A

Mini-Project Report

On

"Smart IoT Based Plant Watering System with Mobile Application Integration"

**Submitted in Partial Fulfillment of
the Academic Requirement for the
Award of Degree of**

BACHELOR OF TECHNOLOGY
in
Electronics and Communication Engineering
Submitted by

A. SAI SUDHEER	**(21R01A04B6)**
Y. SRI MANIKANTA	**(21R01A04C1)**
Y. SURYA PRAKASH	**(21R01A04C9)**

Under the esteemed guidance of

Dr. K. Praveen kumar
Associate Professor
Department of Electronics and Communication Engineering

CMR INSTITUTE OF TECHNOLOGY
(UGCAUTONOMOUS)
Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC
Kandlakoya (V), Medchal Dist-501401
www.cmrithyderabad.edu.in
2024-25

CMR INSTITUTE OF TECHNOLOGY

(UGCAUTONOMOUS)

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUH, Hyderabad

Accredited by NBA and NAAC

Kandlakoya (V), Medchal Dist-501401

www.cmrithyderabad.edu.in

CERTIFICATE

This is to certify that an Industry Oriented Mini-Project Report entitled with ***"Smart IoT Based Plant Watering System with Mobile Application Integration"*** is being submitted by

A. SAI SUDHEER	**(21R01A04B6)**
Y. SRI MANIKANTA	**(21R01A04C1)**
Y. SURYA PRAKASH	**(21R01A04C9)**

to JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B.Tech in Electronics &Communication Engineering and is a record of a bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University forward of another degree or diploma.

Project Supervisor | **HOD, ECE**

)r. K. Praveen Kumar) | **(Dr. K. Niranjan Reddy)**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, Director, **Dr. G. Madhusudhana Rao**, Principal and **Dr. K. Niranjan Reddy**, Head of Department, Dept of Electronics & Communication Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We would like to thank our project supervisor**, Dr. K. Praveen Kumar**, Associate Professor, Department of ECE for the guidance and support, especially for the valuable ideas and knowledge shared to us throughout the Project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of their references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or in directly for every step towards success.

A. SAI SUDHEER	**(21R01A04B6)**
Y. SRI MANIKANTA	**(21R01A04C1)**
Y. SURYA PRAKASH	**(21R01A04C9)**

Declaration

We **A. SAI SUDHEER (21R01A04B6), Y. SRI MANIKANTA (21R01A04C1) and Y. SURYA PRAKASH (21R01A04C9)** of the Real Time/Societal Research Project Report entitled as ***“Smart IoT Based Plant Watering System with Mobile Application Integration”*** hereby declared that the matter embodied in this project is the genuine work done by us only and has not been submitted either to the university or to any university/institute for the fulfillment of the requirement of any course of study.

A. SAI SUDHEER	**(21R01A04B6)**
Y. SRI MANIKANTA	**(21R01A04C1)**
Y. SURYA PRAKASH	**(21R01A04C9)**

INDEX

ACKNOWLEDGEMENT **i**

DECLARATION **ii**

INDEX **iii**

ABSTRACT **v**

LIST OF FIGURES **vi**

LIST OF ABBREVATIONS **vii**

LIST OF TABLES **viii**

CHAPTER-1 INTRODUCTION **1**

1.1 INTRODUCTION 1

1.2 LITERATURE SURVEY 2

1.3 BLOCK DIAGRAM 5

CHAPTER-2 INTRODUCTION TO EMBEDDED SYSTEMS **6**

2.1 CLASSIFICATION 7

2.2 OTHER COMMON PARTS FOUND ON MANY EMBEDDED SYSTEMS 8

2.3 DESIGN PROCESS 8

2.3.1 SPECIFICATION 9

2.3.2 SYSTEM-SYNTHESIS 9

2.3.3 IMPLEMENTATION-SYNTHESIS 9

2.3.4 PROTOTYPING 9

2.3.5 APPLICATIONS 11

CHAPTER-3 ARDUINO **12**

3.1 INTRODUCTION OF ARDUINO 12

3.2 DIGITAL PINS 15

3.3 ANALOG PINS 15

3.4 POWER PINS 16

3.5 OTHER PINS 16

3.6 ATMEGA328 17

3.7 FEATURES 18

3.8 AVR CPU CORE 19

3.9 OVERVIEW 20

3.10 ALU – ARITHMETIC LOGIC UNIT 22

3.11 STATUS REGISTER 22

3.12 STACK POINTER 24

3.13 INTERRUPT RESPONSE TIME 25

3.14 AVR MEMORIES 25

3.15 INTERRUPT 27

3.16 ARDUINO CHARACTERISTICS 29

CHAPTER-4 HARDWARE COMPONENETS 32
4.1 LCD (LIQUID CRISTAL DISPLAY) 32
4.1.1 INTRODUCTION 32
4.1.2 FEATURES 33
4.1.3 SHAPES AND SIZES 34
4.1.4 ELECTRICAL BLOCK DIAGRAM 35
4.1.5 POWER SUPPLY FOR LCD DRIVING 35
4.1.6 PIN DISCRIPTION 36
4.1.7 CONTROL LINES 36
4.1.8 LOGIC STATUS ON CONTROL LINES 37
4.1.9 WRITING DATA TO THE LCD 37
4.1.10 READ DATA FROM DATA LINES ON LCD 38
4.1.11 ENTERING TEXT 38
4.1.12 INTIALIZATION BY INSTRUCTIONS 39
4.2 REGULATED POWER SUPPLY 40
4.2.1 INTRODUCTION 40
4.2.2 BLOCK DIAGRAM 40
4.2.3 STEP-1 TRANSFORMATION 41
4.2.4 TRANSFORMERS 42
4.2.5 STEP UP TRANSFORMERS 43
4.2.6 STEP DOWN TRANSFORMER 43
4.2.7 BATTER POWER SUPPLY 44
4.2.8 STEP-2 RECTIFIERS 45
4.2.9 RECTIFIERS 45
4.2.10 BRIDGE FULL WAVE RECTIFIER 46
4.2.11 STEP-3 FILTERATION 47
4.2.12 FILTERS 47
4.3 CAPACITORS 47
4.3.1 INTRODUCTION OF CAPACITOR 47
4.3.2 OPERATION OF CAPACITOR 48
4.4 STEP-4 REGULATION 49
4.4.1 VOLTAGE REGULATOR 49
4.5 RESISTORS 50
4.5.1 THEORY OF OPERATION 51
4.6 LED 52
4.6.1 WORKING 53
4.7 SOIL MOISTURE SENSOR 54

CHAPTER-5 SOFTWATRE COMPONENTS 55
5.1 ARDUINO SOFTWARE 55
5.1.2 WHAT YOU WILL NEED 55
5.1.3 LAUNCH AND BREAK 57

CHAPTER-6 RESULTS 60

CHAPTER-7 CONCLUSION 61

CHAPTER-8 REFERENCES 62

Abstract

The Smart IoT-Based Plant Watering System with Mobile Application Integration aims to revolutionize plant care through the seamless integration of advanced IoT technologies and user-friendly mobile interfaces. The system is equipped with soil moisture sensors, temperature, and humidity monitors that continuously track environmental conditions to determine the optimal watering requirements for plants. An intelligent algorithm processes the data to automate the watering process, ensuring precise and efficient water usage while maintaining optimal plant health. The mobile application acts as a central hub, providing users with real-time updates on environmental metrics and system status. It enables users to manually override the automated system when necessary, customize watering schedules, and receive notifications for system alerts or plant care tips. This feature-rich solution addresses the challenges of modern lifestyles by reducing the time and effort required for plant maintenance while promoting sustainable water management practices. The system is ideal for residential, commercial, and agricultural applications, offering scalability and adaptability to diverse plant care needs. By combining automation, IoT connectivity, and user-centric mobile control, the Smart IoT-Based Plant Watering System redefines traditional plant care, making it more convenient, efficient, and accessible to users across various skill levels and time constraints.

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NUMBER
1.1	BLOCK DIAGRAM OF PROPOSED SYSTEM	6
2.1	EMEBEDDED DEVELOPMENT LIFE CYCLE	11
3.1	STRUCTURE OF ARDUINO BOARD	13
3.2	ARDUINO BORAD	14
3.3	PIN CONFIGURATION OF ATMEAGA328	19
3.4	BLOCK DIAGRAM AVR CPU CORE	20
3.5	CORE ARCHITECTURE	21
3.6	AVR STATUS REGISTER	23
3.7	SPH AND SPL - STACK POINTER HIGH AND LOW REGISTER	25
3.8	DATA MEMORY MAP	26
4.1	ADDRESS LOCATIONS FOR A 1X16 LINE LCD	32
4.2	SHAPES OF LCD	34
4.3	ELECTRICAL BLOCK DIAGRAM OF LCD	36
4.4	POWER SUPPLY OF LCD DRIVING	37
4.5	PIN DESCRIPTION OF LCD	38
4.6	INITIALIZATION BY INSTRUCTIONS	39
4.7	BLOCK DIAGRAM OF REGULATED POWER SUPPLY	40
4.8	CIRCUIT DIAGRAM OF REGULATED POWER SUPPLY	41
4.9	STEP DOWN TRANSFORMER	43
4.10	HI-WATT 9V BATTERY	44
4.11	PENCIL BATTERY OF 1.5V	44
4.12	BRIDGE RECTIFIER: A FULL-WAVE RECTIFIER USING 4 DIODES	45
4.13	CONSTRUCTION OF A CAPACITOR	46
4.14	ELECTROLYTIC CAPACITOR	46
4.15	CURRENT FLOW IS ALTERNATING BETWEEN ZERO	47
4.16	THE INITIAL SHORT BURST	47
4.17	CAPACITOR IS USED TO DECOUPLE A CIRCUIT	48
4.18	VOLTAGE REGULATOR	49
4.19	RESISTOR	51
4.20	COLOURS OF RESISTORS	51
4.21	LED	52
4.22	ELECTRICAL SYMBOL & POLARITIES OF LED	52
4.23	SOIL MOISTURE SENSOR	54
5.1	ARDUINO AND ARDUINO CABLE	55
5.2	FOLDER OF THE ARDUINO SOFTWARE DOWNLOAD	56
5.3	BLINK EXAMPLE SKETCH	57
5.4	TOOLS SECTION IN ARDUINO SOFTWARE	57
5.5	THE SERIAL/COM PORT	58
5.6	ARDUINO SOFTWARE CODE	58
6.1	PROTOTYPE OF PROPOSED SYSTEM	60

ABBREVATIONS

ABBREVIATION	FULL FORM
IOT	INTERNET OF THINGS
PA	PORT A
PB	PORT B
PC	PORT C
PD	PORT D
RS	REGISTER SELECT
RW	READ/WRITE
DB	DATA BUS
LCD	LIQUID CRYSTAL DISPLAY
CGRAM	CHARACTER GENERATOR RAM
AVCC	ANALOG VOLTAGE COMMON COLLECTOR
AREF	ANALOG REFERENCE
EEPROM	ELECTRICALLY ERASABLE PROGRAMMABLE READ-ONLY MEMORY
ADC	ANALOG-TO-DIGITAL CONVERTER
PWM	PULSE WIDTH MODULATION
SRAM	STATIC RANDOM ACCESS MEMORY
ROM	READ-ONLY MEMORY
SPI	SERIAL PERIPHERAL INTERFACE
USART	UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER/TRANSMITTER
I2C	INTER-INTEGRATED CIRCUIT
TWI	TWO-WIRE INTERFACE
ICSP	IN-CIRCUIT SERIAL PROGRAMMING
CPU	CENTRAL PROCESSING UNIT
ALU	ARITHMETIC LOGIC UNIT
SP	STACK POINTER
LED	LIGHT EMITTING DIODE
PLD	PROGRAMMABLE LOGIC DEVICE
ASIC	APPLICATION-SPECIFIC INTEGRATED CIRCUIT
FPGA	FIELD PROGRAMMABLE GATE ARRAY
UART	UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER
VHDL	VHSIC HARDWARE DESCRIPTION LANGUAGE
MCU	MICROCONTROLLER UNIT
USB	UNIVERSAL SERIAL BUS
DC	DIRECT CURRENT
AC	ALTERNATING CURRENT

LIST OF TABLES

TABLE NUMBER	TABLE NAME	PAGE NUMBER
3.1	Reset and Interrupt Vectors in ATmega328	24
3.2	Reset and Interrupt Vectors Placement in ATmega328 and ATmega328P	28
3.3	Stack Pointer Instructions	28

CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

The need for effective and efficient plant care is growing as urbanization increases and modern lifestyles leave people with limited time to manage routine tasks. Plants require consistent attention, particularly regarding their watering needs, as inadequate or excessive watering can adversely affect their growth and health. To address this challenge, the **Smart IoT-Based Plant Watering System with Mobile Application Integration** offers an innovative solution by combining IoT technology, automation, and a user-friendly mobile interface.

This system integrates soil moisture sensors, temperature, and humidity monitors to continuously analyse environmental conditions and determine the precise watering needs of plants [1]. By leveraging IoT connectivity, the system ensures automated and accurate irrigation, promoting healthy plant growth while conserving water [2].

A dedicated mobile application provides users with real-time data and allows them to monitor and control the system remotely. The app includes features such as customizable watering schedules, manual overrides, and alert notifications, ensuring convenience and flexibility in plant care [3]. This user-centric design makes the system highly suitable for individuals with busy schedules, frequent travellers, or those with limited gardening expertise [4].

The Smart IoT-Based Plant Watering System contributes to sustainable living by optimizing water usage and reducing resource wastage [5]. Its scalability and adaptability make it applicable to a variety of environments, including homes, offices, and agricultural settings [6]. By integrating cutting-edge technology with plant care, this system offers a comprehensive solution to the challenges of modern plant maintenance.

1.2 Literature survey

Singh, R. (2021). IoT-Based Smart Plant Watering System for Indian Climate Conditions. Journal of Smart Systems, 5(3), 45-50.

The paper "IoT-Based Smart Plant Watering System for Indian Climate Conditions" by Singh (2021) explores a smart irrigation system designed to optimize water usage and provide appropriate hydration to plants under Indian climate conditions. The study focuses on integrating Internet of Things (IoT) technology with mobile application capabilities to monitor soil moisture levels and automate watering processes based on real-time data. The system utilizes a capacitance-based soil moisture sensor that is connected to an Arduino microcontroller to measure soil moisture levels accurately. This sensor setup allows for precise monitoring of soil hydration, which is crucial in the varied climate conditions of India—characterized by both periods of intense monsoon rains and dry spells. The IoT aspect involves transmitting data from the sensors to a mobile application using Wi-Fi connectivity, allowing users to control and monitor the watering system remotely. The research highlights the system's ability to save water by reducing unnecessary watering cycles, thereby making it more sustainable and efficient. The system employs weather data to adjust watering schedules, taking into account factors like temperature, humidity, and soil type. This integration helps in better managing the water resources by providing just enough water when needed, minimizing waste. The implementation of the smart system was tested in different locations across India, showing significant results in reducing water wastage and improving plant health. The study concludes that IoT-based smart watering systems are a promising solution for Indian farmers to address the challenges posed by erratic climate patterns. By leveraging mobile applications, users can receive alerts when watering is required, check soil moisture levels remotely, and adjust settings as needed, providing a tailored approach to plant care.

Kumar, A. (2023). Mobile Application Integration for IoT-Based Plant Watering System. IEEE Conference on Automation and Robotics, 12, 67-72.

The paper "Mobile Application Integration for IoT-Based Plant Watering System" by Kumar (2023) explores the integration of mobile applications with IoT technologies to create an intelligent plant watering system. The study primarily focuses on using Internet of Things (IoT) devices and mobile apps to automate and monitor the watering process for plants. This system is designed to be efficient in Indian climate conditions, where water scarcity and harsh environmental conditions can impact the growth and health of plants. Kumar’s research highlights the use of sensors such as soil moisture sensors, temperature sensors, and humidity sensors to monitor the soil conditions. The system collects data from these sensors in real time to determine the moisture level in the soil. Based on this data, the system can automatically control water distribution through connected irrigation systems, ensuring that plants receive the optimal amount of water at the right times. The mobile application acts as an interface for users to set preferences, monitor plant health, and receive alerts about watering schedules and soil conditions. The study emphasizes the importance of mobile application integration in enhancing the user experience by providing a convenient platform for monitoring and controlling the watering system. The integration of mobile applications also allows for remote control, enabling users to manage their watering system from anywhere, which is particularly useful for individuals with busy schedules or those who travel frequently. Moreover, the paper discusses the use of artificial intelligence (AI) techniques to predict watering needs based on environmental data and historical plant performance. This predictive approach helps in optimizing water usage, reducing waste, and preventing over or under-watering. The integration of AI helps the system adapt to different plant types and climatic conditions, making it versatile and suitable for diverse agricultural needs. The practical implications of this research are significant, especially for areas prone to water shortages or irregular rainfall patterns. By leveraging IoT and mobile technology, the system can potentially transform traditional gardening and agriculture, making it more sustainable and efficient.

Ramesh, S. (2022). Artificial Intelligence in IoT Plant Watering Systems. *International Journal of Smart Agriculture*, 7(1), 33-40.

The paper "Artificial Intelligence in IoT Plant Watering Systems" by Ramesh (2022) examines the integration of artificial intelligence (AI) with IoT-based plant watering systems, focusing on enhancing efficiency and sustainability in agriculture and urban gardening. The study explores how AI can be utilized to improve the accuracy and effectiveness of plant watering, particularly in varying environmental conditions such as the Indian climate. Ramesh discusses the use of AI algorithms to predict watering needs based on environmental data collected by IoT sensors. The system gathers real-time information on soil moisture, temperature, humidity, and light levels, which are crucial for determining the water requirements of plants. By processing this data through machine learning models, the system can make informed decisions about when and how much water to supply to different plants. This predictive approach helps prevent over-watering, which can lead to soil erosion, nutrient leaching, and water wastage, especially in water-scarce regions like India. The integration of AI in IoT plant watering systems enables the system to adapt dynamically to changing environmental conditions. For example, if there is a sudden drop in soil moisture due to a high-temperature spell, the AI system can adjust the watering schedule accordingly. This adaptive mechanism is crucial for maintaining optimal soil moisture levels, even as weather patterns fluctuate. Ramesh highlights how this capability not only saves water but also reduces the labor required for manual monitoring and intervention, which is often a barrier for urban and rural gardeners alike. Furthermore, Ramesh discusses the role of AI in plant health monitoring. The system can analyze various parameters such as leaf color, growth rate, and pest infestations to provide alerts to users about potential issues that may require attention, like diseases or nutrient deficiencies. By integrating this feature with the watering system, users can receive recommendations for corrective actions, such as adjusting water volume or adding fertilizers, thus maintaining the overall health of their plants. The paper also examines the implementation challenges and solutions in deploying such systems. Ramesh points out the importance of ensuring data security and privacy, given the sensitive nature of agricultural data. Moreover, the system's scalability across different regions and its adaptability to various plant species and soil types are discussed, underlining the need for customization to local conditions. Overall, Ramesh's paper underscores the potential of AI in revolutionizing IoT-based plant watering systems by making them smarter, more efficient, and environmentally friendly. The integration of AI not only enhances plant care but also promotes sustainable water use, which is critical for addressing global challenges like water scarcity and climate change.

1.3 Block diagram

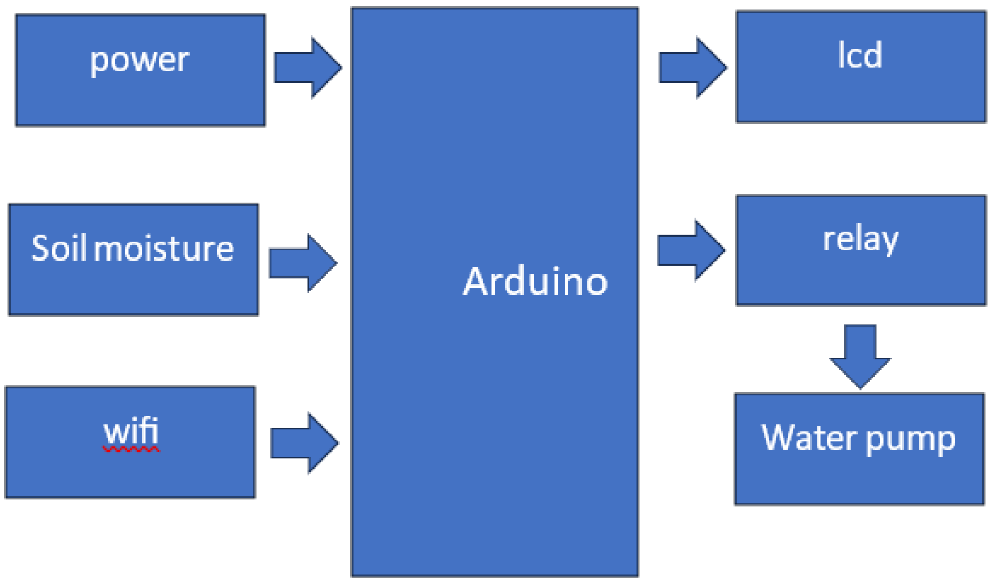


Fig 1.1: Block diagram of Proposed system

CHAPTER 2
INTRODUCTION TO EMBEDDED SYSTEMS

Many embedded systems have substantially different design constraints than desktop computing applications. No single characterization applies to the diverse spectrum of embedded systems. However, some combination of cost pressure, long life-cycle, real-time requirements, reliability requirements, and design culture dysfunction can make it difficult to be successful applying traditional computer design methodologies and tools to embedded applications. Embedded systems in many cases must be optimized for life-cycle and business-driven factors rather than for maximum computing throughput. There is currently little *tool* support for expanding embedded computer design to the scope of holistic embedded system design. However, knowing the strengths and weaknesses of current approaches can set expectations appropriately, identify risk areas to tool adopters, and suggest ways in which tool builders can meet industrial needs. If we look around us, today we see numerous appliances which we use daily, be it our refrigerator, the microwave oven, cars, PDAs etc. Most appliances today are powered by something beneath the sheath that makes them do what they do. These are tiny microprocessors, which respond to various keystrokes or inputs. These tiny microprocessors, working on basic assembly languages, are the heart of the appliances. We call them embedded systems. Of all the semiconductor industries, the embedded systems market place is the most conservative, and engineering decisions here usually lean towards established, low risk solutions. Welcome to the world of embedded systems, of computers that will not look like computers and won't function like anything we are familiar with.

2.1 CLASSIFICATION

Embedded systems are divided into autonomous, real time, networked & mobile categories.

Autonomous systems

They function in standalone mode. Many embedded systems used for process control in manufacturing units& automobiles fall under this category.

Real-time embedded systems

These are required to carry out specific tasks in a specified amount of time. These systems are extensively used to carry out time critical tasks in process control.

Networked embedded systems

They monitor plant parameters such as temperature, pressure and humidity and send the data over the network to a centralized system for on line monitoring.

Mobile gadgets

Mobile gadgets need to store databases locally in their memory. These gadgets imbibe powerful computing & communication capabilities to perform realtime as well as nonrealtime tasks and handle multimedia applications. The embedded system is a combination of computer hardware, software, firmware and perhaps additional mechanical parts, designed to perform a specific function. A good example is an automatic washing machine or a microwave oven. Such a system is in direct contrast to a personal computer, which is not designed to do only a specific task. But an embedded system is designed to do a specific task with in a given timeframe, repeatedly, endlessly, with or without human interaction.

Hardware

Good software design in embedded systems stems from a good understanding of the hardware behind it. All embedded systems need a microprocessor, and the kinds of microprocessors used in them are quite varied. A list of some of the common microprocessors families are: ARM family, The Zilog Z8 family, Intel 8051/X86 family, Motorola 68K family and the power PC family. For processing of information and execution of programs, embedded system incorporates microprocessor or micro- controller. In an embedded system the microprocessor is a part of final product and is not available for reprogramming to the end user. An embedded system also needs memory for two purposes, to store its program and to store its data. Unlike normal desktops in which data and programs are stored at the same place, embedded systems store data and programs in different memories. This is simply because the embedded system does not have a hard drive and the program must be stored in memory even when the power is turned off. This type of memory is called ROM. Embedded applications commonly employ a special type of ROM that can be programmed or reprogrammed with the help of special devices.

2.2 OTHER COMMON PARTS FOUND ON MANY EMBEDDED SYSTEMS

- UART& RS232
- PLD
- ASIC's& FPGA's
- Watch dog timer etc.

2.3 DESIGN PROCESS

Embedded system design is a quantitative job. The pillars of the system design methodology are the separation between function and architecture, is an essential step from conception to implementation. In recent past, the search and industrial community has paid significant attention to the topic of hardware-software (HW/SW) codesign and has tackled the problem of coordinating the design of the parts to be implemented as software and the parts to be implemented as hardware avoiding the HW/SW integration problem marred the electronics system industry so long. In any large scale embedded systems design methodology, concurrency must be considered as a first class citizen at all levels of abstraction and in both hardware and software. Formal models & transformations in system design are used so that verification and synthesis can be applied to advantage in the design methodology. Simulation tools are used for exploring the design space for validating the functional and timing behaviors of embedded systems. Hardware can be simulated at different levels such as electrical circuits, logic gates, RTL e.t.c. using VHDL description. In some environments software development tools can be coupled with hardware simulators, while in others the software is executed on the simulated hardware. The later approach is feasible only for small parts of embedded systems. Design of an embedded system using Intel's 80C188EB chip is shown in the figure. Inorder to reduce complexity, the design process is divided in four major steps: specification, system synthesis, implementation synthesis and performance evaluation of the prototype.

2.3.1 SPECIFICATION

During this part of the design process, the informal requirements of the analysis are transformed to formal specification using SDL.

2.3.2 SYSTEM-SYNTHESIS

For performing an automatic HW/SW partitioning, the system synthesis step translates the SDL specification to an internal system model switch contains problem graph& architecture graph. After system synthesis, the resulting system model is translated back to SDL.

2.3.3 IMPLEMENTATION-SYNTHESIS

SDL specification is then translated into conventional implementation languages such as VHDL for hardware modules and C for software parts of the system.

2.3.4 PROTOTYPING

On a prototyping platform, the implementation of the system under development is executed with the software parts running on multiprocessor unit and the hardware part running on a FPGA board known as phoenix, prototype hardware for Embedded Network Interconnect Accelerators.

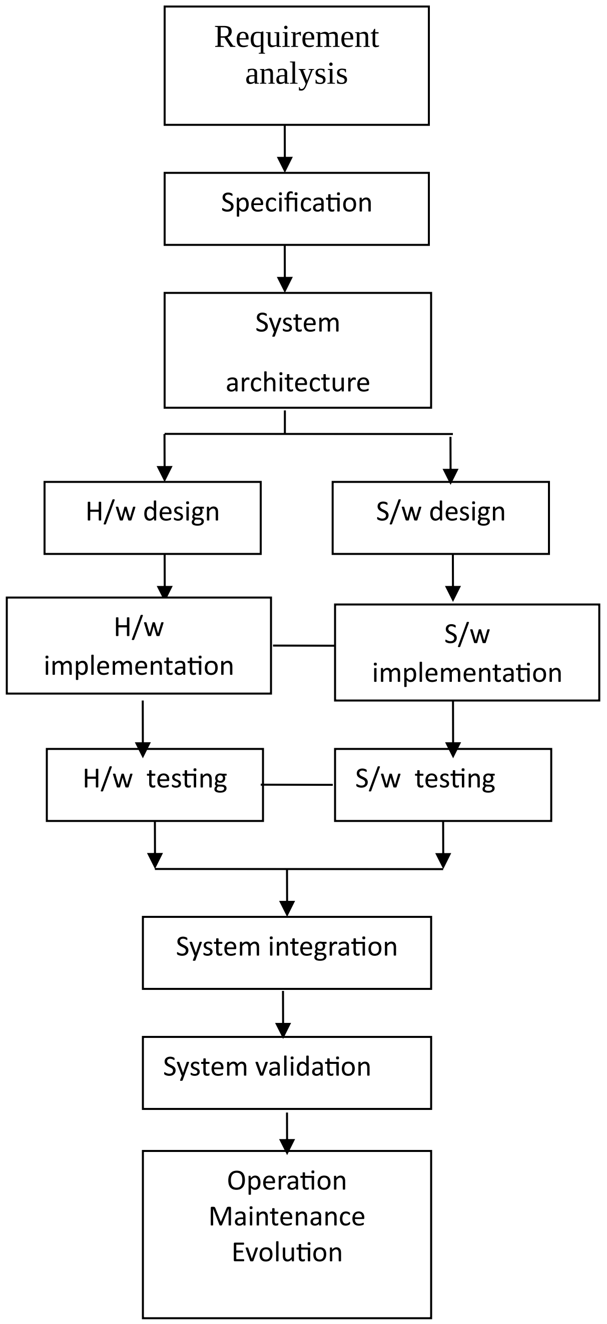


Fig 2.1: Embedded Development Life Cycle

2.3.5 APPLICATIONS

Embedded systems are finding their way into robotic toys and electronic pets, intelligent cars and remote controllable home appliances. All the major toy makers across the world have been coming out with advanced interactive toys that can become our friends for life. ‘Furby’ and ‘AIBO’ are good examples at this kind. Furbies have a distinct life cycle just like human beings, starting from being a baby and growing to an adult one. In AIBO first two letters stands for Artificial Intelligence. Next two letters represents robot. The AIBO is robotic dog. Embedded systems in cars also known as Telematic Systems are used to provide navigational security communication & entertinment services using GPS, satellite. Home appliances are going the embedded way. LG electronics digital DIOS refrigerator can be used for surfing the net, checking e-mail, making video phone calls and watching TV.IBM is developing an air conditioner that we can control over the net. Embedded systems cover such a broad range of products that generalization is difficult. Here are some broad categories.

- **Aerospace and defence electronics**: Fire control, radar, robotics/sensors, sonar.
- **Automotive**: Autobody electronics, auto power train, auto safety, car information systems.
- **Broadcast & entertainment**: Analog and digital sound products, camaras, DVDs, Set top boxes, virtual reality systems, graphic products.
- **Consumer/internet appliances**: Business handheld computers, business network computers/terminals, electronic books, internet smart handheld devices, PDAs.
- **Data communications:** Analog modems, ATM switches, cable modems, XDSL modems, Ethernet switches, concentrators.
- **Digital imaging**: Copiers, digital still cameras, Fax machines, printers, scanners.
- **Industrial measurement and control:** Hydro electric utility research & management traffic management systems, train marine vessel management systems.
- **Medical electronics:** Diagnostic devices, real time medical imaging systems, surgical devices, critical care systems.
- **Server I/O:** Embedded servers, enterprise PC servers, PCI LAN/NIC controllers, RAID devices, SCSI devices.
- **Telecommunications**: ATM communication products, base stations, networking switches, SONET/SDH cross connect, multiplexer.
- **Mobile data infrastructures**: Mobile data terminals, pagers, VSATs, Wireless LANs, Wireless phones.

CHAPTER 3
ARDUINO

3.1 INTRODUCTION OF 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.

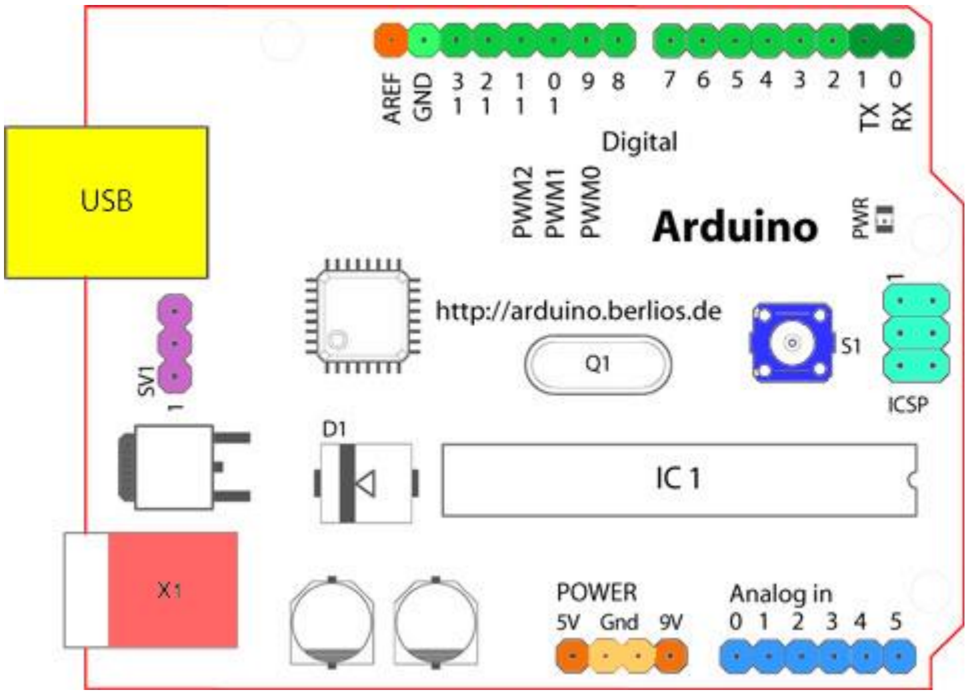


Fig 3.1 : Structure of Arduino Board

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)

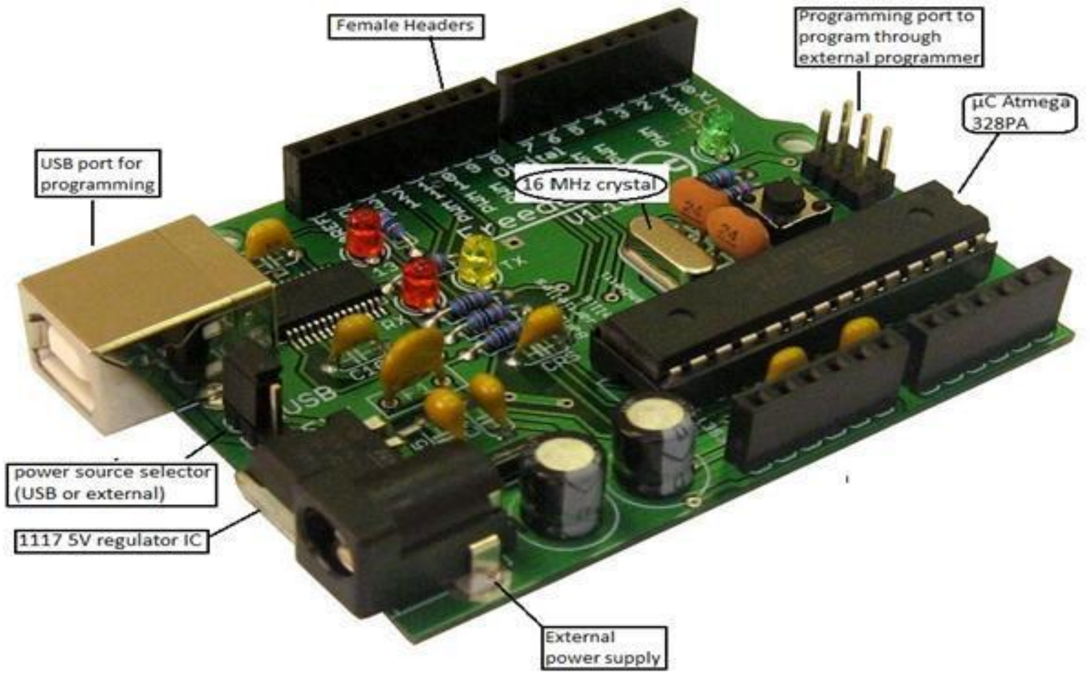


Fig 3.2: Arduino Board

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)

3.2 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. See the attach Interrupt() function for details.
- **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 Diecimila 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.

3.3 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^{2}C: 4 (SDA) and 5 (SCL).** Support I^{2}C (TWI) communication using the Wire library (documentation on the Wiring website).

3.4 POWER PINS

- **VIN** (sometimes labeled "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** (Diecimila-only) : A 3.3 volt supply generated by the on-board FTDI chip.
- **GND:** Ground pins.

3.5 OTHER PINS

- **AREF:** Reference voltage for the analog inputs. Used with analog Reference().
- **Reset:** (Diecimila-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.

3.6 ATMEGA328

Pin diagram

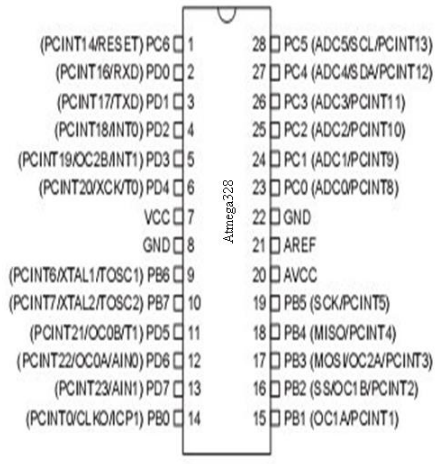


Fig 3.3: Pin Configuration of Atmega328

The ATmega32 microcontroller pin descriptions are crucial for understanding its functionality and usage in various applications. Here's a summarized description of the VCC, GND, Port A, Port B, Port C, Port D, Reset, XTAL1, XTAL2, AVCC, and AREF pins, condensed into a two-page format.

1. **VCC**: This is the digital supply voltage pin for the ATmega32 microcontroller. It powers the internal circuits and ensures proper functioning of the microcontroller when connected to a suitable power source.
2. **GND**: The ground pin provides the reference point for all signals in the microcontroller. It's essential for the proper operation of digital and analog circuits.
3. **Port A (PA7-PA0)**: Port A serves two primary functions. When not using the A/D Converter, it acts as an 8-bit bi-directional I/O port with internal pull-up resistors. The output buffers have high sink and source capability. When used as analog inputs for the

A/D Converter, the Port A pins can provide accurate analog readings from sensors. These pins are tri-stated upon reset.

4. **Port B (PB7-PB0)**: Port B is another 8-bit bi-directional I/O port, similar to Port A, with internal pull-up resistors. It also provides symmetrical drive characteristics and is tri-stated during reset. Port B is used for special functions in the ATmega32, such as communication interfaces, timers, and interrupt inputs.
5. **Port C (PC7-PC0)**: Port C is an 8-bit bi-directional I/O port with internal pull-up resistors. It shares characteristics similar to Port A and B but also integrates functions of the JTAG interface (PC5, PC3, and PC2). These pins have pull-up resistors activated during a reset if the JTAG interface is enabled. Port C is also tri-stated during reset.
6. **Port D (PD7-PD0)**: Like the other ports, Port D is an 8-bit bi-directional I/O port with internal pull-up resistors. It serves various special functions of the ATmega32, such as communication, timer outputs, and interrupts. Similar to Port A, B, and C, Port D pins are tri-stated during reset.
7. **Reset (Reset Input)**: This pin generates a reset when a low level is applied for a duration longer than the minimum pulse length. It ensures that the microcontroller is reset, even if the clock is not running.
8. **XTAL1**: Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. This pin receives the clock signal from an external crystal or resonator.
9. **XTAL2**: Output from the inverting Oscillator amplifier. It sends the clock signal back to the microcontroller, allowing it to function at the desired clock speed.

10. **AVCC**: The AVCC pin provides the supply voltage for Port A and the A/D Converter. It should be connected to VCC even if the ADC is not used, and if used, it must be connected through a low-pass filter to ensure stable voltage supply.
11. **AREF**: AREF is the analog reference pin for the A/D Converter. It should be connected to a stable voltage level to provide a reference for analog-to-digital conversion.

These pin descriptions are essential for designing circuits that involve the ATmega32 microcontroller, ensuring proper interfacing with sensors, other microcontrollers, or communication interfaces. For more detailed information and pin assignments, refer to the ATmega32 datasheet provided by Atmel or Microchip, or visit their official website.

3.7 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

3.8 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.

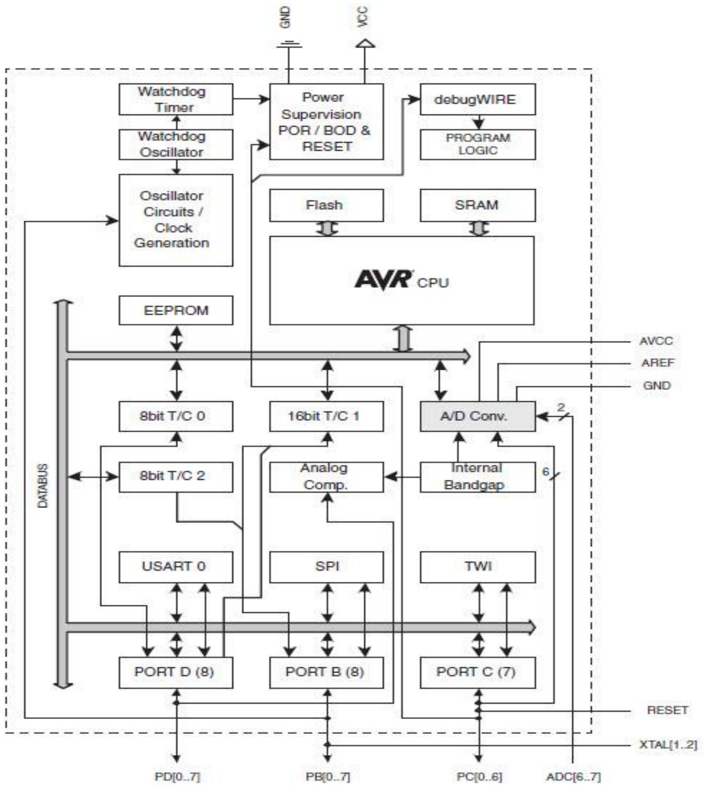


Fig 3.4: Block Diagram of AVR CPU Core

3.9 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.

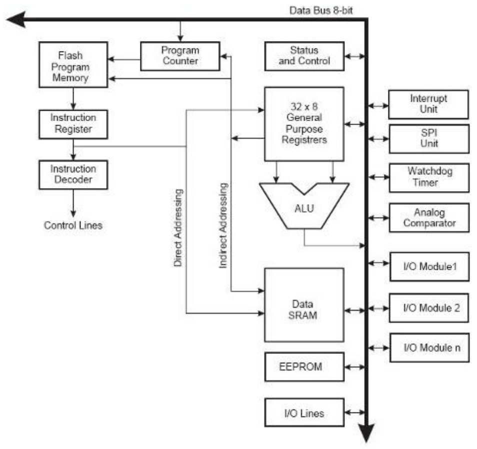


Fig 3.5: core architecture

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 next instruction 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.

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. The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation. Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction. Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section. During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps. A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

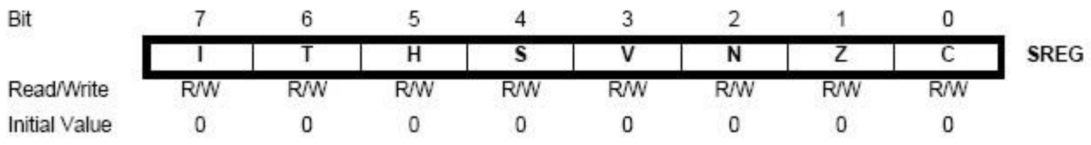
3.10 ALU – ARITHMETIC LOGIC UNIT

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.

3.11 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:



Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 3.6 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, S = N□□ V

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.

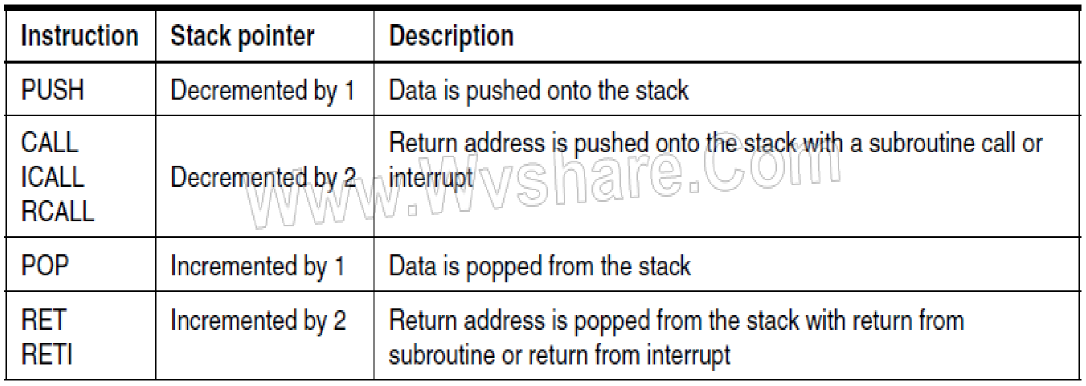
Bit 0 – C: Carry Flag

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

3.12 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.

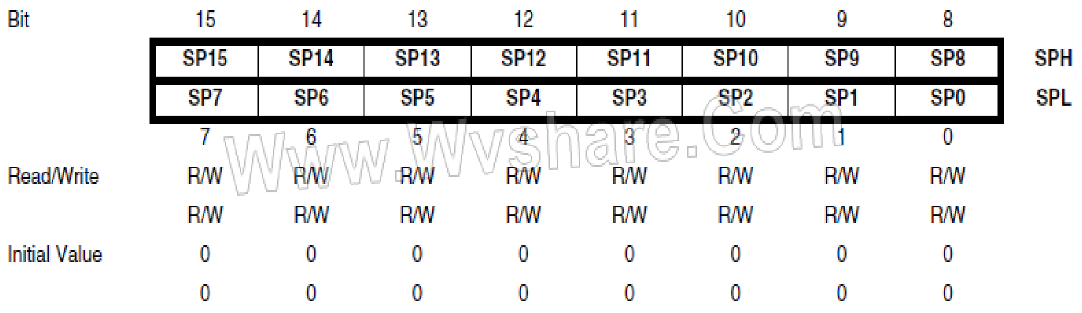
The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM



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

Table 3.1 Stack Pointer instructions

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 data space 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.



Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figure 3.7 SPH and SPL - Stack Pointer High and Low Register

3.13 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.

3.14 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 post-increment, 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.

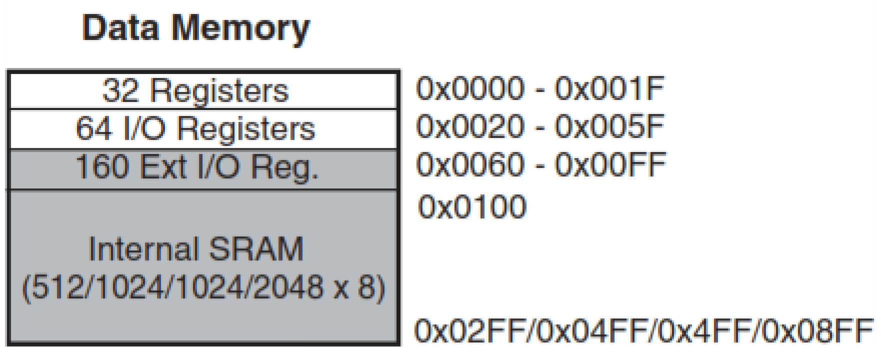


Figure 3.8 Data Memory Map

3.15 Interrupts

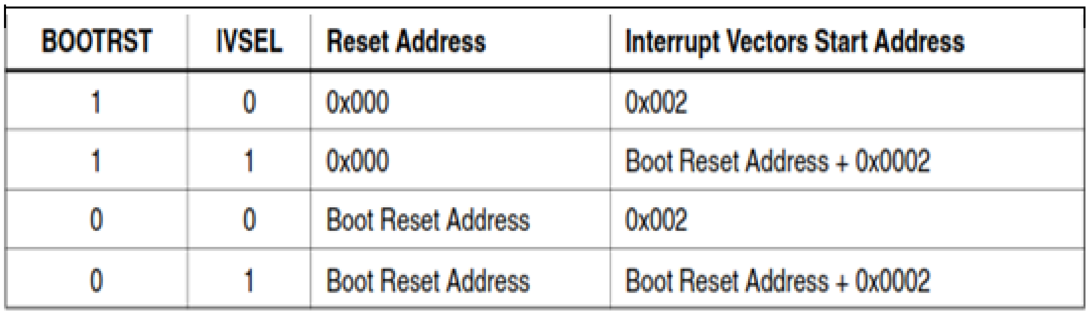
This section describes the specifics of the interrupt handling as performed in the Atmega328. In Atmega328Each Interrupt Vector occupies two instruction words and the Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

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

Table 3.2 Reset and Interrupt Vectors in ATMEGA 328 and ATMEGA 328P

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. 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.



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

Table 3.3 Reset and Interrupt Vectors Placement in ATmega328 and ATmega328P

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 16 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.

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.

- 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.

3.16 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 2.1mm center-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 SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Serial Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, **on Windows, a .inf file is required**. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. For SPI communication, use the SPI library.

CHAPTER 4
HARDWARE COMPONENTS

4.1 LCD (LIQUID CRYSTAL DISPLAY)

4.1.1 Introduction:

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

A program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an controller is an LCD display. Some of the most common LCDs connected to the controllers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Many microcontroller devices use 'smart LCD' displays to output visual information. LCD displays designed around LCD NT-C1611 module, are inexpensive, easy to use, and it is even possible to produce a readout using the 5X7 dots plus cursor of the display. They have a standard ASCII set of characters and mathematical symbols. For an 8-bit data bus, the display requires a +5V supply plus 10 I/O lines (RS RW D7 D6 D5 D4 D3 D2 D1 D0). For a 4-bit data bus it only requires the supply lines plus 6 extra lines(RS RW D7 D6 D5 D4). When the LCD display is not enabled, data lines are tri-state and they do not interfere with the operation of the microcontroller.

4.1.2 Features:

(1) Interface with either 4-bit or 8-bit microprocessor.

(2) Display data RAM

(3) 80x08 bits (80 characters).

□(4) Character generator ROM

(5). 160 different 5007 dot-matrix character patterns.

(6). Character generator RAM

(7) 8 different user programmed 5007 dot-matrix patterns.

(8).Display data RAM and character generator RAM may be

Accessed by the microprocessor.

(9) Numerous instructions

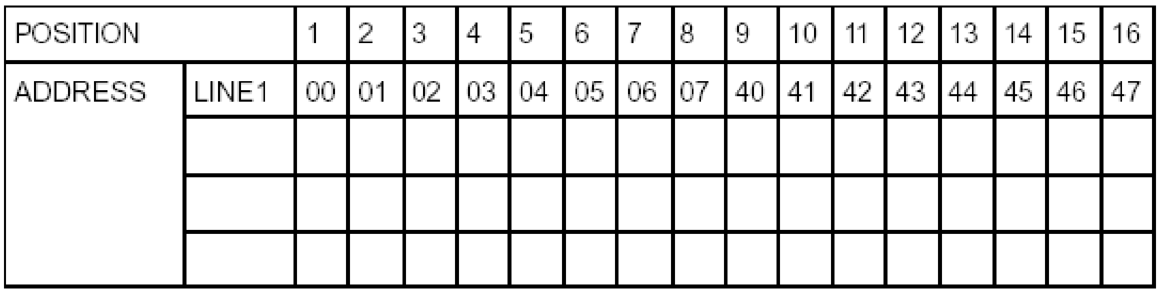
(10) .Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF,

Blink Character, Cursor Shift, Display Shift.

(11). Built-in reset circuit is triggered at power ON.

(12). Built-in oscillator.

Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:



POSITION		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ADDRESS	LINE1	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

Fig 4.1 : Address locations for a 1x16 line LCD

4.1.3 Shapes and sizes:

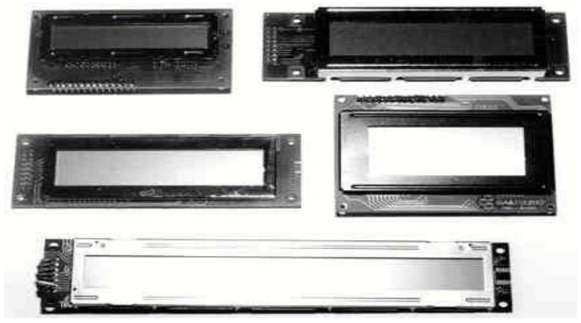


Fig 4.2 : shapes of LCD

Even limited to character based modules, there is still a wide variety of shapes and sizes available. Line lengths of 8,16,20,24,32 and 40 characters are all standard, in one, two and four line versions.

Several different LC technologies exists. “supertwist” types, for example, offer Improved contrast and viewing angle over the older “twisted nematic” types. Some modules are available with back lighting, so that they can be viewed in dimly-lit conditions. The back lighting may be either “electro-luminescent”, requiring a high voltage inverter circuit, or simple LED illumination.

4.1.4 Electrical block diagram:

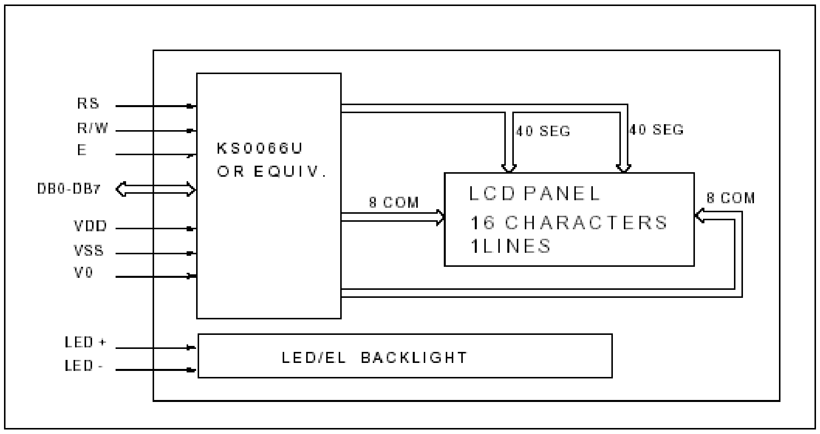


Fig 4.3 : Electrical block diagram of LCD

4.1.5 Power supply for lcd driving:

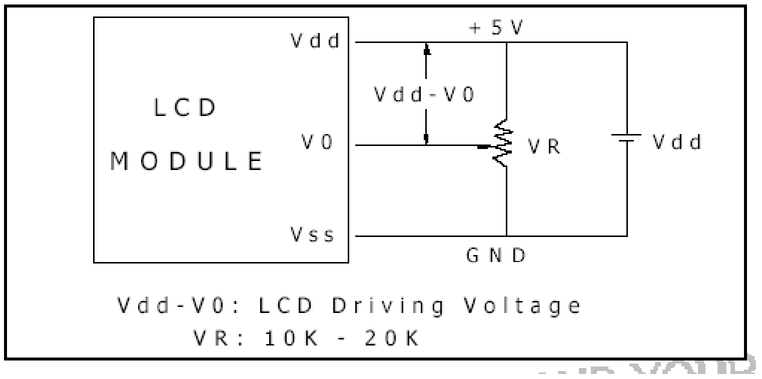


Fig 4.4 : Power supply of lcd driving

4.1.6 PIN DESCRIPTION:

`Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections).

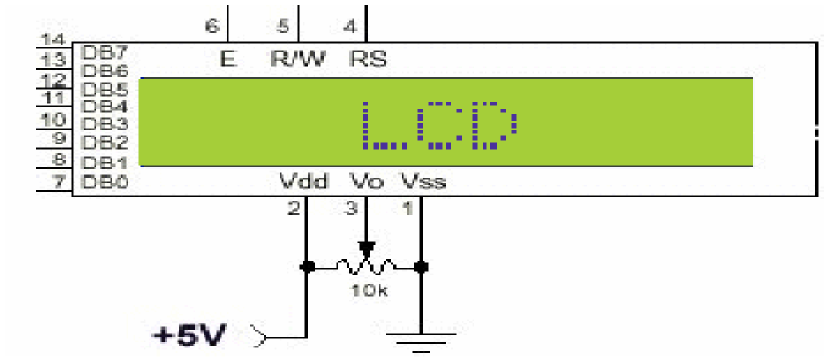


Fig 4.5 : Pin description of LDC

4.1.7 CONTROL LINES:

EN:

Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

RS:

Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which sould be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

RW:

Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

4.1.8 Logic status on control lines:

- E - 0 Access to LCD disabled

- 1 Access to LCD enabled

• R/W - 0 Writing data to LCD

- 1 Reading data from LCD

• RS - 0 Instructions

4.1.9 Writing data to the LCD:

1) Set R/W bit to low

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

4.1.10 Read data from data lines (if it is reading)on LCD:

1) Set R/W bit to high

2) Set RS bit to logic 0 or 1 (instruction or character)

3) Set data to data lines (if it is writing)

4) Set E line to high

5) Set E line to low

4.1.11 Entering Text:

To make it easier to enter characters and commands manually, switching to hexadecimal simplifies the process. This method uses hexadecimal rotary switches where "On = 0" and "Off = F," making it straightforward to understand and set which bits are active. The table of characters shows codes in both binary and hexadecimal format, with the most significant bits across the top and the least significant bits down the side. This setup helps in identifying ASCII standard characters along with exceptions like Japanese Katakana symbols.

For the character display, the first 16 codes (from 00000000 to 00001111, $00 to $0F) refer to the CGRAM (Character Generator RAM). This area can store user-defined graphics characters, enabling functions such as bar graphs, flashing symbols, and animated characters. Once these codes are programmed, they display distinctive symbols instead of blanks, showcasing the module's advanced capabilities.

Codes from 00010000 to 00011111 ($10 to $1F) are not used and display blank characters on the screen. ASCII characters, which are more commonly used, start at 00100000 ($20) and end with 01111111 ($7F). Codes in the range 10000000 to 10011111 ($80 to $9F) are also not used, while 10100000 to 11011111 ($A0 to $DF) include Japanese characters. The control for the display, such as setting RS (Register Select) to "high" and triggering E (Enable) to send characters, requires proper sequence and timing to function correctly, as detailed in the initialization procedure for the LCD unit.

If power conditions for normal operation aren't met, a specific initialization process must be executed to prepare the LCD unit for use. This process includes setting up the Character Generator RAM properly, ensuring that all configurations are accurate for displaying characters as intended. The flexibility offered by these modules allows for diverse character representation, enhancing their use in applications beyond simple text display, as shown by the detailed control over display elements and character sets.

4.1.12 Initialization by Instructions:

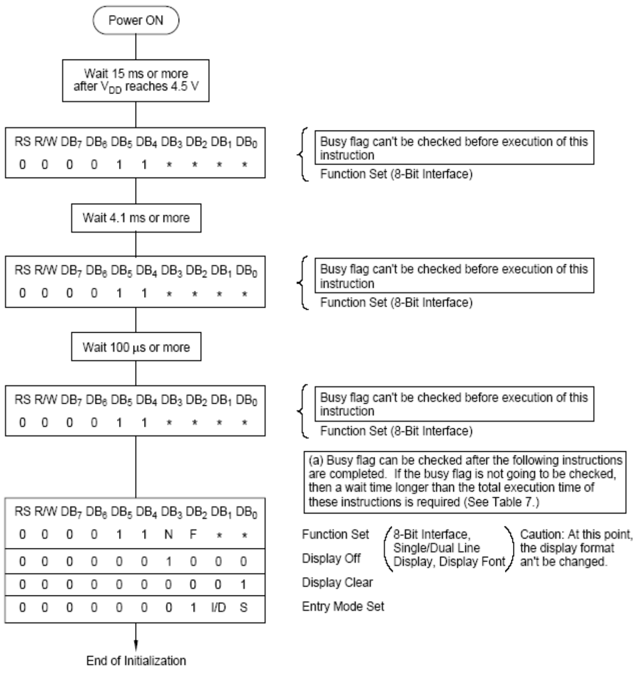

Power ON
Wait 15 ms or more after V_DD reaches 4.5 V
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
Wait 4.1 ms or more
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
Wait 100 µs or more
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
Busy flag can't be checked before execution of this instruction
Function Set (8-Bit Interface)
(a) Busy flag can be checked after the following instructions are completed. If the busy flag is not going to be checked, then a wait time longer than the total execution time of these instructions is required (See Table 7.)
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 N F * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 I/D S
Function Set
Display Off
Display Clear
Entry Mode Set
8-Bit Interface, Single/Dual Line Display, Display Font
Caution: At this point, the display format an't be changed.
End of Initialization

Fig 4.6 : Initialization by instructors

If the power conditions for the normal operation of the internal reset circuit are not satisfied, then executing a series of instructions must initialize LCD unit. The procedure for this initialization process is as above show.

4.2 REGULATED POWER SUPPLY:

4.2.1 Introduction:

Power supply is a supply of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

A power supply may include a power distribution system as well as primary or secondary sources of energy such as

- Conversion of one form of electrical power to another desired form and voltage, typically involving converting AC line voltage to a well-regulated lower-voltage DC for electronic devices. Low voltage, low power DC power supply units are commonly integrated with the devices they supply, such as computers and household electronics.
- Batteries.
- Chemical fuel cells and other forms of energy storage systems.
- Solar power.
- Generators or alternators.

4.2.2 Block Diagram:

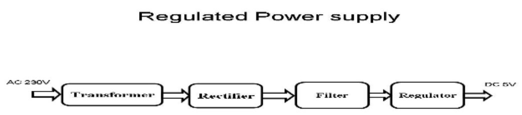


Fig 4.7 : Block diagram of Regulated Power Supply

The basic circuit diagram of a regulated power supply (DC O/P) with led connected as load is shown in fig:

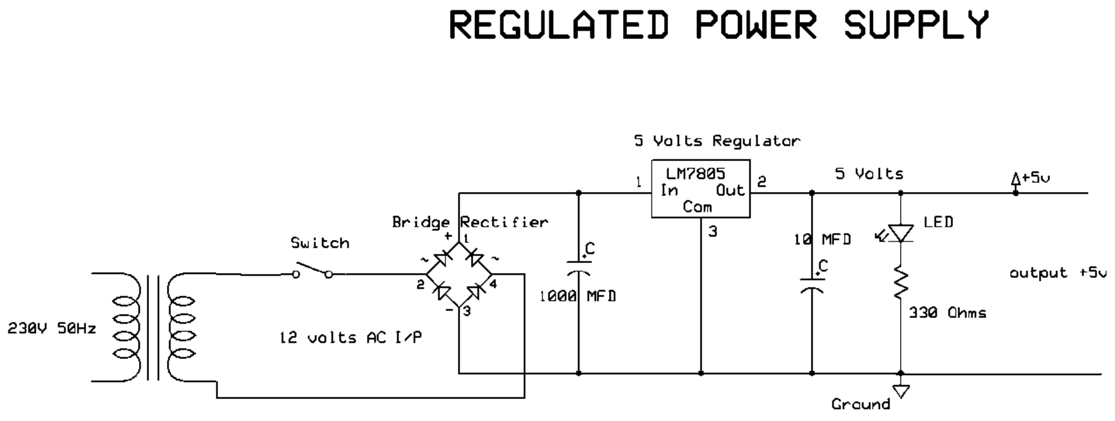


Fig 4.8 : circuit diagram of Regulated Power Supply

The components mainly used in above figure are

- 230V AC MAINS
- TRANSFORMER
- BRIDGE RECTIFIER(DIODES)
- CAPACITOR
- VOLTAGE REGULATOR(IC 7805)
- RESISTOR
- LED(LIGHT EMITTING DIODE)

The detailed explanation of each and every component mentioned above is as follows:

4.2.3 Step 1: Transformation: The process of transforming energy from one device to another is called transformation. For transforming energy we use transformers.

4.2.4 Transformers:

A transformer is an electrical device that transfers energy between circuits through inductively coupled conductors, maintaining the same frequency during the transfer. It operates based on the principle of mutual induction, where a varying current in the primary winding creates a changing magnetic flux in the core, inducing an electromotive force (EMF) in the secondary winding. This induced EMF allows voltage to be transferred to the secondary winding, which powers a connected load. When current flows through the primary winding, it generates a magnetic field resembling the shape of a bar magnet. As the current increases, the lines of magnetic force expand outward, and when the current decreases, they contract inward. If a secondary coil is placed near the primary, the moving magnetic field lines cut across the turns of the secondary coil, inducing voltage within it. With an alternating current (AC) supply, such as the 50 Hz mains, this induction occurs 50 times per second. The primary winding is the input coil, while the secondary winding is the output coil. The transformer's operation relies on mutual induction, which forms the basis for converting voltages up or down. A step-down transformer reduces the voltage from the primary to the secondary winding, as depicted in related diagrams, and is commonly used in various applications to match voltage levels for devices or systems.50S

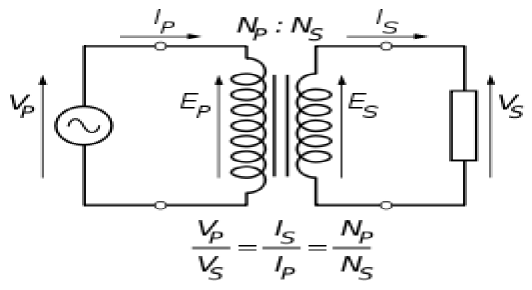


Fig 4.9 : Step Down Transformer

The voltage induced in the secondary is determined by the TURNS RATIO.

$$\frac{\text{primary voltage}}{\text{secondary voltage}} = \frac{\text{number of primary turns}}{\text{number of secondary turns}}$$

In transformers, the voltage ratio between primary and secondary windings is directly proportional to their turns ratio. For example, a secondary with half the turns of the primary will have half its voltage. If the primary has 5000 turns and the secondary 500 turns, the turns ratio is 10:1, and a 240V primary voltage produces a 24V secondary voltage. Assuming no losses, the power supplied by the primary equals the power drawn by the secondary load.

The transformer core, often made of laminated metal sheets, enhances magnetic coupling and reduces eddy currents. At higher frequencies, cores may use iron dust or no core at all. Transformers operate only on AC, as the constantly changing current creates the moving magnetic field necessary for induction. DC cannot induce voltage because of its steady current and field.

Some transformers include an electrostatic screen between primary and secondary windings to reduce interference between equipment and the power supply. Additionally, transformers are used for impedance matching, ensuring efficient power transfer between circuits with different impedance levels.

We can use the transformers as step up or step down.

4.2.5 Step Up transformer:

In case of step up transformer, primary windings are every less compared to secondary winding. Because of having more turns secondary winding accepts more energy, and it releases more voltage at the output side.

4.2.6 Step down transformer:

Incase of step down transformer, Primary winding induces more flux than the secondary winding, and secondary winding is having less number of turns because of that it accepts less number of flux, and releases less amount of voltage.

4.2.7 Battery power supply:

A battery is a type of linear power supply that offers benefits that traditional line-operated power supplies lack: mobility, portability and reliability. A battery consists of multiple electrochemical cells connected to provide the voltage desired.

Fig 4.10 : Hi-Watt 9V battery

Dry-cell batteries, like the carbon-zinc variety, consist of carbon plates, zinc plates, and an electrolyte paste layered to achieve the desired voltage. These batteries typically provide voltages such as 1.5V, 3V, and 9V. During discharge, zinc converts to zinc salt, and manganese dioxide is reduced, maintaining a stable 1.5V output.

Rechargeable lead-acid batteries use lead and lead dioxide electrodes in sulfuric acid. A fully charged cell produces 2.06–2.14V, with a 12V car battery comprising six cells in series. During discharge, lead converts to lead sulphate, and the process reverses during charging.

Nickel-cadmium (Ni-Cd) batteries are also rechargeable and sealed. They offer advantages such as constant voltage, long service life, and the ability to be stored charged or uncharged. In Ni-Cd cells, nickel oxide is oxidized, and cadmium oxide is reduced during charging, ensuring steady performance. These batteries are valued for their reliability and high current capacity.

Fig 4.11 : Pencil Battery of 1.5V

4.2.8 Step 2: Rectification

The process of converting an alternating current to a pulsating direct current is called as rectification. For rectification purpose we use rectifiers.

4.2.9 Rectifiers:

A rectifier is an electrical device that converts alternating current (AC) to direct current (DC), a process known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid-state diodes, vacuum tube diodes, mercury arc valves, and other components.

A device that it can perform the opposite function (converting DC to AC) is known as an inverter.

When only one diode is used to rectify AC (by blocking the negative or positive portion of the waveform), the difference between the term diode and the term rectifier is merely one of usage, i.e., the term rectifier describes a diode that is being used to convert AC to DC. Almost all rectifiers comprise a number of diodes in a specific arrangement for more efficiently converting AC to DC than is possible with only one diode. Before the development of silicon semiconductor rectifiers, vacuum tube diodes and copper (I) oxide or selenium rectifier stacks were used.

4.2.10 Bridge full wave rectifier:

The Bridge rectifier circuit is shown in figure, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure. The circuit has four diodes connected to form a bridge. The ac input voltage is applied to the diagonally opposite ends of the bridge. The load resistance is connected between the other two ends of the bridge. For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance R_L and hence the load current flows through $R_{L.}$ For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance R_L and hence the current flows through R_L in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.

Input Output

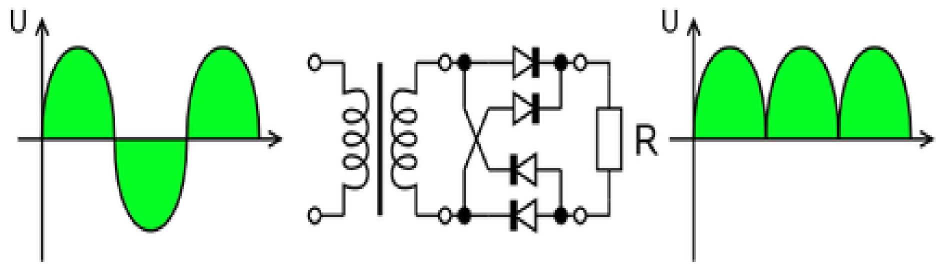


Fig 4.12 : Bridge rectifier: a full-wave rectifier using 4 diodes

4.2.11 Step 3: Filtration

The process of converting a pulsating direct current to a pure direct current using filters is called as filtration.

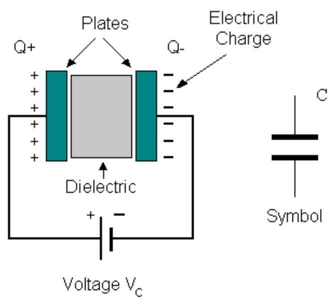
4.2.12 Filters:

Electronic filters are electronic circuits, which perform signal-processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones.

4.3 CAPACITORS

4.3.1 Introduction to Capacitors:

The **Capacitor** or sometimes referred to as a Condenser is a passive device, and one which stores energy in the form of an electrostatic field which produces a potential (static voltage) across its plates. In its basic form a capacitor consists of two parallel conductive plates that are not connected but are electrically separated either by air or by an insulating material called the Dielectric. This flow of electrons to the plates is known as the Charging Current and continues to flow until the voltage across the plates (and hence the capacitor) is equal to the applied voltage Vcc. At this point the capacitor is said to be fully charged and this is illustrated below.



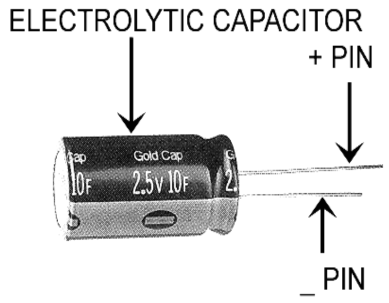


Fig 4.13 : Construction Of a Capacitor

Fig 4.14 : Electrolytic Capacitor

Units of Capacitance:

Microfarad (µF) 1µF = 1/1,000,000 = 0.000001 = 10^{-6} F

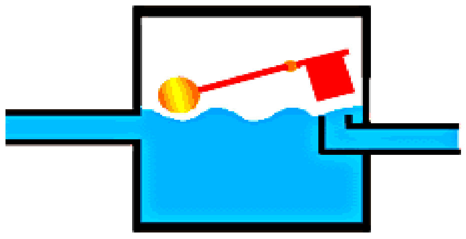
Nano faraday (nF) 1nF = 1/1,000,000,000 = 0.000000001 = 10^{-9} F

Pico farad (pF) 1pF = 1/1,000,000,000,000 = 0.000000000001 = 10^{-12} F

4.3.2 Operation of Capacitor:

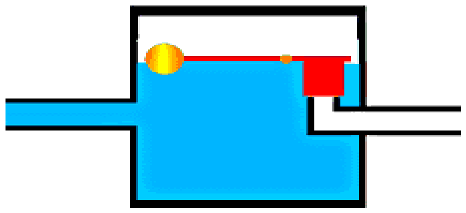
Think of water flowing through a pipe. If we imagine a capacitor as being a storage tank with an inlet and an outlet pipe, it is possible to show approximately how an electronic capacitor works.

First, let's consider the case of a "coupling capacitor" where the capacitor is used to connect a signal from one part of a circuit to another but without allowing any direct current to flow.



If the current flow is alternating between zero and a maximum, our "storage tank" capacitor will allow the current waves to pass through.

Fig 4.15 : current flow is alternating between zero



However, if there is a steady current, only the initial short burst will flow until the "floating ball valve" closes and stops further flow.

Fig 4.16 : the initial short burst

So a coupling capacitor allows "alternating current" to pass through because the ball valve doesn't get a chance to close as the waves go up and down. However, a steady current quickly fills the tank so that all flow stops.

A capacitor will pass alternating current but (apart from an initial surge) it will not pass dc.

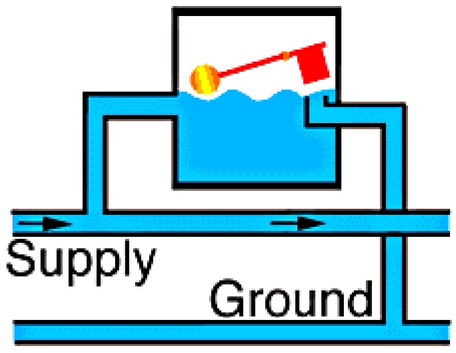


Fig 4.17 : capacitor is used to decouple a circuit

Where a capacitor is used to decouple a circuit, the effect is to "smooth out ripples". Any ripples, waves or pulses of current are passed to ground while Dftable

c. Flows smoothly.

4.4 Step 4: Regulation

The process of converting a varying voltage to a constant regulated voltage is called as regulation. For the process of regulation we use voltage regulators.

4.4.1 Voltage Regulator:

A voltage regulator (also called a ‘regulator’) with only three terminals appears to be a simple device, but it is in fact a very complex integrated circuit. It converts a varying input voltage into a constant ‘regulated’ output voltage. Voltage Regulators are available in a variety of outputs like 5V, 6V, 9V, 12V and 15V. The LM78XX series of voltage regulators are designed for positive input. For applications requiring negative input, the LM79XX series is used. Using a pair of ‘voltage-divider’ resistors can increase the output voltage of a regulator circuit.

It is not possible to obtain a voltage lower than the stated rating. You cannot use a 12V regulator to make a 5V power supply. Voltage regulators are very robust. These can withstand over-current draw due to short circuits and also over-heating. In both cases, the regulator will cut off before any damage occurs. The only way to destroy a regulator is to apply reverse voltage to its input. Reverse polarity destroys the regulator almost instantly. Fig: below shows voltage regulator.

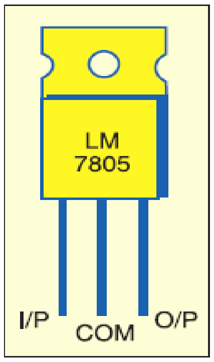


Fig 4.18 : voltage regulator

4.5 Resistors:

A resistor is a two-terminal electronic component that produces a voltage across its terminals that is proportional to the electric current passing through it in accordance with Ohm's law:

$$V = IR$$

Resistors are elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).

The primary characteristics of a resistor are the resistance, the tolerance, maximum working voltage and the power rating. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance is determined by the design, materials and dimensions of the resistor.

Resistors can be made to control the flow of current, to work as Voltage dividers, to dissipate power and it can shape electrical waves when used in combination of other components. Basic unit is ohms.

4.5.1 Theory of operation:

Ohm's law:

The behavior of an ideal resistor is dictated by the relationship specified in Ohm's law:

$$\mathbf{V = IR}$$

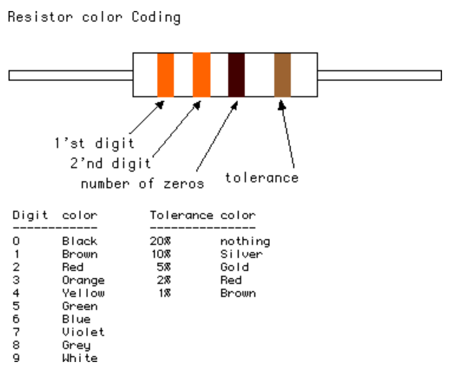
Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) through it where the constant of proportionality is the resistance (R).

Power dissipation:

The power dissipated by a resistor (or the equivalent resistance of a resistor network) is calculated using the following:

$$P = I^2 R = IV = \frac{V^2}{R}$$

Fig 4.19 : Resistor



Digit	color	Tolerance	color
0	Black	20%	nothing
1	Brown	10%	Silver
2	Red	5%	Gold
3	Orange	2%	Red
4	Yellow	1%	Brown
5	Green		
6	Blue		
7	Violet		
8	Grey		
9	White		

Fig 4.20 : colours of resistors

4.6 LED:

A light-emitting diode (LED) is a semiconductor light source. LEDs are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. The internal structure and parts of a led are shown below.

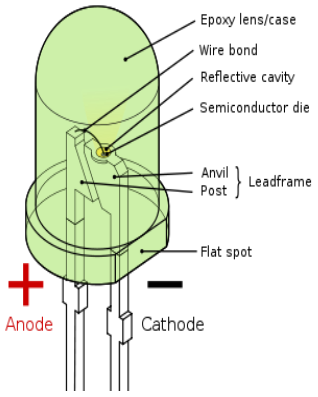


Fig 4.21 : LED

4.6.1 Working:

An LED (Light Emitting Diode) operates on the principle of electroluminescence, where light is emitted when a current passes through a semiconductor diode. When the diode is forward biased, electrons from the negative terminal move into the conduction band and recombine with holes in the positive terminal. This recombination releases energy in the form of photons, producing light. LEDs are compact, highly efficient, and robust compared to traditional light sources. They offer advantages like lower energy consumption, longer operational lifespans, and faster switching times. These attributes make LEDs ideal for various applications, including automotive lighting, traffic signals, displays, and communication technologies. However, they require precise current control and effective heat dissipation to ensure stability and longevity. While initially more expensive than conventional lighting, advancements in technology have made LEDs increasingly accessible for widespread use.The electrical symbol and polarities of led are shown in fig:

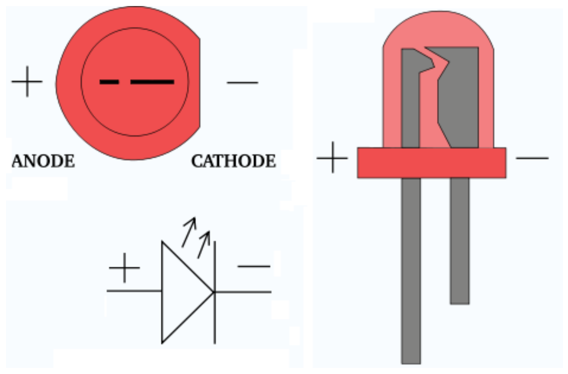


Fig 4.22 : Electrical Symbol & Polarities of LED

LED lights have a variety of advantages over other light sources:

- High-levels of brightness and intensity
- High-efficiency
- Low-voltage and current requirements
- Low radiated heat

- High reliability (resistant to shock and vibration)
- No UV Rays
- Long source life
- Can be easily controlled and programmed

Applications of LED fall into three major categories:

- Visual signal application where the light goes more or less directly from the LED to the human eye, to convey a message or meaning.
- Illumination where LED light is reflected from object to give visual response of these objects.
- Generate light for measuring and interacting with processes that do not involve the human visual system.

4.7 Soil moisture Sensor :

The soil moisture sensor is a vital component of the Smart IoT-Based Plant Watering System, as it measures the volumetric water content in the soil to ensure precise irrigation. When the soil moisture falls below a set threshold, the system automatically activates the water pump to irrigate plants, conserving water and promoting optimal plant health. Integrated with a mobile application, the sensor enables users to remotely track soil conditions, receive alerts, and analyze historical moisture data for improved irrigation strategies. This integration ensures a user-friendly and efficient approach to modern gardening.

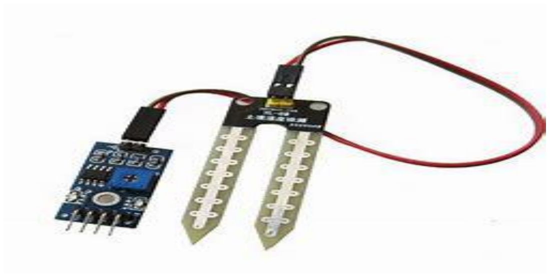


Fig 4.23 : Soil Moisture Sensor

CHAPTER 5
SOFTWARE COMPONENTS

5.1 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

5.1.2 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 Arduino-compatible microcontroller to your computer (check out this USB buying guide if you're not sure which cable to get).

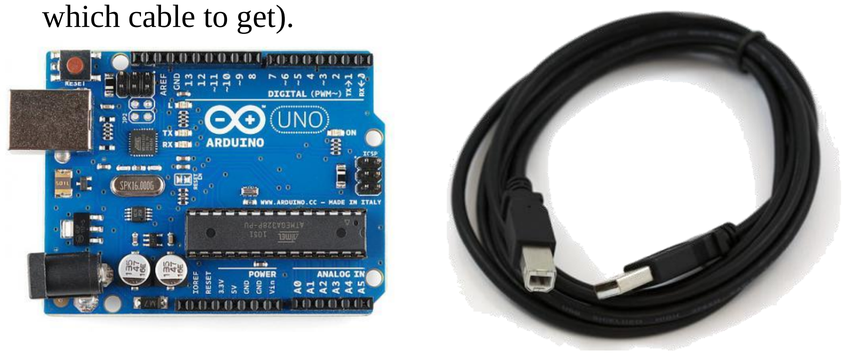
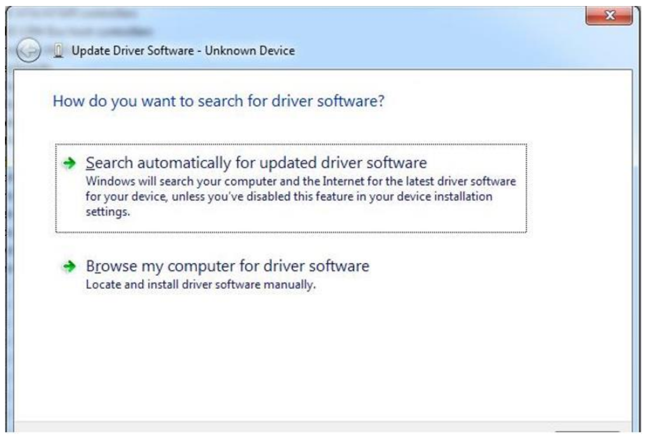


Fig 5.1 : Arduino and Arduino cable

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


Update Driver Software - Unknown Device

How do you want to search for driver software?

→ Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

→ Browse my computer for driver software
Locate and install driver software manually.

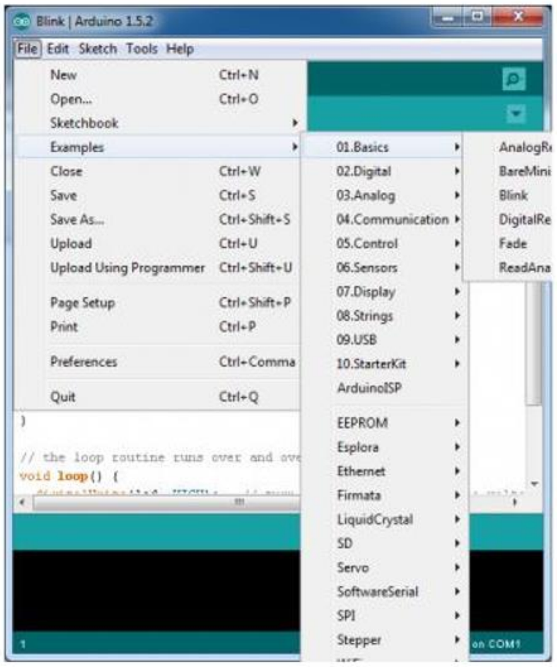
Fig 5.2 : folder of the Arduino Software download

- 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

5.1.3 LAUNCH AND BLINK!

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



```
}

// the loop routine runs over and ove
void loop() {
```

Fig 5.3 : Blink example sketch

- Select the type of Arduino board you're using: Tools > Board > your board type

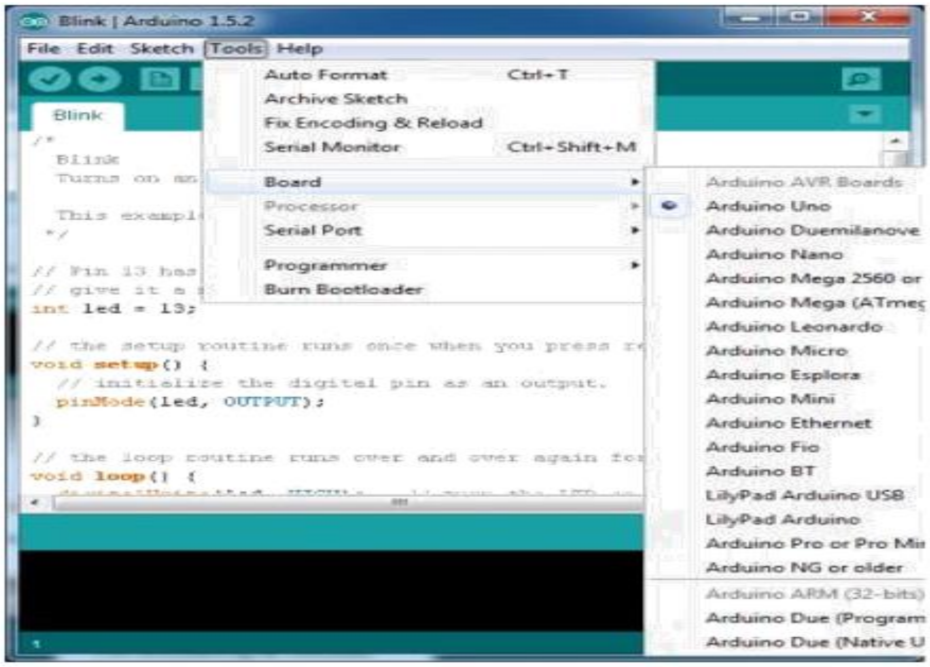
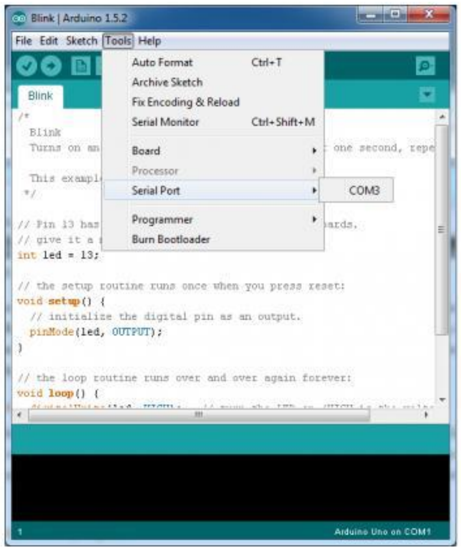


Fig 5.4 : Tools section in Arduino software

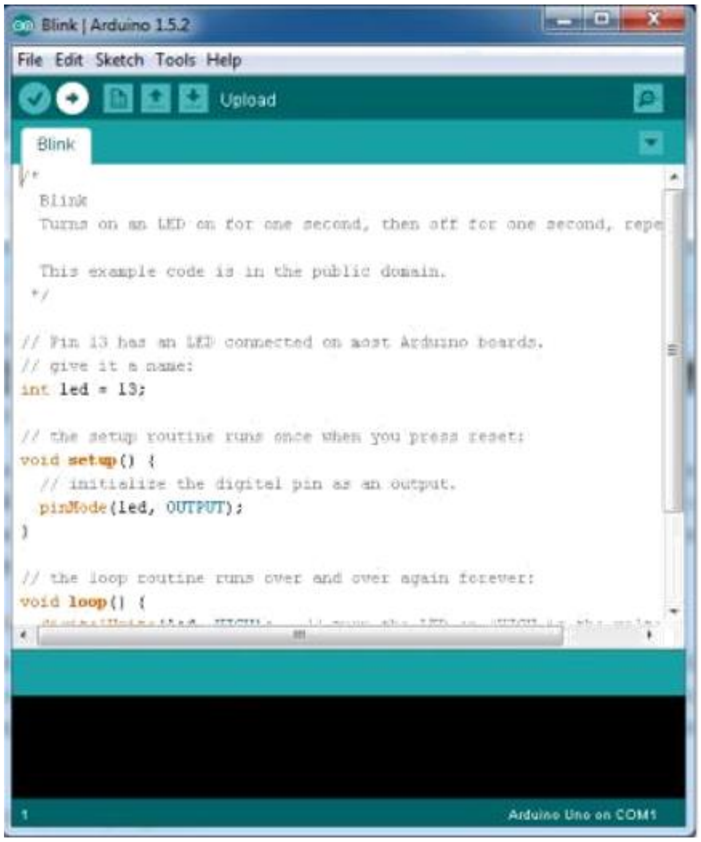
- Select the serial/COM port that your Arduino is attached to: Tools > Port > COMxx



```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
```

Fig 5.5 : the serial/COM port



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
```

Fig 5.6 : Arduino software code

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. The one that disappeared is your Arduino.

With your Arduino board connected, and the Blink sketch open, press the 'Upload' button

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.

If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!

CHAPTER 6
RESULT

The results of a "Smart IoT-Based Plant Watering System with Mobile Application Integration" demonstrate significant advancements in water management, plant health, and user convenience. The system utilizes sensors to monitor soil moisture and environmental conditions, ensuring plants receive optimal hydration. The integration of a mobile application enhances user experience by allowing remote control and monitoring. Notifications and alerts ensure plants are cared for without the need for constant human oversight, minimizing labor requirements and making the system particularly advantageous in urban gardening or large-scale farming contexts. Automation of the watering process improves plant health by maintaining consistent soil moisture levels, reducing the risks of overwatering or underwatering, which could lead to plant stress or disease. Additionally, the scalability of such systems allows their application across diverse environments, from small gardens to extensive agricultural setups. This innovative approach also promotes environmental stewardship by optimizing water usage and integrating energy-efficient IoT devices. The user-friendly design of mobile interfaces further simplifies operations, encouraging widespread adoption of technology-driven irrigation systems. Collectively, these results highlight the system's role in enhancing agricultural productivity, conserving resources, and providing practical solutions for modern gardening challenges.

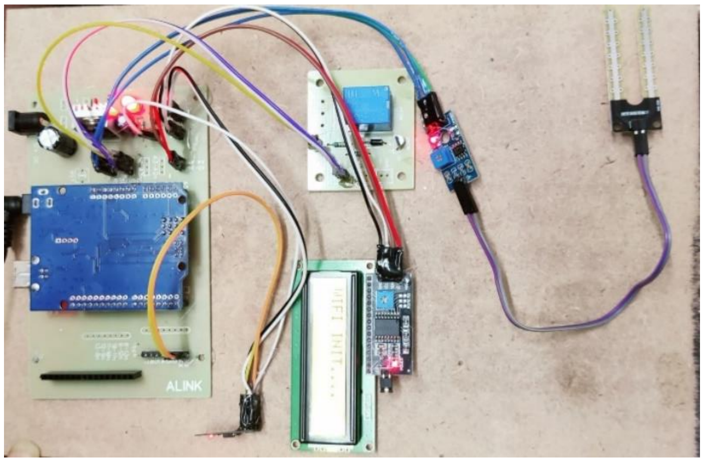


Fig 6.1 : Prototype of proposed system

CHAPTER 7
CONCLUSION

The **Smart IoT-Based Plant Watering System with Mobile Application Integration** proves to be a valuable innovation, benefiting both environmental sustainability and user experience. According to the results, the system not only optimizes water usage and enhances plant health but also demonstrates significant potential to improve the quality of life for elderly users. By simplifying gardening tasks, it bridges the gap between technology and older adults, showing that IoT applications are accessible and user-friendly. The system's ability to monitor soil and environmental conditions, coupled with automated watering, helps maintain consistent plant care, reducing risks associated with overwatering or underwatering. Furthermore, the mobile application facilitates easy interaction with the system, empowering elderly users to engage in gardening activities independently, which contributes to mental well-being by fostering recreation and hobbies they enjoy.

In conclusion, the project successfully integrates IoT technology with practical applications in gardening, demonstrating its impact on water conservation, productivity, and user well-being. Its potential to support cultural and personal dimensions, particularly for elderly individuals, underscores the importance of continuing innovation in this domain.

CHAPTER 8

REFERENCES

1. **Singh, R. (2021).** IoT-Based Smart Plant Watering System for Indian Climate Conditions. *Journal of Smart Systems*, 5(3), 45-50.
2. **Kumar, A. (2023).** Mobile Application Integration for IoT-Based Plant Watering System. *IEEE Conference on Automation and Robotics*, 12, 67-72.
3. Ramesh, S. (2022). Artificial Intelligence in IoT Plant Watering Systems. *International Journal of Smart Agriculture*, 7(1), 33-40.
4. **Deshmukh, P., et al.** (2021). Challenges and Solutions in IoT Device Deployment for Smart Watering Systems in Remote Areas. *IEEE Access*, 9, 118345-118356.
5. **Gupta, V., et al.** (2023). Applications of IoT in Smart Urban Agriculture: Case Study on Plant Watering Systems in Smart Cities. *Journal of Urban Technology*, 20(2), 120-130.
6. **Tripathi, S., & Kumar, P. (2023).** *Role of IoT in Water Management for Sustainable Agriculture.* Journal of Water Resources Management, 11(2), 30-40.
7. **Verma, A., & Sharma, P. (2023).** *Smart Irrigation Systems for Enhancing Crop Productivity using IoT and AI.* Journal of Agricultural Engineering, 28(1), 10-20.
8. **Rao, P., & Joshi, A. (2022).** *Integration of IoT with Mobile Applications for Smart Irrigation Management in Agricultural Fields.* International Journal of Agricultural Engineering and Technology, 9(3), 50-60.
9. **Patel, V., & Agarwal, A. (2022).** *IoT-Based Smart Irrigation System Using Wireless Sensors and Mobile Applications.* Journal of Precision Agriculture, 23(4), 102-110.
10. **Choudhary, S., & Bhatnagar, V. (2022).** *A Survey on IoT-Based Smart Irrigation Systems in India: Challenges and Future Directions.* International Journal of Advanced Computer Science and Applications, 13(5), 293-301.
11. **Sharma, A., & Patel, S. (2023).** *Development and Implementation of IoT-Based Smart Plant Watering System in Urban Agriculture.* Smart Cities and Urban IoT Journal, 15(2), 80-90.