Set B

45

April - 24

Page no - 17

Scoring Indicators

COURSE NAME : EMBEDDED SYSTEMS

COURSE CODE : 6201B QID : 2102240182

Q No	Scoring Indicators	Split score	Sub Total	Total score
	PART A			9
I. 1	Washing machine,printer,scanner,digital camera etc(any two)	0.5*2	1	
I. 2	Actuators: It is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion). Actuator acts as an output device.		1	
I. 3	Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS etc. (wireless) and RS-232, USB, Parallel port etc. (wired) (any two)	0.5*2	1	
I. 4	Program Counter: The PC stores a program memory address that contains the location of the next instruction.		1	
I. 5	PORTx, DDRx & PINx.(any two)	0.5*2	1	
I. 6	Real Time clock		1	
I. 7	PORT A		1	
I. 8	Kernel: A kernel is a central component of an operating system that acts as an interface between the user applications and the hardware. It contains a set of system libraries & services		1	
I. 9	task		1	
	PART B			24
II. 1	. The main components of software involves editor, compiler, assembler, linker, debugger and emulator. • The code you write in C, Java, Python programming languages will be saved in a text file in the editor.		3	

	• A compiler is used to translate high level language program into machine code. • An assembler is used to translate an assembly language program into machine code. • A debugger is a tool used for testing and debugging purposes. It scans the code thoroughly and removes the errors and bugs, and identifies the places where they occur. • A linker, also called a link editor, is a tool that takes one or more object files and combines them to develop a single executable code. • The main task of the emulator is to make the embedded system act like a real system in a simulation environment. .			
II. 2	Onboard Communication Interface: The communication interface channel which interconnects the various components within an embedded system. Eg: Serial interfaces like I2C, SPI, UART etc	Expl(2) +eg(0.5*2)	3	
II. 3	Types of Memory: RAM(Random Access Memory) or Temporary memory: It is the data or working memory of the controller/processor. If the power is turned off then its contents will be lost forever. It is a volatile momory. It is of 2 types: • SRAM (Static RAM): • DRAM (Dynamic RAM): ROM(Read Only Memory) or Fixed memory: It is the code or program memory of the controller/processor. It retains the content even the power applied to the chip turns off. It is a non - volatile momory. It is of 3 types: • Masked ROM: It is an one time programmable memory. These are hardwired memory devices found on system & are factory programmed. It contains pre-programmed set of instruction and data and it cannot be modified by end user. • PROM (PROGRAMMABLE ROM): It is also an one time programmable memory. This memory device comes in an un-programmed state i.e. at the time	1.5*2	3	

	of purchased it is in an unprogrammed state and it allows the user to write his/her own program or code into this ROM. • EPROM (ERASABLE-AND-PROGRAMABLE ROM. The erase operation in case of an EPROM is performed by exposing the chip to a source of ultraviolet light, but is a time consuming process. • EEPROM(Electrically Erasable and Programmable ROM). It is same as EPROM, but the erase operation is performed electrically. The erasing process is faster compared to EPROM. • Flash: Flash memory is the most recent advancement in memory technology. Flash memory devices are high density, low cost, nonvolatile, fast (to read, but not to write), and electrically reprogrammable.			
II. 4	AVR family: AVRs are generally classified into 4 broad groups: • Classic AVR (AT90SXXXX):This is the original AVR chip which has been replaced by newer chips. Some members of this family are – AT90S2313, AT90S2323, AT90S4433. • Mega AVR (ATMegaXXXX): These are powerful micro controllers with more than 120 instructions and lots of different peripheral capabilities. Some members of this family are – ATMega8, ATMega16, ATMega32, ATMega64, ATMega128. • Tiny AVR (ATTiny XXXX): Micro controller in this group have less instructions and smaller packages when compared to mega family. Some members of this family are – ATTiny13, ATTiny25, ATTiny44, ATTiny84. • Special Purpose AVR: They are made with special capabilities for specific applications. Examples for special capabilities are USB controller, CAN controller, LCD controller, Zigbee, Ethernet controller, FPGA and advanced PWM. Some members of this family are – AT90CAN128, AT90USB128, AT90PWM128.(any three)	3*1	3	
II. 5	Data Types: In C programming, data types are declarations for variables. This determines the type and size of data associated with variables. 1) Unsigned char: It is an 8-bit data type that takes the value in the range of 0-255(00-FFH).		3	

	2) Signed char: It is an 8-bit data type that uses the Most Significant Bit(MSB) i.e D7 bit to represent +ve or -ve value. As a result only 7 bits are available for use, giving the rang -128 to +127. 3) Unsigned int: It is an 16-bit data type that takes the value in the range of 0-65,535(0000- FFFFH). . 4) Signed int: It is an 16-bit data type that uses the Most Significant Bit(MSB) i.e D15 bit to represent +ve or -ve value. As a result only 15 bits are available for use, giving the rang -32,768 to +32,767. 5) Other data types: • If we want values greater than 16-bit use long data types. • If we want deal with fractional numbers use float & double data types			
II. 6	`#include <avr/io.h>` `int main(void)` `{` `  DDRB = 0xFF;  //port b is output` `  DDRC = 0x00;  //port c is input` `  PORTB = 0x00;` `  while(1)` `  {` `  if (PINC & (1<<2))` `  {` `  PORTB = 0x00;` `  }` `  else` `  {` `  PORTB = 0xFF;` `  }` `  }` `  return 0;` `}`		3	
II. 7	`#include <avr/io.h>` `int main(void)` `{` `  DDRD = 0xFF;  //port D output` `  DDRA = 0x00;  //port A input, LM 34 connected to ADC0` `  ADCSRA = 0x87;  //ADC enable, ck/128` `  ADMUX = 0xE0;  //2.56 V ref, ADC0 single ended left justified data` `  while(1)` `  {` `  ADCSRA |= (1<<ADSC);  //start conversion` `  while((ADCSRA & (1<<ADIF))==0); //wait for EOC flag ADIF` `  PORTD = ADCH;  //high byte to port B` `  }` `  return 0;` `}`		3	

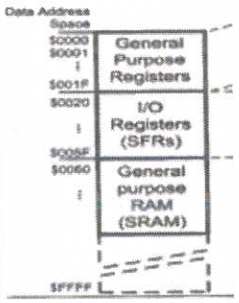
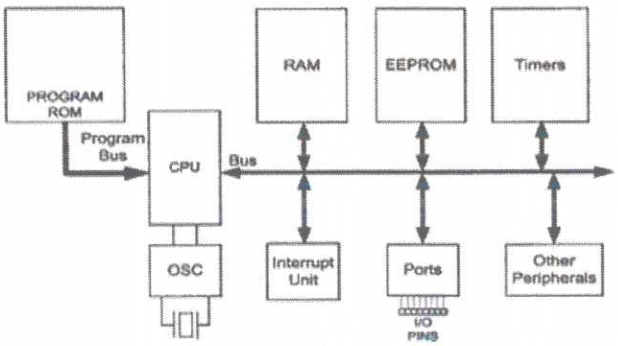
II. 8	`#define F_CPU 1000000UL //1 MHz clock` `#include <avr/io.h>` `#include <util/delay.h>` `int main(void)` `{` `DDRB = 0xFF; //port B configured as output port` `while(1)` `{` `PORTB = 0x01; //PORTB Pin0 made high, Pin1 made low` `_delay_ms(1000); //call delay` `PORTB = 0x02; //PORTB pin0 made low, pin1 made high` `_delay_ms(1000); //call delay` `}` `return 0;` `}`		3	
II.9	List popular Real Time Operating Systems(RTOS): 1. FreeRTOS (Amazon) 2. Zephyr (Linux Foundation) 3. MQX (Philips NXP / Freescale) 4. Keil RTX (ARM) 5. µC/OS (Micrium) 6. LynxOS (Lynx Software Technologies) 7. Integrity (Green Hills Software) 8. embOS (SEGGER) 9. ThreadX (Microsoft Express Logic) 10. Neutrino (BlackBerry(any six)	6*0.5	3	
II.10	The basic functions of a Real-Time kernel are listed below: • Task/Process management: Deals with setting up the memory space for the task, loading task code to memory, allocating system resources, setting up Task Control Block(TCB), termination of a task. TCB is used for holding the information corresponding to a task. • Task/Process scheduling: Deals with sharing the CPU among the various task. A kernel application called 'Scheduler' handles the task scheduling. Scheduler is actually an algorithm implementation which perform the efficient & optimal scheduling of tasks. • Task/Process synchronisation: It enables the tasks to mutually share the resources like buffers, I/O devices etc.	3*1	3	

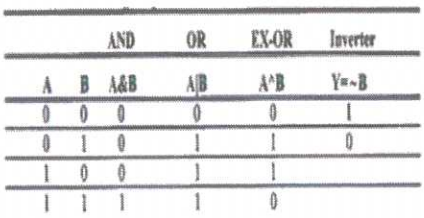
	• Error/Exception handling: Deals with handling errors/ exceptions raised during the execution of tasks like insufficient memory, timeouts, bus error, divide by zero etc.. • Memory management: RTOS uses a block based memory allocation technique, instead of a dynamic memory allocation in GPOS. RTOS kernel uses memory blocks of fixed size & is allocated for a task on a need basis. These blocks are stored in ' Free Buffer Queue'. • Interrupt handling: Deals with handling various types of interrupts. It can be: • Synchronous interrupt: They occur in sync with currently executing task. Usually software interrupt fall into this category. • Asynchronous interrupt: They occur not in sync with currently executing task & are usually generated by external devices. • Time management: Deals with accurate time management of RTOS. It is provided with a high resolution Real-Time Clock(RTC) hardware chip & is programmed to interrupt the processor at a fixed rate. This timer interrupt is called 'Timer Tick'(any three)			
	PART C			42
III. 1		7*1	7	7

Paramater	General Purpose Computer	Embedded System
Basic	It is a general purpose electronic device used to perform different types of tasks.	It is a specialized computer system that used to perform one or a few specific tasks.
System hardware	A computer typically consists of a CPU, storage unit, and I/O units.	Embedded system are designed with a microcontroller which consists of a CPU, memory unit, and I/O interface on a single IC chip.
Processing power	High processing power.	Relatively low processing power.
Storage capacity	High storage capacity or memory to store data and information on the system.	Less memory capacity as compared to computers.
Size	Generally larger in size.	Smaller in size than computers.
Cost	More expensive than embedded systems.	Less expensive.
Operating system	Computers use a full-featured operating system to run.	Embedded systems use a specialized operating system to run.
Software development tools	For computers, the general purpose development tools can be used to develop computer software.	The development of software for embedded systems requires specialized and expert tools.
Maintenance & updates	Computers need regular maintenance and updates.	Embedded systems do not require much maintenance and updates.

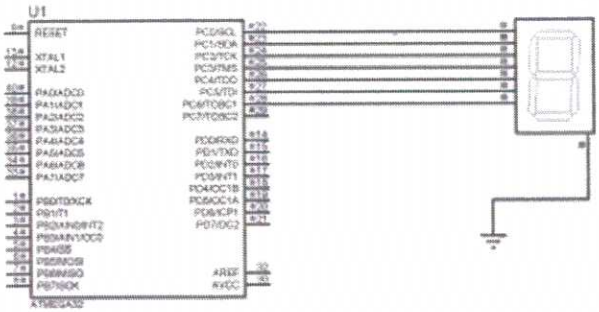
III. 2	Communication Interface: They are essential for communicating with various subsystems of the embedded system and with the external world. The communication interface can be of two types: • Onboard Communication Interface: The communication interface channel which interconnects the various components within an embedded system. Eg: Serial interfaces like I2C, SPI, UART. • External Communication Interface: It is responsible for data transfer between the embedded system and other devices or modules. The external communication interface can be either wired media or wireless media and it can be a serial or parallel interface. Eg: Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), RS-232, USB etc Universal Asynchronous Receiver Transmitter (UART): It is an asynchronous form of serial data transmission.UART uses start and stop bits with actual data bits, which defines the start and end of data packet. UART works under full duplex communication mode. Inter-integrated-circuit (I2C): I2C is a serial communication interface developed by Philips Semiconductors. I2C interface is a master to slave communication interface Serial Peripheral Interface (SPI): It is one of the serial communication interface developed by Motorola. It is a 4-wire protocol namely MOSI (Master Out Slave In), MISO (Master In Slave Out, SS (Slave Select), and SCLK (Serial Clock). MOSI and MISO are the data lines. SPI is a full duplex communication interface. It is a synchronous form of serial data transmission RS232 (Recommended Standard 232): It is a full duplex, wired, asynchronus mode serial communication interface. It extend the UART communication signal for external data communication. Universal Serial Bus (USB): It is a common interface that allows the connection between external devices and controllers. USB protocol sends and receives the data serially between host and external peripheral devices through data signal lines D+	3.5*2	7	7

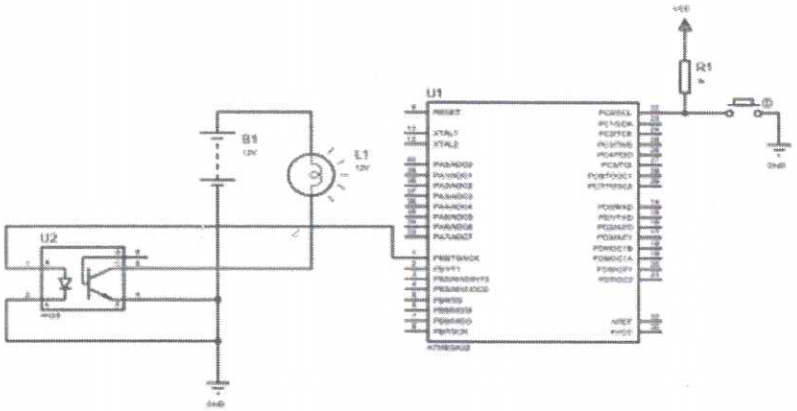
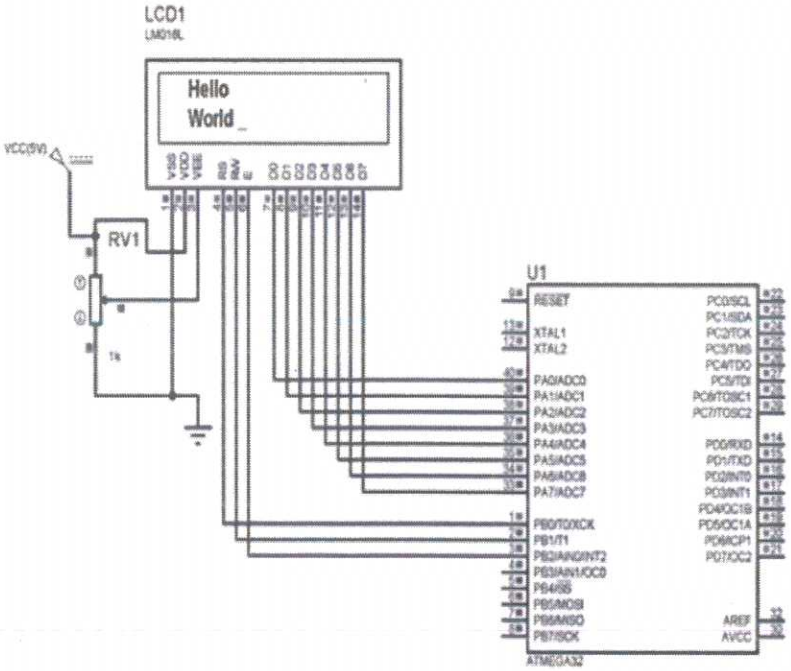
	and D-. Apart from two data lines, USB has VCC and Ground signals to power up the device. Bluetooth: it is a Low-cost short-distance radio communications standard. Bluetooth is wireles and replaces cable technology. It uses radio waves for transmitting and receiving data. It operates at 2.4GHz of radio spectrum. Wireless Fidelity(Wi-Fi): It is the popular wireless communication technique for networked communication of devices. Wi-Fi operates at 2.4GHz or 5GHz of radio spectrum. Wi-Fi supports Internet Protocol (IP) based communication.			
III. 3	AVR data Memory: Data memory store data . this is composed of 3 parts: General Purpose Registers , I/O memory and internal data SRAM. General Purpose Registers: • In ATmega32 there are 32 general purpose registers (R0-R31). • All registers are 8-bit. • Located in lowest location of memory from $00 to $1F. • General purpose registers are used by all arithmetic and logical operations, to store information temporarily. • Each registers can perform function of accumulator in other microcontrollers. I/O Memory or Specific Function Registers(SFRs): • It is dedicated to special functions such as timer, serial communication, I/O port, ADC etc. • AVR I/O memory is made of 8-bit registers. • The function of each memory location is fixed by CPU designer at time of design because it is used for control of micro controller or peripherals . • All of the AVR have at least 64Bytes of I/O memory . This section is called standard I/O memory. • Address of standard I/O memory is $0020 to $005F. • In AVRs with more than 32 pins there is also an extended I/O memory, which contains registers for controlling extra ports and peripherals. Internal data SRAM: • It is used for storing data and parameters by AVR programmers.	Fig(3) +expl(4)	7	7

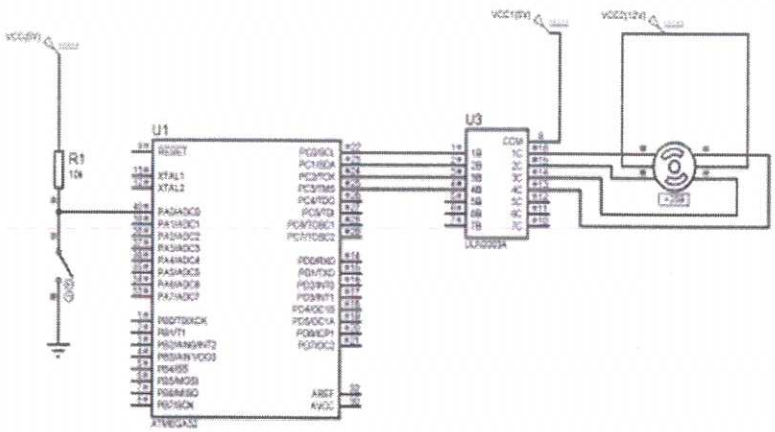
	• It is also called as scratch pad. • Each location is 8 bit and can be directly accessed by its address. • Size of SRAM can vary from chip to chip. In ATmega32 SRAM is 2KB size having address $0060 to $085F. 			
III. 4	 I/O ports: AT Mega32 has four ports(Port A,Port B, port C and Port D) having 32 pins. Oscillator: AT Mega32 has an internal oscillator for its clock. By default it is set to operate an internal caliberated oscillator of 1 MHz. Timer/counter: It is used to count an event or to generate time delays between two operations. AT Mega32 consist of two 8 bit(Timer0 & Timer2) and one 16 bit timer/counter(Timer1). Watchdog timer: A watchdog timer is a simple countdown timer which is used to reset a microprocessor after a specific interval of time. Interrupt unit: AVR ATmega32 consist of 21 interrupt sources out of which three are external. Memory: AT Mega32 consist of three different memory sections. • Program ROM: It is used to store program or codes. AT Mega32 has 32KB as Program ROM	Fig(4) +expl(3)	7	7

	• EEPROM: It is also a non volatile memory used to store data. AT Mega32 has 1 KB of EEPROM. • Data RAM: It is a volatile memory used to store data. AT Mega32 has 2 KB of Data RAM. Other Pheripherals: It includes: ADC interface: USART(universal synchronous asynchronous receiver transmitter• SPI(Serial Peripheral Interface)etc..			
III. 5	**Logical Operations(Bit-wise):** It includes **AND(&), OR(|), EX-OR(^), inverter(~), shift right(>>), shift left(<<).**  AND OR EX-OR Inverter A B A&B A|B A^B Y=~B 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 Eg 1 AND portB DDRB=0XFF; PORTB=0x35 & 0x0F; 35H 00110101 &0FH 00001111 05H 00000101 Eg 2 OR portC DDRC=0XFF; PORTC=0x35 | 0x0F; 35H 00110101 | 0FH 00001111 3FH 00111111 Eg 3 EX-OR portD DDRD=0XFF; PORTD=0x35 ^ 0x0F; 35H 00110101 ^0FH 00001111 3AH 00111010 Eg 4 Inverting portB DDRB=0xFF; PORTB= ~0x35 35H 00110101 ~CAH 11001010		7	7
III. 6	Interrupts in Atmega32: When ever any device want to communicate, the device notifies it by sending an interrupt signal. Upon receiving an interrupt signal, the microcontroller stops the task and respond to the device. Sources of Interrupts in the AVR: AVR ATmega32 consist of 21 interrupt sources out of which three are external. The remaining are internal interrupts which supports the peripherals like USART, ADC, timer etc. There are many sources of interrupts in the AVR, depending on which peripheral is connected to the chip. Some of the important interrupt sources are, • Two interrupts for each timer. • Three interrupts for external hardware which are INT0, INT1, and INT2 respectively. The three external hardware interrupts are on pins PD2, PD3, and PB2		7	7

	• Serial communication has 3 interrupts. • ADC interrupts Interrupt Service Routine(ISR): The program associated with the interrupt is called interrupt service routine. Generally for every interrupt there is a fixed location in memory that holds the address of ISR. This group of memory location set aside for handling ISR is called interrupt vector table. Enabling and Disabling an interrupt: Upon reset, all interrupts are disabled. The interrupts must be enabled by software for the microcontroller to respond to them. The D7 bit of the SREG (Status Register) is used to enable or disable interrupts globally. Steps in enabling an Interrupts : • Bit D7 (I – flag) of SREG register must be set to HIGH to enable all interrupts. • This is done with the SEI instruction. • If I=1, each interrupts is enabled by setting to HIGH the IE(Interrupt Enable) flag bit for that interrupts. The TIMSK(Timer Interrupt Mask) register is used to enable and disable different timer interrupts. These hardware interrupts are enabled by setting the 3 bits in the GICR (General Interrupt Control Register). Enabling Timer Interrupts: The timer interrupts are handled using the TOV(Timer Overflow Flag) in the TIFR register and the TOIE(Timer Overflow Interrupt Enable) bit in the TIMSK register. We must enable both these interrupts flags to handle timer interrupts. Also we must enable or set the global interrupt flag(I) in the status register. Enabling External Hardware Interrupts: The ATMega32 has 3 external interrupts INT0 (PORTD.2), INT1 (PORTD.3) and INT2 (PORTB.2). These hardware interrupts are enabled by setting the 3 bits in the GICR (General Interrupt Control Register). These along with global interrupt flag(I) is set for an interrupt to be responded.			

III. 7	 #include<avr/io.h> #include<util/delay.h> unsigned char seg_7[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F}; // initializing an array to store the hex value to be displayed int main(void) { DDRC = 0xFF; //port C is output PORTC = 0x00; int i= 0; while(1) { PORTC = seg_7[i]; _delay_ms(1000); i = i+1; if(i>=10) { i=0; } } return 0; }	Fig(4) +code(3)	7	7
III. 8	``` #include <avr/io.h> int main(void) { DDRB = 0xFF; //port b is output DDRC = 0x00; //port c is input PORTB = 0x00; while(1) { if (PINC & (1<<2)) { PORTB = 0x00; } else { PORTB = 0xFF; } } return 0; } ```	Fig(4) +code(3)	7	7

				
III. 9	 • LCDs are used for displaying status or parameters in embedded systems. • LCD 16x2 is a 16-pin device that has 8 data pins (D0-D7) and 3 control pins (RS, RW, EN). The remaining 5 pins(VSS, VDD, VEE, LEDA, LEDK) are for the supply and backlight for the LCD. • The control pins help us configure the LCD in command mode or data mode. They also help configure read mode or write mode and also when to read or write. • The port pins to which LCD is connected are configured as an output pins.	Fig(4) +expl(3)	7	7

	Initialize LCD: • Power ON the LCD • Wait for 15 ms ('Power ON' initialization time for LCD) • Send 0x38 command to initialize 2 lines, 5x8 matrix, 8-bit mode LCD • Send any 'Display ON' command (0x0E, 0x0C) to LCD • Send 0x06 command (increment cursor) to LCD Command Write function: • Send the command value to the LCD16x2 data port. • Make RS pin low, RS = 0 (command reg.) • Make RW pin low, RW = 0 (write operation) • Give high to low pulse at the Enable (E) pin of a minimum delay of 450 ns. Data write function: • Send command to the data port. • Make the RS pin High, RS = 1 (Data reg.) • Make the RW pin Low, RW = 0 (Write operation) • Give high to low pulse at the Enable (E) pin Display String function • This function takes a string (an array of characters) and sends one character at a time to the LCD data function till the end of the string. A 'for loop' is used for sending a character in each iteration. A NULL character indicates end of the string			
III. 10	 Interfacing of Stepper Motor: • It is a widely used device to transmit electrical pulses into mechanical movements. • Applications: Disk drives, Dot matrix printers, Robotics • Stepper motor is a brushless DC motor that divides the full rotation angle of 360° into a number of equal steps.	Fig(4) +expl(3)	7	7

	• The motor is rotated by applying a certain sequence of control signals. The speed of rotation can be changed by changing the rate at which the control signals are applied. • Stepper motor commonly have rotating permanent magnetic part called rotor(or shaft) surrounded by a stationary electromagnetic winding part called stator• As the current flow through the stator winding, it get energized & causes the rotor magnet to deflect or rotates. • The direction of rotation is determined by the current flowing through the stator windings. • Stepper motors can be driven in two different patterns or sequences. These are: ▪ Full Step Sequence: In the full step sequence, two coils are energized at the same time and motor shaft rotates. ▪ Half Step Sequence: In Half mode step sequence, motor step angle reduces to half the angle in full mode. • A Unipolar Stepper Motor is interfaced to ATmega32 through ULN2003A driver IC, because the AVR lacks sufficient current to drive the stepper motor windings. • A microcontroller can be used to apply different control signals to the motor to make it rotate according to the need of the application. • The port to which the stepper motor driver is connected has to be configured as an output port.			
III. 11	• Micro C/OS-II is a simple, easy to use real time kernel written in C language of embedded application development . • It is a commercial real-time kernel from Micrium Inc. • It is available for different family of processors/controllers ranging from 8 bit to 64 bit like ARM family of processors, Intel 8085/x86/8051, Freescale 64K series etc. • It features: ▪ Multitasking: It is capable of executing multiple application simultaneously without slowing down the system.	4+3	7	7

	▪ Priority based pre-emptive task scheduling: It chooses the process with the highest priority and runs it until it completes or is stopped by a process with a higher priority. µC/OS-II Applications: It is used in many embedded systems, including ▪ Avionics ▪ Medical equipment and devices ▪ Data communications equipment ▪ Mobile phones, Personal Digital Assistants(PDAs) ▪ Industrial controls ▪ Consumer electronics ▪ Automotiv			
III. 12	Selection criteria for RTOS: The factors include the functional & non-functional requirements for selection an RTOS. Functional Requirements: These are requirements that specify the desired functionality or operations of a RTOS. It includes: • Processor Support: It is essential to ensure that the processor should support the OS under consideration. • Memory Requirement: Since embedded system is memory constrained, it is essential to have a minimal ROM & RAM requirements for an OS under consideration. • Real time capabilities: It is essential to analyse the real time capabilities of an OS under consideration, as task/process scheduling policies play an important role in real time behaviour. • Kernel & Interrupt Latency: The kernel of the OS may disable interrupts while executing certain services & it may lead to interrupt latency. This latency should be minimal for a high response system. • Inter Process Communication & Task Synchronization: This is a kernel dependent parameter & certain kernel implement policies for avoiding issues in resources sharing.		7	7

	• Modularization Support: It allows the developers to choose essential modules & recompile OS image for functioning. • Support for Networking & Communication: This include driver support for networking & communication interfaces. • Development Language Support: Certain OS requires run timr libraries for application written in languages like Java, C#, .Net etc. Non Functional Requirements: They are not related to the software's functional aspect. It concentrates on the expectation and experience of the user. • Custom developed or off the shelf: Custom-developed OS are specifically designed and developed to meet an embedded system's specific needs, while off-the-shelf OS are pre-built OS that can be purchased and used immediately. • Cost: It include developing cost, buying cost, maintenance cost etc. • Development and debugging tools availability: It is a critical factor for selecting an OS, as some OS may be superior in performance, but availability of development & debugging tools are limited. • Ease of use: It specify how easy to use a commercial OS. • After sales: It includes bug or error fixing, patch updation, online support are important factor for selecting an OS.			