**Final Report – Project 2**

**Khalid AlAwadhi - Poorn Mehta**

**S.E.C.U.R.E.R.O.O.M**

(**S**uper **E**ncrypted **C**ourse project which is a **U**seful **R**oom **E**mbedded system that communicates **R**emotely **O**ver the air and it’s **O**bviously **M**arvelous)

**GitHub Link:** [**https://github.com/K5Ma/APES\_Prj2.git**](https://github.com/K5Ma/APES_Prj2.git)

**Documentation Folder Link:** [**https://drive.google.com/open?id=1lEjhavnkyza8AwUQry0yn8fN5KxVqA8M**](https://drive.google.com/open?id=1lEjhavnkyza8AwUQry0yn8fN5KxVqA8M)

**Key Learnings**

**Khalid AlAwadhi**

* This Project was one of the most challenging ones throughout my career, and I learned some very good things while working hard on the same – to make it fully functional.
* My part was to create a central structure, to provide a base for all of the inter-node, inter-thread, and inter-task communication, as well as to have a solid support for the Logging. Starting from developing base on BeagleBone – I learned the importance of proper use of **errno** as it can provide some excellent functionalities – related to system, and failures – which is very useful while debugging.
* I learned thoroughly, about implementing Interrupt based reception through UART on BeagleBone. It was a difficult task at an initial point, because I didn’t have any idea how the Linux system will go about talking to Hardware with Interrupts.
* We have used Bluetooth Communication to setup a link between Control Node and the Remote Node, and I learned some key concepts like Bluetooth Pairing, Bonding, Connecting etc. while trying to interface the same with my nodes.
* Moreover, since my home has quite a few active Bluetooth Devices, I learned about the importance staying in close proximity – for my communication to work. Many times, the communication was failing because the noise from other devices on same Frequency was quite a lot.
* One other important learning was how to deal with two completely different nodes – running different multitasking algorithms/platforms/OS (Linux vs FreeRTOS) – and to make them talk to each other without any issues. I had to implement some novel approaches to get this done.
* Software Flow Control was something that I never really required in the projects I have worked on before, but this project demanded it. It was because the processing speed, time slicing, etc. were completely different for both boards.
* Developing a customized protocol – as a layer on a standard protocol, is complex – but very effective procedure as well. I did it in order to have reliable communication of not just strings, but standard C structures – between nodes. Formatting them into a Byte Stream, Appending proper testing Bytes with Them, Identifying them, and Decode the same on the other side – it was a challenging task which taught me a lot of things.
* Handling IPC on both Nodes, while also keeping track of sent and received messages – required a lot of time, as well as coding and debugging skills. Proper use of Mutex Locks, and Critical Sections was often found out to be an absolute requirement.
* When I started to Integrate the individual I/O Device Driver source files wrote by my teammate, I found that often – the drivers weren’t working as intended, because of some strange issues caused by FreeRTOS’s intense time slicing. Functions such as vTaskDelay() often rendered rather random behavior, finding a workaround of which took a lot of creativity.

**Poorn Mehta**

* For this project, I developed standalone driver source codes for all the I/O devices. It proved to be much more difficult than what I thought – since multitasking of FreeRTOS, the somewhat erratic behavior of some specific I/O devices etc., were quite tough to resolve and it consumed a great amount of time. Apart from technical learnings, I have gained substantial knowledge about project management – from experience, while working on this Project.
* Fault Tolerance was probably one of the key areas which taught me some very good lessons on clean coding, having lesser dependency on shared variables, making the portions of code atomic – while designing others to be multitasking friendly, and other such relevant fields.
* In case of NFC Module, detecting that the Module was offline proved to be difficult, especially when trying to scan an NFC Tag. This took place since the module itself doesn’t have anything built-in to check the availability of the Tag, and thus, it just waits for tag to show up. Transmitting another command was overriding the previous reading tag, and thus, both commands had to take turns for execution.
* There were some very specific requirements for a few I/O devices, to drive which – I had to dig much deeper in documentations, to fully understand and implement some functionalities.
* Detecting the End of Response for Electronic Paper was really tricky – because it would take anywhere from a few milli seconds to a few seconds – to even start responding back. Moreover, it doesn’t have a start or end frame or byte – which can be used to detect the valid response length. I had to develop a unique algorithm to deal with this highly dynamic response pattern.
* At some instances, I/O Devices wouldn’t work properly – because they might not have been connected to a solid ground. This was a quite low-level fault – consuming a good deal of time. Eventually, this led me to connect every Power Connection of the TIVA Board, and of all the I/O Devices on the same node, using breadboard.
* Driving Servo proved to be a challenge, because of undocumented erratic behavior. It works well at various PWM frequencies, but the position to duty cycle correlation changes widely for each of those frequencies. After multiple observations, and trial & error cycles, I was able to figure out the correct values to drive servo in specific positions.
* While writing driver for communicating with BME280 Sensor through SPI bus, I learned that some sensors require ‘pulse’ on the Slave Select pin, before they begin to communicate. This was vaguely mentioned in the datasheet with loose terms, and was counterintuitive – which resultantly took me many hours to resolve the error, and finally get the interface done.
* At some point, one part of my code had partial recursive function calls, which cause the Task to run out of stack at some specific points. I gained knowledge about some key concepts on stack management while resolving this issue.
* Overall, I learned many things in this Project, however I must say that it was a bit – over the head, owing to the ‘size’ of the project itself.

**Project Plan**

****















**Test Results**

****

****

****

****

****