The Open University of Sri Lanka
Faculty of Engineering Technology
Department of Electrical and Computer Engineering

Study Programme	: Bachelor of Science Honours in Engineering
Name of the Examination	: Final Examination
Course Code and Title	**: EEX5346 Embedded Systems**
Academic Year	: 2023/24
Date	: 28th February 2025
Time	: 930-1230hrs
Duration	: **3 hours**

General Instructions

1. Read all instructions carefully before answering the questions.

2. This question paper contains three (3) questions in SECTION A and three (3) questions in SECTION B on **eight (8)** pages.

3. Answer all questions in SECTION A.[60 Marks], and answer any Two questions from SECTION B.[40 Marks].

4. Answers for each question should commence from a new page.

5. Refer to the Annexure of the ESP32 power modes given on page five (5) to calculate power consumption, if Required.

6. Refer to the Annexure of the useful equations given on page six(6), if Required.

7. Refer to the Annexure of the Arduino programming syntax given on page seven(7) to write any Arduino programming code.

8. This is a Closed Book Test (CBT).

9. Answers should be in clear handwriting.

10. Do not use Red colour pen.

Section A: Answer ALL questions [60 Marks]

The following description is about the **Smart Snack Management System (SSMS)**. Your task is to analyse the following specifications and design the **SSMS** by Providing IoT solutions for a smart city.

*SSMS consists of the many interconnected **Smart Vending Machines(SVM)** that are strategically placed throughout the city to create personalised and effective solutions for managing people's snack needs. These vending machines are not just standalone units but part of a **centralised IoT network** that communicates in real-time with the inventory management system. Each vending machine is equipped with **touchscreen interfaces, NFC/RFID payment systems**, and **IoT sensors** that monitor stock levels, product expiration dates, environmental conditions (e.g. temperature and humidity), and machine health. When a consumer approaches the vending machine, it displays personalised recommendations based on their shopping history and dietary preferences. The customer selects a product, taps the phone to pay via NFC, and the machine dispenses the item instantly. The system uses **AI Techniques** to analyse sales data and predict demand, optimising inventory levels.*

SVM also consists of actuators (sensory elements) like lighting, sound, and aroma diffusers, which are strategically used to enhance customer cravings. Different lighting techniques create a welcoming atmosphere, while background music like upbeat tunes, or softer melodies, influences the moving pace. Using tempting aromas near vending machines evokes positive emotions and encourages customers to make purchases.

*The **centralised management platform** provides real-time insights into sales trends, machine performance, and customer preferences. Operators can use this data to optimise snack offerings, adjust pricing strategies, and plan maintenance schedules. For example, if a particular machine frequently experiences technical issues, the system can alert the maintenance team before a breakdown occurs.*

Propose a design for the **SSMS** by executing an IoT solution to provide a better Snack experience. Accordingly, answer the following questions.

[Q1]

(i) From the global perspective, draw an edge-fog-cloud IoT architecture diagram for the proposed **SSMS** solution and provide a brief explanation. Clearly indicate what the edge devices, gateways, fog devices, connectivity, cloud services, and applications are. [16 Marks]

(ii) Write a program using pseudo-code for the algorithm for the following.

a. When there is an emergency breakdown, alert the available maintenance staff members to attend the repair job.

b. When there is a shortage of snacks in the vending machine, alert the nearest supplier to refill the particular vending machine with the snacks.

[12 Marks]

(iii) Briefly describe how the Watchdog timer gets implemented in the **SSMS**. [7 Marks]

[Q2]

Draw a Software architecture for the **SVM** and briefly explain each operation. (hint: show how Interrupt Service Requests (**ISR**) and tasks interact)

[15 Marks]

[Q3]

(i) Identify three (03) security concerns imposed on the proposed **SSMS** system and Briefly explain what steps could be taken to mitigate identified concerns.

[6 Marks]

(ii) Briefly describe two (02) main challenges encountered and how to overcome them when designing **SSMS** by optimising design metrics.

[4 Marks]

SECTION B: Answer any TWO questions. [40 Marks]

[Q4]

(i) Describe the function, Interrupt Service Routine(**ISR**), and Tasks in an Embedded System. [4 Marks]

(ii) Draw an event-driven state machine diagram for **SVM** tasks and describe how they interact with the states of an **RTOS**.

[8 Marks]

(iii) Refer to the Arduino programming instructions and write a program for an event-driven state machine in Q4.(ii). State the comments where necessary.

[8 Marks]

[Q5]

(i) Assuming there are multiple controllers connected to a bus for an **SVM**, draw a diagram and briefly explain the operation of the Daisy Chain Bus arbitration technique. [4 Marks]

(ii) Propose a suitable wireless connectivity architecture for the **SSMS** and justify your selection. [6 Marks]

(iii) Considering remote Person Detection(**PD**) node is installed to detect a person and send a RF signal to **SVM** to activate the sensory elements once every second using an **ESP32** single-core model, and assuming data is transmission takes 550ms. The **PD** sensor consumes 25mA and takes 35ms to perform each detection. Compute the energy consumption over one month. Refer **ESP32** datasheet, if required. (Hint: Apply suitable power optimisation techniques)

[10 Marks]

[Q6]

(i) Briefly describe the static and dynamic characteristics of the sensors. [4 Marks]

(ii) Draw a hardware architecture for the **SVM** and show how sensors and other components interact with a microcontroller (**ESP32** preferred). [6 Marks]

(iii) Draw a diagram and briefly describe Dynamic Voltage and Frequency Scaling (**DVFS**) where the processor runs at three different voltages, 2.5V, 4.0V, and 5.0V. In addition, the maximum clock frequencies are 25MHz, 40MHz, and 50MHz for each voltage respectively. Assume energy consumption is 40nJ per cycle at 5.0V and tasks need to execute 10^9 cycles within 25s. Power consumption is zero when shut down the processor.

[10 Marks]

----------**END**----------

Annexure

ESP32 Power consumption

Table 8: Power Consumption by Power Modes

Power mode	Description			Power Consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 17 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered on.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP coprocessor is powered on.			150 μA
	ULP sensor-monitored pattern			100 μA @1% duty
	RTC timer + RTC memory			10 μA
Hibernation	RTC timer only			5 μA
Power off	CHIP_PU is set to low level, the chip is powered off.			1 μA

Table 9: Association of sleep pattern

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
Sleep pattern	Association sleep pattern			ULP sensor-monitored pattern	-
CPU	ON	ON	PAUSE	OFF	OFF
Wi-Fi/BT baseband and radio	ON	OFF	OFF	OFF	OFF
RTC memory and RTC peripherals	ON	ON	ON	ON	OFF
ULP co-processor	ON	ON	ON	ON/OFF	OFF

Table 17: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

Useful Equations

Energy consumption (E)= sleep energy x sleep time + active energy x active time

Dynamic power consumption:

$$P = \alpha C_L V^2_{dd} f$$

(CMOS constant values where (α) is the switching activity, (C_L) is the load capacitance),

Voltage- V_{dd}, power- P ; Clock frequency- f ; run-time of- t;
Number of parallel operations- β.

Energy consumption:

$$E = P \times t$$

New frequency(f') of reducing frequency (f) by a factor of (β):

$$f' = f / \beta$$

The power (P) reduces per operation by:

$$P' = P / \beta^2$$

Where (P') is new reduced power.

Syntax of selected instructions of the Arduino programming

Structure & Flow

Basic program structure

```
void setup () {
    // Runs once when sketch starts
}
void loop () {
   //Runs repeatedly
}
```

Control Structures

```
If (x < 5) {……} else {……}
while (x < 5) {……}
for ( int i = 0; i< 10; i++) {……}
break;       // Exit a loop immediately
continue;    // Go to next iteration
switch (var) {
case 1:
  ……
break;
   case 2:
 ……..
break;
   default:
 ……..
}
return x;    // x must match return type
return;      // For void return type
```

Function Definitions

<ret. type><name> (params) {……}
e.g. int double (int x) {return x*2;}

Operators

General operators

= assignment
+ add - subtract
* multiply / divide
% modulo
== equal to != not equal to
<less than > greater than
<= less than or equal to
>= greater than or equal to
&& and || or
! not

Compound operators

++ increment
-- decrement
+= compound addition
-= compound subtraction
*= compound multiplication
/= compound division
&= compound bitwise and
|= compound bitwise or

Bitwise Operators

& bitwise and | bitwise or
^ bitwise xor ~ bitwise not
<< shift left >> shift right

Pointer Access

& reference: get a pointer
* dereference: follow a pointer

Variables, Arrays, and Data

Data types

bool	true	|	false
char	-128	-	127, 'a' '$' etc.
unsigned char	0	–	255
byte	0	–	255
int	-32768	-	32767
unsigned int	0	–	65535
word	0	–	65535
long	-2147483648	–	2147483647
unsigned long	0	–	4294967295
float	-3.4028e+38	–	3.4028e+38

double currently same as float
void return type: no return value

Strings

```
char str1[8] =
    {'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
    // Includes \0 null termination
char str2[8] =
    {'A', 'r', 'd', 'u', 'i', 'n', 'o', };
    //Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123 decimal
0b01111011 binary
0173 octal – base 8
0x7B hexadecimal – base 16
123U force unsigned
123L force long
123UL force unsigned long
123.0force floating point
1.23e61.23*10^6 = 1230000

Qualifiers

Static persists between calls
volatilein RAM (nice for ISR)
const read-only
PROGMEMin flash

Arrays

```
byte mypins [] = {2, 4, 8, 3, 6};
int myInts [6];          // Array of 6 ints
myInts[0] = 42;          // Assigning first
                         // Index of myInts
myInts[6] = 12;          // ERROR1 Indexes
                         // are 0 though 5
```

Built – in Functions

Pin Input / Output
Digital I/O – pins 0 – 13 A0 – A5
PinMode(pin,
{ INPUT | OUTPUT | INOUT_PULLUP})
int **digitalRead**(pin)
digitalWrite(pin, {HIGH | LOW})

Analog In – pins A0 – A5
Int **analogRead** (pin)
analogReference(
{DEFAULT | INTERNAL | EXTERNAL})

PWM Out – pins 3 5 6 9 10 11
analogWrite(pin, value) // 0-255

Advanced I/O
tone(pint, freq_Hz, [duration_msec])
noTone(pin)
shiftOut(dataPin, clockPin,
{MSBFIRST | LSBFIRST}, value)
shiftIn(dataPin, clockPin,
{MSBFIRST | LSBFIRST})
unsigned long pulseIn(pin,
{HIGH | LOW}, [timeout_usec])

Time
unsigned long millis()
// overflows at 50 days
unsigned long micros()
// overflow at 70 minutes
delay(msec)
delayMicroseconds(usec)

Math
min(x, y)**max**(x, y) **abs**(x)
sin(rad) **cos**(rad) **tan**(rad)
sqrt(x) **pow**(base, exponent)
constain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)

Random Numbers
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)

Bits and Bytes
lowByte(x) **highByte**(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn) // bitn: 0=LSB 7=MSB

Type Conversions
char(val)**byte**(val)
int(val) **word**(val)
long(val) **float**(val)

External Interrupts
attachInterrrupt(interrupt, func,
{LOW | CHANGE | RISING | FALLING})
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries

serial – comm.with pc or via RX/TX
begin(long speed) // up to 115200
end()
int **available()** // #bytes available
int **read()**// -1 if none available
int **peek()** // Read w/o removing
flush()
print(data)println(data)
write(byte) write(char*string)
write(byte*data, size)
serialEvent() // called if data rdy

softwareSerial.h – comm. on any pin
softwareSerial (rxpin, txpin)
begin(long speed) // up to 115200
listen() // only 1 can listen
isListening() // at aitme.
read, peek, print, println, write
// Equivalent to serial library

EEPROM.h – access non-voLatile memory
byte read(addr)
write(addr, byte)
EEPROM[index] // Access as array

Servo.h – control servo motors
attach(pin, [min_usec, max_usec])
write(angle)// 0 to 180
writeMicroseconds(us)
// 1000-2000; 1500 is midpoint
int read()// 0 to 180
bool attached()
detach()

wire.h – I C communication
begin() // Join a master
begin(addr) // Join a salve @ addr
requestFrom(address, count)
beginTransmission(addr) // step 1
send(byte) // step 2
send(char*string)
send(byte*data, size)
endTransmission() // step 3
int avaialable() // #bytes available
byte receiver() // get next byte
onReceive(handler)
onRequest(handler)