**COP 5615: Distributed Operating Systems Principles**

**Internet of Things Support in Xinu**

**Fall 2016**

**Term Project Report**

**Group 28**

**Team Members:**

Dheeraj Kumar Dogiparthi, Sreeharsha Poluru, Sainath KPS, Anirudh Sarma Bhaskara, Ashwin A. Nair

**All Emails:**

ddheeraj@ufl.edu**,** [spoluru@ufl.edu](mailto:spoluru@ufl.edu), [sainath@ufl.edu](mailto:sainath@ufl.edu), [ani2404@ufl.edu](mailto:ani2404@ufl.edu), nairashwin952013@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 | The I/O design is based on providing abstractions for different physical pins as meaningful devices. This infers that the user does not interacts with the pins but accesses them as devices. These devices are generally accessed by high level abstractions which in turn accesses the underlying physical GPIO Pins.  Usage of interrupts has been done for taking digital inputs. This method is used instead of conventional polling method.  **8** |
| IoT-specific concerns your design addressed, including but not limited to Energy | * The main concern addressed by the usage of interrupts is that energy was conserved because if the polling method had been used there would have been a continuous consumption of power from the device since it is frequently scanning for digital inputs. * Our device driver codes are generic and scalable allowing integration of additional devices to the system with minimal effort. * Since the temperature sensor doesn’t sense the temperature continuously, unless the user requests, this application can be considered, highly energy efficient.     **8** |
| Xinu I/O Interface implementation and testing | We implemented circuit design using breadboard involving photo resistors, buzzer, LEDs and temperature sensor interfaced with BBB.  We tested the interface by manipulating different registers by hardcoding register values and by using abstracted functions to deduce that they both work.  **8** |
| Design of IoT Description Language, Language processing and code generation | **Indicate**: JSON  **Source**: A team member has written Parser Code  **Design:**  IOT Description language has been developed using JSON (Java Script Object Notation) which is a lightweight data-interchange format. It is convenient for machines to parse and generate.  The DDL file gives the necessary information to generate the high-level drivers in an automated manner. Our design comprised of physical devices interfaced with BBB and virtual devices over the physical devices. The interface specific to each device consists of multiple topics to which the process may subscribe to, read, and write. Some topics are readable some topics are writable and some topics are subscribed to. This depends on which device the topic is associated with which is the information encoded into the DDL file in static form. Each device has certain attributes called signals and we have the topics manipulating/updating those signals. The signals can be analog, digital, and derived.  **8** |
| Implementation and testing of IoT Description Language, Language processing and code generation | **Implementation:**  Implementation of the parser was in Python using JSON Objects. JSON consists of nested key value pairs which is suitable to our design needs as we can use indexing for each device and their respective topics easily. The parser parses each topic specific to each device in the DDL file and converts them into high level functions which become interfaces for the applications including internal and external.  **Testing: -**   * Kept changing the parameters in JSON and generated multiple codes. * Tested if the higher-level functions are generated properly.   **8** |
| Implementation and testing of overall on-board driver code (upper- and lower-level drivers, including generated code) | Implementation: -  The upper-level driver codes were generated by the DDL parser from the DDL file.  The lower level drivers were written by referring the hardware manual for ARM Architecture.  Testing: -  We tested the device using different inputs both analog and digital. These include photo resistors, LEDs, blinking LEDs, buzzers, relays of buzzers and LEDs under all possible configurations  **8** |
| Did you use the same existing device driver structure and mechanisms in Xinu? | No, we implemented it partially. The existing driver structure was not apt for the subscriber model we implemented.  **8** |
| Approximate % driver code generated with respect to overall on-board driver code | 65% |
|  |  |
| 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. | We chose the HTTP protocol to abstract the BBB as a device on its own and for accessing it from the cloud remotely.  **8** |
| 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. | Edge stores all the addresses for Xinu and the active ports on which they listen to and Cloud interacts with the Edge only and the addresses for Edge are stored in cloud. The Cloud store the values of the addresses ac TCP port numbers and the IP addresses for the Edge application.   * We have specified the edge and cloud addresses for the device in the DDL file we made. * Xinu registers to and communicates only with those IP addresses. * The communication between the Edge and BBB (Beagle Bone Black Board) starts with a UDP broadcast from the Edge to identify the devices in the smart space which BBB responds with their IP addresses to notify about its presence to the Edge.   **9** |
| Give the details of the externalization abstractions design. | * Our project involves the cloud, edge and Xinu device (BBB) in such a way that the edge acts as an intermediary between the cloud and BBB. We have a TCP (Transmission Control Protocol) connection established between the Edge and Cloud and a UDP (User Datagram protocol) connection between the edge application and BBB. * We have our front end developed in HTML which sends a HTTP request to the back-end which is being hosted by a server technology developed in nodejs which successively communicates with our Xinu devices using UDP (User Datagram Packets) which is an inherent property in the Xinu device due to inclusion of “net.h” library in the Xinu folder. * Device externalization happens from cloud through edge via a message which is in the format <device name> <function>.   **9** |
| 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. | We implemented one of the abstractions using the subscriber model because of which continuous polling for input is avoided.  IOT Specific Concerns Addressed: -   * The applications that the user uses need not know the inherent information/details regarding the GPIO pins. * We are providing the user with two levels of abstractions in which in one the user is provided with direct access to the GPIO pins which enables the user to avoid manipulation of registers resident in the BBB and usage of assembly language to perform data operations among registers like data transfer, etc. * The second level of abstractions equips us by accessing the GPIO pins from web in which every function discussed above is assigned a unique identifier which enables the user to not memorize the functions being used to access the set of interfaced devices. * The major area where we address the energy issue is by utilizing interrupts for the digital part of the project. This enables us to avoid polling for input periodically i.e. to scan for the input data periodically which is a computationally intensive process. * Since the temperature sensor doesn’t sense the temperature continuously, unless the user requests, this application can be considered, highly energy efficient.   **8** |
| Describe your on-board IoT devices Demo App. | **Devices**: describe any of the basic, composite and virtual devices used.  **App:**   * The functionalities covered in on-board IOT devices demo app is we have a provided a button on board the circuit connected to BBB which enables us to switch on and off the buzzer. Also by using the button we can toggle the multiple LEDs connected to BBB. This on-board demo app also involves the attribute when the temperature around the temperature sensor increases above a certain threshold the buzzer goes off sending an alert to the user. * Physical devices like LEDs, Buzzers, Buttons, and temperature sensors cumulatively combine a holistic alert system which sets off an alarm whenever the temperature varies beyond the defined range of temperature. This application can be integral to measuring room temperatures in thermal power plants or nuclear reactors where a constant range of temperature must be maintained. Also, it also helps in developing smoke detectors. * Our virtual device functionality is the receipt of temperature alerts and basic and composite devices involve the different devices interfaced in an individual and combinatorial manner respectively.   **8** |
| Describe your web-based IoT devices Demo App. | * The app demo is used to operate different devices interfaced with the BBB. We have provided buttons by using HTML which executes this functionality by switching on the LEDs (Light Emitting Diodes) connected to BBB. Every button has been assigned a unique color specific to the LED it operates. Apart from this, we have a temperature sensor interfaced with the BBB whose current temperature we can obtain by using the button->get temperature. * A web-based Dashboard has also been provided which informs the user about the current operating status of all the devices interfaced with BBB. This helps make the user aware about the power consumption of the system at a given point of time and helps in optimizing the power consumption levels for the given system.   **9** |

**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)?

Some of the challenges our group faced includes: -

* Reading from the ADC pins was time- consuming.
* We should have been provided with the information regarding the pin configurations. It would have been little easy if we were given some assignments to work on reading from hardware, i.e. working with digital and analog pins on beagle board.

* We should have been provided with the information regarding the low-level registers and hardware details involved in the project since it required an extensive research to learn about these intricacies relevant to the project.
* Xinu doesn’t have popular utility functions such as string functions, datatype conversion and other functions. As we couldn’t adopt an object-oriented programming approach the task was made relatively difficult.
* Generating the code form the DDL parser was difficult.

**3. Overall Experience**

Overall experience-Good

* The overall experience while building the project was progressively constructive, innovative and an intensive learning experience for each group member.
* To develop different modules of a project in a coherent manner and to consequently make them compatible with each other was a pragmatically challenging task.
* Adherence to time constraints after the software planning for the project was formulated was difficult.
* Familiarity with the hardware structure and architecture of the Beagle Bone Black (BBB) Board, formation of electric circuits in a breadboard, awareness about operational functionalities for a temperature sensor.
* Acquainting with the sensitivity of LEDs and light sensors to pertinent voltage ranges.
* Coordination among all group members whilst simultaneously relying on the proficient attributes apposite to every distinct team member.
* The experience made us realize the importance of implementation of object-oriented languages and how it has significantly changed basic programming development, which C was not, enabling us to retrospect our programming skills at a grassroots level.
* Efficient perception about the lab assignments, given earlier, played a key role in the development of the project code. Hence, the efforts we put in throughout our semester writing codes for our three assignments were not rendered obsolete but were rather an essential part to our project implementation.
* The experiment being based on Cloud and Internet of Things was surprisingly disparate from the conventional projects we have worked on and so it was new for everyone.
* Register manipulations and flags set/offset while carrying out a variety of operations and equipping ourselves with a holistic knowledge relevant to General Purpose Input-Output (GPIO) Pins including their relevant functions, the devices they can be interfaced with and much more.