COP 5615: Distributed Operating Systems Principles

Internet of Things Support in Xinu

Fall 2016

Term Project Report

Group 18

Ramona Vaz, Monisha Mathew, Dilip Kunderu, Jayakrishna Thota, Akshat Bhardwaj

rvaz@ufl.edu, monisham@ufl.edu, jaygre@ufl.edu, akshatbhardwaj@ufl.edu, dilipkvsr@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	***Completeness: 9*** An intelligent prototype designed to provide an overview of how XINU Operating System can be tailored to fit into today's IoT domain's changing dynamics. The system has been designed to be effectively scalable with a composite nature accommodating both analog and digital sensors and actuators. The user can remotely read the system state. In our context, the ambient temperature readings as read by the analog temperature sensors are notified to the user. This interaction with the user is smartly used to trigger system changes. For instance, below a certain temperature threshold, the LED is triggered to glow.

IoT-specific concerns your design addressed, including but not limited to Energy	***Completeness: 9*** 1. Scalability, Future scope and Interoperability: A gateway application that can be expanded across platforms with a scalable nature in terms of number and types of devices (sensors and actuators) 2. Cloud: Open Source technologies used. MEAN Stack, with Node.js application hosted on Heroku, mLab providing Database-as-a-service 3. User Interface and Features: A hybrid application with speech recognition and text to speech capabilities (using Apache Cordova plugin) developed on Ionic Framework for android users. In addition, a physical actuator in the form of an LED as a part of the 'thing' reflecting temperature changes
Xinu I/O Interface implementation and testing	***Completeness: 10*** 1. Low level driver implementation: • ADC is controlled using the AIN0 pin to which a temperature sensor is connected. The FIFO registers are set appropriately and the IRQ is defined in the Configuration file in the /config folder. The /device/adc folder has the low level device driver functions that manipulate the switch table. They are adcread, adcinit, syscontrol, adchandler. • GPIO is primarily used on the GPIO2 base register by manipulating output enable, Data in, Data out and Data Clearout offsets.The LED and the tactile latching switch are connected to GPIO2 pins. 2. High level driver implementation: • DDL JSON parsed using a stand-alone Java application • Dynamic generation of High level driver file for all the devices connected to BBB through the same stand-alone Java application 3. User Datagram Protocol: • Static IP assignment to the BBB on the router LAN setup

	• UDP client registration and server registration with the IP of the Edge device

Design of IoT Description Language, Language processing and code generation	***Completeness: 10***
	
	JSON Based
	Source: Used Project document as reference
	Design:
	{
	"ddl": {
	"device": [
	{
	"name": "LED_rgb",
	"pin": [
	{
	"id": "gnd_v",
	"index": "0",
	"type": "D",
	"mode": "out",
	"level": "high"
	}
	]
	},
	{
	"name": "power_btn",
	"pin": [
	{
	"id": "pow_btn",
	"index": "7",
	"type": "D",
	"mode": "in",
	"level": "0"
	}
	]
	},
	{
	"name": "temp_sensor",
	"pin": [
	{
	"id": "temp_sen",
	"index": "5",
	"type": "A",
	"mode": "in",

	"level": "0.899" }] }] } }
Implementation and testing of IoT Description Language, Language processing and code generation	***Completeness: 10*** The JSON Object was designed considering the nature of the devices constituting the Internet of Things and their functionalities in our system. The syntax and correctness of the JSON Object was tested using online editors like - *http://www.jsoneditoronline.org/*
Implementation and testing of overall on-board driver code (upper- and lower-level drivers, including generated code)	***Completeness: 9*** 1. Low level driver implementation: • ADC is controlled using the AIN0 pin to which a temperature sensor is connected. The FIFO registers are set appropriately and the IRQ is defined in the Configuration file in the /config folder. The /device/adc folder has the low level device driver functions that manipulate the switch table. They are adcread, adcinit, syscontrol, adchandler. • GPIO is primarily used on the GPIO2 base register by manipulating output enable, Data in, Data out and Data Clearout offsets.The LED and the tactile latching switch are connected to GPIO2 pins. 2. High level driver implementation: • DDL JSON parsed using a stand-alone Java application • Dynamic generation of High level driver file for all the devices connected to BBB through the same stand-alone Java application
Did you use the same existing device driver structure and mechanisms in Xinu?	Yes
Approximate % driver code generated with respect to overall on-board driver code	***Completeness: 10*** 100%

	The complete device driver file (system/deviceoperations.c) has been designed to be dynamically generated based on the DDL JSON file input to the stand-alone Java application. The Java Application reads the DDL JSON file, defines the scope of the device driver (for each type – Analog, Digital; Input, Output) and stitches in only the required low level read/write functions as the cases in the switch statements of the high level read/write functions. Java Collections like ArrayLists and HashMaps, JSONObject Class, and a few custom classes tailored to manipulate the expected JSON structure have been extensively used in the logic implementation to achieve driver code generation.
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.	***Completeness: 8*** At the device level, the individual sensors and actuators connected to the BBB are blanketed in the abstraction layer provided. The broad categories identified are Analog and Digital, and each are further categorized as Input and Output. Considering that the Prototype has only one active BBB, there was no pressing need for an abstraction layer for the communication between the BBB and the edge device. Although, the UDP interaction can have the data packaged with the device ID to be scalable and still have the capability handling the intricacies of the code and logic.
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.	***Completeness: 9*** UDP server registration is done with the IP and listening Port of the Edge device as the parameters. In our case, the Edge device is a PC running a Python UDP server/client. For the UDP client registration, there is no IP and listening Port parameters required. There is no direct interaction between the BBB and the cloud. The Edge device mediates all interactions with the BBB.
Give the details of the externalization abstractions design.	***Completeness: 8*** As mentioned before, at the device level, the individual sensors and actuators connected to the BBB are blanketed in the abstraction layer provided.

	The broad categories identified are Analog and Digital, and each are further categorized as Input and Output. The prototype design currently needs to be expanded to accommodate this layer to have more IoT devices on-board in a more structured way.
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.	***Completeness: 9*** • ADC is controlled using the AIN0 pin to which a temperature sensor is connected. The FIFO registers are set appropriately and the IRQ is defined in the Configuration file in the /config folder. The /device/adc folder has the low level device driver functions that manipulate the switch table. They are adcread, adcinit, syscontrol, adchandler. • GPIO is primarily used on the GPIO2 base register by manipulating output enable, Data in, Data out and Data Clearout offsets.The LED and the tactile latching switch are connected to GPIO2 pins.
Describe your on-board IoT devices Demo App.	***Completeness: 10*** **Devices**: describe any of the basic, composite and virtual devices used. **App:** LED,Tactile Switch , Temperature Sensor
Describe your web-based IoT devices Demo App.	***Completeness: 10*** A hybrid application with speech recognition and text to speech capabilities (using Apache Cordova plugin) developed on Ionic Framework for android users which uses AngularJs for all the heavy-lifting like making REST calls to the Nodejs application hosted on Heroku. We are using Nodejs as the driver for the MONGO DB connection hosted on mLab.com using Database-as-a-Service.

	The app on invocation of the tab “CHATS” prompts the user to ask for the ‘weather’. Thereafter the REST ‘GET’ call is invoked to fetch the temperature which is checked against a threshold value(in our case 26 deg C) and writes a value of “1” using ‘POST’ to the LED_STATUS which makes the LED glow when the temperature dips beneath the threshold at the temperature sensor connected to the BBB.

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)?

- The team needed time to come up to speed with the technical aspects of the implementation, mapping the project requirements to project deliverables took up a major portion of the time.
- The analog lower device driver implementation required a lot of hardware expertise/experience which we really struggled with. The resources/code for this implementation could’ve been pointed out earlier so that the project effort was more manageable.
- We adopted a modular approach in which each individual was assigned a portion of the project. The most challenging part was the integration of the independent pieces, given the tight deadlines.
- The communication using UDP protocol took some time to figure out, it would’ve been great if this was dealt with in class.
- With a major chunk of the project being, understanding the code in C written for a constrained environment, the IDE was sorely missed and a debugging errors took away a lion’s share of the time.

3. Overall Experience

Overall experience. Describe your overall experience good or bad.

The overall experience has been great!
The learning curve has been phenomenal for all the team members on this project. We now know what it takes to build a complete end-to-end IoT application albeit a simple working prototype. We were stretched to our limits in all phases of the project right from decrypting the project requirements, fiddling with existing implementations of drivers, reading up ARM technical documentation, finalizing the design decisions for frameworks/languages used in Cloud applications and the Edge, getting UDP to work on the BBB with XINU and the end-to-end integration of all the pieces of the puzzle. A sense of satisfaction is now prevalent at having completed the project in the timeframe given, however we believe we could've achieved and explored much more if we had labs that aligned with project deliverables.

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.