Roll No: | | | | | | | | | | | |

PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi | Approved By BCI
Bengaluru

Mid Term Examinations - October 2026

Date: 05/10/2026 **Time:** 11:45 AM - 01:15 PM

Course Code: CIT3403 **Course Name:** Embedded Systems for IoT

Semester: Sixth Semester **Max. Marks:** 50 **Weightage:** 50%

CO - Levels	CO1	CO2
Marks	46	44

PART A: Answer Following Questions. 10 M

Q.N.	Questions	M	CO	BT
1	Addressing modes specify how an instruction identifies the data or operand on which it has to operate. They provide different ways of accessing data, such as immediate, register, direct, and indirect addressing. The appropriate addressing mode makes data access convenient and efficient in an assembly language program.	2	CO1	BT2
2	n parallel communication, multiple bits are transmitted simultaneously through multiple lines. In serial communication, bits are transmitted sequentially over a communication line. Parallel communication is suitable for short-distance high-speed data transfer, whereas serial communication is commonly used for communication between embedded devices.	2	CO1	BT2
3	Data types specify the type and storage requirements of data used in an embedded program. Functions divide a program into smaller, manageable sections and allow specific operations to be reused. Together, they improve the structure and readability of an Embedded C program.	2	CO1	BT2
4	The Arduino IDE provides an environment for writing, compiling, and uploading sketches to an Arduino board. It also helps in identifying programming errors and communicating with the board through the serial monitor.	2	CO2	BT2
5	Digital I/O deals with discrete logic levels, generally HIGH and LOW, and is used for devices such as switches and LEDs. Analog input reads a range of voltage levels from devices such as temperature or light sensors using the Arduino's analog input pins.	2	CO2	BT2

PART B : Answer Any 1 Following Questions. 10 M

Q.N.	Questions	M	CO	BT

ANSWER KEY

PTO (Please Turn Over)

6	Assembly language is a low-level programming language in which instructions are represented using mnemonics. It provides direct control over the registers, memory, and other resources of a microcontroller. Structure of an Assembly Language Program A typical assembly language program consists of the following elements: Label: A label identifies a particular location in the program and is useful as a target for branch instructions.Mnemonic/Instruction: It specifies the operation to be performed by the microcontroller, such as data transfer, arithmetic, or logical operation.Operand: It specifies the data or location on which the instruction operates.Comment: Comments are included to describe the purpose of instructions and improve program readability.A simple representation is: LABEL: MNEMONIC OPERAND ; COMMENTAddressing Modes Addressing modes specify how the operand of an instruction is accessed. 1. Immediate Addressing: The actual data is given directly in the instruction. Example: MOV A, #25HHere, 25H is the data to be loaded into the accumulator. 2. Register Addressing: The operand is stored in a register. Example: MOV A, R1The contents of register R1 are transferred to A. 3. Direct Addressing: The instruction directly specifies the memory location containing the data. Example: MOV A, 30H4. Indirect Addressing: A register contains the address of the data to be accessed. Example: MOV A, @R0Here, R0 contains the address of the required data.	10	CO1	BT2
7	parallel port provides a group of digital input/output lines through which a microcontroller communicates with external devices. Each line can generally be used to read an input signal or provide an output signal. Digital Input For input operation, a device such as a push button or digital sensor is connected to an I/O pin. The microcontroller reads the logic level present at the pin. For example: Logic 0 may represent LOW.Logic 1 may represent HIGH.The program can then take the required action based on the input condition. Digital Output For output operation, devices such as LEDs, buzzers, or relays can be connected to the I/O port. The microcontroller sends a HIGH or LOW signal through the corresponding pin. For example: HIGH → LED ONLOW → LED OFFBasic Interfacing Sequence Input Device → I/O Port → Microcontroller → I/O Port → Output Device The microcontroller reads the input, processes the required condition, and produces the corresponding output. Example Application Consider a push-button-controlled LED. When the push button is pressed, the microcontroller reads a HIGH or LOW signal from the input port and changes the output port accordingly to switch the LED ON or OFF. Importance of Parallel Ports Parallel ports are useful for: Reading digital sensorsInterfacing switchesControlling LEDsOperating relaysInterfacing displays and other digital devicesThus, parallel port programming provides a simple method for allowing a microcontroller to sense external digital signals and control external devices.	10	CO1	BT2

PART C: Answer Any 1 Following Questions. 10 M				
Q.N.	**Questions**	**M**	**CO**	**BT**

ANSWER KEY

8	he Arduino development toolchain provides the required environment for developing and transferring programs to an Arduino board. The Arduino IDE is the main software used for this purpose. 1. Writing the Sketch The required program, called a sketch, is written in the Arduino IDE. It generally contains the setup() and loop() functions. 2. Compilation When the Verify/Compile option is selected, the IDE checks the program for syntax and other compilation errors. The sketch is converted into a form that can be executed by the microcontroller. 3. Selecting the Board and Port Before uploading, the appropriate Arduino board and the connected communication port are selected in the IDE. This ensures that the program is transferred to the correct hardware. 4. Uploading The Upload option transfers the compiled program from the computer to the Arduino board through the USB connection. The microcontroller stores the program in its program memory. 5. Program Execution After uploading, the Arduino starts executing the sketch. The setup() function executes once, followed by continuous execution of the loop() function. 6. Debugging If the program does not work as expected, errors can be identified by checking the code, observing output values, and using tools such as the Serial Monitor. 7. Testing The connected sensors, LEDs, motors, or other devices are tested to verify that the program produces the expected operation. Basic Development Flow Write Sketch → Compile → Select Board/Port → Upload → Execute → Test/Debug Thus, the Arduino toolchain provides a simple process for developing, testing, and transferring embedded applications to an Arduino board.	10	CO2	BT2
9	Digital input is used to read two-state signals such as HIGH and LOW. Example: A push button can be connected to a digital input pin. Arduino reads whether the button is pressed or released. 2. Digital Output Digital output is used to control devices that require ON/OFF signals. Example: An LED can be connected to a digital output pin. Arduino can switch it ON using HIGH and OFF using LOW. 3. Analog Input Analog input is used to read continuously varying signals from sensors. Examples: Temperature sensor Light sensor Potentiometer The Arduino converts the analog signal into a digital value that can be processed by the program. 4. PWM Output Pulse Width Modulation (PWM) is used to control the average power supplied to a device by varying the duty cycle of a digital pulse. PWM can be used for: Controlling LED brightness Controlling motor speed Generating variable output levels Example In an automatic lighting system, an LDR can provide an analog input to Arduino. Based on the measured light intensity, Arduino can vary the brightness of an LED using PWM.	10	CO2	BT2

PART D: Answer Any 1 Following Questions. 10 M				
Q.N.	**Questions**	**M**	**CO**	**BT**
10	UART (Universal Asynchronous Receiver Transmitter) is a commonly used serial communication method for transferring data between a microcontroller and another device. It transfers data serially, generally without requiring a separate clock signal between the communicating devices. Basic UART Communication UART communication generally uses two main signal lines: TX – TransmitRX – ReceiveThe TX line of one device is connected to the RX line of the other device and vice versa. Data Transmission During transmission, the data is converted into a serial stream of bits. A typical UART frame contains: Start bitData bitsOptional parity bitStop bitThe receiver identifies the start of the frame and then reads the data bits according to the configured communication settings. Data Reception During reception, the UART receives the serial bit stream through the RX line. It identifies the start bit, collects the data bits, checks the frame if required, and makes the received data available to the microcontroller. Important Parameters UART communication is configured using parameters such as: Baud rateNumber of data bitsParityNumber of stop bitsBoth communicating devices should use compatible settings. Applications UART is commonly used for: Microcontroller-to-PC communicationGPS modulesBluetooth modulesDebugging and monitoringCommunication between embedded devices	10	CO1	BT2

ANSWER KEY

11	Embedded C is a programming language approach widely used for developing software for microcontrollers. It combines the structured programming features of C with the ability to work closely with hardware resources. 1. Variables Variables are used to store values required during program execution. For example, a variable may store a sensor reading, counter value, or status information. int temperature;2. Data Types Data types specify the kind of data stored in a variable. Common data types include: charintfloatunsigned intIn embedded systems, selecting an appropriate data type helps in using memory efficiently. 3. Functions Functions divide a program into smaller sections, with each function performing a particular task. They improve program organization, readability, and reusability. For example: void readSensor() { // sensor reading operations }4. Control Statements Control statements determine the sequence in which program instructions are executed. Important types include: Conditional statements: if(condition) { // statements }Loops: while(condition) { // statements }They can be used for decision-making and repeated operations. 5. Hardware Interaction Embedded C programs can be used to read inputs from sensors, process the information, and control output devices such as LEDs, motors, and relays. Importance of Embedded C Embedded C provides a structured way to develop microcontroller programs while allowing interaction with hardware resources. Its use of variables, functions, data types, and control statements makes embedded programs easier to organize and maintain.	10	CO1	BT2

PART E: Answer Any 1 Following Questions. 10 M				
Q.N.	**Questions**	**M**	**CO**	**BT**
12	Arduino can be used to automatically control a fan according to the surrounding temperature. The system consists of a temperature sensor, Arduino board, transistor/relay driver, and DC fan. 1. Components and Interfacing Temperature sensor: Measures the surrounding temperature and provides an electrical signal to the Arduino. Arduino: Reads the sensor value and compares it with a predefined temperature limit. Relay/Transistor driver: Acts as an interface between the Arduino and the fan because the Arduino cannot directly supply sufficient current to the fan. Fan: Turns ON or OFF according to the control signal from Arduino. Basic flow: Temperature Sensor → Arduino → Relay/Driver → Fan 2. Sequence of Operation The temperature sensor continuously senses the surrounding temperature. Arduino reads the sensor output through an appropriate input pin. The program converts/processes the sensor reading to determine the temperature. The measured temperature is compared with a predefined threshold, for example, 30°C. If the temperature is above 30°C, Arduino activates the relay/driver and the fan turns ON. If the temperature falls below 30°C, Arduino deactivates the relay/driver and the fan turns OFF. This process is repeated continuously inside the loop() function. 3. Role of the Arduino Program The Arduino program performs three main functions: Configures the sensor and fan-control pins in setup(). Reads the temperature continuously in loop(). Uses an if-else condition to control the fan according to the measured temperature. A simple program structure is: int sensorPin = A0; int fanPin = 7; void setup() { pinMode(fanPin, OUTPUT); } void loop() { int temperature = analogRead(sensorPin); if (temperature > 500) { digitalWrite(fanPin, HIGH); // Fan ON } else { digitalWrite(fanPin, LOW); // Fan OFF }	10	CO2	BT2

ANSWER KEY

13	n Arduino shield is an additional hardware board that can be mounted on an Arduino board to provide extra functionality without designing the complete interface from the beginning. Types of Arduino Shields 1. Communication Shields These shields provide communication capabilities that may not be available on the basic Arduino board. Examples include: Wi-Fi shields Ethernet shields GSM shields They are useful for connecting Arduino to networks and IoT platforms. 2. Motor Driver Shields Motor driver shields provide suitable interfaces for controlling motors. They are used in: Robotics Automated vehicles Conveyor systems 3. Sensor Shields These shields provide convenient connections for different sensors and simplify sensor interfacing. They can be used for collecting: Temperature data Light intensity Humidity Motion information 4. Network Shields Network shields allow Arduino to communicate with other devices or Internet services through wired or wireless networks. Role in IoT Shields extend Arduino's capabilities and make it easier to build application-specific systems. For example: Sensor → Arduino → Wi-Fi Shield → Internet/Cloud In a smart home application, sensors can collect environmental data, Arduino can process it, and a communication shield can transmit the information to a remote monitoring system.	10	CO2	BT2

ANSWER KEY