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

035

Study Programme	: Bachelor of Technology Honours in Engineering/ Bachelor of Science Honours in Engineering
Name of the Examination	: Final Examination
Course Code and Title	**: EEX5346 Embedded Systems**
Academic Year	: 2022/23
Date	: 15th February 2024
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 **three (3)** pages, from page 2 to page 4.

3. Answer all questions in SECTION A.[60 Marks], 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 on 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 Water Management System (SWMS)**. Your task is to analyse the following specifications and design the **SWMS** *by Providing IoT solutions for a smart city.*

Water is a crucial necessity for every aspect of life. Inadequate water management poses risks and impacts everyone. The growing populations in agriculture and industries are putting more strain on water resources, especially with climate change affecting water levels. Recognising this challenge, utilizing technological advancements like IoT for water supply management can prove beneficial. This approach can aid in efficiently handling our limited water resources and ensuring their equitable availability to all.

It is recommended to propose an IoT-based smart water management system for overseeing the consumer water supply in Sri Lanka, involving approximately 4 million connected households. This system aims to offer features like real-time consumption monitoring, early detection of leaks or irregular patterns, and the generation of alerts for swift actions. Additionally, the system is designed to automate both private and public lawn(i.e. parks) watering, scheduling watering during periods of lower water consumption or in compliance with municipal water restrictions.

On a city-wide scale, IoT technology can effectively manage water supply equipment, providing status reports on factors such as open/closed gates, operational status, reservoir levels, and output speed versus input. The system allows remote and automatic control of gates or pumps in real-time based on a variety of flow analytics data. Vibration measurements can be utilized to identify and predict potential equipment failures, enabling proactive dispatch of repair teams before any breakdown occurs. These operational enhancements directly contribute to increased efficiency. The data collected from the central hub is utilized to predict future water usage, enabling authorities to plan water distribution based on anticipated demands.

Propose a design for the **SWMS** by executing an IoT solution to provide water to everyone and reduce risk in all aspects of life. Accordingly, answer the following questions.

[Q1]

(i) Draw a diagram and briefly explain the IoT architecture of the proposed **SWMS** solution. Clearly indicate what the edge devices, gateways, connectivities, cloud services, and applications are. [13 Marks]

(ii)

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

a. Dispatching an appropriate repair team to address issues such as water leakage or water supply distribution to respond to emergency repair operations.
b. Scheduling lawn watering during periods of reduced water consumption.

[12 Marks]

(iii) Draw a block diagram and show how to integrate each input and output of each module/unit in the consumer and water distribution ends.

[5 Marks]

[Q2]

Design the following user-friendly interface to visualise the water consumption in real-time, set consumption thresholds, and receive notifications/alerts about leaks or unusual usage patterns.

(i) Mobile interfaces for consumers and repair team. [7 Marks]

(ii) PC dashboard for the authority and central monitoring station to control and monitor the water distribution process. [8 Marks]

[Q3]

(i) Identify three (03) security concerns imposed on the proposed system.

[6 Marks]

(ii) Briefly explain what steps could be taken to mitigate identified concerns.

[9 Marks]

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

[Q4]

(i) Briefly describe the operation of the MQTT broker. [4 Marks]

(ii) Create a flow chart and briefly describe the process of measuring water flow in a pipeline. Use a water flow sensor that generates digital output in litres/minute, and employ an ESP32 Wi-Fi module to publish this data to an MQTT broker.

[6 Marks]

(iii) Refer to the Arduino programming instructions and write a program for Q4.(ii) Process. State the comments where necessary [10 Marks]

[Q5]

(i) List two (02) short range and two (02) long range wireless technologies and compare their range, cost, power, bandwidth, and throughput for the proposed **SWMS** solution in section A. [4 Marks]

(ii) Select a wireless interconnection architecture for the proposed **SWMS** and justify your selection. [6 Marks]

(iii) Briefly explain the choice of persistent storage and interfacing technique used for the **SWMS** sensor node. [4 Marks]

(iv) Compare and contrast the general purpose Operating System and Real Time Operating System. [6 Marks]

[Q6]

The Water Control Embedded System (**WCE**) in the **SWMS** offers functionalities, such as monitoring water pressure, detecting free-running (unintentionally left open) water taps, identifying water leakages, and more. It is designed to generate alerts in response to these conditions. Accordingly, answer the following questions.

(i) List the sensors and controllers required for the **WCE** system and justify your selection. [5 Marks]

(ii) Draw a hardware architecture for the **WCE** system and show how sensors and other components interact with the ESP32 microcontroller. [5 Marks]

(iii) Considering power optimization, compute the energy consumption for the **WCE** system over one month. Assume that microcontroller module undergoes an immediate transition from sleep to wake, transmitting data within 530ms. Each sensor requires 15ms for a single measurement and consumes 20mA during typical operation. Refer ESP32 datasheet, if required.
(*Hint: E= sleep energy x sleep time + active energy x active time*) [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

Annexure

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;       // ERROR! 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)