Project 2 – worth 75 Points

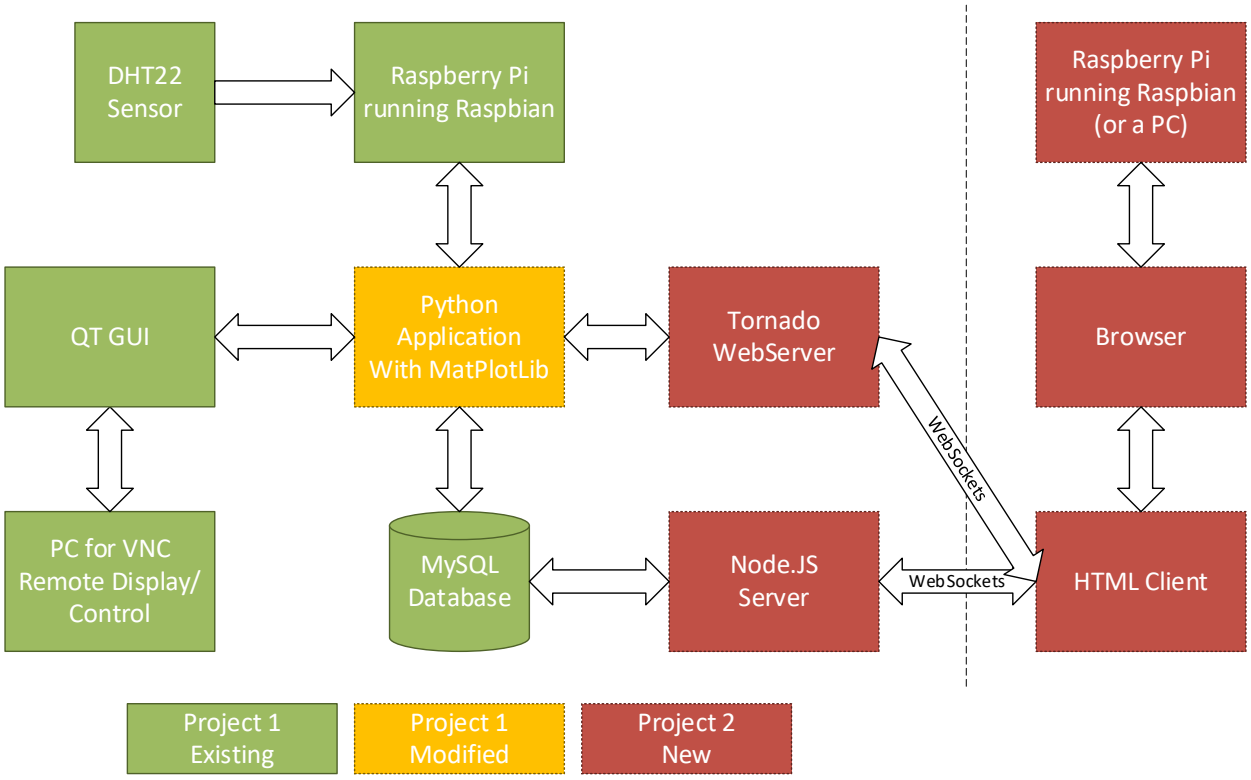
Part 1. AWS Development Account – 5 points

Sign up for an AWS account for use in future projects. I recommend signing up for the AWS Student Starter account at: https://www.awseducate.com/registration#APP_TYPE

Once you sign up, there may be a day or two delay for approval. Once you receive your account approval, please paste proof of AWS account access (your approval e-mail or similar) into a PDF called AWSAccount.PDF and include that PDF in your Repo for project 2.

Part 2. Tornado and Node.JS servers for an HTML client – 70 points

- This project will allow you to use several new technologies in addition to the RPi3, DHT22, QT, Python, MatPlotLib, and MySQL from the prior Project 1. The new tools include Node.JS, WebSockets (using the Python Tornado server and the Node.JS WebSocket package), and an HTML-based UI for a remotely connected client.
- The system will look something like this:



- Your Project 1 code will be used as a basis for beginning Project 2.
- All functions of your Project 1 code and UI should continue to work after Project 2 is complete and when Project 2 is active.
- A single script on the Project 1 Raspberry Pi should start the Python App, the Tornado WebServer, and the Node.JS server to prepare for remote or local queries from an HTML client and to begin sensor reads.

The HTML client you will develop will be running remotely on a browser (on another Pi or a remote PC) and will have the following capabilities:

- At startup, the HTML page should establish WebSocket communications with both the Tornado WebServer and the Node.JS Webserver.
- A button that when pressed retrieves a current reading from the DHT22 sensor via the running Python application from Project 1. If the DHT22 does not respond, an error should be returned to the HTML page for display, otherwise, the current temperature and humidity should be returned for display. The path for this communication is HTML->WebSockets->Tornado->Python App->DHT22 and back again.
- A button that when pressed contacts a Node.JS server that retrieves the last temperature and humidity reading from the MySQL database established in Project 1. This data should be displayed in the HTML client page. The flow should be HTML->Node.JS->MySQL with no Python interaction required.
- A button should be provided to turn local displays of temperature from F to C.
- A button that tests network responsiveness by requesting the last 10 MySQL humidity entries in the MySQL database and returns them to the HTML client for tabular display, along with a timestamp indicating the start, end, and duration of that transaction. Then, another query should be made through the Tornado Webserver to have the Python App provide the same data, along with the same timestamp. The two data sets and timing measures should be displayed side by side in the HTML page with an indication of which dataset is from which query.
- All error conditions should be considered and allowed for. Documentation should indicate which errors are being monitored.
- For 5 extra credit points, another pair of buttons should retrieve graphs of the 10 last database temperature OR humidity readings from your Python App and display them on the HTML Client page.
- I recommend using HTML, CSS, and JQuery UI for developing your web client. Please contact me for approval before you use other approaches.

References:

- Node.js talking to MySQL: https://www.w3schools.com/nodejs/nodejs_mysql.asp
- Node.js WebServer example: https://www.pubnub.com/blog/nodejs-websocket-programming-examples/
- Python-Tornado-HTML example: https://os.mbed.com/cookbook/Websockets-Server
- Refer to the Node.JS lecture and ensure you have up to date versions of Node and NPM on your Raspberry Pi.
- Tornado is just one of several Python-based webservers.
    - http://www.tornadoweb.org/en/stable/
    - Python standard library choices: BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer
    - Many other choices, many levels of complexity: https://wiki.python.org/moin/WebServers

Project Delivery

- The code should be yours and your team's work
  - Cite any sources for any Code from the Web; should include the URL of the resource you took it from
  - You may not directly use code from other teams, although they may give you advice or suggestions
  - If someone (students or class staff) helps you on part of your code, give credits in comments and the README and identify which code was provided
- Even though this is a prototype, I'd like to see well-structured Python and Node.JS code
- The project must run natively on an RPi3 development system
- The code must be well commented
  - A typical comment template can be copied (forked) from example.py in my GitHub repo at https://github.com/brmjr9/eid-fall2018
  - Comment/Docstring header for each file identifying the author and file description
  - Comments/Docstrings at any functions or classes including description, input, output
  - Comments for purpose of data structures or complex transactions (usually this is a why and not a how)
- Turn in a GitHub repo link (one submission per team) containing your project with
  - Any code files needed to run the project (not including standard libraries)
  - A README.md (markdown text file) including:
    - Title (i.e. EID Project 2)
    - Names of the developers/students on your team
    - A section called **Installation Instructions**
      - We should be able to follow the instructions to run your project on my RPi3 system (for Project 1,2,3,4 – not the Super-Project)
    - A section called **Project Work**
      - On a multi-person project, include who was responsible for what parts
    - A section called **Project Additions**
      - Describes any features you've added that are above the project requirements
  - Remember to include proof of AWS Signup

Grading Rubric – Total 75 points (+ 5 extra credit)

- Project must be turned in via GitHub URL
- AWS Proof of signup in Repo – 5 points
- README.md structured as reviewed above – 10 points (-2 per missing element)
- Demonstration of features by executing project – 30 points
  1. Reading from sensor on demand in HTML – 5 points
  2. Reading from MySQL via Node.JS in HTML – 5 points
  3. Temperature unit change on HTML – 5 points
  4. Transaction comparison – 5 points
  5. Error handling – 5 points
  6. Continuing operation of original Project 1 code – 5 points

- Well commented and structured code – 30 points – 10 for Node.JS, 10 for HTML, 10 for Python (-2 to 4 for poor commenting, -2 to 4 for poor structure)
- For 5 extra credit points, another pair of buttons on the HTML Client page should retrieve graphs of the 10 last database temperature OR humidity readings from your Python App and display them in the HTML.
- 15% Grade penalty if turned in late (accepted for one week, then 0 points awarded)

RPi3 Handling Notes:

- As with all electronics, follow ESD guidelines to avoid damaging hardware
- Whenever possible – use the Linux command line or GUI to shut down or reboot the Pi, avoid just pulling power – may corrupt the system
- If you see a yellow lightning bolt on the RPi3 GUI, you are low on power (using 2 power supplies may be needed for the Pi and an attached display for instance)
- Keep a Git or other backup of your SD card and your work – they can and do corrupt