VirginiaTech
Invent the Future

Assignment 2

Book Inventory System

Due Date: Sunday, October 21, 2018 @ 23:55

ECE 4564 - Network Application Design

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

Learning Objectives

Message Broker
- AMQP Protocol
- RabbitMQ
- Direct Exchange

Callback Routines

JSON

Python Encapsulation

Multi-threading

Data Persistence
- noSQL Database (MongoDB)

Bluetooth

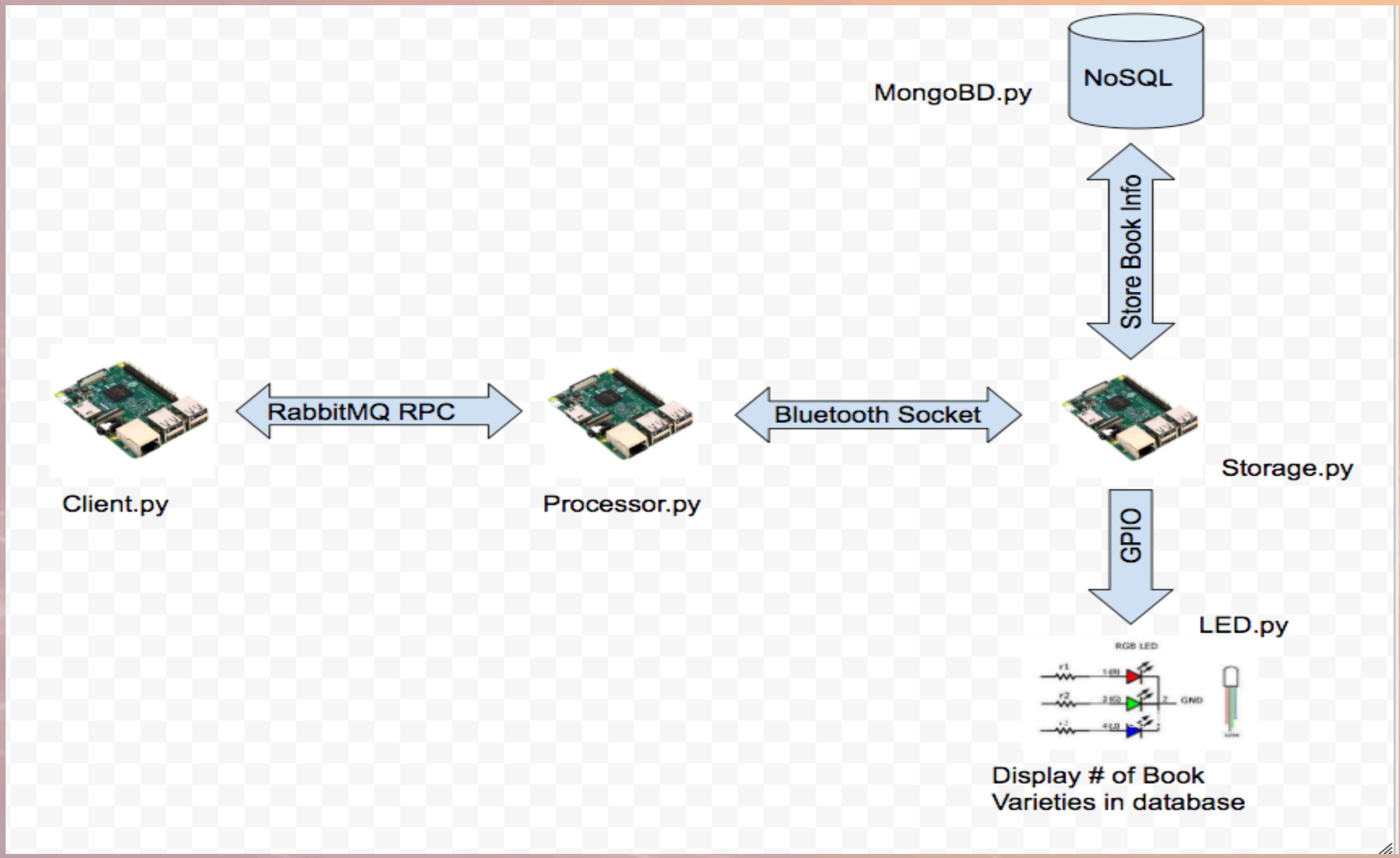
Raspberry Pi GPIO

Book Inventory System

System supports

- Sending Book Info to client via Bluetooth and RabbitMQ
- Accepting Book Info from client over Bluetooth and RabbitMQ

System Overview



Assignment Overview

- Client Rpi (client.py)
  - Uses RabbitMQ RPC to send and receive messages from Processor RPi
  - Allows user to control storage info by proper CMDs (ADD/BUY/SELL/COUNT/LIST/DELETE)

- Processor RPi (processor.py)
  - Setup RabbitMQ RPC server receives and sends message to Client Rpi.
  - Uses Bluetooth Serial over RFCOMM to send and receive messages to Storage RPi

Assignment Overview

- Storage Rpi (storage.py, MongoDB.py, LED.py)
  - Service that manages
    - Messages sent from processor RPi
    - Message requests to processor RPi
  - Indicates # of book varieties in storage using RGB LED
  - Setup Bluetooth server to receive and sent msgs to Processor Rpi.
  - Maintains all book info in local NoSQL database (MongoDB).

Important Note

- Encapsulation
  - MongoDB related functions (Add book info, increase/decrease stock count, etc.) should be encapsulated in a class inside MongoDB.py file.
  - LED related functions should be encapsulated in a class inside LED.py file.
- Multi-thread
  - Stoarge Rpi should be able to process msg cmd and display book variety number simultaneously.
- MongoDB – will be used to check data format during validation
  - Database name: db
  - Collection name: collection

Client RPi

Client RPi

Processor RPi

RPC

Messages via RabbitMQ

- The client RPi sends CMDs and receives replies from Processor Rpi(bridge) via RabbitMQ
- Message format: Python Dictionary
- Client CMDs:
  - ADD
  - BUY
  - SELL
  - DELETE
  - COUNT
  - LIST

Client Rpi Request Payloads

- – ADD/DELETE/COUNT

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

Client Rpi Request Payloads

- BUY/SELL

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

- LIST

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

Client Rpi Received Answer Payloads

- ADD/BUY/SELL/COUNT/DELETE

{

' Msg ' : <Response Msg>

}

- LIST

{

' Msg ' : < Response Msg>,

’ Books ' : [<Book 1 Info> , <Book 2 Info>. . . .]

}

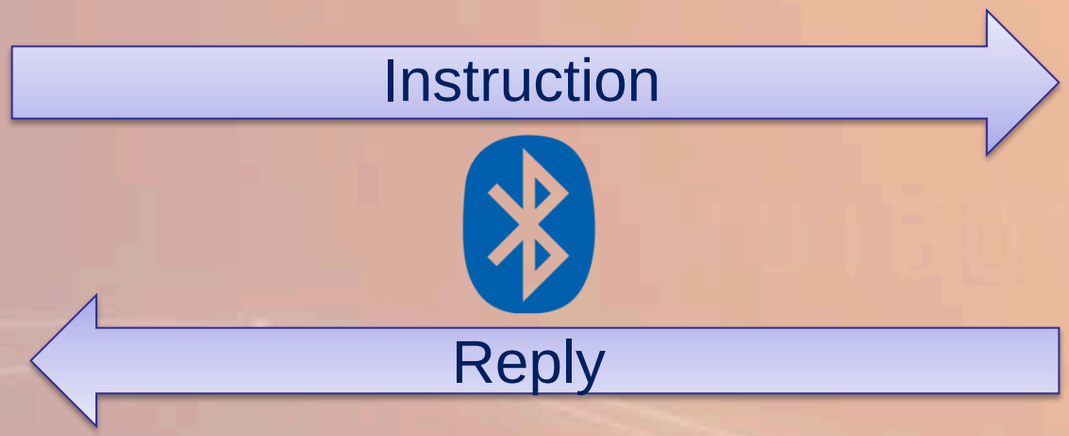
Answer Payload Response Msgs

- 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 the provided sample files.

Processor Rpi

Processor RPi



Storage RPi

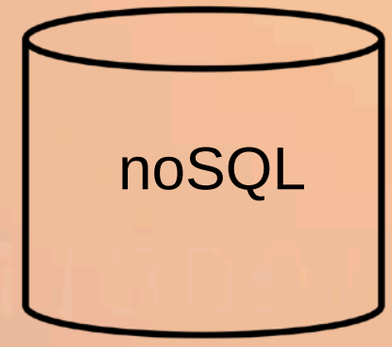
- Filename : processor.py
- Receive message instruction from client Rpi via RabbitMQ
- Sends message instruction to the Storage Rpi via Bluetooth
- Receives replies from the Storage RPi via Bluetooth
- Sends reply to Client Rpi via RabbitMQ

RabbitMQ

- Need to use RabbitMQ RPC (callback queue)

```
Callback queue Example:
result = channel.queue_declare(exclusive=True)
callback_queue = result.method.queue
channel.basic_publish(exchange='',
                                routing_key='rpc_queue',
                properties=pika.BasicProperties(
                                 reply_to = callback_queue,
                               ),
                                        body=request
        )
```

Storage Rpi



- Processes message command
  - Place message in noSQL (MongoDB) database
  - Get proper book info
- Displays how many kinds of books in MongoDB with RGB LED
- Above two procedures should be able to run simultaneously (Multi-threading)

GPIO - LED

Indicates system status:

One Red Blink – 100 kinds of books in storage

One Green Blink – 10 kinds of books in storage

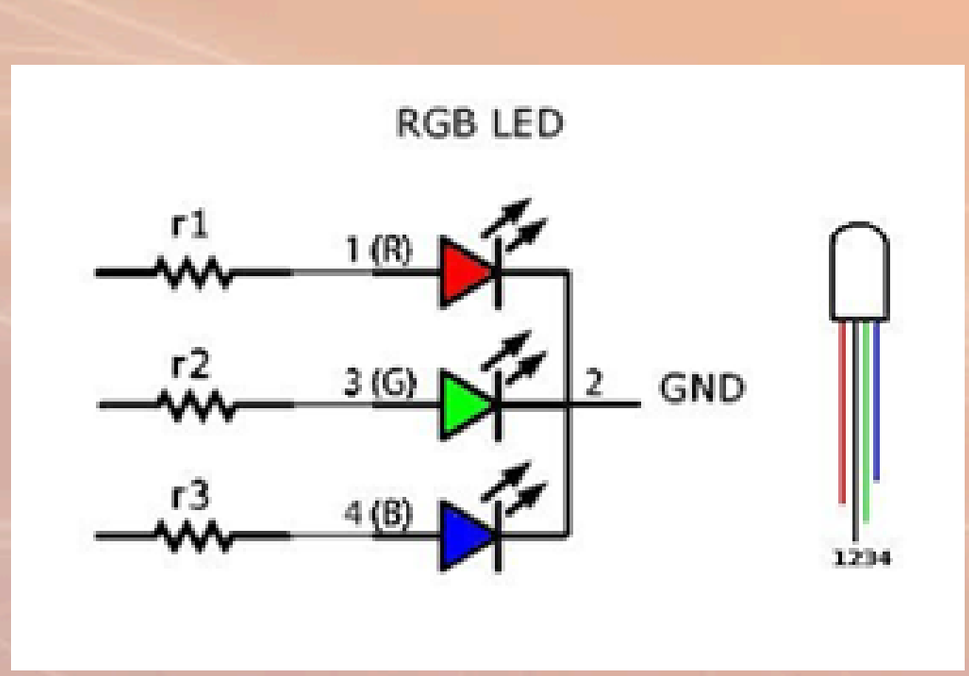
One Blue Blink – 1 kind of book in storage

For example:

# of book varieties = 12

1 green blink followed by

2 blue blink



Suggested libraries:

Rpi.GPIO

GPIO Zero

Note: use timing delays

MongoDB Element Format

Database Name: db

Collection Name: collection

MongoDB Element Format

```
{
    # "_id" is generated by MongoDB
    "_id" : ObjectId("5baecbdf74fece788cc46a8b"),
    "stock" : <BOOK COUNT>,
    "Name" : <BOOK Name>,
    "Author" : <AUTHOR NAME>
}
```

Grading

GTA will provide grading rubric

Python Style

Follow style guide PEP0008 when writing and commenting your code

https://www.python.org/dev/peps/pep-0008/

What You Turn In

All assignments must be submitted through Canvas, no later than the due date of Oct 21 2018 @ 23:55

Your assignment should be a single .tgz file which contains the following:

- All source code you wrote for this assignment
  - Client.py
  - Processor.py
  - Storage.py
  - MongoDB.py
  - LED.py
- Readme
  - System syntax
  - Libraries
  - Description of team member contributions

Assignment References

RabbitMQ Tutorials

- RabbitMQ – Callback Queue Tutorial

Python Docs

- PyMongo

MongoDB

- JSON and BSON

Bluetooth

- RFCOMM Socket

Academic Integrity

- For this assignment, it is expected that a team's work is their own.
- The code you turn in must be your own (i.e. you need to have written your assignment).
- You are allowed to copy and paste example code from other websites, but you must include a comment in your code that attributes the website you copied the code from (i.e. original author's name and URL to the original code).
- You can discuss the assignment with other teams.
- However, you cannot just tell another team the answer to a particular problem.

Final Thoughts

In many cases, engineers are expected to just make things work given a particular design constraint (e.g. software package to use or are limited to a particular hardware platform).

You will likely run into similar situations in this class while designing and implementing your assignments and project.

.

When you're stuck, try searching online for a solution. Many times others have tried something similar and documented their experiences for others to learn and benefit from

Do not publically post answers to assignments, or example code until after the assignment due date.

Contact your instructor or GTA as soon as you encounter a problem you're unable to solve.

Don't wait until right before the assignment is due.