Embedded System & Microcontroller Application (4351102) - Summer 2025 Solution

Milav Dabgar

May 12, 2025

Question 1(a) [3 marks]

State the features of ATmega32.

Solution

ATmega32 Features:

Table 1. ATmega32 Features

Feature	Description
Flash Memory	32KB programmable memory
SRAM	2KB internal SRAM
EEPROM	1KB non-volatile data storage
I/O Pins	32 programmable I/O lines
Timers	3 flexible timer/counters
ADC	10-bit 8-channel ADC

Additional Specifications:

- **Operating Voltage**: 2.7V to 5.5V range
- **Clock Speed**: Up to 16 MHz operation
- **Communication**: USART, SPI, I2C interfaces

Mnemonic

"Fast SRAM Enjoys Input Timers And Communication"

Question 1(b) [4 marks]

Write criteria for choosing microcontroller.

Solution

Selection Criteria:

Table 2. Selection Criteria

Criteria	Consideration
Processing Speed	Clock frequency requirements
Memory Size	Program and data storage needs
I/O Requirements	Number of pins needed
Power Consumption	Battery life considerations
Cost	Budget constraints
Development Tools	Compiler and debugger availability

System Considerations:

- **Application Type**: Real-time vs general purpose
- **Communication Needs**: Serial, parallel, wireless protocols
- **Package Size**: Space constraints in final product

Mnemonic

"Processing Memory I/O Power Cost Development Application Communication Package"

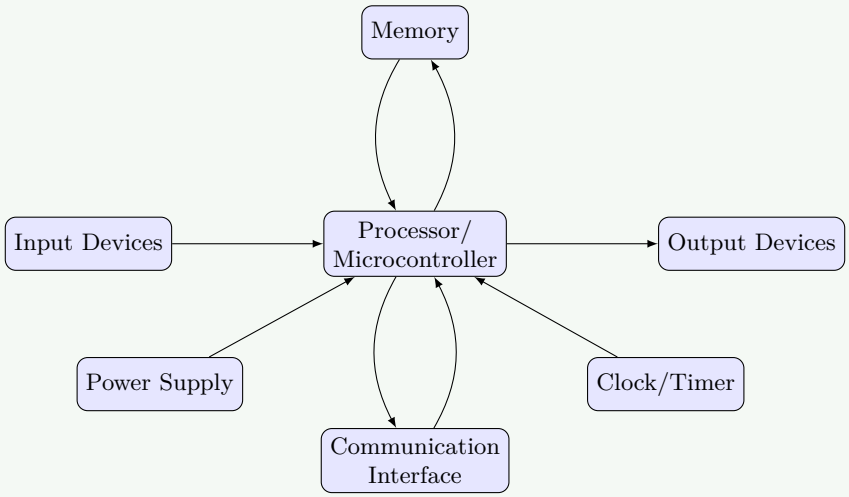
Question 1(c) [7 marks]

Draw and explain general block diagram of embedded system.

Solution

General Block Diagram:

Figure 1. Embedded System Block Diagram



Block Functions:

- **Processor**: Central processing unit executing instructions
- **Memory**: Stores program code and data temporarily
- **Input Devices**: Sensors, switches providing system input
- **Output Devices**: Actuators, displays showing results
- **Communication**: Interfaces for external device connectivity
- **Power Supply**: Provides stable voltage to all components
- **Clock/Timer**: Synchronizes system operations and timing

Mnemonic

"Processors Memory Input Output Communication Power Clock"

OR

Question 1(c) [7 marks]

Define real time operating system and explain its characteristics.

Solution

Real Time Operating System (RTOS): Operating system designed to process data and events within strict time constraints.

Table 3. RTOS Characteristics

Characteristic	Description
Deterministic	Predictable response times
Preemptive	Higher priority tasks interrupt lower ones
Multitasking	Multiple tasks run concurrently
Fast Context Switch	Quick task switching capability
Priority Scheduling	Tasks executed based on priority
Interrupt Handling	Efficient interrupt processing

Key Concepts:

- **Hard Real-time**: Missing deadline causes system failure
- **Soft Real-time**: Missing deadline degrades performance
- **Time Constraints**: Operations must complete within deadlines

Mnemonic

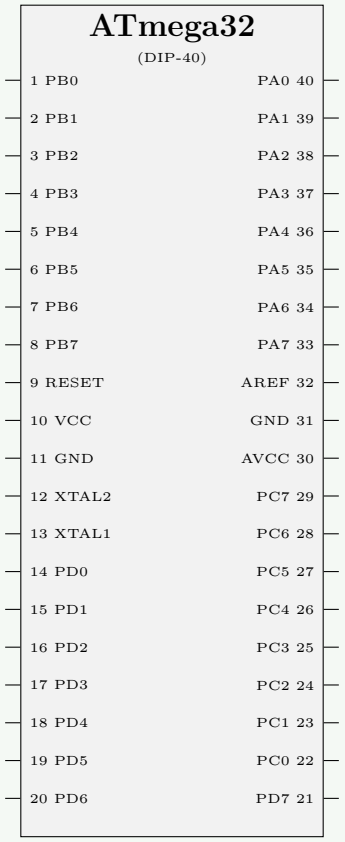
"Deterministic Preemptive Multitasking Fast Priority Interrupt"

Question 2(a) [3 marks]

Draw pin diagram of ATmega32.

Solution

ATmega32 Pin Configuration:



Mnemonic

"Port B A Reset Vcc Ground Crystal Port D C"

Question 2(b) [4 marks]

Explain status register of ATmega32.

Solution

Status Register (SREG):

Table 4. SREG Bit Configuration

Bit	Name	Function
Bit 7	I	Global Interrupt Enable
Bit 6	T	Bit Copy Storage
Bit 5	H	Half Carry Flag
Bit 4	S	Sign Bit
Bit 3	V	Overflow Flag
Bit 2	N	Negative Flag
Bit 1	Z	Zero Flag
Bit 0	C	Carry Flag

Functions:

- **Flags Update**: Automatically set/cleared by ALU operations
- **Conditional Branching**: Used for program flow control (e.g., BREQ, BRCS)

- **Interrupt Control**: I-bit enables/disables all interrupts

Mnemonic

"I Think Half Sign Overflow Negative Zero Carry"

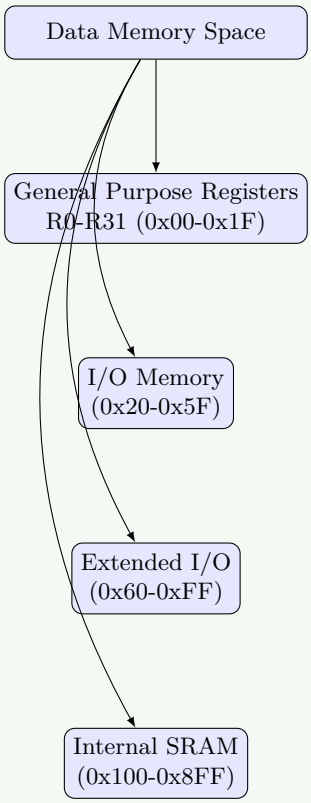
Question 2(c) [7 marks]

Explain data memory of ATmega32 in detail.

Solution

Data Memory Organization:

Figure 2. Data Memory Space



Memory Sections:

- **General Purpose Registers**: 32 registers (R0-R31) for fast data operations, mapped to lowest addresses.
- **I/O Memory**: 64 registers for peripheral control (Port, Timer, UART, etc.), accessible via IN/OUT instructions.
- **Extended I/O**: Reserved for future use or extended peripheral control in some AVRs.
- **Internal SRAM**: 2KB (2048 bytes) of volatile memory for variables and stack.
- **Stack**: Implementation grows downward from high memory addresses towards low addresses.

Mnemonic

"General I/O Extended SRAM Address Stack"

OR

Question 2(a) [3 marks]

Write functions of DDRx, PINx and PORTx registers.

Solution

I/O Port Registers:

Table 5. I/O Registers

Register	Function
DDRx	Data Direction Register - configures pin direction
PINx	Pin Input Register - reads current pin logical state
PORTx	Port Output Register - writes data/pull-up control

Operations:
- **DDRx**: Set bit to 1 for Output, 0 for Input.
- **PINx**: Read accesses physical pin voltage level.
- **PORTx**: When DDRx=1, drives High/Low. When DDRx=0, enables internal Pull-up (if 1).

Mnemonic

"Direction Input Output"

OR

Question 2(b) [4 marks]

Explain different I/O registers associated with EEPROM in AVR.

Solution

EEPROM Registers:

Table 6. EEPROM Registers

Register	Function
EEAR	EEPROM Address Register (High/Low bytes)
EEDR	EEPROM Data Register
EECR	EEPROM Control Register

EECR Control Bits:
- **EERIE**: EEPROM Ready Interrupt Enable
- **EEMWE**: EEPROM Master Write Enable (Safety mechanism)
- **EEWE**: EEPROM Write Enable (Strobe to write)
- **EERE**: EEPROM Read Enable

Process: To write, ensure no write is in progress, set address and data, set EEMWE, then set EEWE within 4 clock cycles.

Mnemonic

"Address Data Control Ready Master Write Read"

OR

Question 2(c) [7 marks]

Explain different ways of connecting clock sources to the AVR.

Solution

Clock Source Options:

Figure 3. Clock Sources

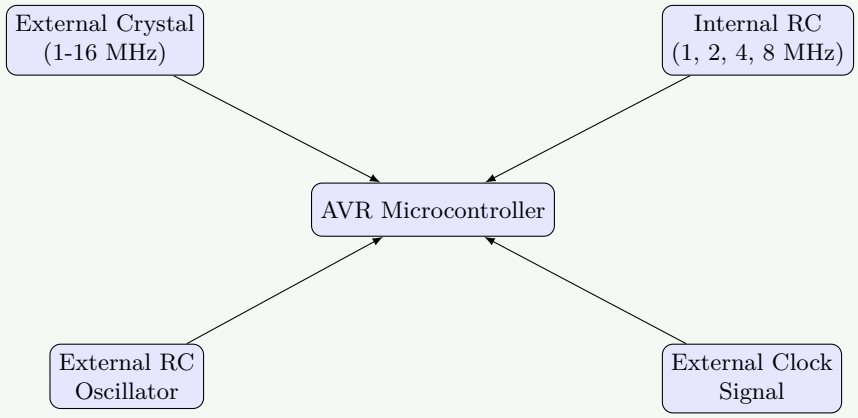


Table 7. Clock Source Types

Source	Description
External Crystal	High precision quartz crystal connected to XTAL1/XTAL2. Capacitor needed.
External RC	Resistor-Capacitor circuit. Low cost, less precise.
Internal RC	Built-in oscillator. No external components. 1MHz default, up to 8MHz.
External Clock	Drive XTAL1 with external logic signal.

Configuration:

- **Fuse Bits**: CKSEL3:0 and SUT1:0 fuses determine the active clock source and startup time.
- **Usage**: Crystal for timing-critical (UART), Internal RC for cost/power saving.

Mnemonic

"Crystal RC Internal External Fuse Startup Frequency Components"

Question 3(a) [3 marks]

Write function of registers associated with Timer 1.

Solution

Timer 1 Registers (16-bit):

Table 8. Timer 1 Registers

Register	Function
TCNT1	Timer/Counter Register - Holds current count value (16-bit)
TCCR1A/B	Control Registers - Sets mode, prescaler, compare output
ICR1	Input Capture Register - Captures TCNT1 value on event
OCR1A/B	Output Compare Registers - Values to compare with TCNT1
TIMSK	Interrupt Mask - Enables Overflow/Compare/Capture interrupts
TIFR	Interrupt Flag - Indicates pending interrupts

Mnemonic

"Timer Control Input Output Mask Flag"

Question 3(b) [4 marks]

Discuss steps to program Timer0 in Normal mode.

Solution

Programming Steps used for Normal Mode:

1. **Configure Mode**: Write to TCCR0 to select Normal Mode (WGM01:0 = 00).
2. **Load Initial Value**: Write starting count to TCNT0.
3. **Interrupts**: If using interrupts, set TOIE0 in TIMSK and enable global interrupts (sei).
4. **Start Timer**: Write non-zero prescaler value to Clock Select bits (CS02:0) in TCCR0.
5. **Monitor/Handle**: Wait for TOV0 flag in TIFR (polling) or handle in ISR.

```
TCCR0 = 0x00;        // Stop timer
TCNT0 = 0x00;        // Clear count
TCCR0 = 0x01;        // Start: Normal mode, No prescaler
while(!(TIFR & 1));  // Wait for overflow
TIFR = 0x01;         // Clear flag
```

Mnemonic

"Set Select Load Enable Start"

Question 3(c) [7 marks]

Write a C program to receive bytes of data serially and put them on PORT A. Set the baud rate at 9600, 8-bit, and 1 stop bit.

Solution

Program:

```
#include <avr/io.h>

void USART_Init() {
    // Set baud rate to 9600 (assuming 8MHz clock)
    // UBRR = (F_CPU / (16 * Baud)) - 1 = (8M / (16 * 9600)) - 1 = 51
    UBRRH = 0x00;
    UBRRL = 51;

    // Enable receiver
```

```
    UCSRB = (1<<RXEN);

    // Set frame format: 8 data, 1 stop bit
    // URSEL must be 1 to write to UCSRC
    UCSRC = (1<<URSEL) | (3<<UCSZ0);
}

unsigned char USART_Receive() {
    // Wait for data to be received (RXC flag)
    while(!(UCSRA & (1<<RXC)));
    // Get and return received data from buffer
    return UDR;
}

int main() {
    DDRA = 0xFF;        // Configure PORTA as output
    USART_Init();       // Initialize USART

    while(1) {
        PORTA = USART_Receive();  // Receive and display on Port A
    }
    return 0;
}
```

Configuration Details:

- **Baud Rate**: 9600 bps requires UBRR=51 at 8MHz.
- **Frame Format**: 8-bit data (UCSZ1:0 = 11), 1 stop bit (USBS = 0).
- **Reception**: Polling RXC bit in UCSRA ensures valid data is read.

Mnemonic

"Initialize Receive Display Loop"

OR

Question 3(a) [3 marks]

Write function of registers associated with Serial Communication in AVR.

Solution

USART Registers:

Table 9. USART Registers

Register	Function
UDR	USART Data Register - Buffer for Tx/Rx data
UCSRA	Control/Status A - Flags (RXC, TXC, UDRE), error bits
UCSRB	Control/Status B - Enable Rx/Tx, Interrupts (RXCIE), 9th bit
UCSRC	Control/Status C - Frame format (Data bits, Parity, Stop bits)
UBRR	Baud Rate Register - Sets communication speed

Mnemonic

"Data Control Status Baud"

OR

Question 3(b) [4 marks]

Discuss steps to program AVR to transfer data serially.

Solution

Steps for Serial Transmission:

1. **Baud Rate**: Calculate and load value into UBRRH/Low registers.
2. **Frame Format**: Set data bits, parity, and stop bits in UCSRC (e.g., 8N1).
3. **Enable Tx**: Set TXEN bit in UCSRB to enable transmitter.
4. **Wait for Buffer**: Monitor UDRE flag in UCSRA to ensure UDR is empty.
5. **Send Data**: Write byte to UDR register to start transmission.

```
void USART_Transmit(unsigned char data) {
    // Wait for empty transmit buffer
    while(!(UCSRA & (1<<UDRE)));
    // Put data into buffer, sends the data
    UDR = data;
}
```

Mnemonic

"Baud Enable Format Wait Load"

OR

Question 3(c) [7 marks]

Write a C program to toggle only the PORTB.4 bit continuously every 2 ms. Use timer 1, Normal mode, and no prescaler to create the delay. Assume XTAL=8MHz.

Solution

Timer 1 Delay Calculation:

- **Target Delay**: 2 ms
- **Frequency**: 8 MHz
- **Machine Cycle**: $1/8\text{MHz} = 0.125\mu\text{s}$
- **Counts Required**: $2\text{ms}/0.125\mu\text{s} = 16000$ counts
- **Timer Start Value**: $65536 - 16000 = 49536$ (0xC180)

Program:

```
#include <avr/io.h>
#include <avr/interrupt.h>

// ISR for Timer1 Overflow
ISR(TIMER1_OVF_vect) {
    // Toggle PORTB.4
    PORTB ^= (1<<4);

    // Reload Timer for 2ms
    TCNT1 = 49536;
}

int main() {
    // Set PORTB.4 as output
    DDRB |= (1<<4);

    // Initialize Timer1
```

```
    // Normal Mode (WGM13:0 = 0000)
    TCCR1A = 0x00;

    // Start Timer with No Prescaler (CS12:0 = 001)
    TCCR1B = (1<<CS10);

    // Load initial count
    TCNT1 = 49536;

    // Enable Timer1 Overflow Interrupt
    TIMSK |= (1<<TOIE1);

    // Enable Global Interrupts
    sei();

    while(1) {
        // Main loop does nothing, ISR handles toggling
    }
    return 0;
}
```

Mnemonic

"Configure Timer Calculate Enable Loop"

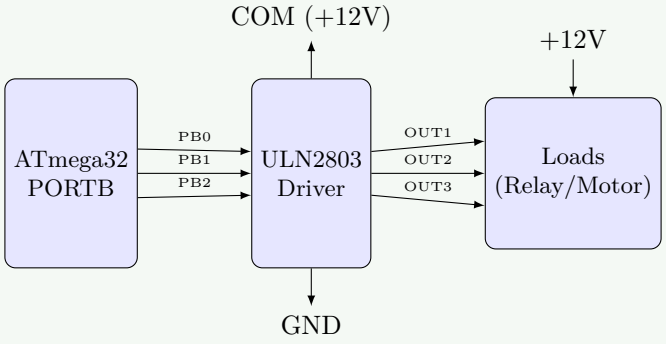
Question 4(a) [3 marks]

Draw interfacing diagram of ULN2803 with ATmega32.

Solution

ULN2803 Interfacing:

Figure 4. ULN2803 Interface



Connections:

- **Inputs**: Connect ATmega32 pins directly to ULN2803 inputs (1B-8B).
- **Outputs**: Open collector outputs drive loads connected to positive supply.
- **Common**: Pin 10 to COM (Load Supply) for flyback diodes, Pin 9 to GND.

Mnemonic

"Input Output Common Supply Ground"

Question 4(b) [4 marks]

Write an AVR C program to get a byte of data from Port B, and then send it to Port C.

Solution

Program:

```c
#include <avr/io.h>

int main() {
    // Configure Directions
    DDRB = 0x00;    // Port B as Input
    DDRC = 0xFF;    // Port C as Output

    // Enable Pull-ups on Input Port
    PORTB = 0xFF;

    while(1) {
        // Read data from PINB register
        unsigned char data = PINB;

        // Write data to PORTC register
        PORTC = data;
    }
    return 0;
}
```

- **Direction**: DDRB=0 (In), DDRC=1 (Out).
- **Pull-ups**: Writing 1 to PORTB when DDRB=0 enables internal pull-ups.
- **Transfer**: Continuous loop reads PINB and updates PORTC.

Mnemonic

"Configure Enable Read Write"

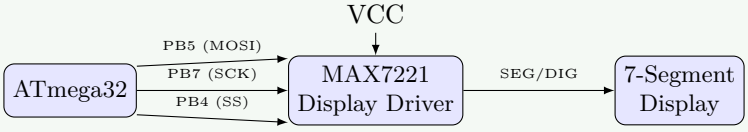
Question 4(c) [7 marks]

Draw and explain interfacing of MAX7221 with ATmega32.

Solution

MAX7221 Interface:

Figure 5. MAX7221 Interface



Interface Details:

- **Protocol**: SPI (Serial Peripheral Interface). 3-wire: Data (DIN), Clock (CLK), Load (CS).
- **Function**: Drives up to 8 common-cathode 7-segment displays.
- **Multiplexing**: Handles scanning circuitry internally.
- **Current Control**: Set via external resistor (ISET pin).

Programming:

1. Pull LOAD (CS) Low.
2. Shift in 16-bit word (Address + Data) via DIN on CLK rising edge.

3. Pulse LOAD High to latch data.

Mnemonic

"SPI Data Clock Load Multiplex Current Program"

OR

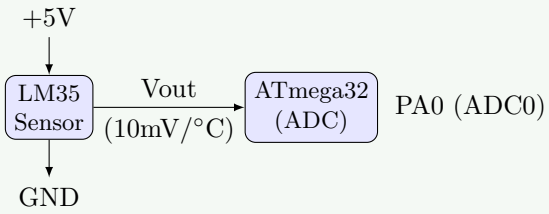
Question 4(a) [3 marks]

Draw interfacing diagram of LM35 with ATmega32.

Solution

LM35 Interface Circuit:

Figure 6. LM35 Connection



Specifications:

- **Output**: Linear voltage proportional to temperature (10mV/°C).
- **Connection**: Vout connected to any ADC channel (e.g., PA0).
- **Conversion**: $V_{in} \times \frac{1024}{V_{ref}} = \text{ADC Value}$.

Mnemonic

"VCC OUT GND Linear"

OR

Question 4(b) [4 marks]

Write an AVR C program to monitor bit 5 of port C, if it is HIGH, send 55H to Port B; otherwise, send AAH to Port B.

Solution

Program:

```c
#include <avr/io.h>

int main() {
    DDRC &= ~(1<<5);    // PC5 as Input
    PORTC |= (1<<5);    // Enable Pull-up on PC5
    DDRB = 0xFF;        // Port B as Output

    while(1) {
        // Check if PC5 is High
        if(PINC & (1<<5)) {
```

```
            PORTB = 0x55;   // Pattern 01010101
        } else {
            PORTB = 0xAA;   // Pattern 10101010
        }
    }
    return 0;
}
```

Logic:
- **Masking**: `PINC & (1<<5)` isolates bit 5.
- **Condition**: If result is non-zero (True), bit is High.

Mnemonic

"Monitor Conditional Output Loop"

OR

Question 4(c) [7 marks]

Discuss registers used to program SPI in the AVR.

Solution

SPI Registers:

Table 10. SPI Registers

Register	Function
SPCR	Control Register - EN, Int, Order, Master/Slave, Mode
SPSR	Status Register - Interrupt Flag (SPIF), Collision (WCOL)
SPDR	Data Register - Shift register for transmission/reception

SPCR Bits:
- **SPE**: SPI Enable.
- **MSTR**: Master (1) or Slave (0) Select.
- **SPR1:0**: Clock Rate Select (fosc/4, /16, /64, /128).
- **CPOL/CPHA**: Clock Polarity and Phase (Mode 0-3).

Usage: Write to SPDR starts transmission (Master). Read SPDR after SPIF is set retrieves received data.

Mnemonic

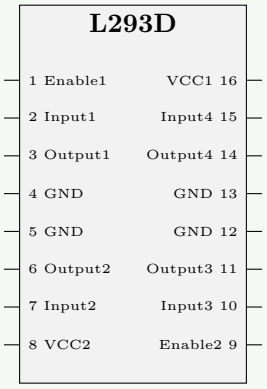
"Control Status Data Configure Enable Write Wait Read"

Question 5(a) [3 marks]

Draw pin diagram of L293D motor driver IC.

Solution

L293D Pinout:



Mnemonic

"Enable Input Output Supply Ground"

Question 5(b) [4 marks]

Draw and explain ADMUX register.

Solution

ADMUX (ADC Multiplexer Selection Register):

Table 11. ADMUX Register

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Bit Descriptions:

- **REFS1:0**: Reference Selection Bits.
  - 00: AREF, Internal Vref turned off.
  - 01: AVCC with external capacitor at AREF pin.
  - 11: Internal 2.56V Voltage Reference.
- **ADLAR**: ADC Left Adjust Result. If 1, result is left-adjusted (for 8-bit precision).
- **MUX4:0**: Analog Channel and Gain Selection Bits. Selects which ADC pin (ADC0-ADC7) is connected to the ADC.

Mnemonic

"Reference Adjust Multiplex Channel"

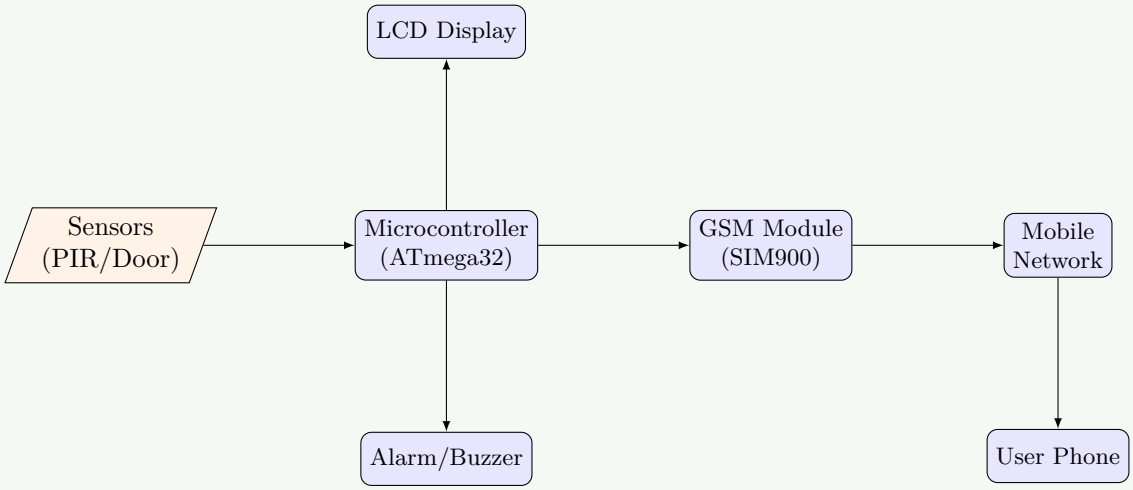
Question 5(c) [7 marks]

Explain the block diagram of GSM based security system.

Solution

GSM Security System:

Figure 7. GSM Security Block Diagram



System Components:

- **Sensors**: PIR sensor detects motion, Door switches detect entry.
- **Microcontroller**: Processes sensor inputs. If intrusion detected:
  1. Activates Alarm/Buzzer.
  2. Sends AT commands to GSM Module.
  3. Updates Status on LCD.
- **GSM Module**: Communicates with cellular network via UART. Sends SMS/Call to user.
- **User Mobile**: Receives alert ("Intruder Detected!").

Mnemonic

"Sensors Microcontroller GSM Mobile Alarm Display Power Operation"

OR

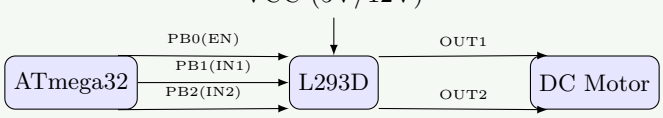
Question 5(a) [3 marks]

Draw circuit diagram to interface DC motor with ATmega32 using L293D motor driver.

Solution

DC Motor Interface:

Figure 8. DC Motor with L293D



Control Logic:

- **Enable**: High (1) to activate driver channel.
- **Direction**: IN1=1, IN2=0 (Forward); IN1=0, IN2=1 (Reverse).
- **Stop**: IN1=0, IN2=0.

Mnemonic

"Enable Direction Output Power Control"

OR

Question 5(b) [4 marks]

Draw and explain ADCSRA register.

Solution

ADCSRA (ADC Control and Status Register A):

Table 12. ADCSRA Register

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Bit Functions:

- **ADEN**: ADC Enable. Must be set to 1 to use ADC.
- **ADSC**: Start Conversion. Write 1 to start a single conversion.
- **ADATE**: Auto Trigger Enable.
- **ADIF**: Interrupt Flag. Set when conversion completes.
- **ADIE**: Interrupt Enable. Activates ADC interrupt if Global Interrupts are enabled.
- **ADPS2:0**: Prescaler Select. Divides system clock (e.g., /128 for 8MHz crystal) to get ADC clock (50-200kHz).

Mnemonic

"Enable Start Auto Interrupt Prescaler Configure"