NATIONAL UNIVERSITY OF SINGAPORE

IS4151/IS5451 – AIoT Solutions and Development

(Semester 2: AY2024/25)

MOCK PRACTICAL EXAM

Time Allowed : 2 Hours 30 Minutes

(with additional 10 Minutes Reading Time)

INSTRUCTIONS TO STUDENTS

1. This assessment paper contains **SIX** (6) tasks and comprises **SEVEN** (7) printed pages, including this page.
2. This is an **OPEN BOOK** assessment.
3. At the end of the assessment, return this assessment paper in its entirety. Do **NOT** retain or redistribute the assessment paper.
4. This assessment is conducted with your own laptop, i.e., Bring Your Own Device (BYOD). It is your responsibility to ensure that your laptop has been set up according to the IS4151/IS5451 software requirements.
5. During the 10 minutes reading time, you are **NOT** allowed to commence typing.
6. All required resources can be found in the zip archive file **MOCKPE.zip** that can be downloaded from the Canvas Files tool folder: Assessments > Mock Practical Exam.
7. Place all the source files into a single ZIP archive file and name the ZIP file using your NUS student number in the format "AXXXXXXXX.zip". Upload this ZIP file to the Canvas Assignment: Assessments > Mock Practical Exam at the end of the assessment.
8. You are required to arrive at your designated venue 20 minutes before the start of the assessment from 8:40 am onwards to set up your laptop.

STUDENT NUMBER:

This portion is for examiner's use only

Task	Maximum Possible Marks	Marks Obtained	Remark
A	2		
B	2		
C	6		
D	4		
E	3		
F	3		
Total	**20**		

Opening Narrative

In this series of six tasks, you will be building a smart AIoT system that trains a clustering model using ambient brightness data obtained from a light sensor to determine whether the current ambient environment is bright or dim. The clustering model is then used to construct a control application for a smart lighting system that automatically turns the light on and off depending on the ambient brightness threshold determined by the clustering model.

Task A – Wireless Node Device (2 marks)

1. Design and implement a 2.4 GHz radio wireless node device using a micro:bit that is attached with a Grove light sensor module via a Grove shield. The Grove light sensor module measures the ambient brightness and returns an analogue value in the range of 1-650.

2. The node is to be associated with another micro:bit acting as a 2.4 GHz radio wireless node controller via a handshaking process and thereafter reports its ambient brightness upon request.

3. The node device is to complete the handshaking process by responding to a "handshake" command sent by the controller.

4. After handshaking, the node device is to report its ambient brightness measured by the Grove light sensor module by responding to a "sensor=light" command sent by the controller.

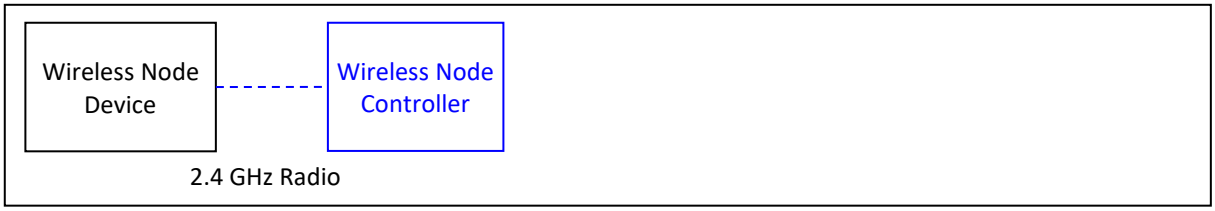
5. The current ambient brightness should be displayed on the 5x5 LEDs of the micro:bit at all times using a graph with a max value of 650.

Task B – Wireless Node Controller (2 marks)

1. Design and implement a 2.4 GHz radio wireless node controller using a micro:bit that is to be attached to a Raspberry Pi via wired serial USB. In theory, the controller must be able to control and interact with multiple node devices. But for the purpose of this task, you only need to implement one node device.

2. The controller is to receive a "handshake" command from the Raspberry Pi and relay the "handshake" command to micro:bit devices in the same radio group that are within maximum transmission range. Thereafter, the controller should wait and collect the enrolment data before sending back to the Raspberry Pi.

3. After the handshaking process has been completed, the controller is to receive a "cmd:sensor=light" command from the Raspberry Pi at periodic interval. For each command received, the controller is to relay the "sensor=light" command to the node devices that are associated with it and then collect the senor data before sending back to the Raspberry Pi.



Task C – Integrated Hub (6 marks)

1. Design and implement an integrated hub using a Raspberry Pi that performs the three main functions of i) 2.4 GHz wireless radio hub; ii) sensor data relay to the cloud server; and iii) smart light actuation.

2. **2.4 GHz wireless radio hub** – After the integrated hub has initialised, it should initiate the handshaking process with 2.4 GHz radio wireless node devices that are in the same radio group and within maximum transmission range. This done by sending the "handshake" command to the 2.4 GHz radio wireless node controller via wired serial USB. The integrated hub will then wait for the controller to send back the device name of all nearby node devices.

   After the handshaking process has completed, the integrated hub will transition to sensor data collection mode at 1 sec interval. This is done by sending the "sensor=light" command to the controller and then wait for the controller to send back the sensor data of all nearby node devices.

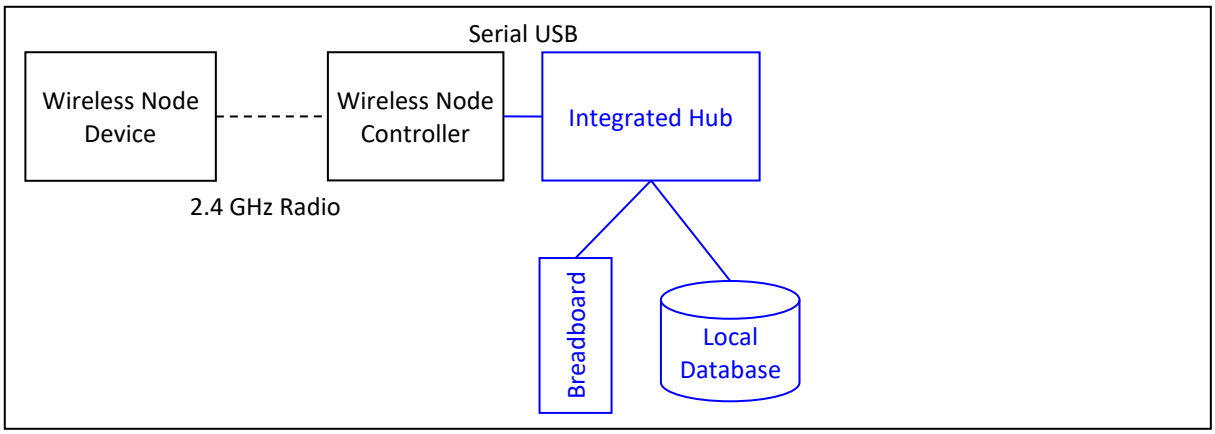
   The sensor data are to be persisted into a local relational database such as SQLite or MySQL on the Raspberry Pi with the following columns:

   - id – INTEGER, primary key
   - devicename – CHARACTER(5)
   - light – INTEGER, actual value of the Gove light sensor module
   - timestamp – DATETIME, current timestamp at the juncture the sensor data is inserted into the database table.

3. **Sensor data relay to the cloud server** – The integrated hub should relay the sensor data in the local database to the cloud server, i.e., your laptop, at 10 sec interval using REST web service hosted on the cloud server. Only new sensor data should be relayed.

4. **Smart light actuation** – The integrated hub should use the MQTT protocol to receive the smart actuation command published by a control application running on the cloud server at a suitable topic with a public broker.

   If the "on" command is received, a red colour LED on the breadboard attached to the Raspberry Pi should be turned on steady. If the "off" command is received, the red colour LED should be turned off steady.



Task D – Cloud Server 4 marks)

1. Design and implement a cloud server using your laptop that performs the two main functions of i) publishing global sensor data; ii) exposing REST web service.

2. **Publishing global sensor data** – The cloud server should publish all the global sensor data retrieved from the relational database on the laptop in the form of a webpage sorted in descending order of id. The following attributes should be shown in the web page:

   - id
   - devicename
   - light
   - timestamp

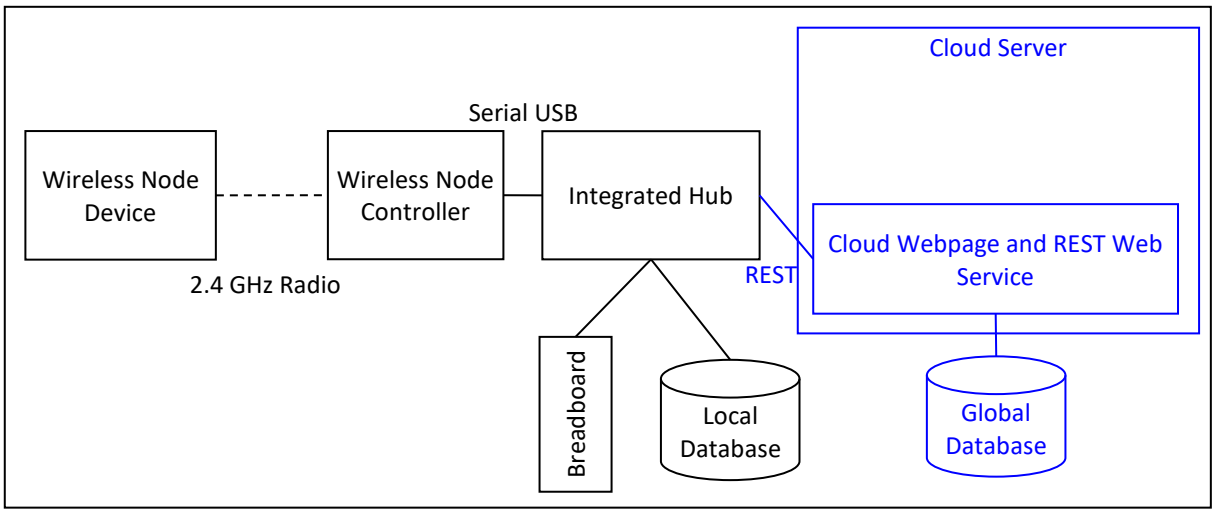
   The webpage should ideally auto refresh by itself at 5 sec interval.

3. **REST web service** – A total of three REST web service methods are required.

   A method for a caller to retrieve all the global sensor data retrieved from the database consisting of the same four attributes.

   A method for a caller, in this case the integrated hub on the Raspberry Pi, to relay sensor data to the cloud server consisting of the same four attributes. The sensor data are to be persisted into a global relational database such as SQLite or MySQL on the laptop with the same four columns.

   A method for a caller, in this case the control application, to pass in an ambient brightness value (1-650) and obtain a cluster label of either “0” or “1” using a pretrained cluster model that is loaded from a file.

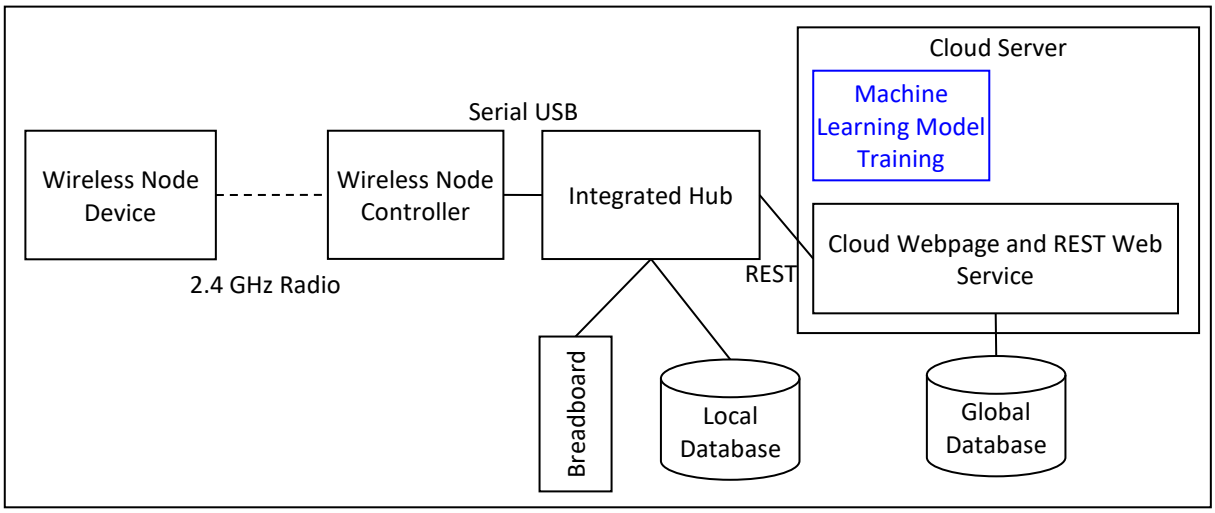


Task E – Machine Learning (3 marks)

1. Retrieve all the global sensor data from the database to train a clustering model using only the sensor data, i.e., the ambient brightness value originally harvested from the Grove light sensor, with K-means ($K = 2$).

2. In order to determine which cluster corresponds to low/high ambient brightness, you should compute the within cluster mean and standard deviation of brightness.

3. Store the trained model into a file such that it can be loaded back to predict the cluster label of a new ambient brightness values. This model will be deployed via the REST web service of the cloud server as per Task D.3.



Task F – Smart Control Application (3 marks)

1. Design and implement a control application that is to be run on the cloud server, that is, your laptop.

2. The control application is run from the terminal with an input argument representing a new ambient brightness value.

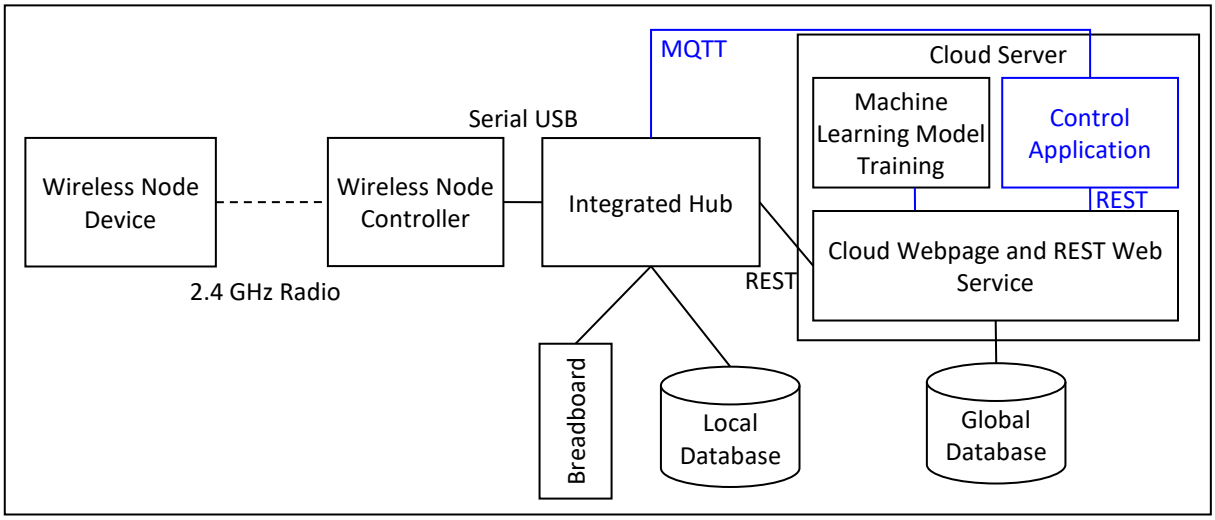
In Python, the following code fragment can be used to retrieve the input argument:

```
import sys
arg = sys.argv[1] # 0 is the name of the Python program
```

The control application should then call the web service method in Task D.3 to obtain the cluster label.

3. Depending on whether the cluster label corresponds to low/high ambient brightness, the control application should send the appropriate command to the integrated hub via the MQTT protocol at a suitable topic with a public broker:

    - Low ambient brightness – "on" command
    - High ambient brightness – "off" command



- End of Paper -