**COP 5615: Distributed Operating Systems Principles**

**Internet of Things Support in Xinu**

**Fall 2016**

**Term Project Report**

**Group 37**

Aman Yadav, Satyam Sinha, Akash Divakar Gore, Abishek Sethuraman, Aneesh Khandekar

[amannagarro@ufl.edu](mailto:amannagarro@ufl.edu)**,** [satyam268@ufl.edu](mailto:satyam268@ufl.edu)**,** [akashgore@ufl.edu](mailto:akashgore@ufl.edu)**,** [abisheksethuraman@ufl.edu](mailto:abisheksethuraman@ufl.edu)**,** [aneeshk@ufl.edu](mailto:aneeshk@ufl.edu)

**1. Describe your project using this table**

|  |  |
| --- | --- |
| **Part** | **Indicate Completeness (give a no. from 1-10), followed by Description** |
| Xinu I/O Interface design | **7**  At XINU level the generic read and write IO abstractions are called which is then channeled to the appropriate device driver handling the device.  This ensures the current XINU implementation does not get affected by our system architecture. |
| IoT-specific concerns your design addressed, including but not limited to Energy | **8**  Our system is a sentient efficient system.  The manual monitoring system only requests data when required and uses that data by showing it to the user. Thus, every poll to the device is a meaningful poll.  Also, a XINU process polls data from the system devices at specific time intervals, all of which is then stored on the Mongo Database hosted on the cloud.  This information is then relayed as a trend to the user upon his request from the UI from the database. Thus, every data request made to the device is used by the system making it sentient efficient. |
| Xinu I/O Interface implementation and testing | **9**  The MQTT-like publisher/subscriber model used for machine-machine communication makes use of newly created XINU shell commands to allow devices seamlessly subscribe/unsubscribe to events without impacting the system architecture.  These shell commands allow a user monitoring the system at the device level to expand/compress the system scope without rebooting the system. |
| Design of IoT Description Language, Language processing and code generation | **7**  **Indicate**: XML-, JSON, Other-based  **Source**: any open source used? Indicate the Github or other s/w package name.  **Design:** An XML basedDDL is used to represent each beaglebone device and its connected devices in the overall system.  The DDL has been created per the DDL standards specified in the DDL paper.  **Device Level Information:**  DDL contains information about each device connected in the node.  This includes the device name, and Pin Numbers onto which each individual device is connected in the system.  The formula tag contains device specific formula to convert device signal into reading.  **Controller Level Information:**  The IP and port number of the BBB is specified in the DDL which is also parsed on the edge to provide a representation of the entire node on the edge. |
| Implementation and testing of IoT Description Language, Language processing and code generation | **7**  This DDL is parsed at XINU compile time using javac which is set up on the machine hosting XINU.  The DDL is first validated against the .xsd and then parsed upon which the Device Name, pin number formula tag is read and used to generate the device driver code. This formula used ensures the final output of each device represents a final reading and not a value in voltage.  The same DDL is also parsed on the edge to get the IP address of the controller, using which UDP messages can be successfully sent to the devices connected to the controller. |
| Implementation and testing of overall on-board driver code (upper- and lower-level drivers, including generated code) | **8**  The device driver code has been successfully implemented using the existing device driver structure. Analog and digital sensors and actuators have been identified as discrete devices. Different devices of the same type are distinguished using the device minor numbers.  Device driver code has been developed independent of the pin on which a device is connected on the BeagleBone. The user is given the flexibility of connecting a device on any pin number and this is specified in the DDL.  Extensive testing showed that the high level I/O functions invoke the appropriate device driver code from the device table and part of the driver code from the DDL is also successfully generated. |
| Did you use the same existing device driver structure and mechanisms in Xinu? | Yes or No. If No, explain.  Yes. |
| Approximate % driver code generated with respect to overall on-board driver code | Give % only here, no description.  20% |
|  |  |
| Which device externalization abstraction have you chosen (which existing technology or any new ideas)? You may, or may not explain the reason for your choice. | The edge and the cloud have been implemented using node.js.  Mongo DB has also been used over the edge to store data received from the devices. This data can then be used for data analytics, visualization and to enhance system functioning. |
| How, where, and when do you specify the edge and cloud addresses of the device? Explain how device configuration and initialization are done including device externalization. | **9**  The same URLs can be used for any beaglebone controller attached to the system. If the DDL of the newly added device is being provided to the edge server in a directory, the edge parses it and creates entries into the database, to service requests for newly added beaglebones.  Therefore, in our application the same abstract methods namely: getTemperature,getProximity etc can be used for all new/existing beaglebones present in the system.  Minimal changes would be required at the cloud end to provide new routes to the edge server. |
| Give the details of the externalization abstractions design. | **8**  The cloud and the edge are flexible enough to accommodate new devices and controllers into the system.  In our application the same abstract methods namely: getTemperature,getProximity can be used for all new/existing beaglebones present in the system. |
| Describe the implementation of the abstractions (how they connect to the actual device), and discuss any IoT-specific concern (including energy) that may have been addressed by your implementation. | **9**  The system integrates with M2M communication using MQTT like publisher subscriber design pattern.  **Machine to Machine model:**  Actuators which are not on the BBB containing the sensor system can be subscribed/unsubscribed from the topics at any point in XINU at run time via subscribe/unsubscribe shell commands which communicates to EDGE via UDP.  This ensures the nodes of the system do not require a reboot on changes to subscribe/unsubscribe events.  A process running on XINU sensor system monitors for a specific change in data and publishes data to the specific topic. The edge then sends over this data to all the subscribers which results in the corresponding actuator’s action.  **Request Based Model(via Http):**  Here, the user requests for device data from the UI which only then polls the device for data and sends it back to the user.  Flexibility:  If the application to be implemented using this system needs to be more energy efficient, beaglebone can be unsubscribed from all the topics, only allowing for device data from UI.  This enables data polling only via the manual part enabling energy efficiency. |
| Describe your on-board IoT devices Demo App. | **8**  **Devices**: describe any of the basic, composite and virtual devices used.  **App: Industrial IoT in a manufacturing plant**  A chemical/product manufacturing plant has a lot of boilers preparing products at the same time. This process is then repeated in batches.  In a scenario where an ingredient must be added into the boiler only when the current solution has reached a specific temperature, manual intervention might be required to start a conveyor belt hosting this new ingredient and insert it into the boiler and continue the manufacturing process.  Our application automates this process by using a set of temperature and proximity sensors on each boiler.  The device containing the conveyor belt is subscribed to 2 events:” temperature\_threshold” and “proximity\_acheived”.    The boiler BBB consisting of temperature sensor senses for temperature threshold and upon reaching it, publishes data to the temperature threshold event.  The actuator BBB receives this data and triggers on the conveyor belt.  When the ingredient reaches the boiler, the proximity sensor publishes data to a proximity\_acheived event allowing the actuator device to stop the conveyor.  The ingredient can then be added to the boiler without human intervention.  Manual controls are also provided to the user to check for the system monitoring. |
| Describe your web-based IoT devices Demo App. | **9**  The web based IoT application is a simple web application that communicates to cloud through a restful api, and displays data in real time.  Web application -> RESTful -> Cloud -> RESTful -> Edge -> UDP request -> XINU and response goes in the other direction using same protocols. |

**2. Challenges**

Challenges your group faced. What was the most time consuming parts of the project? what piece(s) would you have really liked to have us provide to you so the total effort is more manageable (again, if any)?

It was a fantastic experience overall. The interfacing over the cloud and edge when implemented using good coding practices.

Implementing hardware level changes proved to be difficult because of minimal information present on the same.

Better ideas could have been implemented had the course syllabus covered information regarding networking protocols available on XINU and basic level information about the power of XINU.

We came to know about XINU shell commands right at the end which we then put to good use, though it would have been better to have known about such features right from project offset.

**3. Overall Experience**

Overall experience. Describe your overall experience good or bad.

It was a good learning experience.

**4. Effort Distribution**

Report only if effort was considered by any member of the group to not be even. In this case a table showing the names, ID’s, and percentage of effort should be provided.