00047

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

Study Programme	: Bachelor of Technology Honours in Engineering
Name of the Examination	: Final Examination
Course Code and Title	**: EEX5346 Embedded Systems**
Academic Year	: 2021/22
Date	: 09th March 2023
Time	: 930-1230 hrs
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 **three(3)** pages, from page 2 to page 4.

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

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

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

6. Refer to the Annexure of the Arduino programming syntax given on page six(6) to write any Arduino programming code.

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

8. Answers should be in clear handwriting.

9.Do not use Red colour pen.

Section A:answer ALL questions [60 Marks]

The following description is about the **Smart Ambulance Service (SAS)**. Your task is to analyse the following specifications and design the Smart ambulance service *by Giving IoT solutions for a healthcare emergency service.*

"COVID-19 has caused a disaster situation all over the world in the recent past by putting healthcare services to their maximum capacity. With Intensive Care Units (ICUs) overflowing with patients, several smart technologies have been deployed, including smart Ambulance service.

The Smart Ambulance service system allows the most suitable ambulance to attend to the patient. Ambulance selection is done based on the factors like minimal response time and less travel time to the patient's location and from the patient's location to the hospital. Furthermore, the system selects the most suitable hospital based on the patient's health condition and the available services of the hospital. In order to speed up the transportation of the patient, the system incorporates special functions like an "emergency corridor" for emergency service vehicles to pass through in traffic conditions by adjusting the traffic-control operations.

These ambulances are equipped with patient health monitoring systems to constantly monitor the patient's health conditions like pulse, blood pressure, etc. The patient's health condition will be updated online to selected hospitals and monitoring stations for them the make necessary preparation without any delay.

Emergency vehicles' engines status is also monitored and technical problems are automatically notified to the control centre so they can make fast arrangements for a substitute vehicle if a vehicle breakdown happens.

This system maintains the history logs like patient's medical records, ambulance service records, etc. to provide efficient service."

Propose a design of the SAS by executing the IoT services to bring emergency healthcare services to save more human lives. Accordingly, answer the following questions.

[Q1]

(i) Draw a diagram and briefly explain the end-to-end architecture of the proposed SAS IoT application. Clearly indicate what the edge devices, gateways, connectivities, cloud services, and applications are.

[5 Marks]

(ii) Draw separate flowchart(s) to represent each algorithm for the Ambulance dispatch operation and select a suitable hospital for the patient after dispatching.

[10 Marks]

(iii) Write a program using pseudo-code for two algorithms in Q1(ii) by considering their high-level operations. [10 Marks]

(iv) Draw a block diagram and show how to integrate each input and output of each module/unit in the ambulance. [5 Marks]

[Q2]

(i) Identify three security concerns imposed on the proposed system. [6 Marks]

(ii) Briefly explain what steps could be taken to mitigate identified concerns. [9 Marks]

[Q3]

Design the following dashboard to visualise the data in the SAS IoT application.

(i) Mobile interface for ambulance and patients. [7 Marks]

(ii) PC interface for the doctors and monitoring station to monitor data. [8 Marks]

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

[Q4]

Estimate the power consumption for the following description. (Hint: by considering the ESP32- power modes of deep-sleep, modern-sleep, active, etc..).

(i) Consider a proximity sensor node in a vehicle sending a distance measurement of the vehicle ahead **once every 1 second** using an ESP32 single-core module. Let's assume **data is transmitted within 650**ms. The system is implemented with different ESP32-power modes, including **deep sleep**. The proximity sensor will take 25ms **to perform a single measurement** and consume **15mA** in its typical operation. Draw a power profile (Chart of power consumption vs time indicating each operational stage). Assume that system instantaneously transitioned from sleep to wake. [5 Marks]

(ii) Calculate the duty cycle (System operational time period) in Q4(i). [3 Marks]

(iii) Compute the energy consumption for one-month duration in Q4(i). (Hint: E= sleep energy x sleep time +active energy x active time) [6 Marks]

(iv) Let's assume the vehicle measures the distance using four(4) sensors from four(4) directions and it uses ESP32 dual-core module. Propose a power optimization method for the same operation in Q4(i) for the new system and estimate new energy consumption for a one-month period. [6 Marks]

[Q5]

(i) Briefly explain what Industrial IoT(IIOT) is and how it differs from consumer IoT.

[4 Marks]

(ii) List six(6) challenges faced when applying the IoT concept to the SAS application.

[6 Marks]

(iii) Write an Arduino IDE program that determines the distance to an object using Ultrasonic Sensor. The Sensor has below inputs and outputs. State the comments where necessary.

- Trigger Pulse Input – Trigger Pin of Ultrasonic Sensor.
- Echo Pulse Output - Echo Pin of Ultrasonic Sensor.

[10 Marks]

[Q6]

"Emergency Corridor(EC)" is activated by detecting the smart ambulance within a range of wireless sensor networks (WSN) located near the traffic signals. It ensures traffic controllers respond quickly in emergency situations. The EC overrides the traffic control operation by allowing the emergency vehicle to pass through. RF communication technology is used for detecting Emergency vehicles.

(i) Draw a block diagram of a wireless sensor node and briefly describe the operation of each component of the diagram.

[6 Marks]

(ii) Each sensor node in the network requires a buffer/storage. It is required due to the hop-by-hop transition of data when multiple sensor nodes send data to other sensor nodes. Draw a diagram to show how the buffer/storage interfaces with a sensor nodes. Select a suitable interfacing technique.

[4 Marks]

(iii) Briefly explain how the wireless interconnection architectures are used in proposed Emergency corridor sensor networks (ECSN).

[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

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
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in 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;  // ERROR! Indexes
                 // are 0 though 5
```

Built-in Functions

```
Pin Input/Output
Digital I/O - pins 0-13 A0-A5
  pinMode(pin,
    {INPUT|OUTPUT|INPUT_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(pin, 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()
  // Overflows 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)
constrain(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
attachInterrupt(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 a time.
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 - I²C communication
begin()     // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```