University of Colorado Boulder

ECEN 5783 – Embedded Interface Design (Fall-2019)

Final Project Report

Project Details	Description
Created By:	**Rushi James Macwan & Poorn Mehta**
Project Title:	**Industrial Plant Monitoring and Control System**
Platform Used:	**Raspberry Pi 3Bs**
Report Submission Date:	***Dec 11th, 2019***
GitHub Repo	***Link***

Credits: *All codes utilized for completing this project have been duly credited and referenced.*

Table of Contents

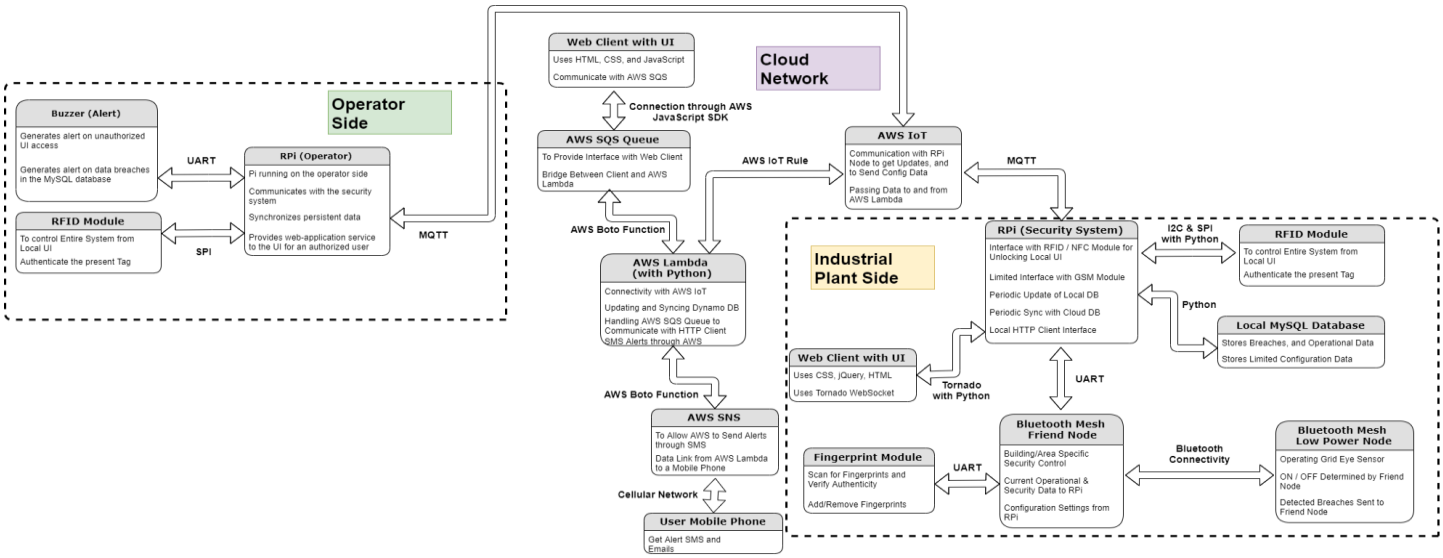
Abstract

The proposed super project for the ECEN 5783 Embedded Interface Class has been built around the simple notion and philosophy of integration of IoT (connected devices) with the cloud across a huge area (i.e. an Industrial Manufacturing Plant). The prime inspiration behind this project is the fact that the end-user has control over the workplace – where critical data (e.g. number of employees working inside the plant at a specified point in time) can be traced. That said, the second inspiration revolves around the very idea that the user has the freedom to monitor and control the plant activities – probably from anywhere across the globe – as long as the user is connected to the internet or has a cellular network. Moreover, the design and objectives of this project are proposed with the premises that the user or the workplace may not have a consistent supply of internet services (assuming that are intermittent service breakdowns). Given that, it is highly critical of the system ends to meet consistently with the persistent data that rests on the cloud and the same for the data image that exists locally. Synchronization primitives in coherent merger with the build-up of cloud networking will allow the user to monitor/control the workplace.

Credits: *Please note, the above abstract was directly borrowed from here which was previously prepared for this project. It is added here strictly for informational purposes. It has been added to give an overview of the project again before moving into the details of its implementation and demonstration for end-semester.*

Final System Architecture Diagram & Statement

The final system architecture diagram can be found below. The diagram portrays some of deviations from the project proposal made earlier at the time of Project-4 submission. The deviation is discussed at length in the upcoming section.



The final system architecture comprises of three essential system blocks that together give an acting form to this project. The three essential system blocks are as below:

1. Operator Side
2. Industrial Plant Side
3. Cloud Network

Operator Side:

The operator side is the initial part of the project that communicates with the real part of the project that is involved in actually sensing and interacting with the industrial workplace for which it has been designed. The operator side transmits authentication messages for users that are logged into the system for operating and managing the industrial plant statistics on the other side (i.e. Industrial Plant Side).

The operator side involves a Raspberry Pi running Raspbian and is interfaced with sensors like RFID sensor and a buzzer.

The operator side essentially involves the following tasks:

1. Authenticate a remote operator (user) by requesting them to scan their RFID/NFC tags/keys
2. Produce an alert for unauthorized login attempts into the system on the remote operator side
3. Connect with the AWS Cloud IoT Core for communication with the industrial plant side

Industrial Plant Side:

The industrial plant side is involved in performing similar tasks as the operator side but is actually associated with monitoring the industrial plant where it has been deployed. The industrial plant side involves a Bluetooth Mesh implementation for a low-power node interfaced with a Bluetooth friend node. It involves communicating with a grid eye sensor and other authentication modules as on the operator side. To be specific, it involves implementation of an RFID sensor, Fingerprint sensor, MySQL logical database and a buzzer.

The industrial plant side essentially involves the following tasks:

1. Authenticate a local operator (user) by the same measures as on the operator side
2. Produce an alert by the same measures as on the operator side
3. Connect with BT Mesh nodes for interacting with the grid eye sensor to detect human movement in the plant
4. Connect with the AWS Cloud IoT Core for communication with the industrial plant side
5. Surface an HTML page for real-time plant monitoring status

Cloud Network:

This is the bridge between the above two segments and played a crucial role in project integration. The AWS cloud consists of an interaction that begins with the IoT Core and thereafter is associated with internal AWS implementations (e.g. Lambda Function, AWS SNS/SQS) that provides the capabilities to store logged in user data and generates alert messages/emails for any anomalies on the industrial plant side.

Please Note: *The original final architecture diagram can be found in the repository here.*

Project Deviation Statement

The project deviation statement exposes the differences that were observed in the implementation of this project and the proposal that was made earlier at the time of Project-4 submission.

The deviations reported for the final demonstration revolve around a set of missing and added features that do not change the project outcome but affect its proposed implementation. The detailed explanation of the project deviation can be found below:

a. **Missing Features:** The set of missing features that were observed for this project are as below:
   1. The implementation of GSM was removed both the physical domains – remote operator side and the industrial plant side.
   2. The implementation of a custom Linux kernel node and its Bluetooth based intercommunication was totally removed in the final demonstration.
   3. The local web-client on the remote operator side was removed.
   4. Synchronized web-client support was removed which reduced the level of implementation complexity and time-to-release.
   5. The implementation of NFC sensor was removed.
   6. The implementation of fingerprint sensor and MySQL database support was removed on the remote operator side.

b. **Features that work differently:** On the lower end, there were some features that were implemented differently than previously expected:
   1. The NFC sensor on the remote operator side was just implemented to test if the user was authenticated into the system while the RFID would allow for user-login and registration.
   2. Synchronized web-client support was removed. However, a general web-client was available through AWS Cloud support with privileges reserved by the operators (users) on the industrial plant side.

Third-party Code Used Statement

Please note, while we did write every single line of this script on our own, we could not have made it without extensive support, tutorials, and the most helpful QnA found online. We have tried to credit as many resources as possible - but it is certainly possible that we might miss a few ones.

It should be noted that we only drew 'knowledge' and 'methods' from these references - the logical implementation is of our own. That is - we have NOT directly copied chunks of codes. Rather, we understood the underlying implementation of these resources, and designed our own implementation. Finally, this project is developed while heavily utilizing the knowledge gained in previous projects of the subject EID (Embedded Interface Development - CU Boulder), and therefore, the references are also inherited.

Please Note: *Please see the* ***APPENDIX section*** *for all the original references utilized for completing this project.*

Project Observations Statement

Some of the detailed observations made throughout this project implementation are as follows:

Instances better than expected…

Hard to find, but some of these instances were:

1. Some typical instances were with finding a breath of open-source project resources for implementing our project.
2. We had much better support from the work done in the previous EID projects during the semester which helped us a lot in speeding up our work.
3. Building the little modules in its own privacy allowed us to understand better its working and how problems could arise while performing integration.

Instances worse than expected…

It is easier to find such instances and here we are with some of the top ones!

1. Typically, while the project integration and testing was coming into picture, it was pretty worse than expected since those little modules that would work great in its own privacy would require a lot of effort to interface it with a bunch of other modules.
2. The implementation of the custom Linux kernel module on the remote operator side was one such issue and it became increasingly challenging to integrate it with the Raspbian node that would connect it to the cloud.
3. It was really difficult to implement the AWS Cloud IoT Core – Node integration since this was one such area where multiprocessing was a must thing and sharing global variables across cores was a challenging task – even though we could implement that in the previous project implementations.

Instances differently than expected…

Some of these instances were:

1. We were thinking of implementing the remote operator side with much more complexity. However, with increased complexity and reduced availability of time, we did implement it differently while eliminating the redundancy.
2. We had to reduce the number of UI features than had proposed. But, it still has had good support.
3. We planned to begin integration much earlier than when we actually were able to do it.

Appendices

Appendix – Third-party source code references:

https://pimylifeup.com/raspberry-pi-rfid-rc522/
https://www.tutorialspoint.com/python3/python_multithreading.htm
https://stackoverflow.com/questions/663171/how-do-i-get-a-substring-of-a-string-in-python
https://stackoverflow.com/questions/18072557/tornado-send-message-periodically
https://stackoverflow.com/questions/15766767/sharing-data-structures-between-2-python-processes
https://stackoverflow.com/questions/19790570/using-a-global-variable-with-a-thread
https://www.geeksforgeeks.org/multiprocessing-python-set-1/
https://docs.python.org/2/library/multiprocessing.html
https://docs.python.org/3.4/library/multiprocessing.html?highlight=process
https://pymotw.com/2/multiprocessing/basics.html
https://stackoverflow.com/questions/17377426/shared-variable-in-pythons-multiprocessing
https://blog.ruanbekker.com/blog/2019/02/19/sharing-global-variables-in-python-using-multiprocessing/
https://eli.thegreenplace.net/2012/01/04/shared-counter-with-pythons-multiprocessing
https://pymotw.com/2/multiprocessing/communication.html
https://www.geeksforgeeks.org/multiprocessing-python-set-2/
http://www.korznikov.com/2014/07/python-multiprocessing-global-variables.html
https://stackoverflow.com/questions/7894791/use-numpy-array-in-shared-memory-for-multiprocessing
https://docs.python.org/3/library/multiprocessing.shared_memory.html
https://research.wmz.ninja/articles/2018/03/on-sharing-large-arrays-when-using-pythons-multiprocessing.html
https://www.reddit.com/r/learnpython/comments/4u0xh8/fastest_way_to_share_numpy_arrays_between/
https://stackoverflow.com/questions/21290960/how-to-share-a-string-amongst-multiple-processes-using-managers-in-python
https://www.reddit.com/r/learnpython/comments/3osasv/shared_string_value_between_multiprocessing/
https://lowpowerlab.com/2013/01/17/raspberrypi-websockets-with-python-tornado/
https://www.tornadoweb.org/en/stable/
https://pypi.org/project/tornado/
https://www.hackster.io/innat/raspberry-pi-websocket-acef41
https://opensource.com/article/18/6/tornado-framework
http://niltoid.com/blog/raspberry-pi-arduino-tornado/
https://www.tornadoweb.org/en/stable/ioloop.html
https://stackoverflow.com/questions/49259055/python-tornado-periodiccallback-at-specific-time
https://www.tornadoweb.org/en/branch4.4/ioloop.html
https://www.programcreek.com/python/example/58932/tornado.ioloop.PeriodicCallback
https://stackoverflow.com/questions/24089086/unexpected-tornado-ioloop-periodiccallback-behavior
https://stackoverflow.com/questions/37119652/python-tornado-call-to-classmethod
https://stackoverflow.com/questions/14586038/nested-web-service-calls-with-tornado-async
https://stackoverflow.com/questions/52574715/python-tornado-sending-websocket-messages-from-another-class
https://stackoverflow.com/questions/32445705/how-do-i-send-a-websocket-message-in-tornado-at-will/32446100
https://stackoverflow.com/questions/6623113/is-it-possible-to-send-a-message-to-all-active-websocket-connections-using-eith
https://stackoverflow.com/questions/12479054/how-to-run-functions-outside-websocket-loop-in-python-tornado
https://www.tornadoweb.org/en/stable/ioloop.htmltornado.ioloop.IOLoop.add_timeout
https://docs.aws.amazon.com/iot/latest/developerguide/sdk-tutorials.html
https://docs.aws.amazon.com/greengrass/latest/developerguide/setup-filter.rpi.html
https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-raspi-setup.html
https://www.balena.io/blog/use-a-raspberry-pi-to-communicate-with-amazon-aws-iot/
https://medium.com/@sajithswa/connect-your-raspberry-pi-with-aws-iot-28920dbc0e88

https://medium.com/coinmonks/architecting-your-iot-app-using-raspberry-pi-and-aws-b8b89e3ac39a
https://www.freecodecamp.org/news/how-to-measure-temperature-and-send-it-to-aws-iot-using-a-raspberry-pi-d6f7b196ec35/
https://cloudncode.blog/2017/11/07/make-your-first-iot-device-via-aws-iot-service-and-raspberry-pi/
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/examples.html
https://stackoverflow.com/questions/17774768/python-creating-a-shared-variable-between-threads
https://www.pythonforthelab.com/blog/handling-and-sharing-data-between-threads/
https://www.codeproject.com/Questions/1176500/Python-how-do-I-share-a-variable-between-two-threa
https://kite.com/python/examples/4035/threading-share-a-global-variable-between-two-threads
https://hackernoon.com/synchronization-primitives-in-python-564f89fee732
https://stackoverflow.com/questions/9436757/how-does-multiprocessing-manager-work-in-python
https://stackoverflow.com/questions/tagged/multiprocessing-manager?sort=votes&pageSize=50
https://stackoverflow.com/questions/50869656/multiple-queues-from-one-multiprocessing-manager?rq=1
https://stackoverflow.com/questions/50869656/multiple-queues-from-one-multiprocessing-manager/50934524
https://stackoverflow.com/questions/46717246/calling-methods-in-multiprocessing-manager-class-from-another-class?rq=1
https://www.instructables.com/id/Raspberry-Pi-Arduino-Serial-Communication/
https://classes.engineering.wustl.edu/ese205/core/index.php?title=Serial_Communication_between_Raspberry_Pi_%26_Arduino
https://oscarliang.com/raspberry-pi-and-arduino-connected-serial-gpio/
https://oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/
https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino
https://forum.arduino.cc/index.php?topic=633792.0
https://github.com/NicoHood/NicoHood.github.io/wiki/Arduino---Raspberry-Pi-Serial-Communication-Protocol-via-USB-and-C-C
https://raspberrypi.stackexchange.com/questions/67840/send-data-from-python-to-arduino-through-serial-port
https://electronicshobbyists.com/control-arduino-using-raspberry-pi-arduino-and-raspberry-pi-serial-communication/
https://www.raspberrypi.org/forums/viewtopic.php?t=41055
https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c
https://electrosome.com/uart-raspberry-pi-python/
https://pimylifeup.com/raspberry-pi-serial/
https://embeddedlaboratory.blogspot.com/2017/03/serial-communication-in-raspberry-pi.html
https://raspberrypi.stackexchange.com/questions/57443/how-to-use-uart-with-python
https://github.com/boto/boto3
https://github.com/aws/aws-iot-device-sdk-python
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs-examples.html
https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html
https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html
https://www.alienvault.com/documentation/usm-anywhere/deployment-guide/notifications/setup-sns-topic.htm
https://mkdev.me/en/posts/how-to-send-sms-messages-with-aws-lambda-sns-and-python-3
https://stackoverflow.com/questions/4528099/convert-string-to-json-using-python
https://russell.ballestrini.net/setting-region-programmatically-in-boto3/
https://stackoverflow.com/questions/4528099/convert-string-to-json-using-python
https://www.w3schools.com/jquery/
https://www.w3schools.com/js/default.asp
https://os.mbed.com/cookbook/Websockets-Server
https://www.pubnub.com/blog/nodejs-websocket-programmingexamples/
https://www.w3schools.com/nodejs/nodejs_mysql.asp
http://www.tornadoweb.org/en/stable/
https://www.w3schools.com/js/js_json_parse.asp
https://stackoverflow.com/questions/21634918/extract-a-float-from-a-string-in-javascript
https://www.w3schools.com/jsref/jsref_tostring_number.asp

https://api.jquery.com/jquery.parsejson/
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_SendMessage.html
https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/SQS.html#receiveMessage-property
https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/sqs-examples-send-receive-messages.html
https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/sqs-examples-using-queues.html
https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/SQS.html#deleteMessageBatch-property
https://stackoverflow.com/questions/951021/what-is-the-javascript-version-of-sleep
https://stackoverflow.com/questions/10073699/pad-a-number-with-leading-zeros-in-javascript