ECE 4564 Spring 2018
Assignment 2 Rubric
Bluetooth Ordering System

===
Rubric
===

Points	Description
5 pts	No output besides checkpoint printout.
5 pts	Client, server, processor, and led are initialized with the required command line parameters.
10 pts	Proper use of "rmq_params.py" and "menu.py"
10 pts	Client, server, processor, and led run without crashing or need to restart.
10 pts	All RMQ queues and exchanges are declared with "auto_delete=True".
10 pts	No code modifications required at all during validation.
10 pts	GPIO-LED Status Indicator
15 pts	Bluetooth RFCOMM Serial Communication
25 pts	RabbitMQ Communication

==
Grading
==

- Most grading will occur through checkpoint print statements.

- Examples of input and expected output have been posted on Canvas. Take a look at them.

- The system's output does not have to match the example output *exactly*. This means the output will not be run through a regex script that requires that you have commas, periods, colons, etc. in exact locations. That being said, the closer the system gets to matching the example output, the higher the chance is of receiving a good grade.

- Notice there is an ***essential component*** to the system: Exchanging menu and order information over serial Bluetooth. If this does not work during validation, the system cannot be validated and it will most likely receive a grade of 0.

- The 10 points for "no code modification" is meant for easy one line fixes. Submissions that require code modification as a result of non-minor syntax or non-minor logic issues will receive a 0.

- Basic guideline: If it takes more than a few seconds to fix the issue, the submission will receive a 0.

- Malicious code included with a submission that breaks or takes advantage of the system performing the validations will be considered an honor code violation.

==
Submission
==

- Submissions must be "tar.gz" or ".tgz" file type. Either is fine, they are the same file types.

- Do **not** try to trick Canvas by "zipping" your submission and changing the extension from ".zip" to ".tar.gz" or ".tgz".

- Use the "menu.py" file posted on Canvas. Should be located in the same directory as "server.py"

- Use the "rmq_params.py" posted on Canvas. A copy of this should be located in the same directory as "led.py", "processor.py", "client.py", and "server.py".

- All files should be located in the root directory of the submission after "un-taring" the submission. This is important for validation.

- Include a report/readme file (txt preferred over pdf) that contains information about special libraries used or special configuration and the contributions of each group member as well as their names and emails.

==
Remote Validation
==

- Validation/grading will initially happen remotely shortly after the deadline of the assignment.

- If no major **unidentifiable** problems occur during remote validation, it will ***not*** be necessary to sign up for a validation time slot and validate in person.

- All submissions will be run on a Raspberry Pi 3. The following will be installed on the Raspberry Pi:
    - pika – pip3
    - pybluez – pip3
    - libbluetooth-dev – sudo apt-get install
    - rabbitmq-server – sudo apt-get install

- If your system requires any other library besides the ones above, please send the GTA an email before the deadline and include that information in the report/readme file.

==
In Person Validation
==

- In person validation time slots (15 mins each) will also be posted shortly after the deadline.

- If major **<u>unidentifiable</u>** issues occur during remote validation, expect to receive an email notifying you of the need to validate in person in the SWEL (Durham 348).

    - Notified groups should sign up for an in person validation time slot on Canvas.

    - All group members should attend the in person validation if it is necessary.

    - Group members who have a time conflict should notify the GTA as soon as possible. Points will be deducted from individual group member's grade for failure to do this.

- In person validations that occur in the SWEL require a wireless internet connection via Eduroam.

- There will be no time to set up an Eduroam connection during a designated validation time. Please ensure this works before attending your group's in person validation time.

- Each group should bring their own set of hardware, Raspberry Pis with imaged microSD cards, power supplies, Ethernet cables, etc. In person validation will occur on the group's hardware.

- There are 4 HDMI monitors with keyboards and mice. One pair of the monitors, keyboards, and mice will be reserved for the group currently validating. The other pair is reserved for the group next in line to setup/prepare for their Raspberry Pis for validation.

- To save time and make the process more efficient, it is highly recommended that groups use VNC or SSH via direct Ethernet or Eduroam instead of the HDMI monitors.

- The GTA will need to SSH wirelessly via Eduroam into the group's Raspberry Pis. Have the IP addresses and passwords ready.

- The GTA will then upload the group's code that was submitted to Canvas and test the system with the same test cases that were used during remote validation.

==
Command Line Parameters
==

Server:

- python3 server.py
- python3 led.py -s <RMQ IP OR HOSTNAME> -m <GPIO MODE> -r <RED PIN NUMBER> -g <GREEN PIN NUMBER> -b <BLUE PIN NUMBER>

Processor:

- python3 processor.py -s <RMQ IP OR HOSTNAME>

Client:

- python3 client.py -s <RMQ IP OR HOSTNAME> -b <BLUETOOTH ADDRESS>

==
Error Handling
==

- You are encouraged to do as much error handling as possible. These system environment conditions can be assumed and should give you an idea what error handling to prioritize:

    - Client and server will not and should not disconnect until the server has sent a receipt with an order id (queue name) to the client.

    - As long as the Bluetooth server is not busy and is ready to serve a client, another client can connect and place an order while the orders of past clients are processing.

    - All pieces of the system are reliable and do not randomly crash or become unavailable.

    - All pieces of the system are up and running before a client Bluetooth device attempts to connect to the server.

    - Initialization order:
        - server.py
        - led.py
        - processor.py
        - client.py

    - Once subscribed to its server-designated client notification queue, the client will not disconnect/unsubscribe until it receives the third order status notification related to the order being complete.

==
Warnings
==

- If you are writing your code on Windows and then uploading to the Raspberry Pi, make sure the code can run on your Raspberry Pi with Python3.

- While coding the system, you might be tempted to hardcode input command line parameters like the IP address, port number, etc. Try to avoid this. Get argument parsing done first or you might forget to change them once the deadline comes around and you submit.

- You may run into issues with GPIO across different runs related to "Channel already in use" errors. This error usually does not impact the code but to make it go away, make sure "RPI.GPIO.cleanup()" is called before the execution of your code exits.

- You will probably find that the serial Bluetooth examples posted on various places on the internet do not work on the Raspberry Pi. To make it work, follow the guide posted on Canvas related to Serial Bluetooth configuration.

- Make sure you have backups to the files stored on your Raspberry Pi.

==
Final Thoughts
==

- Instead of parsing the arguments yourself, try using Python's built in argparse library!

- Try to figure out if a library/package/api is available through "sudo apt-get install" or "pip3 install" before attempting to manually install/compile/configure something you downloaded from a website instead.

- Each piece of this system that sends and receives data is written in Python. You might find it easier to handle pickled data, or json objects, instead of parsing raw data or strings.

- GPIO MODE: Will be either 10 for "BOARD" or 11 for "BCM" mode. This is important because the numbers refer to different pins depending on the GPIO mode.

- To figure out your Rasperry Pi's Bluetooth address, you can use the "hciconfig" command.

- To figure out your Rasperry Pi's IP address, you can use the "ifconfig" command.

- If you notice a discrepancy between this rubric and its guidelines and the slides presented in class, or have any other questions about this document, email the GTA Kelvin Aviles (kaviles@vt.edu).