
  lcd.setCursor(0, 1);
  lcd.print("Right  ");
  digitalWrite(motor1_pin1, d2);
  digitalWrite(motor1_pin2, d2);
  digitalWrite(motor2_pin1, d2);
  digitalWrite(motor2_pin2, d2);
  delay(1000);
  digitalWrite(motor1_pin1, d1);
  digitalWrite(motor1_pin2, d2);
  digitalWrite(motor2_pin1, d2);
  digitalWrite(motor2_pin2, d1);
  break;
 case 'L':
  lcd.setCursor(0, 1);
  lcd.print("left   ");
  digitalWrite(motor1_pin1, d2);
  digitalWrite(motor1_pin2, d2);
  digitalWrite(motor2_pin1, d2);
  digitalWrite(motor2_pin2, d2);
  delay(1000);
  digitalWrite(motor1_pin1, d2);
  digitalWrite(motor1_pin2, d1);
```

```
      digitalWrite(motor2_pin1, d1);
      digitalWrite(motor2_pin2, d2);
      break;
    case 'S':
      lcd.setCursor(0, 1);
      lcd.print("Stop   ");
      digitalWrite(motor1_pin1, d2);
      digitalWrite(motor1_pin2, d2);
      digitalWrite(motor2_pin1, d2);
      digitalWrite(motor2_pin2, d2);
      delay(1000);
      break;
  }
  d = "";
}
void gyro()
{
  Wire.beginTransmission(MPU1);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU1, 12, true);
  GX = Wire.read() << 8 | Wire.read();              //raw data reading for the angular velocity for SENSOR
1  (sensor that is placed on the shank)
  GY = Wire.read() << 8 | Wire.read();
  GZ = Wire.read() << 8 | Wire.read();


  int X1ANG = map(GX, minVal, maxVal, -90, 90);      // change the raw data into degrees for SENSOR
1  (sensor that is placed on the shank)
  int Y1ANG = map(GY, minVal, maxVal, -90, 90);
```

```
  int Z1ANG = map(GZ, minVal, maxVal, -90, 90);


  GYX = RAD_TO_DEG * (atan2(-Y1ANG, -Z1ANG) + PI);

  GYY = RAD_TO_DEG * (atan2(-X1ANG, -Z1ANG) + PI);

  GYZ = RAD_TO_DEG * (atan2(-Y1ANG, -X1ANG) + PI);


  //Serial.println(GYX);

  lcd.setCursor(0, 1);

  lcd.print("X=" + String(GYX));

  lcd.setCursor(6, 1);

  lcd.print("Y=" + String(GYY));

  lcd.setCursor(11, 1);

  lcd.print("Z=" + String(GYZ));


  if (GYX >= 355 && GYX <= 359 || GYX >= 0 && GYX <= 25 && GYY >= 330 && GYY <= 359 ||
GYY >= 0 && GYY <= 6 && GYZ >= 80 && GYZ <= 220) //&& ((GYY >= 200) && (GYY <= 320)) &&
((GYZ >= 200) && (GYZ <= 320)))

  {

    Motor('S');

    //ss.write('N');

  }

  if (GYX >= 25 && GYX <= 100 && GYY <= 359 && GYZ <= 114 ) // && ((GYY >= 310) && (GYY <=
340)) && ((GYZ >= 80) && (GYZ <= 130)))

  {

    Motor('R');

    //ss.write('R');

  }

  if (GYX >= 300 && GYX <= 359 && GYY >= 330 && GYZ >= 221) // && ((GYY >= 310) && (GYY
<= 340)) && ((GYZ >= 80) && (GYZ <= 130)))

  {
```

```
    Motor('L');

    //ss.write('L');

  }

  if (GYY <= 330 && GYY <= 330 && GYZ >= 160) //&& ((GYY >= 50) && (GYY <= 90)) && ((GYZ
>= 0) && (GYZ <= 50)))

  {

    Motor('F');

    //ss.write('F');

  }

  if (GYX >= 0 && GYX <= 25 && GYY <= 70 && GYZ <= 110 ) //&& ((GYY >= 270) && (GYY <=
345)) && ((GYZ >= 150) && (GYZ <= 185)))

  {

    Motor('E');

    //ss.write('B');

  }

}




void heart(void)

{

  int j = 120;

  int i = 0;

  while (j > 0)

  {

    if (digitalRead(HR) == LOW)

      i++;

    j--;

    delay(14);

  }
```

```
    G_i = i;



}
```