ELECTRIC MOTORS

Introduction to Microcontrollers and its peripheral interface

A **microprocessor** is an integrated circuit (IC) designed to perform computation and control tasks by processing data. It serves as the **central processing unit (CPU)** in computer systems and other digital devices. The primary function of a microprocessor is to execute instructions provided by a program, performing operations like arithmetic, logic, control, and data manipulation.

Parts of a Microprocessor:

1. **Arithmetic and Logic Unit (ALU):**
    - **Function:** Performs arithmetic operations (addition, subtraction, multiplication, division) and logical operations (AND, OR, NOT, etc.) on binary data.
    - **Importance:** It is responsible for the core computations in the microprocessor.
2. **Registers:**
    - **Function:** Small, high-speed storage locations that hold data temporarily for processing.
    - **Types of Registers:**
        - **General-purpose registers**: Used to store data during processing.
        - **Special-purpose registers**: Includes program counter (PC), instruction register (IR), and status registers that help manage the control and flow of data.
3. **Timing and Control Unit:**
    - **Function:** Coordinates the operation of the microprocessor by generating clock signals and controlling the sequence of operations.
    - **Importance:** Ensures that the various components of the microprocessor work synchronously.

Difference between Microprocessor and Microcontroller:

Feature	Microprocessor	Microcontroller
Definition	A single-chip integrated circuit that contains the CPU.	A single-chip device that contains the CPU, memory, and I/O peripherals.

Feature	Microprocessor	Microcontroller
Components	Contains only the **CPU** (ALU, registers, control unit).	Contains **CPU, RAM, ROM, I/O ports**, and timers.
Purpose	Designed for **general-purpose computing** (e.g., PCs, servers).	Designed for **specific control applications** (e.g., embedded systems).
Memory	Requires external memory (RAM, ROM, etc.).	Has **on-chip memory** (RAM, ROM, EEPROM).
Cost	More expensive due to the complexity and external components.	Cheaper, as it integrates memory and peripherals in a single chip.
Applications	**Personal computers**, workstations, servers, complex calculations, and general-purpose systems.	**Embedded systems**, automation, appliances, robotics, IoT devices.

Microprocessor Working:

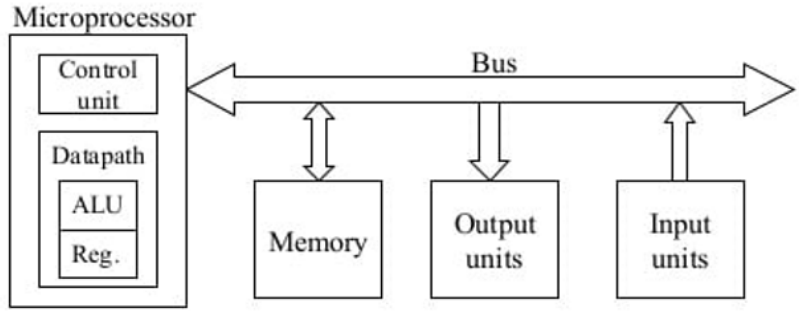
A microprocessor operates by fetching instructions from memory, decoding them, and executing the required operations. The control unit coordinates this process by managing the timing of all actions within the microprocessor.

1. **Fetch**: The instruction is retrieved from memory (RAM or ROM) and placed into the **instruction register**.
2. **Decode**: The instruction is decoded to understand what operation is needed (e.g., arithmetic or logical).
3. **Execute**: The ALU or control unit performs the operation specified by the instruction.
4. **Store**: The result of the operation is stored in registers or memory.

Microprocessor-Based System

A **microprocessor-based system** is a system that uses a microprocessor as the central unit to process data and control external devices. The microprocessor alone is not sufficient to perform real-world tasks, so additional external components are required to form a complete functional system. These external components include memory, input/output (I/O) ports, timers, and other peripherals.

Block Diagram of a Microprocessor-Based System:



Components of a Microprocessor-Based System:

1. **Microprocessor (CPU):**
   The core of the system that performs computations and executes instructions.
2. **Memory:**
   - **ROM:** Stores permanent program code.
   - **RAM:** Temporary storage for running programs.
3. **Input Devices:**
   Devices like sensors, keyboards, etc., that provide data to the microprocessor.
4. **Output Devices:**
   Devices like displays, printers, and actuators that output the processed data.
5. **I/O Ports:**
   Interfaces for data exchange between the microprocessor and external devices.
6. **Timers:**
   Manage time-based operations like delays and periodic interrupts.
7. **ADC/DAC:**
   ADC converts analog signals to digital, while **DAC** converts digital signals to analog.
8. **Communication Modules:**
   Enable data exchange through interfaces like UART, SPI, and I2C.

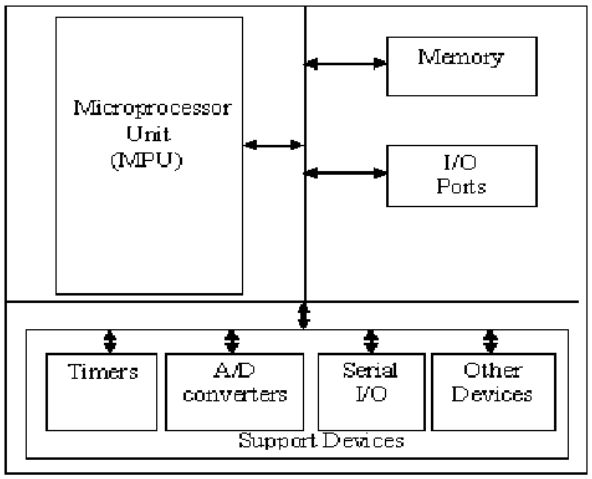
Applications of Microprocessor-Based Systems:

1. **Computers:**
   Serve as the central processing unit in personal computers and laptops.
2. **Automated Control Systems:**
   Used in robotics, manufacturing, and industrial automation.
3. **Consumer Electronics:**
   Found in devices like microwaves, washing machines, and digital cameras.

4. **Automotive Systems:**
   Control engine performance, airbag systems, and infotainment.
5. **Medical Devices:**
   Used in devices like pacemakers, diagnostic equipment, and monitoring systems.
6. **Communication Systems:**
   Enable mobile phones, networking devices, and satellite systems.
7. **Home Appliances:**
   Control smart appliances like thermostats, refrigerators, and vacuum cleaners.

Microcontroller

A **Microcontroller** is a compact, integrated circuit (IC) that processes data based on given instructions, much like a microprocessor. However, it requires fewer external devices since it has many components such as memory, input/output ports, timers, and counters built into the chip itself. Despite this, to perform specific tasks or operations, external devices may still be connected to form a microcontroller-based system.

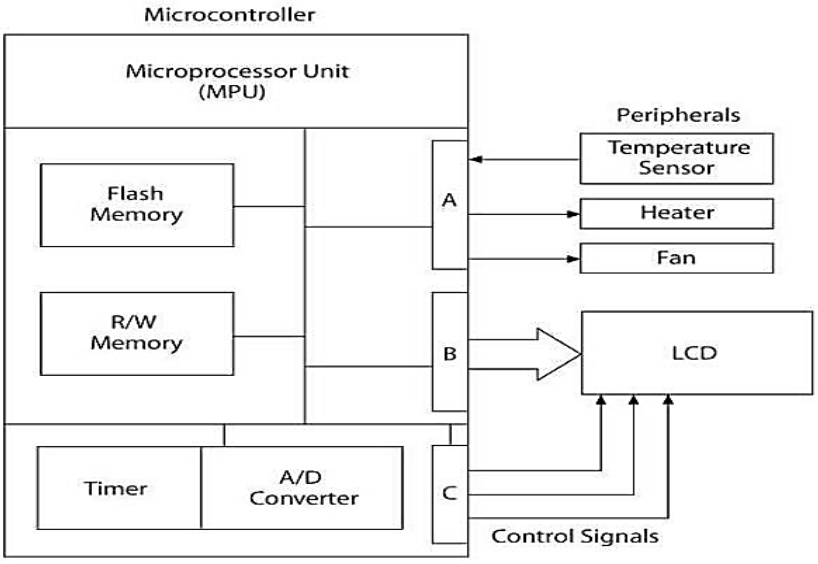


Microcontroller-based system

A **Microcontroller-based system** is created by connecting the microcontroller to external devices or peripherals, which allow it to perform specific tasks or operations.

While the microcontroller contains essential components like memory, I/O ports, timers, and counters internally, external devices such as sensors, actuators, displays, and communication interfaces are needed to complete the system's functionality.

These external components enable the microcontroller to interact with the environment and carry out its intended tasks.



Microprocessor Vs Microcontroller

Feature	Microprocessor	Microcontroller
Definition	It is a CPU that performs arithmetic, logic, and control operations in a computer system.	A single IC with a CPU, memory, and peripherals for specific tasks.
Consists	ALU, resisters, timing and control unit.	Along with microprocessor it consists of all types of memories, ADC, DAC, timers, counters etc.
Components	Requires external RAM, ROM, and I/O devices.	Built-in memory (RAM, ROM), I/O ports, and timers.
Functionality	Used for general-purpose processing.	Designed for dedicated tasks with minimal peripherals.
Complexity	More complex and versatile.	Simpler, optimized for specific applications.
Memory	External memory needed.	Inbuilt memory.
Cost	More expensive due to external components.	Cheaper with integrated components.
Power Consumption	Higher due to complexity.	Lower, optimized for specific tasks.
Applications	PCs, laptops, servers.	Embedded systems, robotics, automation.
Speed	Faster processing.	Slower processing.
Example	Intel Core i7, AMD Ryzen.	8051, Arduino, PIC.

What is Arduino?

Arduino is an **open-source electronics platform** based on **easy-to-use hardware and software**.

- It consists of a **microcontroller-based board** (hardware) and a **user-friendly IDE** (software).
- Designed for beginners and designers who want to use **physical computing** without deep electronics knowledge.

Key Features:

- Reads **analog/digital inputs** (e.g., from sensors) and converts them into outputs (e.g., motor, LED, cloud communication).
- Programs are uploaded via a **USB cable**, no separate programmer required.
- Uses **simplified C++** in the Arduino IDE.
- Standard board layout makes hardware functions easily accessible.

Advantages:

- Easy to use and program.
- Simple interfacing with real-world devices.
- Open-source hardware and software.
- Low cost and cross-platform compatibility.
- Large online support and active community.

ATmega328P – Overview

- **8-bit RISC microcontroller** used in **Arduino UNO**.
- Based on **Harvard Architecture** (separate memory for code and data).
- **RISC (Reduced Instruction Set Computing):** Fast execution with simple instructions.

Memory:

- **32 KB Flash Memory:** Stores code (non-volatile).
- **2 KB SRAM:** Temporary data storage (volatile).
- **1 KB EEPROM:** Retains data even without power (non-volatile).

Key Features:

- **High speed:** Up to **20 MIPS** at **20 MHz**.
- **131 instructions**, mostly single-cycle.
- **On-chip multiplier** for faster arithmetic.
- **Flash life:** 10,000 cycles; **EEPROM:** 100,000 cycles.
- **Data retention:** Up to 100 years at 25°C.

Peripherals:

- **Timers:** Two 8-bit, one 16-bit timer.
- **PWM Channels:** 6 for motor/signal control.
- **ADC:** 10-bit resolution, up to 8 channels.
- **Serial Interfaces:** SPI, I2C, USART.
- **Watchdog timer**, analog comparator.
- **Interrupts and sleep modes** for power saving.

Power & Package:

- **Operating Voltage:** 1.8V – 5.5V
- **Low power consumption:** 0.2 mA in active mode.
- **Package:** 28-pin DIP, 32-pin TQFP, etc.
- **Temperature range:** –40°C to +105°C

GPIO (General Purpose Input/Output):

- GPIO is a **software-controlled interface** found in **microcontrollers, SoCs, and other ICs**.
- These are **programmable pins** with no fixed function, used to connect and control **external devices**.
- Used in microcontrollers, DSPs, FPGAs, PMICs, LED controllers, etc.
- **GPIO Expanders** (via I2C/SPI) increase available GPIOs.

Basic Capabilities:

- Pins can be **enabled/disabled**.
- Configurable as **input or output**.
- Output: Set to **HIGH (1)** or **LOW (0)**.
- Input: Can read values **(HIGH/LOW)**.
- Can act as **interrupt signals** (edge/level triggered).

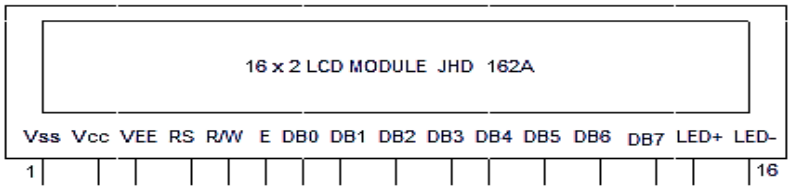
Timers in ATmega328P:

- ATmega328P has **3 timers**: **Timer0**, **Timer1**, and **Timer2**.
- **Timer0 & Timer2** are **8-bit** timers (can count from 0 to 255).
- **Timer1** is a **16-bit** timer (can count from 0 to 65535).
- Timers are used for tasks like **delays, PWM, and event counting**.

Interfacing 16×2 LCD with Arduino UNO

A **16×2 LCD** can display 2 lines with 16 characters each. It's commonly used with Arduino for displaying messages or data.

Key Pins and Connections:



- **Vss (Pin 1):** GND
- **Vcc (Pin 2):** +5V
- **VEE (Pin 3):** Contrast control (via 10K potentiometer)
- **RS (Pin 4):** Register select → Arduino pin 12
- **R/W (Pin 5):** Set to GND (write mode)
- **E (Pin 6):** Enable → Arduino pin 11
- **DB4–DB7 (Pins 11–14):** Data lines → Arduino pins 5, 4, 3, 2
- **LED+ (Pin 15):** Backlight +5V via 560Ω resistor
- **LED- (Pin 16):** GND

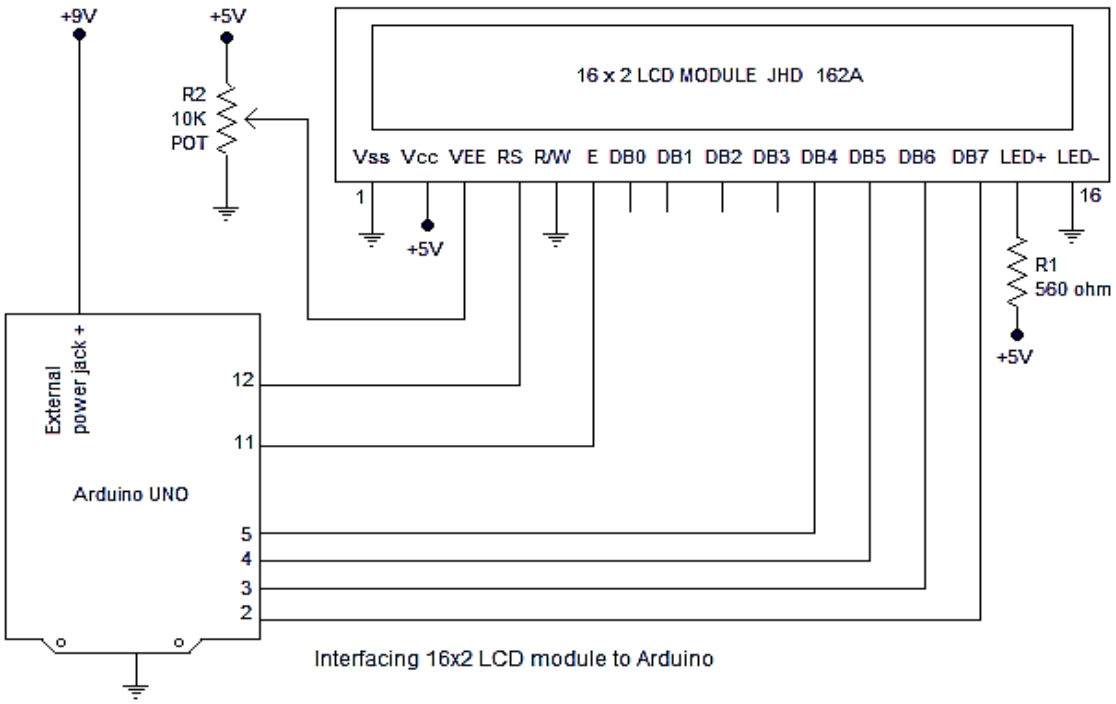
Why 4-bit Mode?

- Saves pins by sending data in 2 parts (nibbles).
- Uses only DB4–DB7.

Power:

- LCD gets +5V from Arduino.
- Arduino powered via USB or external supply.

Circuit diagram – Arduino to 16×2 LCD Module



Interfacing 16x2 LCD module to Arduino

3×3 Keypad Interfacing with Arduino

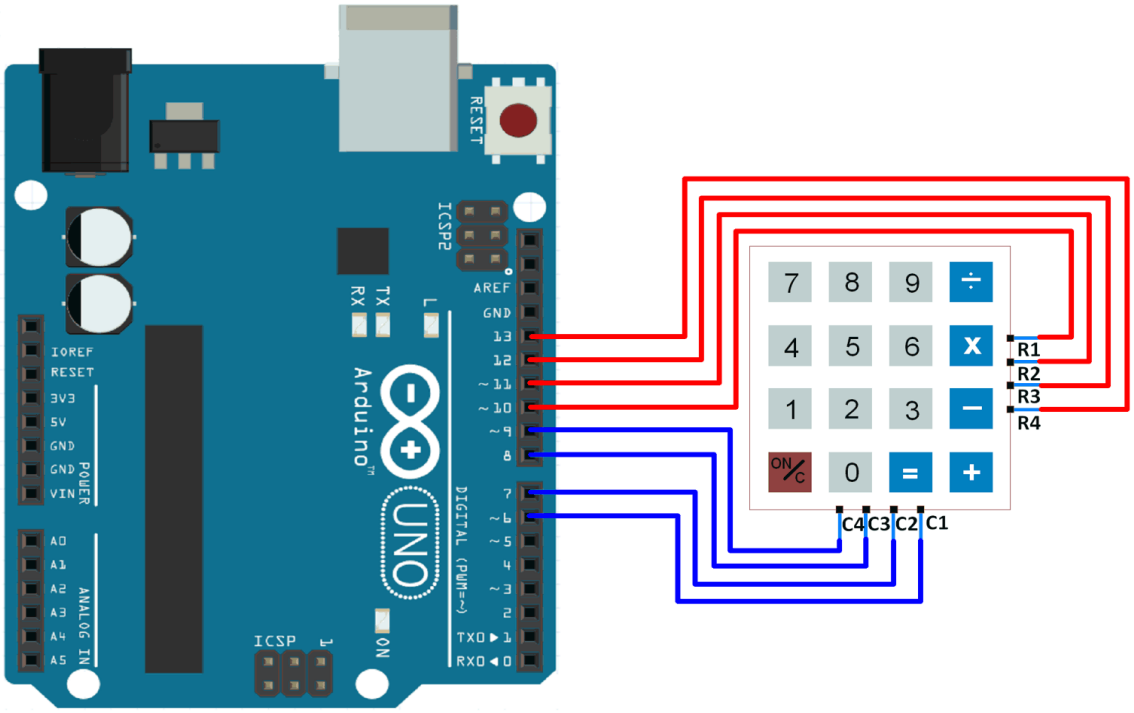
A **3×3 keypad** is used to take input from the user. It has **3 rows and 3 columns**, forming **9 keys**.

How It Works:

- Each key is placed at the intersection of a row and a column.
- When a key is pressed, it connects the corresponding row and column.
- Arduino detects which row and column are connected to identify the pressed key.

Wiring:

- The keypad has **9 pins** (R1, R2, R3, C1, C2, C3).
- These are connected to **9 digital I/O pins** on the Arduino.
- A **Keypad library** is used in Arduino to scan the keys and read input easily.


RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TXD ▶ 1
RXD ◀ 0
DIGITAL (PWM=~)
TX
RX
L
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
Arduino™
UNO
7 8 9 ÷
4 5 6 x
1 2 3 −
ON C 0 = +
R1
R2
R3
R4
C4 C3 C2 C1

Assignment

1. What is ATmega?

ATmega is a family of **8-bit microcontrollers**. These microcontrollers are based on the **AVR architecture** and are used extensively in **embedded systems**. They are known for their **simplicity**, **low power consumption**, and **ease of use**, making them a popular choice for projects involving sensors, motors, and other peripherals. The ATmega328P is one of the most popular microcontrollers in this series and is used in various applications, particularly in Arduino boards like the **Arduino Uno**.

---

2. What is the difference between Microprocessor and Microcontroller?

Feature	**Microprocessor**	**Microcontroller**
Definition	It is a CPU that performs arithmetic, logic, and control operations in a computer system.	A single IC with a CPU, memory, and peripherals for specific tasks.
Consists	ALU, resisters, timing and control unit.	Along with microprocessor it consists of all types of memories, ADC, DAC, timers, counters etc.
Components	Requires external RAM, ROM, and I/O devices.	Built-in memory (RAM, ROM), I/O ports, and timers.
Functionality	Used for general-purpose processing.	Designed for dedicated tasks with minimal peripherals.
Complexity	More complex and versatile.	Simpler, optimized for specific applications.
Memory	External memory needed.	Inbuilt memory.
Cost	More expensive	Cheaper with integrated components.
Power Consumption	Higher due to complexity.	Lower, optimized for specific tasks.
Applications	PCs, laptops, servers.	Embedded systems, robotics, automation, IOT, etc.
Speed	Faster processing.	Slower processing.
Example	Intel Core i7, AMD Ryzen.	8051, Arduino, PIC, raspberry pi

3. What are the major features of the ATmega328P microcontroller?

The ATmega328P microcontroller is packed with features that make it suitable for a wide range of embedded applications:

- **Architecture**: 8-bit AVR RISC (Reduced Instruction Set Computing) architecture, providing fast execution and efficiency.
- **Memory**:
    - 32 KB Flash memory for program storage.
    - 2 KB SRAM for temporary data storage during operation.
    - 1 KB EEPROM for non-volatile data storage.
- **I/O Pins**: 23 programmable I/O pins (used for connecting peripherals like sensors, motors, displays).
- **Timers**: Three timers – two 8-bit timers and one 16-bit timer, used for generating time delays or controlling the timing of operations.
- **PWM Channels**: Six Pulse Width Modulation (PWM) channels to control motor speed or dim LEDs.
- **Analog-to-Digital Converter (ADC)**: A 10-bit ADC with 6 input channels, allowing it to read analog sensors.
- **Communication Interfaces**: Supports USART (for serial communication), SPI (for peripheral devices), and I2C (for sensors and displays).
- **Operating Voltage**: Can operate between 1.8V and 5.5V, providing flexibility for different power sources.
- **Clock Speed**: Up to 20 MHz, allowing for faster processing.
- **Low Power Modes**: Multiple sleep modes to reduce power consumption in battery-operated devices.
- **Watchdog Timer**: For ensuring the device doesn't freeze or enter an infinite loop.
- **Brown-out Detection**: Ensures the microcontroller operates correctly under fluctuating voltage levels.

These features make the ATmega328P an ideal choice for low-power, small embedded systems that need to control devices like motors, sensors, and displays.

---

4. Explain how features of the ATmega328P support embedded system applications.

The ATmega328P microcontroller is specifically designed for embedded system applications due to its compactness, flexibility, and power efficiency. Here’s how its features support such applications:

- **Low Power Consumption**: The ability to operate at low voltages (1.8V – 5.5V) and have low power sleep modes makes it ideal for battery-powered applications, such as remote sensors or IoT devices.
- **Multiple I/O Pins**: The 23 I/O pins can be configured as either input or output, which allows interfacing with various sensors, actuators, and displays.
- **Timers and PWM**: These features are essential for precise timing applications such as controlling the speed of motors, generating audio signals, or creating time delays.
- **Communication Protocols**: With built-in support for USART, SPI, and I2C, the ATmega328P can easily communicate with other devices, sensors, and peripherals. This is crucial for applications such as data collection, remote control, or communication with other devices in a system.
- **Analog-to-Digital Conversion (ADC)**: The 10-bit ADC can read analog signals from sensors, such as temperature, light, or sound sensors, which is critical for applications like environmental monitoring or health devices.
- **Memory Options**: The large flash memory and EEPROM provide storage for both code and data, essential for applications that require data logging or complex control logic.
- **Watchdog and Brown-out Detection**: These ensure system reliability by resetting the microcontroller in case of a malfunction or low voltage, preventing the system from entering an invalid state.
- **Small Form Factor**: The ATmega328P comes in small packages, making it suitable for space-constrained embedded applications.

---

5. Draw and explain the architecture of the ATmega328P

Here's a basic explanation of the architecture (I can also provide a diagram if needed):

- **CPU (8-bit RISC)**: Executes instructions and controls all operations.
- **Program Memory (Flash)**: 32 KB of flash memory to store program code.
- **Data Memory (SRAM)**: 2 KB of SRAM for temporary data storage during operations.
- **EEPROM**: 1 KB of EEPROM to store non-volatile data, such as user settings or logs.
- **Timers/Counters**: Three timers control time-sensitive operations such as PWM for motor control.
- **ADC (Analog-to-Digital Converter)**: 10-bit ADC with 6 input channels to convert analog signals to digital for processing.
- **I/O Ports (Port B, C, D)**: 23 programmable I/O pins for interfacing with external devices.

- **Communication Modules**: USART, SPI, and I2C for serial communication with external devices.

---

6. Explain how ATmega328P supports GPIO function used in motor control.

General Purpose Input/Output (GPIO) pins on the ATmega328P can be used for motor control in the following ways:

- **Digital Output**: Some GPIO pins are configured as digital outputs to drive devices like relays or transistors that control the power to motors.
- **PWM Output**: The ATmega328P has built-in PWM channels (using timers) to vary the voltage supplied to motors. This allows precise speed control.
- **Direction Control**: GPIO pins can be used to control the direction of a motor by switching between different motor driver configurations (e.g., H-Bridge control).
- **Feedback from Motor**: Input pins can read the status of sensors (e.g., encoders) to monitor motor position or speed.

GPIOs on the ATmega328P provide flexibility and allow for efficient motor control in embedded systems.

---

7. Explain the configuration of GPIO pins of the ATmega328P for interfacing peripheral devices like LCDs in a motor control application.

In a motor control application, an LCD (such as a 16x2 LCD) can be interfaced to display motor speed, direction, or status. Here's how the GPIO pins can be configured:

- **Data Pins**: The LCD's data pins (DB4 to DB7) are connected to digital GPIO pins. The data pins are used to send commands or data (such as motor status or speed) to the LCD.
- **RS (Register Select)**: This pin is connected to a GPIO pin and is used to toggle between the data register and the command register.
- **E (Enable)**: This pin is connected to a GPIO pin and triggers the LCD to process the data sent via the data pins.
- **Pin Configuration**: GPIO pins are configured as output pins using the DDRx registers. Commands and data are written to the LCD via the PORTx registers.
- **Contrast Control**: A potentiometer connected to the VEE pin of the LCD is used to adjust the contrast of the display for better visibility.

---

8. Explain the configuration of GPIO pins of the ATmega328P for interfacing peripheral devices like Keypad in a motor control application.

In a motor control system, a 3x3 keypad can be used for user input to control motor speed or direction. Here's how GPIO pins are configured:

- **Rows as Outputs**: The rows of the keypad are connected to GPIO pins configured as outputs. These pins send signals to the keypad.
- **Columns as Inputs**: The columns are connected to GPIO pins configured as inputs with internal pull-up resistors enabled. When a key is pressed, it connects a row to a column.
- **Key Scanning**: The microcontroller scans the rows and detects which column is pressed. This allows it to identify which key is pressed and trigger specific actions like changing motor speed or direction.

---

9. Explain the role of GPIO pins in controlling devices using an Arduino board.

In an Arduino-based system, GPIO pins play a crucial role in interfacing with external devices:

- **Digital Pins**: Used for turning devices on or off (e.g., controlling LEDs, motors, relays) using digitalWrite() functions.
- **Analog Pins**: Used for reading analog sensor values (e.g., temperature, light levels) via analogRead() functions.
- **PWM Pins**: Used for controlling the brightness of LEDs, motor speed, or tone generation through analogWrite() functions.
- **Interfacing with Peripherals**: GPIOs can communicate with devices like LCDs, keypads, and sensors to gather input or display output.
- **Control Logic**: GPIO pins allow the Arduino to implement control logic for systems like motor controllers, alarm systems, or user interfaces.