A

MAJOR PROJECT REPORT ON

DESIGN AND DEVELOPMENT OF A HEAD, BLUETOOTH AND VOICE CONTROLLED WHEELCHAIR

Submitted in partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SUBMITTED BY

KRISHNA KANTH	**218R1A0496**
KURRI NAVYA	**218R1A0497**
KURAMA RAHUL	**218R1A0498**
LINGAMPALLY VENKATA SAI	**218R1A0499**

Under the Esteemed Guidance of

Mrs. G. PRAVALIKA
Assistant professor

CMR
ENGINEERING COLLEGE
EXPLORE TO INVENT

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA)

Kandlakoya(V), Medchal(M), Telangana – 501401

(2024-2025)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA)

Kandlakoya(V), Medchal Road, Hyderabad - 501 401

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the major-project work entitled **"DESIGN AND DEVELOPMENT OF HEAD, BLUETOOTH AND VOICE CONTROLLED WHEELCHAIR"** is being submitted by **KRISHNA KANTH** bearing Roll No **218R1A0496, KURRI NAVYA** bearing Roll No **218R1A0497, KURAMA RAHUL** bearing Roll No **218R1A0498, VENKATA SAI** bearing Roll No **218R1A0499** in B.Tech IV-II semester, Electronics and Communication Engineering is a record Bonafide work carried out during the academic year 2024-25. The results embodied in this report have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE
Mrs. PRAVALIKA

HEAD OF THE DEPARTMENT
Dr. SUMAN MISHRA

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We sincerely thank the management of our college **CMR Engineering College** for providing required facilities during our project work. We derive great pleasure in expressing our sincere gratitude to our Principal **Dr. A. S. REDDY** for his timely suggestions, which helped us to complete the project work successfully. It is the very auspicious moment we would like to express our gratitude to **Dr. SUMAN MISHRA,** Head of the Department, ECE for his consistent encouragement during the progress of this project.

We take it as a privilege to thank our project coordinator **Dr. T. SATYANARAYANA**, Associate Professor, Department of ECE for the ideas that led to complete the project work and we also thank him for his continuous guidance, support and unfailing patience, throughout the course of this work. We sincerely thank our project internal guide **Mrs. PRAVALIKA,** Assistant Professor of ECE for guidance and encouragement in carrying out this project work.

DECLARATION

We hereby declare that the major project entitled **"DESIGN AND DEVELOPMENT OF A HEAD, BLUETOOTH AND VOICE CONTROLLED WHEELCHAIR"** is the work done by us in campus at **CMR ENGINEERING COLLEGE,** Kandlakoya during the academic year 2024-2025 and is submitted as major project in partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING FROM JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD.**

KRISHNA KANTH **(218R1A0496)**

K. NAVYA **(218R1A0497)**

K. RAHUL **(218R1A0498)**

L. VENKATA SAI **(218R1A0499)**

ABSTRACT

This project presents an innovative approach to wheelchair mobility through the design and development of a head, Bluetooth, and voice-controlled wheelchair. This system aims to empower individuals with limited hand dexterity or those who find traditional joystick controls challenging to navigate comfortably.

The proposed solution includes a specialized head-mounted device, Bluetooth connectivity, and a voice recognition system equipped with advanced sensors to detect head gestures and spoken commands. These inputs are translated into precise controls, enabling the user to navigate the wheelchair's direction and speed effectively. By leveraging advanced technology, the project emphasizes the design of the head control system, seamless Bluetooth integration, the development of robust voice-command algorithms, and the implementation of a user-friendly wireless communication system.

The anticipated outcome is a highly functional and intuitive wheelchair system that significantly enhances mobility and independence for individuals with disabilities. This innovative solution has the potential to improve the quality of life for many by providing a more accessible and efficient means of wheelchair control, thus promoting greater autonomy and comfort for users.

CONTENTS

CHAPTERS	PAGE
CERTIFICATE	I
ACKNOWLEDGEMENTS	II
DECLARATION	III
ABSTRACT	i
CONTENTS	ii
LIST OF FIGURES	iii
LIST OF TABLES	iv
CHAPTER-1	
INTRODUCTION	**1**
1.1 OVERVIEW OF THE PROJECT	1
1.2 OBJECTIVE OF THE PROJECT	1
1.3 ORGANIZATION OF THE PROJECT	2
CHAPTER-2	
LITERATURE SURVEY	**4**
2.1 EXISTING SYSTEM	4
2.2 PROPOSED SYSTEM	4
2.3 EMBEDDED INTRODUCTION	5
2.4 WHY EMBEDDED?	9
2.5 DESIGN APPROACHES	10
2.6 COMBINATION OF LOGIC DEVICE	15
CHAPTER-3	
HARDWARE REQUIREMENTS	**17**
3.1 HARDWARE	17
CHAPTER-4	
SOFTWARE REQUIREMENTS	**23**
4.1 SOFTWARE	23
4.2 RESEARCH	26
CHAPTER-5	
WORKING AND COMPONENTS	**29**
5.1 BLOCK DIAGRAM	29
5.2 WORKING	30
5.2.1 INTRODUCTION TO ARDUINO	32

CHAPTERS **PAGE**

5.2.2 BLOCK DIAGRAM 49

5.2.3 INTRODUCTION TO BLUETOOTH MODULE 50

5.2.4 BLUETOOTH MODULE AND WORKING PRINCIPLE 51

5.2.5 INTRODUCTION TO MEMS SENSOR 51

5.2.6 INTRODUCTION TO VOICE CONTROL MODULE 53

5.2.7 BLOCK DIAGRAM 53

CHAPTER-6 **55**

RESULTS 55

ADVANTAGES 56

APPLICATIONS 57

CONCLUSION 57

FUTURE SCOPE 57

REFERENCES **59**

APPENDIX 60

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
2.1	EMBEDDED OS	6
2.2	EMBEDDED SYSTEMS	8
2.3	BLOCKS OF EMBEDDED SYSTEMS	9
2.4	EMBEDDED SYSTEMS HARDWARE	10
2.5	EMBEDDED DESIGN- PROCESS- STEPS	11
2.6	APPLICATION OF EMBEDDED SYSTEMS	14
2.7	LOGIC GATES	15
2.8	EMBEDDED SYSTEMS GROUP	16
3.1	EMBEDDED SYSTEMS HARDWARE BLOCK DIAGRAM	18
3.2	BASIC EMBEDDED STRUCTURE	20
4.1	ARDUINO UNO	23
4.2	DEVICE MANAGER	24
4.3	UPDATE DRIVE SOFTWARE	24
4.4	ARDUINO FILE BASIC	25
4.5	ARDUINO TOOLS BOARD	26
4.6	BLINK ARDUINO TOOLS	26
5.1	BLOCK DIAGRAM OF HEAD, BLUETOOTH AND VOICE CONTROLLED WHEELCHAIR	29
5.2	STRUCTURE OF ARDUINO BOARD	33
5.3	ARDUINO BOARD	33
5.4	PIN CONFIGURATION OF ATMEGA328	36
5.5	AVR BLOCK DIAGRAM	40
5.6	AVR STATUS REGISTER	41
5.7	ARDUINO BLOCK DIAGRAM	49
5.8	BLUETOOTH MODULE	50
5.9	WORKING OF BLUETOOTH MODULE	50
5.10	MEMS SENSOR	52
5.11	BLOCK DIAGRAM OF VOICE CONTROL MODULE	54
6.1	RESULT OF HEAD, BLUETOOTH AND VOICE CONTROLLED WHEELCHAIR	55
6.2	RESULT OF HEAD MOTION CONTEROL WITH CAP	56

LIST OF TABLES

TABLE NO	LIST OF TABLE NAME	PAGE NO
2.1	EMBEDDED SYSTEMS DESIGNS SOFTWARE DEVELOPMENT ACTIVITIES	14
5.2.1	STACK POINTER INSTRUCTIONS	43
5.2.2	RESET AND INTERRUPT VECTORS	45
5.2.3	RESET AND INTERRUPT VECTORS PLACEMENT	47
5.2.4	COMMAND AND FUNCTION OF VOICE CONTROL MODULE	54

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

In recent years, technological advancements have significantly improved the quality of life for individuals with disabilities. One such innovation is the head, Bluetooth, and voice-controlled wheelchair. This cutting-edge technology empowers users with limited hand dexterity or those who find traditional joystick controls challenging.

The core components of this system include a specialized head-mounted device, Bluetooth module, and voice recognition system equipped with advanced sensors. These sensors, such as gyroscopes and microphones, accurately capture head movements and voice commands. The collected data is then processed by a sophisticated control algorithm to extract relevant information and generate precise commands for the wheelchair's motors. The algorithm translates head gestures and spoken instructions into specific actions, such as forward, backward, left, right, and stop.

To ensure seamless communication between the control system and the wheelchair, a robust wireless communication system is implemented. This system transmits control signals from the head-mounted device and voice module to the wheelchair's control unit, By providing greater control and flexibility, this technology can significantly enhance the quality of life for individuals with disabilities. It offers a more intuitive and efficient means of wheelchair control, empowering users to navigate their surroundings with greater ease and independence.

1.2 OBJECTIVE OF THE PROJECT

The objective of the Head, Bluetooth, and Voice-Controlled Wheelchair project is to enhance mobility solutions by integrating multiple control mechanisms to provide users with a more flexible, efficient, and accessible means of transportation. The primary aim is to develop a wheelchair that can be controlled using head movements, voice commands, and Bluetooth-enabled devices, thereby offering users multiple modes of operation based on their comfort and ability. This multi-modal control system is designed to assist individuals with varying levels of mobility impairment, making independent movement more feasible and convenient.

In addition to improving accessibility, the project seeks to enhance the responsiveness and accuracy of wheelchair control by employing advanced sensor-based technologies.

Head control system utilizes motion sensors to detect directional tilts, while the voice recognition system processes spoken commands for movement control. These features work alongside a Bluetooth interface that allows users to control the wheelchair remotely via a smartphone or tablet, ensuring a reliable alternative for navigation. The wheelchair is also designed to operate efficiently in different environments, providing smooth navigation and stability on various surfaces.

The project also aims to improve user safety and comfort through real-time feedback systems and obstacle detection. By integrating smart sensors, the wheelchair can alert users to potential obstacles, preventing accidents and ensuring a secure riding experience. Furthermore, real-time tracking and monitoring capabilities can assist caregivers or healthcare providers in keeping track of the user's movement and well-being.

By combining automation with user-centric design, this project aims to empower individuals with mobility challenges by providing a smarter, more intuitive, and reliable wheelchair system. Ultimately, the goal is to enhance independence, improve daily mobility, and contribute to a more inclusive and technologically advanced assistive device ecosystem.

1.3 ORGANIZATION OF THE PROJECT

The organization of the Head, Bluetooth, and Voice-Controlled Wheelchair project involves a structured approach, starting with the planning and initiation phase where project objectives, scope, and requirements are clearly defined. This includes determining the functionalities, performance metrics, and budget, as well as developing a timeline with key milestones. Following this, the research and development phase begins with an investigation into the appropriate technologies for head movement detection, voice recognition, and Bluetooth-based control integration. This leads to the creation of design concepts and the development of a prototype to test these ideas.

During the hardware development phase, the physical components of the wheelchair are built, including the frame, sensors, motion detectors, and necessary electronic systems for multi-modal control. Concurrently, software development focuses on integrating the various control mechanisms and developing user interfaces for real-time monitoring and navigation assistance. Once the prototype is tested and refined, the project moves into the

deployment phase, where the wheelchair is introduced in selected environments for pilot testing.

Finally, ongoing support and maintenance ensure the system's reliability and address any issues that arise, ensuring a successful and efficient implementation of the smart wheelchair system.

CHAPTER-2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

In the present system, individuals with mobility impairments face challenges in independently controlling their wheelchairs. Traditional wheelchairs rely solely on manual operation, which can be difficult for users with limited upper body strength or dexterity. While some advanced wheelchairs exist, they often lack multiple control options, making them less adaptable to different user needs. The use of the existing system leads to the following problems: Individuals with severe disabilities may struggle to operate a traditional joystick. Lack of alternative control methods limits accessibility for users with varying levels of mobility impairment. Maneuvering a wheelchair in tight spaces can be difficult without precise control. Dependency on caregivers increases when users are unable to operate wheelchairs independently.

All these problems can be addressed in this project through the Head, Bluetooth, and Voice-Controlled Wheelchair system. Users will have multiple control options, including head movement, voice commands, and Bluetooth-enabled remote operation, making wheelchair navigation more accessible. The integration of smart control mechanisms allows for greater independence and ease of use. While effective for basic mobility, traditional wheelchairs rely heavily on manual input, which can be restrictive for individuals with severe physical limitations. The absence of smart control options reduces adaptability in dynamic environments, limiting user autonomy and efficiency.

2.2 PROPOSED SYSTEMS

The advent of assistive technologies has significantly improved the quality of life for individuals with disabilities. However, traditional wheelchair control methods, such as joysticks, often present challenges for users with limited hand dexterity or those who find them cumbersome. To address these limitations, a revolutionary Head, Bluetooth, and Voice-Controlled Wheelchair system is proposed.

This innovative system leverages the power of advanced sensor technology and sophisticated control algorithms to empower users with greater independence and mobility. The core components of the system include a head movement detection module, a voice recognition system, and a Bluetooth-enabled control interface. These sensors accurately

capture head gestures, voice commands, and Bluetooth-based inputs, translating them into precise control commands for the wheelchair.

A sophisticated control algorithm processes the sensor data in real time, extracting relevant information and generating appropriate control signals. The algorithm maps specific head movements, spoken instructions, and Bluetooth commands to various wheelchair functions such as forward, backward, left, right, and stop. By customizing the algorithm to individual user preferences, the system can be tailored to meet diverse needs and abilities.

To ensure seamless communication between the control modules and the wheelchair, a robust wireless communication system is implemented. This system transmits control signals to the wheelchair's control unit, allowing for real-time navigation. By utilizing reliable and low-latency wireless technology, the system ensures a seamless and responsive user experience.

The integration of the multi-control system with a suitable wheelchair platform is crucial. The wheelchair's control unit is modified to receive and process the wireless signals from the head movement, voice commands, and Bluetooth interface, enabling precise control of the wheelchair's motors. Compatibility with various wheelchair models and configurations is ensured to accommodate a wide range of user preferences and requirements.

The potential benefits of this Head, Bluetooth, and Voice-Controlled Wheelchair system are significant. By providing a more intuitive and efficient means of wheelchair control, it can significantly enhance the mobility and independence of individuals with disabilities. Users can navigate their surroundings with greater ease, access public spaces more readily, and participate in social activities with confidence.

Furthermore, the system can be customized to suit individual needs, allowing users to tailor the control scheme to their preferences and abilities. By continuously refining the technology and exploring new possibilities, the goal is to create a truly transformative solution that empowers individuals with disabilities and improves their overall quality of life.

2.3 INTRODUCTION TO EMBEDDED SYSTEMS

An embedded system is a combination of computer hardware and software designed for a specific function or functions within a larger system. The systems can be programmable or

with fixed functionality. Industrial machines, consumer electronics, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

While embedded systems are computing systems, they can range from having no user interface (UI) -- for example, on devices in which the system is designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. User interfaces can include buttons, LEDs and touchscreen sensing. Some systems use remote user interfaces as well.

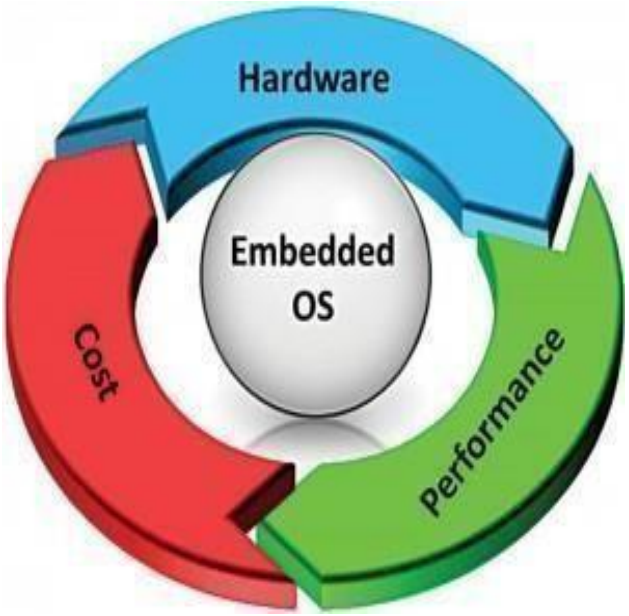

Fig 2.1: Embedded OS

History of embedded systems

Embedded systems date back to the 1960s. Charles Stark Draper developed an integrated circuit (IC) in 1961 to reduce the size and weight of the Apollo Guidance Computer, the digital system installed on the Apollo Command Module and Lunar Module. The first computer to use ICs, it helped astronauts collect real-time flight data.

In 1965, Autonoetic, now a part of Boeing, developed the D-17B, the computer used in the Minuteman I missile guidance system. It is widely recognized as the first mass-produced embedded system. When the Minuteman II went into production in 1966, the D-17B was replaced with the NS-17 missile guidance system of integrated circuits.

In 1968, the first embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system.

By the late 1960s and early 1970s, the price of integrated circuits dropped, and usage surged. The first microcontroller was developed by Texas Instruments in 1971. The TMS 1000 series, which became commercially available in 1974, contained a 4-bit processor, read-only memory (ROM) and random-access memory (RAM), and cost around $2 apiece in bulk orders.

Also, in 1971, Intel released what is widely recognized as the first commercially available processor, the 4004. The 4-bit microprocessor was designed for use in calculators and small electronics, though it required eternal memory and support chips. The 8-bit Intel 8008, released in 1972, had 16 KB of memory; the Intel 8080 followed in 1974 with 64 KB of memory. The 8080's successor, x86 series, was released in 1978 and is still largely in use today.

In 1987, the first embedded operating system, the real-time VxWorks, was released by Wind River, followed by Microsoft's Windows Embedded CE in 1996. By the late 1990s, the first embedded Linux products began to appear. Today, Linux is used in almost all embedded devices.

Characteristics of embedded systems

The main characteristic of embedded systems is that they are task specific. They perform a single task within a larger system. For example, a mobile phone is *not* an embedded system, it is a combination of embedded systems that together allow it to perform a variety of general-purpose tasks. The embedded systems within it perform specialized functions. For example, the GUI performs the singular function of allowing the user to interface with the device. In short, they are programmable computers, but designed for specific purposes, not general ones.

The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers, and generally refer to a CPU that is integrated with other basic computing components such as memory chips and digital signal processors (DSP). Microcontrollers have those components built into one chip.

Additionally, embedded systems can include the following characteristics: comprised of hardware, software and firmware; embedded in a larger system to perform a specific function as they are built for specialized tasks within the system, not various tasks; either microprocessor-based or microcontroller-based -- both are integrated circuits that give the system compute power; often used for sensing and real-time computing.

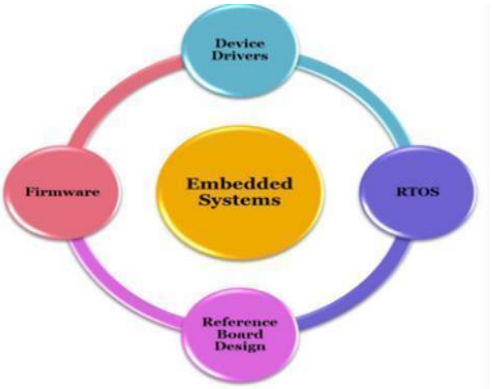

Fig 2.2: Embedded Systems

In internet of things (IoT) devices -- devices that are internet-connected and do not require a user to operate; vary in complexity and in function, which affects the type of software, firmware and hardware they use; and often required to perform their function under a time constraint to keep the larger system functioning properly. Embedded systems vary in complexity, but generally consist of three main elements:

- **Hardware.** The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers, and generally refer to a CPU that is integrated with other basic computing components such as memory chips and digital signal processors (DSP). Microcontrollers have those components built into one chip.

- **Software.** Software for embedded systems can vary in complexity. However, industrial grade microcontrollers and embedded IoT systems generally run very simple software that requires little memory.

- **Firmware.** Embedded firmware is usually used in more complex embedded systems to connect the software to the hardware. Firmware is the software that interfaces directly with the hardware. A simpler system may just have software directly in the chip, but more complicated systems need firmware under more complex software applications and

operating systems.

Fig 2.3: Blocks Of Embedded Systems

2.4 WHY EMBEDDED?

An embedded system is a computer system with a particular defined function within a larger mechanical or electrical system. They control many devices in common use. They consume low power, are of a small size and their cost is low per-unit. Modern embedded systems are often based on micro-controllers. A microcontroller is a small computer on a single integrated circuit which contains a processor core, memory, and programmable input and output peripherals. As Embedded system is dedicated to perform specific tasks therefore, they can be optimized to reduce the size and cost of the product and increase the reliability

and performance. Almost every Electronic Gadget around us is an Embedded System, digital watches, MP3 players, Washing Machine, Security System, scanner, printer, a cellular phone, Elevators, ATM, Vendor Machines, GPS, traffic lights, Remote Control, Microwave Oven and many more. The uses of embedded systems are virtually limitless because every day new products are introduced to the market which utilize embedded computers in a number of ways

Embedded Systems has brought about a revolution in Science. It is also a part of a Internet of Things (IoT) – a technology in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. Let’s make it easy for you. For Example – You are sitting in a train headed to your destination and you are already fifty miles away from your home and suddenly you realise that you forgot to switch of the fan. Not to worry, you can switch it off just by clicking a button on your cell phone using this technology – The Internet of Things.

Well this is just one good thing about IoT. We can monitor Pollution Levels, we can control the intensity of street lights as per the season and weather requirements, IoT can also provide the parents with real-time information about their baby's breathing, skin temperature, body position, and activity level on their smartphones and many other applications which can make our life easy.

Fig 2.4: EMBEDDED SYSTEMS HARDWARE

2.5 DESIGN APPROACHES

A system designed with the embedding of hardware and software together for a specific function with a larger area is embedded system design. In embedded system design, a microcontroller plays a vital role. Micro-controller is based on Harvard architecture, it is an important component of an embedded system. External processor, internal memory and i/o components are interfaced with the microcontroller. It occupies less area, less power

consumption. The application of microcontrollers is MP3, washing machines. Critical Embedded Systems (CES) are systems in which failures are potentially catastrophic and, therefore, hard constraints are imposed on them. In the last years the amount of software accommodated within CES has considerably changed. For example, in smart cars the amount of software has grown about 100 times compared to previous years. This change means that software design for these systems is also bounded to hard constraints (e.g., high security and performance). Along the evolution of CES, the approaches for designing them are also changing rapidly, so as to fit the specialized needs of CES. Thus, a broad understanding of such approaches is missing.

Steps in the Embedded System Design Process

The different steps in the embedded system design flow diagram include the following.

Abstraction

In this stage the problem related to the system is abstracted.

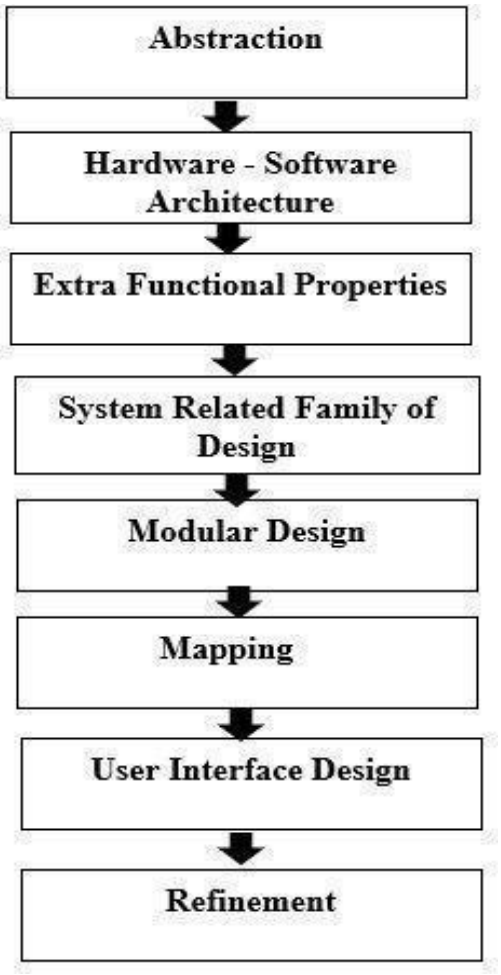

Fig 2.5: Embedded Design-Process-Steps

Hardware – Software Architecture

Proper knowledge of hardware and software to be known before starting any design process.

Extra Functional Properties

Extra functions to be implemented are to be understood completely from the main design.

System Related Family of Design

When designing a system, one should refer to a previous system-related family of design.

Modular Design

Separate module designs must be made so that they can be used later on when required.

Mapping

Based on software mapping is done. For example, data flow and program flow are mapped into one.

User Interface Design

In user interface design it depends on user requirements, environment analysis and function of the system. For example, on a mobile phone if we want to reduce the power consumption of mobile phones, we take care of other parameters, so that power consumption can be reduced.

Refinement

Every component and module must be refined appropriately so that the software team can understand. Architectural description language is used to describe the software design.

- Control Hierarchy Partition of structure
- Data structure and hierarchy
- Software Procedure.

In user interface design it depends on user requirements, environment analysis and function of the system. For example, on a mobile phone if we want to reduce the power consumption of mobile phones, we take care of other parameters, so that power consumption can be reduced. To help countries and health-care facilities to achieve system change and adopt alcohol-based handrubs as the gold standard for hand hygiene in health care, WHO has identified formulations for their local preparation. Logistic, economic, safety, and cultural.

Some examples include:

Design Metrics / Design Parameters of an Embedded System	**Function**
Power Dissipation	Always maintained low
Performance	Should be high

Process Deadlines	The process/task should be completed within a specified time.
Manufacturing Cost	Should be maintained.
Engineering Cost	It is the cost for the edit- test- debug of hardware and software.
Size	Size is defined in terms of memory RAM/ROM/Flash Memory/Physical Memory.
Prototype	It is the total time taken for developing a system and testing it.
Safety	System safety should be taken like phone locking, user safety like engine breaks
	down safety measure must be taken
Maintenance	Proper maintenance of the system must be taken, in order to avoid system failure.

Time to market	It is the time taken for the product/system developed to be launched into the market.

Table:2.1 Embedded System Design Software Development Activities

- **Automobiles**: Modern cars commonly consist of many computers (sometimes as many as 100), or embedded systems, designed to perform different tasks within the vehicle. Some of these systems perform basic utility function and others provide entertainment or user facing functions. Some embedded systems in consumer vehicles include cruise control, backup sensors, suspension control, navigation systems and airbag systems.
- **Mobile phones**: These consist of many embedded systems, including GUI software and hardware, operating systems, cameras, microphones and USB I/O modules.

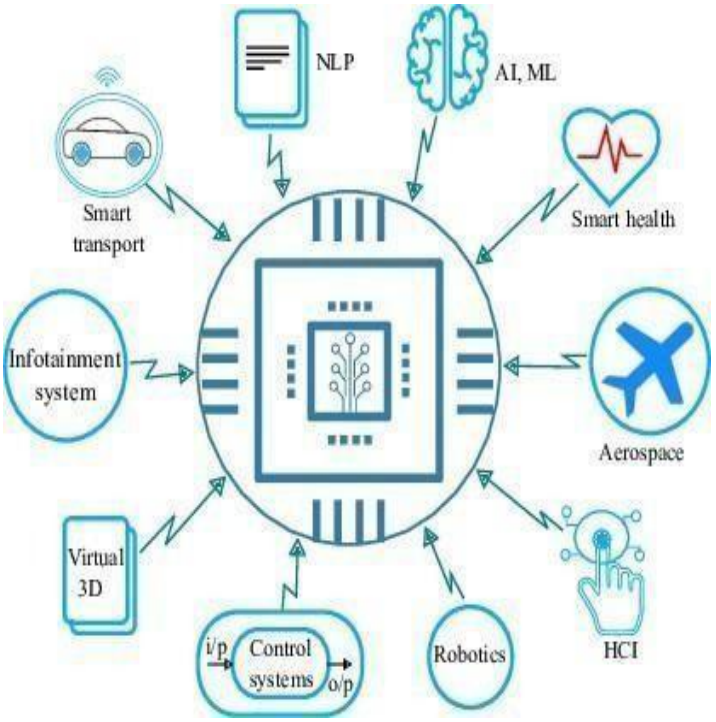

Fig 2.6: Applications Of Embedded Systems

- **Industrial machines**: They can contain embedded systems, like sensors, and can be embedded systems themselves. Industrial machines often have embedded automation systems that perform specific monitoring and control functions.

- **Medical equipment:** These may contain embedded systems like sensors and control mechanisms. Medical equipment, such as industrial machines, also must be very user friendly, so that human health isn't jeopardized by preventable machine mistakes. This means they'll often include a more complex OS and GUI designed for an appropriate UI. The choice of components for the WHO-recommended handrub formulations takes into account cost constraints and microbicidal activity. The following two formulations are recommended for local production with a maximum of 50 litres per lot to ensure safety in production and storage.

2.6 COMBINATION OF LOGIC DEVICES

Logic gates are physical devices that use combinational logic to switch an electrical one (“1”) or zero (“0”) to downstream blocks in digital design. Combinational logic uses those bits to send or receive data within embedded systems. Data bits build into digital words used to communicate with other design blocks within the system. Digital bits and words do this with logic gates in an organized fashion using dedicated address, data, or control signal nodes. Logic gates are the physical devices that enable processing of many 1’s and 0’s

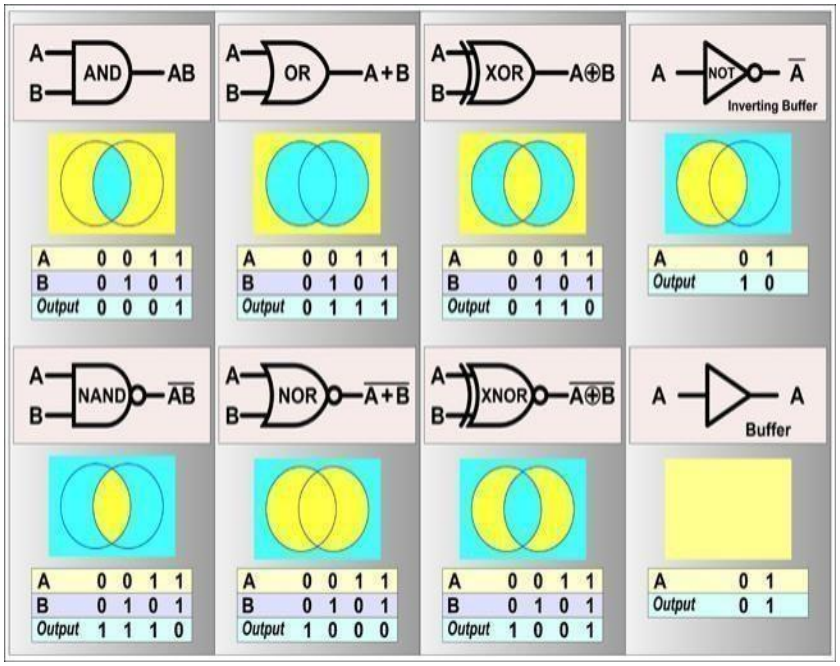

Fig 2.7: Logic Gates

Logic families are collections of integrated circuits containing logic gates that perform functions needed by embedded systems to communicate with one another to drive the design. Logic gates are organized into families relative to the type of material and its operational characteristics. Most logic gates are made from silicon, although some utilize gallium arsenide or other semiconductor materials. The semiconductor material is doped

information throughout an embedded system. Because of its compact size, many millions of transistors combine within very small spaces. This allows millions of gates to operate in compact areas while transmitting and receiving mind-boggling amounts of intelligence through combinational logic. This is all accomplished within a minimal power budget.

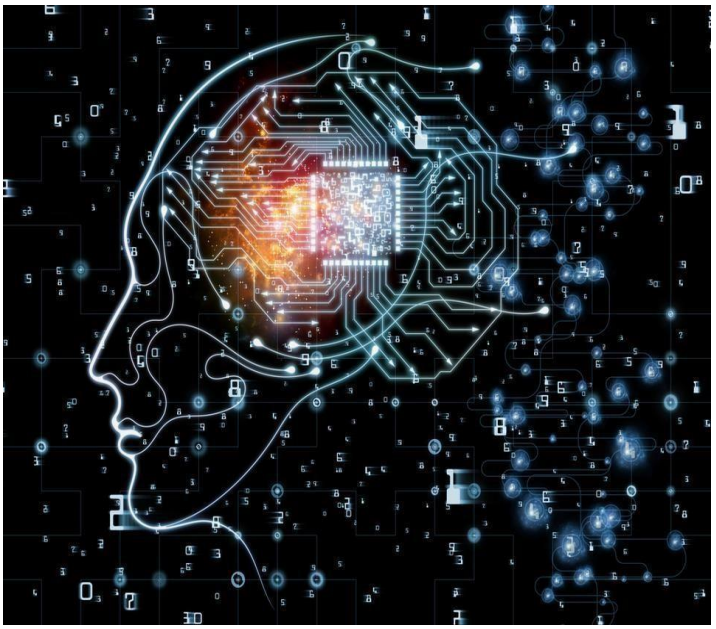

Fig 2.8: Embedded Systems Group

CHAPTER-3
HARDWARE REQUIREMENTS

3.1 HARDWARE

Embedded system hardware can be microprocessor- or microcontroller-based. In either case, an integrated circuit is at the heart of the product that is generally designed to carry out real time computing. Microprocessors are visually indistinguishable from microcontrollers. However, the microprocessor only implements a central processing unit (CPU) and, thus, requires the addition of other components such as memory chips. Conversely, microcontrollers are designed as self-contained systems.

Microcontrollers include not only a CPU, but also memory and peripherals such as flash memory, RAM or serial communication ports. Because microcontrollers tend to implement full (if relatively low computer power) systems, they are frequently used on more complex tasks. For example, microcontrollers are used in the operations of vehicles, robots, medical devices and home appliances. At the higher end of microcontroller capability, the term System on a chip (SOC) is often used, although there's no exact delineation in terms of RAM, clock speed, power consumption and so on.

It is one of the characteristics of embedded and cyber-physical systems that both hardware and software must be taken into account. The reuse of available hard- and software components is at the heart of the platform-based design methodology. Consistent with the need to consider available hardware components and with the design information flow, we are now going to describe some of the essentials of embedded system hardware.

Hardware for embedded systems is much less standardized than hardware for personal computers. Due to the huge variety of embedded system hardware, it is impossible to provide a comprehensive overview of all types of hardware components. Nevertheless, we will try to provide a survey of some of the essential components which can be found in most systems. The choice of components for the WHO-recommended hand rub formulations takes into account cost constraints and microbicidal activity. The following two formulations are recommended for local production with a maximum of 50 litres per lot to ensure safety in production and storage.

Markets and Markets, a business to business (B2B) research firm, predicts that the embedded market will be worth $116.2 billion by 2025. Chip manufacturers for embedded

systems include many well-known technology companies, such as Apple, IBM, Intel and Texas Instruments, as well as numerous other companies less familiar to those outside the field. The expected growth is partially due to the continued investment in artificial intelligence (AI), mobile computing and the need for chips designed for that high-level processing.

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer.[1] These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time.

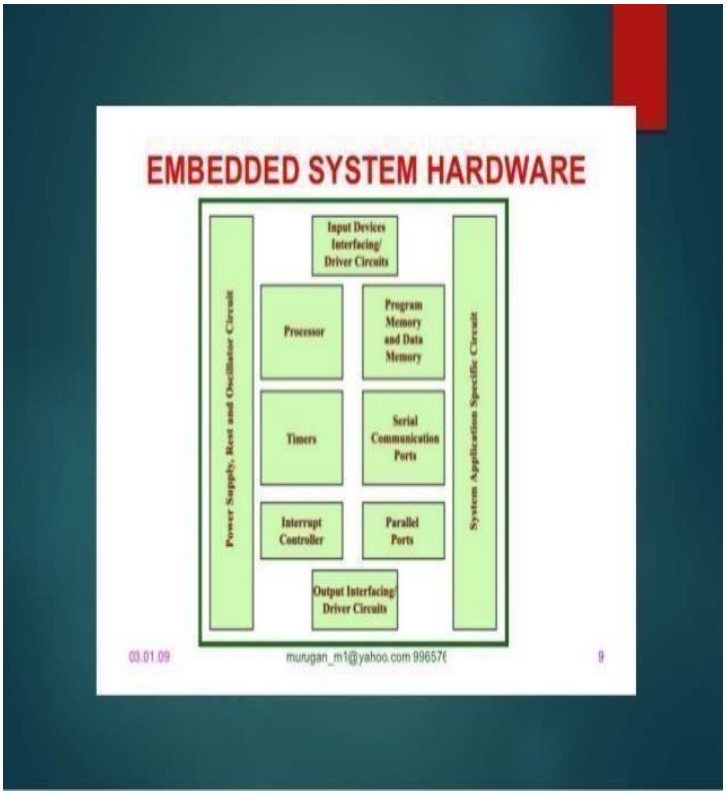

Fig 3.1: Embedded Systems Hardware Block Diagram

Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of system requirements, is a generalisation of this first definition, giving the requirements to be met in the design of a system or subsystem.

Often manufacturers of games will provide the consumer with a set of requirements that are different from those that are needed to run a software. These requirements are usually called the recommended requirements. These requirements are almost always of a significantly higher level than the minimum requirements, and represent the ideal situation in which to run the software. Generally speaking, this is a better guideline than minimum system requirements in order to have a fully usable and enjoyable experience with that software.

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

Architecture

All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory

All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance

of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage

Data storage device requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter

Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals

Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system

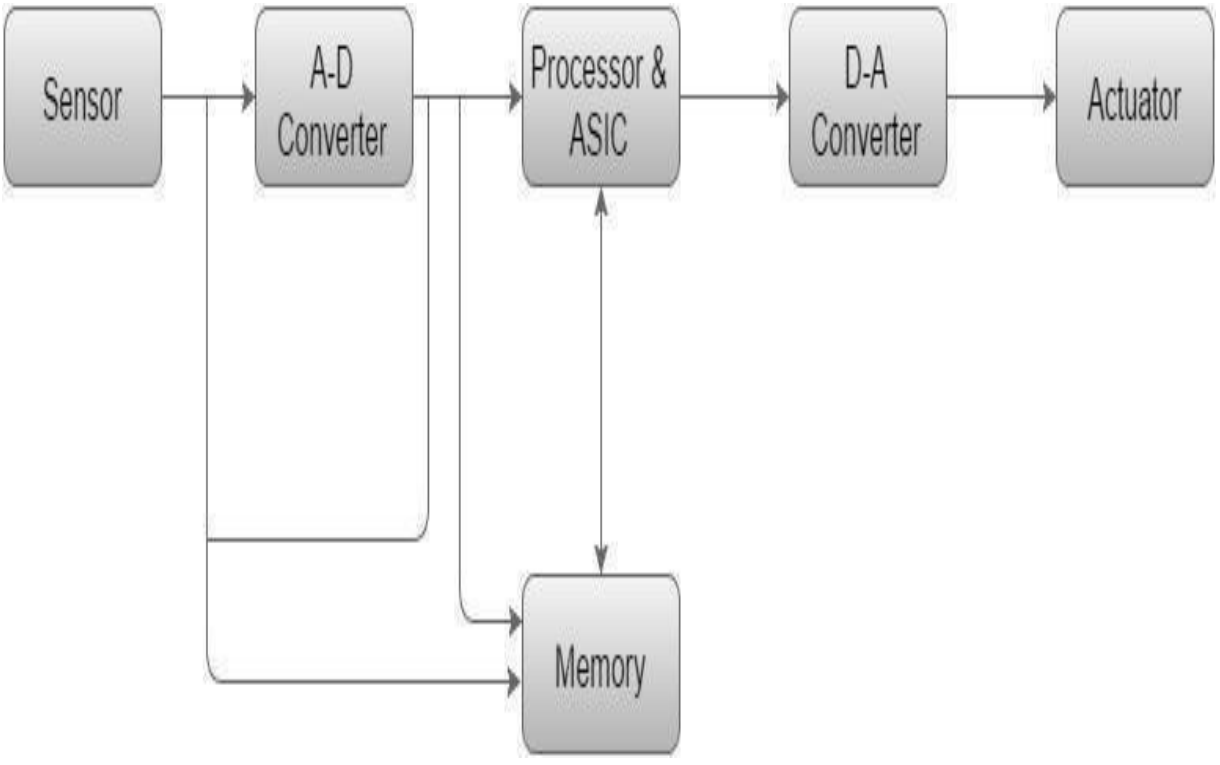

Fig:3.2: Basic Embedded Structure

Sensor − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

- **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor intoa digital signal.
- **Processor & ASICs** − Processors process the data to measure the output and store it to the memory.
- **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data
- **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

In this loop, information about the physical environment is made available through sensors.Typically, sensors generate continuous sequences of analog values. In this book, we will restrict ourselves to information processing where digital computers process discrete sequences of values. Appropriate conversions are performed by two kinds of circuits: sample-and-hold-circuits and analog-to-digital (A/D) converters. After such conversion, information can be processed digitally.Generated results can be displayed and also be used to control the physical environment through actuators. Since most actuators are analog actuators, conversion from digital to analog signals is also needed. This model is obviously appropriate for control applications. For other applications, it can be employed as a first order approximation. In the following, we will describe essential hardware components of cyber-physical systems following the loop structure.

They aren’t a lot different to the requirements for working with non-embedded systems.

A lot depends onthe purpose of the embedded system. You need to understand:

- Requirement set
- Environmental context
- Regulator requirements
- Interface specifications, including choice of hardware, and how to drive that hardware
- Criticality of what you are building, including hazards, and any defined mitigation

- tothose hazards. For instance, is there a safe state to fail to if anything goes wrong.
- Real time constraints, such as cycle times, and time allowable for response. This should also include hysteresis, response of mechanical components, and backlash.
- Real time response from the combination of code, operating system, and hardware youare working with.
- Architecture, including any need for redundancy, diversity, fail safety, voting systems, comparators.
- Platform limitations. Cross compilers, linkers, auto-code generators, etc.
- Choice of operating environment, such as bare metal minimal kernel, real time operating system, or regular operating system.
- Whether you can rely on your tools. In particular, you need to understand how your tools can fail, and how you'd know if something went wrong.
- Communications protocols. Synchronous, and asynchronous communications. Error checking. Error correcting.
- Timing issues, clock rates, reentrancy. Anything that might stop you meeting your realtime response targets, or impact the firing of timers.

Exception handling.

Interrupts. How to handle them. How long they take. What the impact is upon responding to real time response targets.

How to test on the platform you are working with?

As well as that, you also need to know how to code. Choose a language well, and constrain your use of it to assure you meet the limitations you found by considering the above. If required for the domain, use static code checking. If necessary, use formal proof.

CHAPTER-4

SOFTWARE REQUIREMENTS

4.1 ARDUINO SOFTWARE

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they're dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output.

What you will need:

- A computer (Windows, Mac, or Linux)
- An Arduino-compatible microcontroller (anything from this guide should work)
- A USB A-to-B cable, or another appropriate way to connect your Arduinocompatible microcontroller to your computer (check out this USB buying guide if you're not sure which cable to get).

Fig 4.1: Arduino UNO

- An Arduino Uno
- Windows 7, Vista, and XP
- Installing the Drivers for the Arduino Uno (from Arduino.cc)
- Plug in your board and wait for Windows to begin it's driver installation process After a few moments, the process will fail, despite its best efforts

- Click on the Start Menu, and open up the Control Panel
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)".
- While in the Control Panel, navigate to System and Security. Next, click on System Once the System window is up, open the Device Manager.

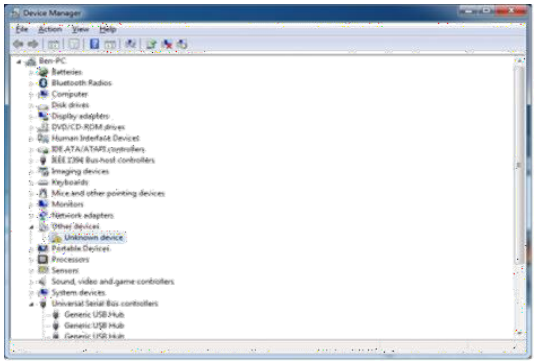

Fig 4.2: Device Manager

- If there is no COM & LPT section, look under 'Other Devices' for 'Unknown Device'.
- Right click on the "Arduino UNO (COMxx)" or "Unknown Device" port and choose the "Update Driver Software" opti Next, choose the "Browse my computer for Driver software" option.

Fig 4.3: Update Driver Software

- Finally, navigate to and select the Uno's driver file, named "ArduinoUNO.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory).
- If you cannot see the .inf file, it is probably just hidden. You can select the 'drivers' folder with the 'search sub-folders' option selected instead.

Windows will finish up the driver installation.

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board!

- Launch the Arduino application
- If you disconnected your board, plug it back in
- Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink
- After a second, you should see some LEDs flashing on your Arduino, followed by the message 'Done Uploading' in the status bar of the Blink sketch.

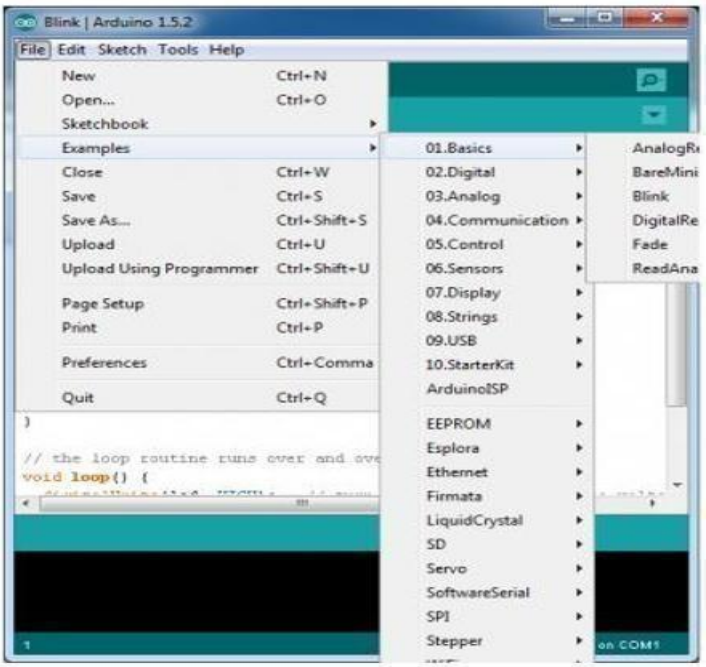

Fig 4.4: Arduino File basics

- If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!
- Select the type of Arduino board you're using: Tools > Board > your board type

- If you’re not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again.
- Select the serial/COM port that your Arduino is attached to: Tools > Port > COMxx

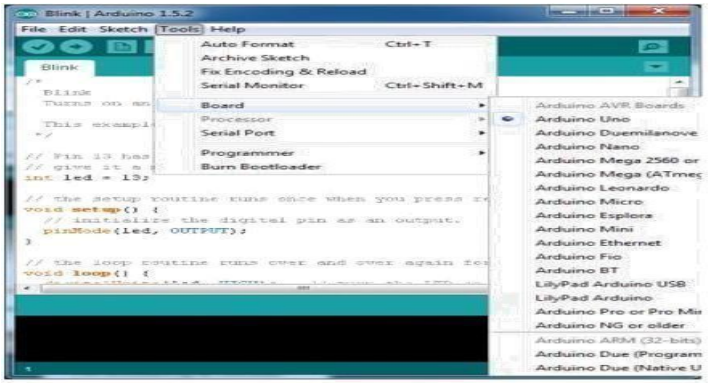

Fig 4.5: Arduino Tools Board

• The one that disappeared is your Arduino.With your Arduino board connected, and the Blink sketch open, press the ‘Upload’ button.

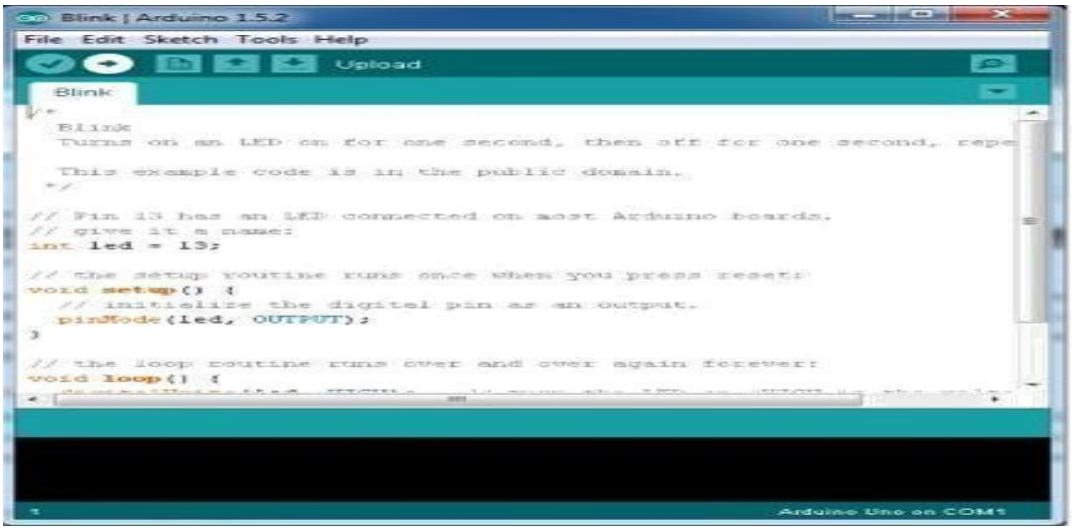

Fig 4.6: Blink Arduino Tools

4.2 RESEARCH

The embedded systems industry was born with the invention of microcontrollers and since then it has evolved into various forms, from primarily being designed for machine control

applications to various other new verticals with the convergence of communications. Today it spans right form small metering devices to the multi-functional smartphones. I will cover the areas that are currently focused for development in embedded systems and state what are the ongoing

Security

Security remains a great challenge even today. Ongoing Research is to sustain physical tampering, mechanisms to trust the software, authenticate the data and securely communicate over internet. With the advent of IoT/IoE, not only the number devices will continue to increase but also will the number of possible attack vectors. Many challenges remain ahead to get the connected devices on a billion scale.

Connectivity

Wi-Fi, BLE, ZigBee, Thread, ANT, etc have been adapted by embedded system experts from considerable time. Head-on competition between these groups is in progress to determine as to who will emerge as the best solution provider to this huge estimated market of IoT/IoE. 4G/5G on low power devices is the ongoing experimentation which will make embedded systems easily and robustly connect to the internet. Communication using GSM/LTE in licensed/unlicensed communication bands with the cloud can change the ball game of IoE all together.

Memory

Various type of volatile/non-volatile memories with variable sizes and speeds are widely available today. Research is more towards organizing them in best possible architecture to reach closer to the design goal of optimal power-performance-cost.

Energy

Power/Battery management has been under focus for some time. Usage of renewable resources to power device's lifetime is currently the challenge that is tried to address; especially for wearables. Optimal power usage to get Longer Battery Life with new Hardware/Software architectural designs will continue for some time.

System

Multicore (Symmetric/Asymmetric) architectures are experimented since long. Addition of GPUs to systems for VR/Gaming/Machine learning is addressed currently. Programmable SOCs (PSOCs**)** - (Configurable Hardware Capability) have been there for a long time now,

but some has not yet gained the momentum. Application specific computer architectures is also in the pipeline in order to optimize the design matrix of power performance-cost.

Performance

Real-time on-board Image/Video/Audio processing, feature enabled cameras, on board machine learning are all currently experimented with varied approaches.

Commercialization of these technologies has already started but there is still some time to get the best out of these technologies and there is lot of scope to make them more user friendly

Other than this, hardening of modular software functionalities (Yes lot of architectures are coming up with hardware performing redundant software functionalities). Ongoing research is to analyse the performance and determine the applications where this strategy can be fruitful.

Networking

Wireless Sensor Networks, Machine to Machine Communication/Interaction, Human Computer Interaction, Security Gateway protocols are still being improved. Light weight algorithms with optimal security will be targeted for embedded systems.

Real Time Operating Systems (RTOS)

Many companies are backing at least one Real Time Open-Source Operating System and there are many out there. Challenge is to cover the wide span of devices, there functionalities and variety of applications.

Application Specific Research Area Medical Entertainment.

CHAPTER -5

WORKING AND ITS COMPONENTS

5.1 BLOCK DIAGRAM

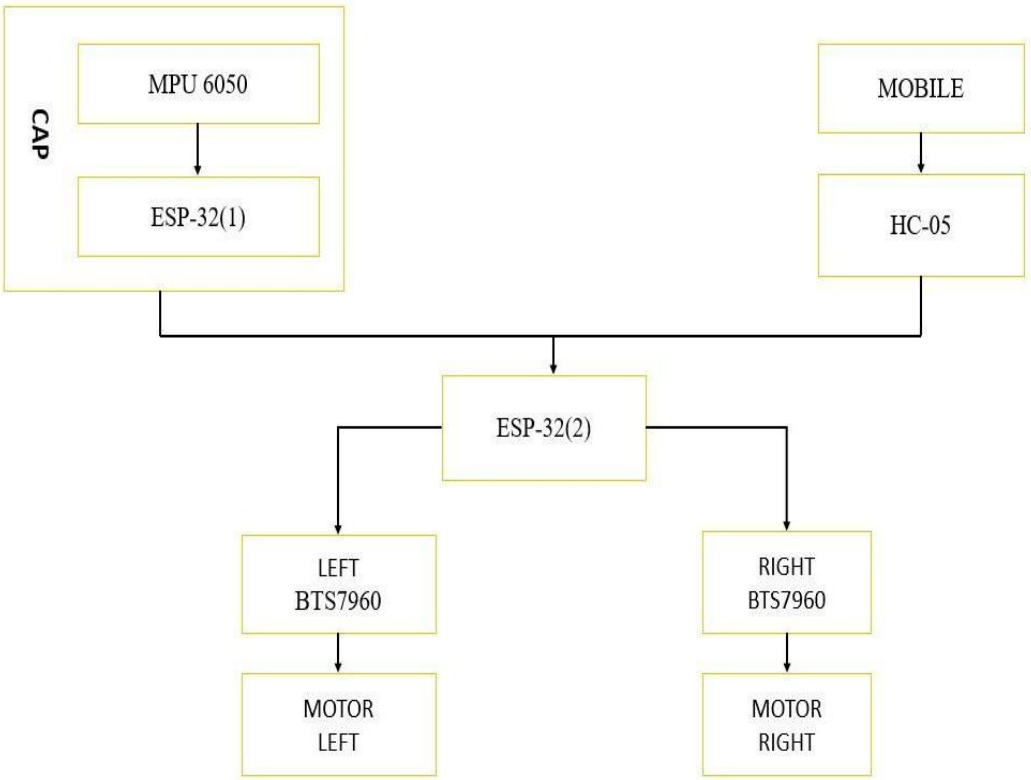

Fig 5.1: Block Diagram OF Head, Bluetooth And Voice Controlled Wheelchair

This block diagram represents a system that controls a motorized device using an ESP32 microcontroller, an MPU6050 sensor, and Bluetooth communication. Here's a breakdown of the components and their interactions:

Components:

Cap Unit:

The system consists of a cap unit, which houses an MPU6050 sensor and an ESP32 (1) microcontroller. The MPU6050 is a motion sensor that detects head movements such as tilting and rotation. This sensor data is processed by ESP32 (1) and transmitted wirelessly using ESP-NOW, a low-latency communication protocol, to the main control unit.

Mobile Communication:

On the other side, there is a mobile communication module that allows users to control the system using a mobile device. The mobile phone connects to an HC-05 Bluetooth module, which is linked to ESP32 (2). This provides an additional means of control, allowing commands to be sent via a smartphone.

Main Control Unit:

The main control unit, ESP32 (2), serves as the central processor. It receives motion data from ESP32 (1) through ESP-NOW and command signals from the mobile via Bluetooth. Based on these inputs, ESP32 (2) processes the data and determines the necessary motor actions. It then sends control signals to two BTS7960 motor drivers, which control the left and right motors.

Motor Control:

The motor drivers receive signals from ESP32 (2) and adjust the movement accordingly. The left motor is controlled by one BTS7960 module, while the right motor is managed by another. This setup allows the system to execute movements such as forward, backward, left, and right turns based on the user's head movements or mobile commands.

Working Principle:

- The user wears a cap equipped with an MPU6050 sensor and ESP32 (1).
- The MPU6050 detects head movements (tilt, rotation, etc.).
- ESP32 (1) sends the processed motion data to ESP32 (2) using ESP-NOW.
- The user can also send commands via a mobile device, which communicates through Bluetooth (HC-05).
- ESP32 (2) processes the received signals and controls the BTS7960 motor drivers.
- The motor drivers adjust the speed and direction of the left and right motors, enabling movement.

5.2 WORKING

The Head, Bluetooth, and Voice-Controlled Wheelchair is a revolutionary assistive technology that empowers individuals with mobility impairments by offering multiple control methods. This innovative system leverages advanced sensor technology and sophisticated control algorithms to provide a more intuitive and efficient means of wheelchair navigation.

The core components of the system include a head movement detection module, a voice recognition system, and a Bluetooth-enabled control interface. These sensors accurately detect head gestures, spoken commands, and Bluetooth-based inputs, generating analog signals. The analog signals are then converted into digital signals by an analog-to-digital converter (ADC) and processed by a microcontroller.

The microcontroller plays a crucial role in interpreting the sensor data and generating appropriate control commands. It employs sophisticated algorithms to analyze the data, filter out noise, and extract relevant information. The control algorithm maps specific head movements, voice instructions, or Bluetooth inputs to corresponding wheelchair actions such as forward, backward, left, right, and stop. By customizing the algorithm to individual user preferences, the system can be tailored to meet diverse needs and abilities.

To ensure seamless communication between the various control modules and the wheelchair, a robust wireless communication system is implemented. The control signals generated by the microcontroller are transmitted wirelessly to the wheelchair's control unit.A wireless communication module, such as Bluetooth or Wi-Fi, is used to establish a reliable connection between the control devices and the wheelchair.

The wheelchair’s control unit receives the wireless signals and decodes the control commands. It then translates the commands into specific motor control signals, which are sent to the wheelchair’s motors. The motors drive the wheels to execute the desired movements, ensuring smooth and responsive mobility.

One of the key challenges in developing a Head, Bluetooth, and Voice-Controlled Wheelchair is ensuring accurate and reliable sensor data. Noise and interference can affect the performance of the sensors, leading to inaccurate control commands. To address this challenge, advanced signal processing techniques are employed to filter out noise and improve the accuracy of sensor readings.

In addition to signal processing techniques, environmental factors and user variability also play a crucial role in ensuring reliable control of the wheelchair. Factors such as ambient noise, varying lighting conditions, and different user head movement patterns can introduce inconsistencies in sensor data. To overcome these challenges, adaptive algorithms can be integrated, allowing the system to learn and adjust to each user's unique movement patterns and voice characteristics. Moreover, incorporating real-time feedback mechanisms, such as haptic feedback or auditory confirmation.

Another important consideration is power consumption. The head movement sensors, voice recognition module, and wireless communication system must be energy-efficient to minimize battery life issues. Low-power components and efficient power management techniques are essential to ensure the system's longevity.

The user interface is also a crucial aspect of the system. A user-friendly interface allows users to customize the system to their preferences and adjust sensitivity settings. Clear visual feedback can help users understand the system's status and make necessary adjustments.

Safety is paramount in the design and implementation of the Head, Bluetooth, and Voice- Controlled Wheelchair. The system should incorporate safety features, such as emergency stop buttons and obstacle detection sensors. Additionally, the wireless communication protocol should be secure to prevent unauthorized access and malicious attacks.

By addressing these challenges and incorporating advanced technologies, the Head, Bluetooth, and Voice-Controlled Wheelchair has the potential to revolutionize the field of assistive technology. It offers a more intuitive and efficient means of wheelchair control, empowering individuals with disabilities to lead more independent and fulfilling lives.

5.2.1 Introduction to arduino:

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they're dry. Arduinos (we use the standard Arduino Uno) are built around an ATmega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output pins, all on a single chip. Unlike, say, a Raspberry Pi, it's designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts.

The Arduino connects to your computer via USB, where you program it in a simple language (C/C++, similar to Java) from inside the free Arduino IDE by uploading your compiled code to the board. Once programmed, the Arduino can run with the USB link back to your computer, or stand-alone without it — no keyboard or screen needed, just power. It offers a more intuitive and efficient means of wheelchair control, empowering individuals with disabilities to lead more independent and fulfilling lives.

Arduino boards are microcontroller-based platforms that enable users to create various electronic projects. The most common board is the Arduino Uno, which features an ATmega328P microcontroller, but there are several other models tailored for specific applications, such as the Arduino Mega, Nano, and Leonardo.

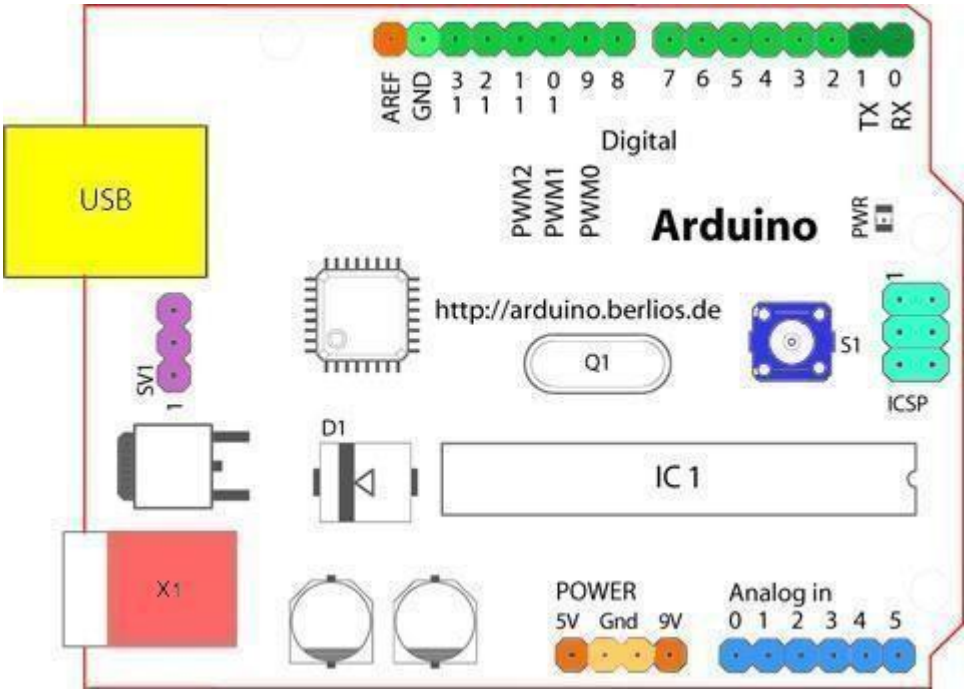

Fig 5.2: Structure of Arduino Board

Looking at the board from the top down, this is an outline of what you will see (parts of the board you might interact with in the course of normal use are highlighted)

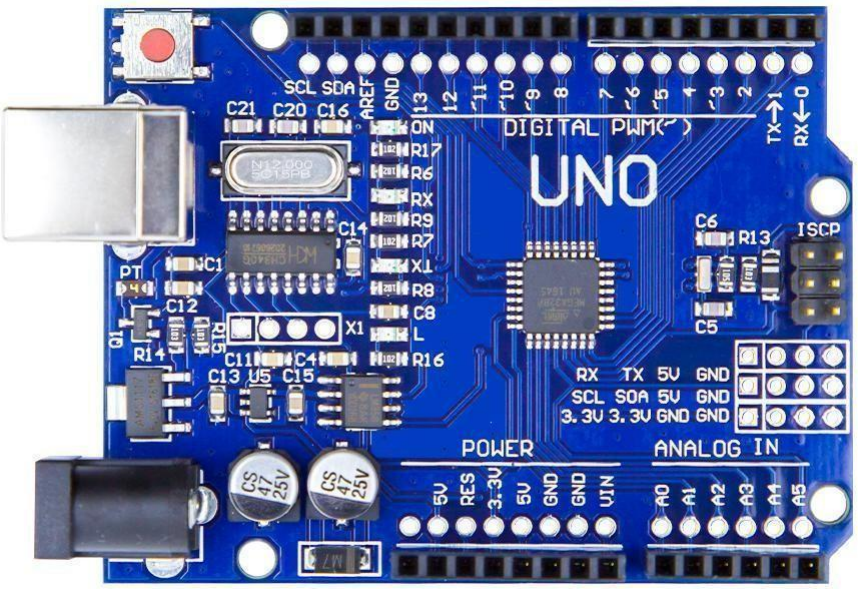

Fig 5.3: Arduino Board

Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (Digital Read and Digital Write) if you are also using serial communication (e.g. Serial.begin).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

DIGITAL PINS

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pin mode(), Digital read(), and Digital write() commands. Each pin has an internal pull-up resistor which can be turned on and off using digital Write() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40mA.

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).

- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

- **PWM: 3, 5, 6, 9, 10, and 11** Provide 8-bit PWM output with the analog write() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT Reset: 7.** (Arduino BT-only) Connected to the reset line of the Bluetooth module.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** On the Decimole and Lilypad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analog Read() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- **I^2C: 4 (SDA) and 5 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website).

Power pins

- **VIN** (sometimes labelled "9V"): The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Also note that the Lily Pad has no VIN pin and accepts only a regulated input.
- **5V:** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3** (Decimole-only) : A 3.3 volt supply generated by the on-board FTDI chip.
- **GND:** Ground pins.

Other pins

- **AREF:** Reference voltage for the analog inputs. Used with analog Reference().
- **Reset:** (decimole-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Atmega328

The ATmega328 is an 8-bit microcontroller based on the AVR architecture. It is popular for its balance of performance, power consumption, and ease of use, making it a favourite among hobbyists and professionals for various electronics projects.

The ATmega328 can be programmed using the Arduino IDE, which simplifies the process with a user-friendly interface and a set of libraries. Users typically write in a simplified version of C/C++. The IDE also provides built-in functions that allow for easy interaction with the microcontroller's hardware features.

Pin diagram

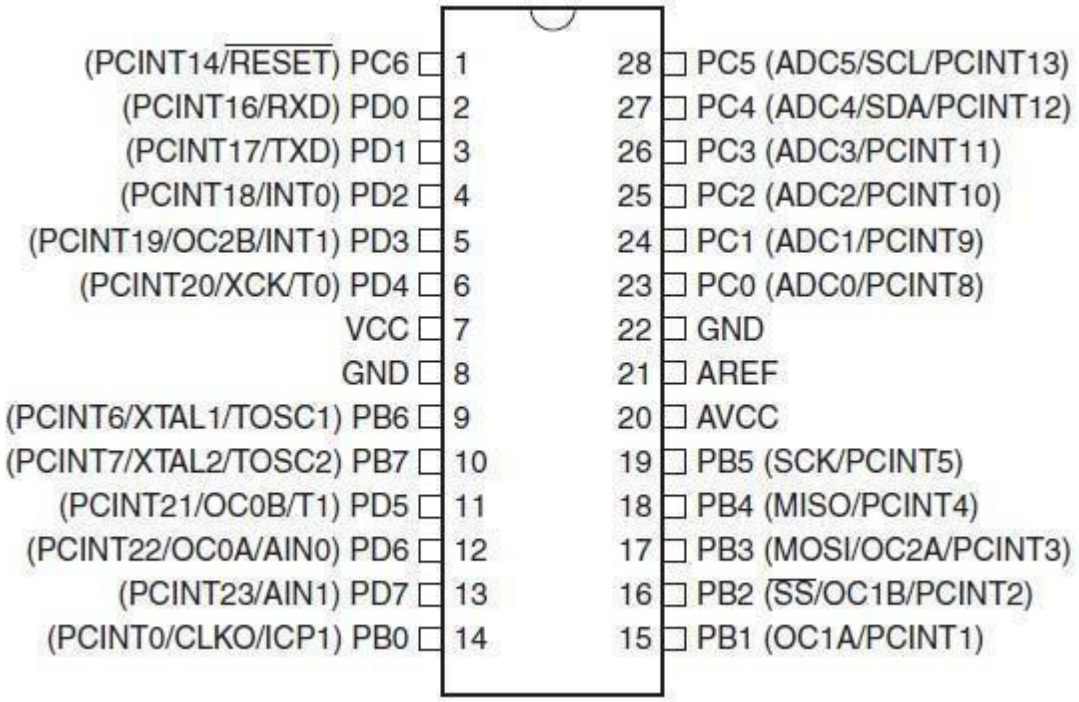

Fig 5.4: Pin Configuration of Atmega328

Pin Description VCC:

Digital supply

voltage. GND:

Ground.

Port A (PA7-PA0):

Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bidirectional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7-PB0):

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega32.

Port C (PC7-PC0):

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. The TD0 pin is tri-stated unless TAP states that shift out data are entered.

Port D (PD7-PD0):

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bidirectional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega32.

Reset (Reset Input):

A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

XTAL1:

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2:

Output from the inverting Oscillator amplifier.

AVCC:

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

AREF:

AREF is the analog reference pin for the A/D Converter.

FEATURES

- 1.8-5.5V operating range
- Up to 20MHz
- Part: ATMEGA328P-AU
- 32kB Flash program memory
- 1kB EEPROM
- 2kB Internal SRAM
- 2 8-bit Timer/Counters
- 16-bit Timer/Counter
- RTC with separate oscillator

- 6 PWM Channels
- 8 Channel 10-bit ADC
- Serial USART
- Master/Slave SPI interface
- 2-wire (I2C) interface
- Watchdog timer
- Analog comparator
- 23 IO lines
- Data retention: 20 years at 85C/ 100 years at 25C
- Digital I/O Pins are 14 (out of which 6 provide PWM output) □ Analog Input Pins are 6.
- DC Current per I/O is 40 mA
- DC Current for 3.3V Pin is 50mA

AVR CPU CORE

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

In order to maximize performance and parallelism, the AVR uses a Harvard architecture with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the nextinstruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory. The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the

Register File, the operation is executed, and the result is stored back in the Register File– in one clock cycle.

The main function of the CPU core is to ensure correct program execution. The AVR CPU is capable to access memories, perform calculations, control peripherals, and handle interrupts.

OVERVIEW

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

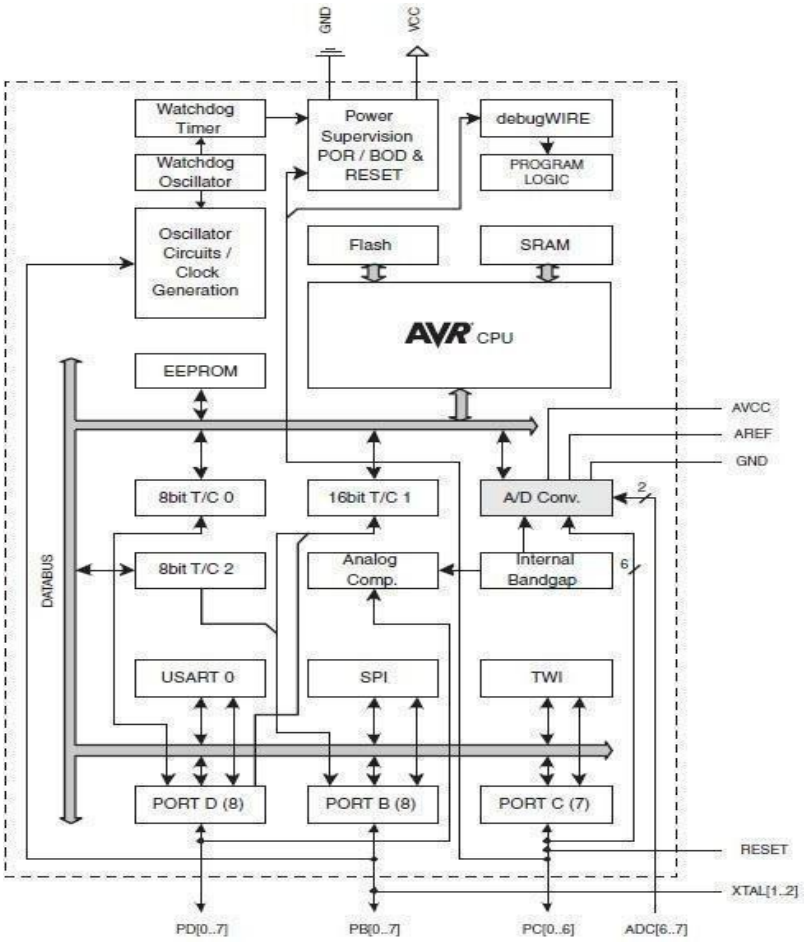

Fig 5.5: Block Diagram

ALU – Arithmetic logic unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code. The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG is defined as:

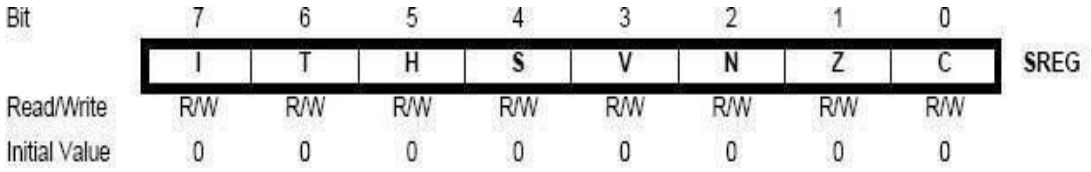

Fig 5.6: AVR status register

Bit 7 – I: **Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

Bit 6 – T: Bit Copy Storage

The Bit Copy instructions BLD (Bit Load) and BST (Bit Store) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

Bit 5 – H: Half Carry Flag

The Half Carry Flag H indicates a Half Carry in some arithmetic operations The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

Bit 4 – S: Sign Bit

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

Bit 3 – V: Two's Complement Overflow Flag

The Two's Complement Overflow Flag V supports two's complement arithmetic.

Bit 2 – N: Negative Flag

The Negative Flag N indicates a negative result in an arithmetic or logic operation.

Bit 1 – Z: Zero Flag

The Zero Flag Z indicates a zero result in an arithmetic or logic operation.

Bit 0 – C: Carry Flag

The Carry Flag C indicates a carry in an arithmetic or logic operation.

Stack pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. Note that the Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM.

The AVR ATmega128A Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the dataspace in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.SPH and SPL - Stack Pointer High and Low Register.

Table 5. 2.1 Stack Pointer instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL , ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

Interrupt response time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles

minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed.

During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

AVR Memories

This section describes the different memories in the ATmega328. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, theATmega328 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory:

The ATmega328 contains 4/8/16/32Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 2/4/8/16K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Loader Section and Application Program Section. The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega328 Program Counter (PC) is 11/12/13/14 bits wide, thus addressing the 2/4/8/16K program memory locations.

SRAM Data Memory:

ATmega328 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 768/1280/1280/2303 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024/2048 locations address the internal data SRAM. The five different addressing modes for the data memory cover: Direct,.

Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In The Register File, Registers R26 to R31 Feature the indirect addressing pointer registers. The direct addressing reaches the entire data space. The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z register

When using register indirect addressing modes with automatic pre-decrement and postincrement, the address registers X, Y, and Z are decremented or incremented. The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 512/1024/1024/2048 bytes of internal data SRAM in the ATmega328 are all accessible through all these addressing modes.

Interrupts

This section describes the specifics of the interrupt handling as performed in the Atmega328. In Atmega328Each Interrupt Vector occupies two instruction words and the Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.Table below shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to versionR2) programmed as a USB- to-serial converter.

Table 5. 2.2 Reset and Interrupt Vectors in ATMEGA 328 and ATMEGA 328P

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0

4	0x0006	PCINTO	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter 2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter 2 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 Compare Match B
14	0x001A	TIMER 1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIME0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART RX Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, TX Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 5. 2.3 Reset and Interrupt Vectors Placement in ATmega328 and ATmega328P

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x002
1	1	0x000	Boot Reset Address + 0x0002
0	0	BootReset Address	0x002
0	1	BootReset Address	Boot Reset Address + 0x002

Arduino with ATmega328

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 1 MHz ceramic resonator, 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.

- Pin out: Added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino. Due that operates with 3.3V. The second one is a not connected pin that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.
- "Uno" means one in Italian and is named to mark the upcoming release of Arduino

1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward.

The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Arduino Characteristics Power:

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1 mmcenter-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN:** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V:** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.
- **IOREF.** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory:

The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of and 1 KB of EEPROM (which can be read and written with the EEPROM library). Serial Communication: The ATmega328 microcontroller features 32 KB of Flash memory, with

0.5 KB reserved for the bootloader, allowing for program storage and execution. It also includes 2 KB of SRAM for temporary data handling and 1 KB of EEPROM, which provides non-volatile storage and can be accessed using the EEPROM library. This memory architecture enables efficient data management for embedded applications.

5.2.2 Block diagram

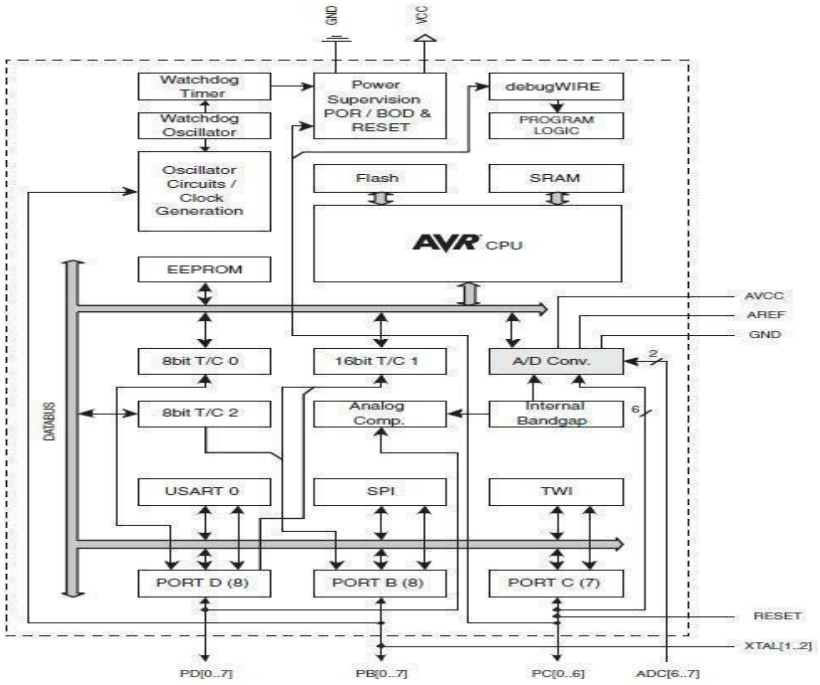

Fig 5.7: Arduino Block Diagram

The block diagram represents the architecture of an AVR microcontroller, showcasing its key functional units and their interconnections. At the core of the microcontroller is the AVR CPU, responsible for executing instructions and managing data flow. The memory architecture includes Flash memory for program storage, SRAM for temporary data, and EEPROM for non-volatile data retention. The system's clock and power management features include an oscillator circuit for clock generation, a power supervision unit with power-on reset (POR) and brown-out detection (BOD), as well as a watchdog timer and watchdog oscillator for system reliability.

The microcontroller integrates various communication interfaces, including a Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) for serial communication, a Serial Peripheral Interface (SPI) for high-speed synchronous data transfer, and a Two-Wire Interface (TWI), also known as I2C, for peripheral communication. It features multiple timers and counters, including 8-bit and 16-bit timer/counters for precise timing applications. The analog subsystem consists of an Analog-to-Digital Converter (ADC) for converting analog signals to digital values, an analog comparator for voltage comparison.

5.2.3 Introduction to bluetooth module

Before the Bluetooth module can be understood completely, it is essential to understand how Wireless communication occurs. Wireless communication occurs by the transference of data over radio waves. By generating a specific radio signal at the source, its effect can be detected at the receiver far from the source, which then identifies it and processes the transmitted information.

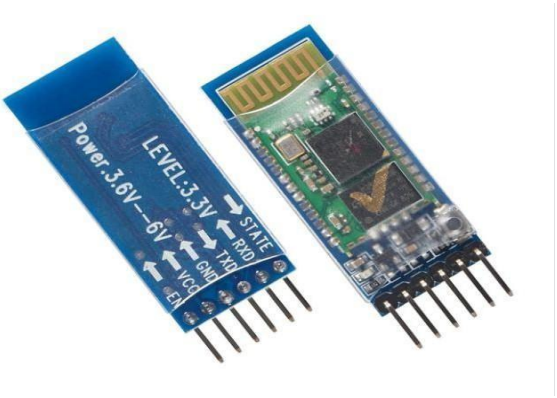

Fig 5.8: Bluetooth Module

In a Bluetooth system, the Bluetooth module, which facilitates wireless communication, generates a signal containing the respective data, which is received by a paired device such as a microcontroller, smartphone, or computer. The received data can then be processed based on the specific application requirements. Thus, a Bluetooth system can be visualized as the sum of the following three components:

• Bluetooth module (transceiver)

• Receiving device (e.g., microcontroller or smartphone)

• Data processing subsystem.

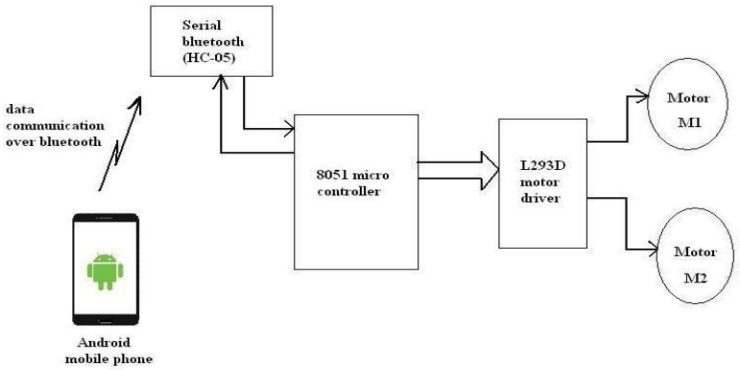

Fig 5.9: Working Of Bluetooth Module

A Bluetooth module consists of a built-in antenna, a transceiver, and an encapsulating material. Bluetooth modules can operate over various classes, with Class 1 offering higher power and range (~100 meters), while Class 2 operates at a shorter range (~10 meters) but with lower power consumption, making it ideal for personal device communication. A Bluetooth-enabled device consists of an antenna, transceiver, and decoder, which continuously scans for paired devices within its range. Once a paired device is detected, the Bluetooth module transmits or receives data for processing. The data processing subsystem provides the means of handling and storing the data.

5.2.4 Bluetooth module and working principle

Bluetooth modules, also referred to as transceivers, facilitate wireless data transmission between devices. They convert radio frequency signals into digital data that can be processed by a microcontroller, smartphone, or another receiving device. Bluetooth modules and paired devices must operate on the same frequency band, typically 2.4 GHz, to establish communication.

A Bluetooth system consists of two main components: a transmitting module and a receiving device. Bluetooth modules function similarly to RFID systems but are designed for two-way wireless communication. The module contains an antenna and an integrated circuit that manages wireless connections and data transmission.

The Bluetooth module uses radio frequency signals to establish connections and transmit data wirelessly. To communicate, it uses a built-in antenna to send and receive signals from other Bluetooth-enabled devices. The data stored and transmitted can be either static or dynamic, depending on the device's role in the communication. Bluetooth memory is primarily read/write, allowing real-time updates and data exchange based on application requirements.

5.2.5 Introduction to mems sensor

A MEMS (Micro-Electro-Mechanical Systems) sensor is an electronic device that detects motion, orientation, and acceleration by utilizing microscopic mechanical and electrical components. MEMS sensors integrate mechanical elements, sensors, actuators, and electronics onto a single chip, allowing for precise movement detection in compact and efficient designs. These sensors consist of microstructures that react to external forces, such as acceleration or tilt, generating electrical signals based on physical motion.

MEMS sensors primarily contain two key components: the sensing element (which detects motion through microscopic mechanical structures) and the signal processing unit (which converts mechanical movement into an electrical output). The sensor calculates acceleration, orientation, or tilt by measuring changes in capacitance, piezoresistive properties, or resonant frequency, depending on the MEMS technology used.

To determine acceleration or orientation, MEMS sensors utilize equations based on Newton's laws of motion. These sensors detect movement in multiple axes, allowing real-time data acquisition for navigation and control applications. For example, if a MEMS accelerometer detects a sudden change in movement within 0.025 seconds at a predefined rate, the system can calculate acceleration based on known parameters, helping adjust device responses accordingly.

MEMS sensors are widely used in motion-sensing applications. They are found in wearable devices, automotive airbag systems, and smartphone gyroscopes. MEMS sensors are also essential for robotic motion tracking, virtual reality systems, and industrial automation. Compared to traditional mechanical sensors, MEMS sensors offer compact size, low power consumption, and high precision, making them ideal for modern mobility solutions.

MEMS sensors are also used in medical applications to monitor patient movements, detect falls, and support prosthetic limb functionality. They play a critical role in improving safety, performance, and user interaction in various industries, making them a fundamental component in smart mobility solutions.

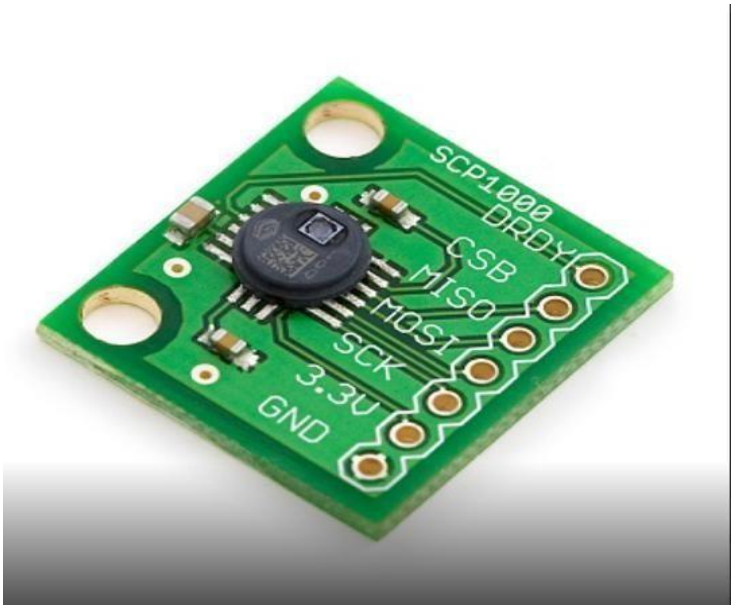

Fig 5.10: MEMS SENSOR

5.2.6 Introduction to voice control module using command working

The **Voice Control Module** is an essential component in modern assistive technologies. It enables users to control electronic systems using voice commands, eliminating the need forphysical interaction. A **voice-controlled system** works by recognizing and processing spoken words, converting them into electrical signals that a microcontroller can interpret and execute as commands.

A **Voice Recognition Module (VRM)** is designed to process voice inputs and match them against predefined commands stored in memory. The module captures the user's speech through a **microphone**, processes it with a built-in **Digital Signal Processor (DSP)**, and then sends the recognized command to the control system.

Voice Command Processing

Voice command processing in a **Voice Control Module** involves multiple stages:

- **Speech Acquisition** – The microphone captures the spoken words.
- **Signal Processing** – The captured signal is filtered and digitized for analysis.
- **Feature Extraction** – Important characteristics of the voice are extracted and compared with pre-recorded commands.
- **Command Execution** – The system identifies the closest match and triggers the corresponding function.

When a user issues a voice command, the module processes the input in real time and activates specific wheelchair functions such as **"Move Forward," "Turn Left," "Stop," or "Reverse."**

5.2.7 Block diagram

The **block diagram** below illustrates the working of the **Voice Control Module** and how it interacts with other components of the wheelchair system.

- **Microphone** – Captures voice input.
- **Pre-Processing Unit** – Converts analog speech to digital data.
- **Voice Recognition Processor** – Matches input with stored commands.
- **Microcontroller Unit (MCU)** – Processes recognized commands and sends control signals.
- **Motor Driver & Actuators** – Executes movements based on processed commands.

Common Voice Commands Used in Wheelchair Control

The Voice Control Module recognizes multiple predefined voice commands:

Command	Function
"Move Forward"	Moves the wheelchair forward.
"Move Backward"	Moves the wheelchair backward.
"Turn Left"	Turns the wheelchair left.
"Turn Right"	Turns the wheelchair right.
"Stop"	Stops all movement.
"Speed Up"	Increases speed.
"Slow Down"	Decreases speed.

TABLE 5.2.4: COMMAND AND FUNCTION OF VOICE CONTROL MODULE

Each command is stored in the Voice Recognition Module (VRM) and matched to predefined motor functions using the microcontroller

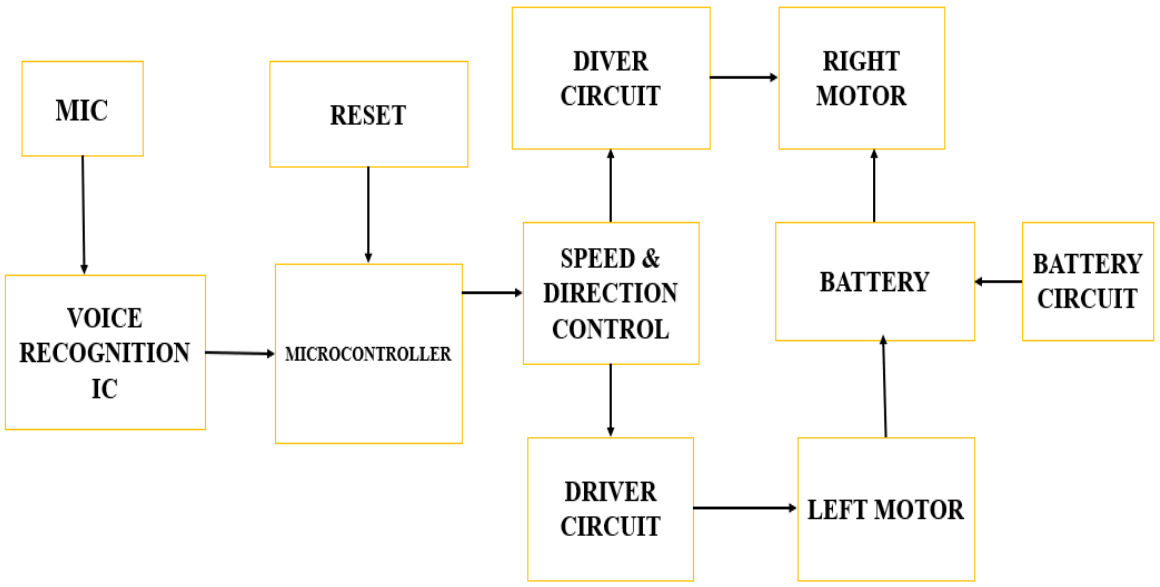

Fig 5.11: Block Diagram Of Voice Control Module Using Command Working

CHAPTER – 6

RESULTS

Implementing a Bluetooth and voice-controlled wheelchair system enhances mobility and independence for users with disabilities. The wheelchair is equipped with a microcontroller, such as the ESP32, which processes commands received via Bluetooth or voice input. A smartphone or dedicated voice module interprets voice commands and transmits control signals to the ESP32, directing the wheelchair's movement. The system integrates motor drivers to regulate speed and direction, ensuring smooth navigation based on user input.

Fig 6.1: Result Of A Head, Bluetooth And Voice Controlled Wheelchair

The Bluetooth module enables wireless communication between the wheelchair and a mobile app, allowing users to control movement through a touchscreen interface or predefined voice commands. The voice recognition system processes spoken instructions, converting them into movement commands for the wheelchair motors. The ESP32 manages

communication, processes data, and executes motor control operations, providing a seamless user experience.

Fig 6.2: Result Of A Head Motion Control With Cap

In the internet-enabled version, the wheelchair can be integrated with cloud platforms for remote monitoring and diagnostics. The ESP32 connects to WiFi, enabling real-time tracking of the wheelchair's status and location. Users or caregivers can access data via a web interface or mobile app, receiving alerts in case of low battery, obstacles detected, or emergency situations. By leveraging platforms like AWS, Firebase, or a custom server, the system ensures accessibility, safety, and reliability, offering enhanced functionality beyond basic mobility control.

ADVANTAGES

The integration of Bluetooth and voice control in a wheelchair offers numerous advantages. First, the voice recognition system enables hands-free operation, providing greater independence for users with mobility impairments. The Bluetooth module ensures seamless wireless communication between the wheelchair and a smartphone, allowing users to control movement effortlessly. The system's ESP32 microcontroller serves as a cost-effective and powerful processing unit, managing voice commands and Bluetooth communication with minimal latency. Additionally, real-time remote monitoring via cloud platforms enhances safety by enabling caregivers to track the wheelchair's status and location. The ability to operate offline using Bluetooth ensures continuous functionality in areas with poor internet connectivity. Moreover, the integration of all these components results in a compact and efficient solution that is easy to deploy and manage, improving accessibility and user

experience.

APPLICATIONS

This system has a wide range of practical applications. It can be used in assistive technology, providing enhanced mobility solutions for individuals with disabilities. In healthcare facilities, it allows caregivers to monitor and control wheelchairs remotely, improving patient care and safety. The system is also ideal for elderly users, offering an easy-to-use interface with voice commands for effortless navigation. Additionally, it can be employed in rehabilitation centers to assist patients recovering from mobility impairments, enabling controlled movement for therapy sessions. The Bluetooth connectivity allows seamless integration with smart home systems, enabling users to control their wheelchair alongside other home automation features. Overall, this system is versatile, serving a range of applications from personal mobility to healthcare and rehabilitation.

CONCLUSION

In conclusion, the integration of Bluetooth and voice control in a wheelchair provides a comprehensive solution for enhancing mobility, accessibility, and user independence. The use of voice recognition ensures hands-free operation, while the Bluetooth module enables seamless wireless control. The ESP32 microcontroller efficiently processes commands, ensuring smooth and responsive movement. The system’s ability to connect to the internet for remote monitoring further enhances safety, making it ideal for users who require assistance. With real-time tracking, secure data storage, and reliable operation, this setup offers a cost-effective and efficient mobility solution. The ability to control the wheelchair remotely and receive alerts makes this system valuable for both personal and healthcare applications.

FUTURE SCOPE

The future scope of this system is vast, with several potential improvements and extensions that can significantly enhance its functionality, user experience, and efficiency. One significant area for enhancement is the integration of AI-based motion detection and obstacle recognition. By leveraging computer vision and machine learning, the system can intelligently detect and respond to obstacles in real-time, making navigation safer and more reliable, especially in dynamic or unfamiliar environments.

Another promising development is cloud integration, which would enable real-time tracking and data analysis. With this, caregivers and healthcare providers could remotely monitor the wheelchair's status, receive alerts, and assess usage patterns. This would be

particularly useful in ensuring user safety and providing timely interventions if abnormal behavior or emergencies are detected. Additionally, the system could be enhanced to support semi-autonomous movement using a combination of AI and advanced sensors, allowing the wheelchair to assist users in navigating complex spaces with minimal manual input.

Energy efficiency is another critical area of improvement. Incorporating battery backup systems or exploring solar charging solutions can extend the operational time of the wheelchair, making it more sustainable and practical for prolonged usage. Solar panels could provide supplementary energy, especially in outdoor settings, reducing reliance on frequent recharging and promoting environmentally friendly usage.

As assistive technology continues to evolve, integrating 5G connectivity could revolutionize the system's capabilities. With ultra-fast data transmission and low latency, 5G would enhance real-time remote monitoring, cloud communication, and responsiveness, enabling seamless interactions between the wheelchair, caregivers, and other smart devices or healthcare systems. This would also open the door to features such as emergency alerts and real-time video streaming for better situational awareness.

Future versions of the system could incorporate multi-modal control mechanisms, including voice commands, gesture recognition, and even brain-computer interface (BCI) technologies. These features would greatly benefit users with severe mobility impairments, providing them with more intuitive and accessible ways to operate the wheelchair. Furthermore, the system could be linked with wearable health devices to continuously monitor vital signs such as heart rate, oxygen levels, and body temperature, transmitting this data to caregivers for proactive healthcare management.

Edge computing can be adopted to handle data processing locally on the device, reducing reliance on the cloud and ensuring faster decision-making. AI algorithms could be personalized to learn user preferences and behavior patterns, optimizing navigation and control for each individual. To ensure data privacy and protection, strong cybersecurity measures like encryption, secure communication protocols, and access controls must be implemented as connectivity expands.

Lastly, a modular and scalable design would allow the system to evolve over time. Users could easily add new features or upgrade existing ones without needing a complete overhaul. This flexibility ensures long-term adaptability and cost-effectiveness as new technologies emerge and user needs change.

REFERENCES

[1]. Ashish Kushwaha, Gaurav Katiyar, & Harshita Katiyar, Hemant Yadav, Saxena 'Bluetooth and Voice-Controlled Wheelchair System'; National Student Conference On "Advances in Electrical & Information Communication Technology" AEICT-2014.

[2]. Hu Jian-ming; Li Jie; Li Guang-Hui, "Assistive Mobility System Based on Bluetooth and Voice Control," Intelligent Networks and Intelligent Systems (ICINIS), 2012 Fifth International Conference on, vol., no., pp.199,201, 1-3 Nov. 2012.

[3]. C. Prabha, R. Sunitha, R. Anitha; Smart Wheelchair Navigation Using Bluetooth and Voice Commands; International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.

[4]. T. Krishna Kishore, T. Sasi Vardhan, N. Lakshmi Narayana "Wheelchair Control Using a Reliable Embedded System with Bluetooth and Voice Recognition" International Journal of Computer Science and Network Security, February 2010.

[5]. Nirav Thakor, Tanmay Vyas, Divyang Shah; Intelligent Wheelchair Control System Based on ARM & Bluetooth; International Journal for Research in Technological Studies ISSN: - Applied (Online) Vol-1, Issue - 1, Dec 2013.

[6]. Raj Kamal, "Embedded System Architecture Programming and Design" (2nd edition), Tata McGraw Hill.

[7]. Sri Krishna Chaitanya Varma, Poornesh, Tarun Varma, Harsha; Voice-Controlled Wheelchair Navigation System Using Bluetooth and Embedded Technology; International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August-2013.

[8]. Ramani1 R, Valarmathy S, Vanitha NS, Selvaraju S, Thiruppathi M, Thangam R. Smart Mobility and Assistance System Based on Bluetooth and Voice Control. Indian Journal Intelligent Systems and Applications. 2013; 5–6.

PENDIX

Appendix-1: Gather Components

Before beginning the project, ensure you have all necessary components:

1. ESP32 Microcontroller
2. Bluetooth Module
3. Voice Recognition Module
4. Motor Driver Module
5. Obstacle Sensors: For detecting objects in the wheelchair's path.
6. Power Supply: (12V battery or rechargeable battery pack).
7. Jumper Wires, Breadboard, Resistors: For prototyping.
8. Smartphone/Device: For controlling the wheelchair via Bluetooth.

Appendix-2: Circuit Design & Wiring

1.1: ESP32 Pin Connections

1. Bluetooth Module: Connect to the ESP32 using UART (TX/RX).
2. Voice Module: Connect to the ESP32 for serial communication.
3. Motor Driver Module: Connect via PWM/GPIO for motor control.
4. Obstacle Sensors: Connect digital pin(s) of the ESP32 to the sensor module.
5. Power Supply: Use a 12V battery, ensuring proper voltage regulation for the ESP32 & modules.

1.2: Power Distribution

1. Ensure the system receives stable power from the wheelchair’s battery.
2. Consider using a backup battery or capacitor to keep the system operational when power is low.

Appendix-3: Setting Up the ESP32Environment.

Appendix-4: Writing the Firmware for ESP32

Appendix-5: Testing the System

Appendix-6: Install the System in Wheelchair

Appendix-7: Final Testing and Optimization in Mobile app

Appendix-8: Monitor and Maintain