Assignment 2 - Book Inventory System

ECE 4564

October 8, 2018

Contents

1 Rubric

- 5 pts - All Bluetooth Mac addr/ip addr are not hard coded into the code
- 5 pts - Use correct MongoDB element data format
- 5 pts - Serialize payload by JSON
- 5 pts - Encapsulated LED/MongoDB functions into an individual class LED.py/MongoDB.py
- 5 pts - Print checkpoints wirh corrct format and required info
- 20 pts - Client can send request payload and get answer payload to processor via RabbitMQ
- 15 pts - Processor can send request and get answer payload to storage via Bluetooth socket connection
- 20 pts - MongoDB
  - 4 pts - Add new item to mongodb
  - 4 pts - Detele item from mongodb
  - 4 pts - Increase/Decrease item stock number
  - 4 pts - List out all database items
  - 4 pts - Get item stock number
- 20 pts - GPIO
  - 15 pts - Use RGB LED to display how many kind of books' information in database **(Not total number of book)**
  - 5 pts - LED display can running with MongDB simultaneously (Multi-thread needed)
  - 5 pts - Turn off all LEDs after program is terminated

2 Guideline

2.1 Required Files

- MongoDB.py
- LED.py
- storage.py
- processor.py
- client.py

2.2 Important Note

- All mongodb funcitons(create, add, delete, find, count, etc.) should be encapsulated in a class inside MongoDB.py.
- All LED functions(RGB LED controlling, etc) should be encapsulated in a class inside LED.py.
- **Do not hard code any absolute path, IP addr, Bluetooth MAC Addr Inside of your code**
- Please clearup your mongodb database everytime you terminate the storage program. If not, your validation may be affected.
- Please taking your Pi with you during validation. You'll lose 5 pts if don't have your Pi
- You'll lose 10 pts for late submission

2.3 CMD Parameters Format

- Client
  - **ADD**: Add book info into storage.
    client.py -proc <Processor IP Addr> -action ADD -book <Book Info>
  - **BUY**: Buy more books.
    client.py -proc <Processor IP Addr> -action BUY -book <Book Info> -count <Bought Count>
  - **SELL**: Sell books.
    client.py -proc <Processor IP Addr> -action SELL -book <Book Info> -count <Sell Count>
  - **DELETE**: Delete book info from storage
    client.py -proc <Processor IP Addr> -action DELETE -book <Book Info>
  - **COUNT**: Get specific book's count in stock
    client.py -proc <Processor IP Addr> -action COUNT -book <Book Info>
  - **LIST**: List all book info inside storage
    client.py -proc <Processor IP Addr> -action LIST
- Processor
  processor.py -storage <Storage Bluetooth Mac Addr> -p <Storage Port Number> -z <Socket Size>
- Storage
  storage.py -p <Port Number> -b <Backlog> -z <Socket Size>

2.4 Book Info Format

- Input Type: JSON String
- Sample
  '{"Name": "The Sign of the Four", "Author": "Arthur Conan Doyle"}'

2.5 Get RPi Bluetooth MAC Addr with Python3

```
from subprocess import check_output
def get_bluetooth_mac_addr():
    addr_info = str(check_output(["hcitool", "dev"]), "UTF-8")
    chunks = addr_info.split('\t')
    mac_addr = chunks[-1][:-1]
    return mac_addr
```

2.6 Bluetooth RFCOMM

- Port Number 1 - 30

2.7 Element Data Format Inside MongoDB

- Format

```
{
    "Name": "YYYY",
    "Author": "XXXX",
    "stock": <Number in stock>
}
```

- Sample
  {"_id" : ObjectId("5baecbdf74fece788cc46a8b"), "stock" : 0, "Name" : "To Kill a Mockingbird", "Author" : "Harper Lee" }

- Note:
  "_id" is generated by MongoDB

2.8 Payload Format

Format: **Python Dictionary**

- Request Payload
  - ADD/DELETE/COUNT

```
{
    'Action': <Action>,
    'Msg': {
        'Book Info': {
            'Name': <Name>,
            'Author': <Author>
        }
    }
}
```

  - BUY/SELL

```
{
    'Action': <Action>,
    'Msg': {
        'Book Info': {
            'Name': <Name>,
            'Author': <Author>
        },
        'Count': <COUNT>
    }
}
```

  - LIST

```
{
    'Action': <Action>,
    'Msg': {}
}
```

- Answer Payload
  - ADD/BUY/SELL/COUNT/DELETE

```
{
    'Msg': <Response Msg>
}
```

  - LIST

```
{
    'Msg': <Response Msg>,
    'Books': [<Book 1>, <Book 2>....]
}
```

  - Response Msg
    - Without Error
      - ADD - OK: <Added Book Info> <MongoBD assigned Book ID>

- BUY/SELL - OK: <Book Info with Current Book Count>
- COUNT - OK: <Current Book Count>
- DELETE - OK: <Book Info>
- LIST - OK: <# of Book Varieties>

* With Error
  - BUY/DELETE/COUNT - ERROR: <Error Info (Book doesn't exist)>
  - SELL - ERROR: <Error Info (Book doesn't exist/No enought book)>

For more details, please take look of provided sample files.

3 Checkpoints

Checkpoint Format: **[Timestamp] Checkpoint Message**

- Storage

```
[Timestamp] Created socket at <Bluetooth MAC Addr> on port <Port Number>
[Timestamp] Listening for client connections
[Timestamp] Accepted client connection from <Bluetooth MAC Addr> on port <Port Number>
[Timestamp] Received Payload: <Question Payload>
[Timestamp] Answer Payload: <Answer Payload>
```

- Bridge

```
[Timestamp] Created rabbitmq at 0.0.0.0
[Timestamp] Awaiting client requests
[Timestamp] Received request payload: <Request Payload>
[Timestamp] Connecting to <Bluetooth MAC Addr> on port <Port Number>
[Timestamp] Received answer payload: <Answer Payload>
```

- Client

```
[Timestamp] <Check if action is valid>
[Timestamp] Request Paylaod: <Request Payload>
[Timestamp] Response: <Response Msg>
```

4 Sample

See sample files