COMP4447/COMP5047 Pervasive Computing

Practice Exam 2023

EXAM CONDITIONS:
This is a RESTRICTED OPEN book exam - specified materials permitted

MATERIALS PERMITTED IN THE EXAM VENUE:
(No electronic aids are permitted e.g. laptops, phones)

One A4 sheet of handwritten and/or typed notes double-sided

MATERIALS TO BE SUPPLIED TO STUDENTS:
None

INSTRUCTIONS TO STUDENTS:

- Answer all questions in the spaces provided on this question paper.
- Additional empty spaces are provided at the end of the booklet. If you use these spaces, please state the exercise and question number clearly.
- Return this question paper and any additional answer booklets or materials.
- Write or draw your final answers in black or blue ink, not pencil.
- Take care to write and draw legibly.
- Please check your examination paper is complete (page numbers in the footer of the page) and you have written your student ID.

Please tick the box to confirm that your examination paper is complete. ☐

Exercise 1. Multiple Answer Questions **2 x 3 = 6 points**

Please select only the correct answers, a maximum of 2 answers can be correct. **Incorrect answers get a -1 point (total limited to 0 points)**.

1. When you drive an LED with your microcontroller, a series resistor must be carefully chosen so that:

   - ☐ You use the correct sampling frequency
   - ☐ You use the correct PWM frequency
   - ☐ You do not make a short circuit
   - ☒ You do not overdraw current

2. Why do we call Mark Weiser's "The Computer for the 21st Century" a seminal paper?

   - ☒ It has impacted so many other research fields
   - ☐ It accurately predicted how exactly the technology work today
   - ☒ It defined the field "Ubiquitous/Pervasive Computing"
   - ☐ It describes cool research

3. With respect to the basic structure for IoT, please select statement(s) that are correct.

   - ☐ Storage can only be connected through the management system.
   - ☒ Connections between IoT objects are part of the network.
   - ☐ Administrators are always given a command line interface and users get a graphical interface.
   - ☒ Edge computing creates a distributed system.

Exercise 2. Evaluation

2 + 4 = 6 points

You have developed a mobile app that displays the step count of users using a wearable sensor. Sensor data is automatically sent to a cloud server where an AI system is used to track the number of steps. The app is meant to show step counts, which are organized into the count per day, per month, week, or even hourly. For this app, you are tasked to do a think-aloud test.

1. Write an abstract task.

   Abstract task: Find step count using the mobile app

2. Write two (2) concrete tasks.

   Concrete task: Find the number steps you walked yesterday

   Find the step count for last hour

Exercise 3. Interfaces

2 + 6 = 8 points

Consider the interface available in a Virtual Reality Environment. Correctly identify 1 interaction style and briefly explain why it describes it the best (max 50 words).

1. Which interaction style best describes this interface? Clearly state 1 interaction style.

   Interaction style: Exploring

2. Briefly explain why it describes it the best (max 50 words).

   Description: The interaction involves a user moving through a virtual space similar to how we explore things in real world.

Note: Sometimes there can be more than one interaction type/style that works well any of those will be correct if explained properly.

Exercise 4. Pervasive Application **10 + 14 + 14 = 38 points**

A simple pervasive system is needed to measure the direction of the sun to control a solar panel.

The system components consist of 1) an Arduino-based microcontroller FireBeetle ESP32 E Board connected to the internet via a local Wi-Fi network, 2) an analog sensor that measures the angle of the sun, 3) a node-red flow to visualize data, and 4) an MQTT broker. The functions and characteristics of the system are:

- The analog sensor measures the direction of the sun.
- The FireBeetle should use the analog sensor to get the direction of the sun and post it to the MQTT broker.
- Node-red flow should read the sun-direction values from the MQTT broker and convert it to an appropriate angle value and display it in the Node-RED debug.

The following design considerations and specifications need to apply to the system:

- The direction of the sun is measured using the analog sensor.
  - The ADC interface has a resolution of 8 bits and operating at 3.3V.
  - The full range of analog interface 0V to 3.3V represents angles 0 to 180 degrees.
  - The sensor is connected to the FireBeetle using wires.
  - Measurements need to be made every 10 seconds.
- The local Wi-Fi network's SSID is "solarWiFi" and the password is your "soalar2023".
- The MQTT broker is operating at the IP "172.23.31.4" port 1888.
  - You can connect to the MQTT broker anonymously.
  - The raw sensor values should be published to the topic "sundirection".
- The Node-RED flow should read the sensor values from the MQTT broker and output them to the debug.
  - In the debug, sun direction should be converted to an angle range (in degrees).

Please use the above specifications to answer following questions.

Exercise 4.1: Draw a block diagram of how the four (4) components of the system are interconnected, clearly stating the network or connection types used, and connected ports (e.g, Serial port, SPI, ..., Bluetooth, Wi-Fi, A0, A1, .., D0, D1,...).

- You should hand draw this in the provided square space below, the quality of the drawing does not matter, but the clarity does.
- The components and connections should be properly labelled.
- You can choose which device should connect to which port as part of your design decisions.

Please refer to how you draw this in your projects.

Exercise 4.2: Write an Arduino program for *the micro-controllers that read the sun direction from the sensor and post them to a topic in the MQTT broker*. You can use the common functions given in Figure 4.1 to build on.

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()
```

Figure 4.1: A set of commonly used Arduino functions that you may find useful. The list is adapted from: `https://github.com/liffiton/Arduino-Cheat-Sheet/`

- Assume the example sketch given below is your `main.cpp`.
- You can assume the libraries `WiFi.h` and `MQTT.h` are available to you.
- The ports, pins, and methods each component connects are up to you, and you should configure them accordingly.
- Sensible comments where applicable will get points for the correct logic if the code is incorrect.
- You can assume both WiFi and MQTT connections are established accurately within 500ms.
- You may delete the parts of the example sketch by striking through (e.g., ~~deleted~~) that are irrelevant to your code.
- If a line needs modification, you can strike through that line and write it with appropriate changes in the space provided between lines.
- Where new lines of code are necessary, you can integrate them into the space of the sketch.
- If you run out of space within the sketch, you can indicate the line numbers and write new lines of code in the empty page provided.

Please check lecture video on guidance on how to answer this point onwards.

```
// Based on various open source materials
// Sample networked application
// Author: Anusha Withana

#include <WiFi.h>
#include <MQTT.h>



IPAddress mqttHost(172, 23, 31, 4);



WiFiClient wifiNet;
MQTTClient mqttClient;

// WiFi setting
const char ssid[] = "YourWifiHotspotName";


const char pass[] = "YourWifiHotspotPW";



void messageReceived(String &topic, String &payload);


unsigned long lastTime = 0;


void setup() {
  Serial.begin(115200);

  pinMode(A0, INPUT);







  Serial.println(ssid);
  WiFi.begin(ssid, pass);



  while (WiFi.status() != WL_CONNECTED) {
      delay(500);
  }
  Serial.println(WiFi.localIP());



  mqttClient.begin(mqttHost, wifiNet);
  mqttClient.onMessage(messageReceived);

```

```
56
57    while (!mqttClient.connect("esp32-mqttClient")) {
58      delay(500);
59    }
60    Serial.println("\nMQTT connected!\n");
61
62
63
64    mqttClient.subscribe("/hello");
65
66
67
68  } // end of function: setup
69
70
71
72  void loop() {
73    mqttClient.loop();
74
75
76
77
78
79
80
81
82
83
84    // publish a message roughly every second.
85    if (millis() - lastTime > 1000) {
86
87
88
89
90      lastTime = millis();
91
92
93
94
95
96      mqttClient.publish("/hello", "world");
97    }
98
99
100 } // end of function: loop
101
102
103 void messageReceived(String &topic, String &payload) {
104   Serial.println("incoming:" + topic + "-" + payload);
105
106
107
108
109 } // end of function: messageReceived
```

Use this space to write lines of code if you ran out of space above. P***lease make sure you mention the line number where the code needs to be inserted into.***

Exercise 4.3: Sketch a Node-RED flow for the above application. You can use the Node-RED nodes listed in Figure 4.2. You can use other nodes if you remember them. Please make sure each node is appropriately connected and the relevant configuration details are accurately listed in your sketch. You can also label the nodes as (A), (B), etc. and list the configuration details on the next page to save space.

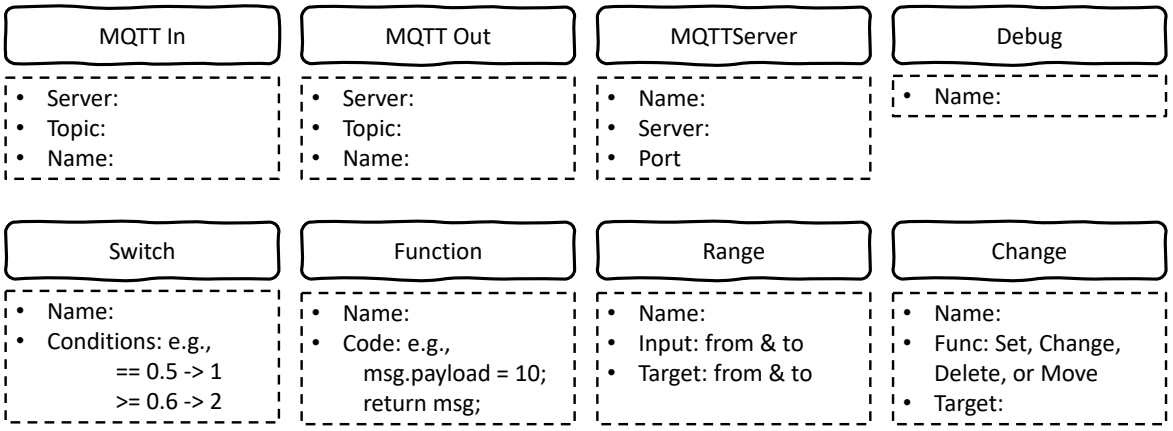

Figure 4.2: A set of example Node-RED nodes, with relevant configuration details.

You can use these two pages for ***rough work or as additional space for answers***. If you do use this as additional space, please ***clearly state the exercise number and the question number***. Otherwise, your answers in this area will not be graded.

END OF EXAMINATION