COP 5615: Distributed Operating Systems Principles

Internet of Things Support in Xinu

Fall 2016

Term Project Report

Group 19

Chujia Liu, Jinhao Zhu, JiangJiang Zhu, Liyue Wang, Danli Wang

halosern@ufl.edu, jiangjiangzhu@ufl.edu, zhujinhao3@ufl.edu, liyuewang@ufl.edu, wangdanli@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	10 Read (devptr, char*, uint32) Write (devptr, char*, uint32)
IoT-specific concerns your design addressed, including but not limited to Energy	10 Though we decided to design a read_int() and write_int() high-level I/O interface at beginning, we found out that when doing communication with RPC, we have to convert integer to char in order to fit in to the data chunk. Thus, we finally decided to use the original I/O interface read() and write which seem to be most energy efficient considering the conversion of format.
Xinu I/O Interface	10

implementation and testing	As we use the original I/O interface, the implementation is the same. We test the interface with GPIO devices as well as ADC devices.
Design of IoT Description Language, Language processing and code generation	**Language**: JSON **Design:** **1 DDL Data Nomenclature** 1. Member and element names all start with upper case. Member and element names use a "_" to separate multiple word names so as to improve readability. 2. Member and element names avoid the use ofabbreviations to enhance readability. 3. Within this document, members are listed in a '[]' collection. 4. Within this document, elements are listed in a '{}' collection. **2 DDL Framework** 2.1 "Devices" "Devices" is a member of DDL file. It contains two kinds of elements. One is sensor and the other is actuator. 2.2 "Description" "Description" is an element of device. It contains the identification information of a device. 2.2.1 "Name" "Name" is an element of "Description". It defines the name of the device. 2.2.2 "Device_Type" "Device_Type" is an element of "Description". It defines the type of the device. 2.2.3 "Verbose_Description" "Verbose_Description" is an element of "Description". It shows the description of the device. 2.3 "Interface" "Interface" is an element of device. It contains the description of the interface of the device. It contains two

	members: “Signal” and “Reading”. 2.4 “Signal” “Signal” is a member of “Interface”. It contains the description of signals transmitted via the interface. 2.4.1 “id” “id” is an elements of “Signal”. It defines the signal id of the device. It starts with ‘s’ and followed by a numerical number. 2.4.2 “Operation” “Operation” is an element of “Signal”. It defines the operation of the signal. The example operation is input or output. 2.4.3 “Type” “Type” is an element of “Signal”. It defines the type of signal. The example type is digital or analog. 2.4.4 “Measurement” “Measurement” is an element of “Signal”. It defines the measurement of signal. 2.5 “Reading” “Reading” is an element of “Interface”. It contains the information of readings converted from the signal. 2.5.1 “id” “id” is an element of “Reading”. It defines the identification of reading. 2.5.2 “From” “From” is an element of “Reading”. It defines the id of signal from which to read 2.5.3 “Type” “Type” is an element of “Reading”. It defines the type of the device. 2.5.4 “Measurement” “Measurement” is an element of “Reading”. It defines what the device measures. For a temperature sensor, it measures temperature. For a noise sensor, it measures

	noise. 2.5.5 “Unit” “Unit’ is an element of “Reading”. It defines the unit of the reading result. For instance, the unit of temperature can be centigrade or fahrenheit. 2.5.6 “Data_Type” “Data_Type” is an element of “Reading”. It defines the return data type of the lower level driver. 2.5.7 “Return_Length” “Return_Length” is an element of “Reading”. It defines the length of the returning data of the lower level driver. 2.5.8 “Computation” “Computation” is an element of “Reading”. It defines how to convert the data returned from the lower level driver. It contains two elements: “Type” and “Expression”. Usually type is formula. “Expression” is the detailed way to convert raw data. **3 DDL Parser (Python)** DDL is based on the concept that the availability of standard device descriptions allows for the development of intelligent environment applications integrating multiple sensors, actuators and complex devices. By representing each device as a service in the service-oriented architecture(SOA), programming the smart space becomes as easy as calling a number of methods from the device services.

Implementation and testing of IoT Description Language, Language processing and code generation	**1 DDL Structure(JSON):**
	Devices
	“Description”: {
	“Name”: // it defines the name of the device
	“Device_Type”: // it defines the type of the device
	“Verbose_Description”: // it shows the description of the device
	},
	“Interface”: {
	“Signal”: [

	{
	"id": // it defines the signal id of the device
	"Operation": // it defines the operation of the signal, input or output
	"Type": // it defines the type of signal, digital or analog
	"Measurement": // it defines the measurement of signal
	}
	],
	"Reading": [
	{
	"id": // it defines the id of reading
	"From": // it defines the id of signal where to read
	"Type": // it defines the type of the device
	"Measurement": // it defines what the device measures
	"Unit": // it defines the unit of the reading result
	"Data_Type": // it defines the the return data type of lower level driver
	"Return_Length": // it defines the length of the return data
	"Computation":
	"Type": // it defines the type of computation
	"Expression": // it defines the expression of computation
	}
	]
	}
	
	2 DDL parser
	DDL Parser helps parse the ddl file and generate device driver as much as possible. Our DDL Parser is implemented by Python which can import JSON file and make an analysis easily.

Implementation and testing of overall on-board driver code (upper- and lower-level drivers, including generated code)	10 The upper-level drivers tmp36read.c, ledread.c ledwrite.c and dht11read.c are automatically generated from DLL. The lower-level drivers are manually written, which are readdht11.c, readtmp36.c, readled.c, writeled.c. The upper- and lower-level drivers are tested with Led, DHT11 temperature sensor and TMP36 temperature sensor.

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	50% We generated all the upper-level of the drivers.
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 use Json to represent the externalization abstraction of device. Each device will have the following information: Name, IP, id, data, status. The reason we choose the abstraction is that Json is well used in web technology
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.	The edge will broadcast a request to all of the devices in its subnet every specified period with “IP” in the data. Each platform will return its ip address and recorded by the edge, this will be the configuration part. Then to the “alive” platform, the edge will send “Read” request, the platform will read all its devices data and send back to edge. The edge then will know what devices are on the platform. This will be the initialization part.
Give the details of the externalization abstractions design.	The externalization abstraction will include the basic information of what the device is and what can the device do.
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	As the initialization and configuration process finished. The edge will have the information of what and how the devices on each platform can be accessed. The devices is specified by the ip of each platform and the id of the device. The implementation will have the generalization of all the devices, also the energy the communication between is request driven which will reduce the energy and the benefit from the generalization of the abstraction design the implementation will also be efficient.

your implementation.	
Describe your on-board IoT devices Demo App.	**Devices**: Led, TMP36, DHT11 **App:** The Led_0 will turn on if the humidity is higher than a certain level. Led_1 will turn on if the temperature of DHT11 is higher than a certain level. Led_2 will turn on if the temperature of TMP36 is higher than a certain level.
Describe your web-based IoT devices Demo App.	The Demo App will have the full control of each platform. It can measure and record each devices' data and turn on/off of Led. We use JAVA to develop the edge on the glassfish4 server and develop the web-based APP on the nginx server. By calling restful service api through AngularJS, the APP can transfer data with the edge.

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)?

1. The most time consuming part is find the way using GPIO in Xinu. Because most of sources online about GPIO is in Linux and Linux has its original device tree but Xinu does not include it. So we take a lot of time to figure out how to use GPIO.
2. Find each GPIO I/O pin. Xinu includes 4 GPIO and each GPIO includes 32bits which means it has 32 pins. So when we want to use one pin as output or input, we need to find pin value and corresponding pin in BBB. So it still takes some time to find corresponding and set the correct value.
3. It is difficult to find out how to communicate the edge and Xinu, then we choose UDP as the way of communication.
4. One of the most consuming part is reading data from sensors. At the beginning, we did not know how to configure the ADC before reading data from the pin. And the information from the AM335x technical reference manual is not explicit enough. I want you to provide us some guide on reading sensor or operating processors. This part is not hard, but we just cannot find the right direction to do that, so if we have more guide we can do better and save more time.
5. This project is the beginning for us to learn IoT. I think it is a good idea to do this project, but the overall experience is not very good. At the beginning, we cost a lot of time on reading sensor, but this part is not the most important part. During the whole process, we always spend our time on some unimportant parts and we cannot find the right direction to do the project. We don't know the purpose of this project. The whole blueprint of IoT is great, but we cannot implementation this by a small project. We cannot find out the purpose or motivation to do this project.

6. One of the challenge is to understand what we should do for the project. It takes us about two weeks to understand the project description. The second challenge is to be proficient in controlling gpio for Xinu. There is little online available xinu gpio material. The third challenge is to come up with appropriate high level interface for IOT. What should DDL generate is also a little bit confused because of the division of high level and low level drivers. I think the project description should be more clear so that we can know what we should do and in a right direction.

3. Overall Experience

Overall experience. Describe your overall experience good or bad.

1. Chujia Liu:
   Working as the team leader of the project. The overall experience of this project is not so good, maybe it's because this year is the first year of the project. The project description is painting a very huge picture whereas the project itself is digging to the ground. Though I believe the professor has his good wills on encouraging us to think creatively but the description does not seem to be a proper guide for it, especially when most of us do not have profound background in IoT development. Most of the time is waste on clarifying and understanding what shall we do, what can/can't we do.

2. DanLi Wang:
   For overall experience, I think it's a little hard for us. The final project makes us understand high-level driver, low-level driver, high-level interface and Cloud-Edge-Beneath and figure out how to make them work. But the project includes some fuzzy information so that we cannot get the correct request of project. And there is fewer information about XINU from website, that cost a lot of time to figure out how the BBB with XINU works at the beginning. But generally, we learnt a lot of knowledge from this project and did all projects by ourselves.

3. Jinhao Zhu:
   The description of the project is hard to understand but the process of understanding the project expands my knowledge.

4. Liyue Wang:

This project is the beginning for us to learn IoT. I think it is a good idea to do this project, but the overall experience is not very good. At the beginning, we cost a lot of time on reading sensor, but this part is not the most important part. During the whole process, we always spend our time on some unimportant parts and we cannot find the right direction to do the project. We don't know the purpose of this project. The whole blueprint of IoT is great, but we cannot implementation this by a small demo. We cannot find out the purpose or motivation to do this project.

5. Jiangjiang Zhu
   It is somewhat torturing because we wasted too much time understanding the purpose of the project. However, we learned a lot of concepts and have a better understanding of operating system and Internet of Things. Hands on design and implementation make us more skillful in doing a hard project of operating system.

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.