README

**Hardware needed:**

Sensors used: Temperature sensor: TMP36, Sound Sensor: Phantom-SR, LED lights as actuators.

Two Beagel Bone Blacks, router and PC to run screen for on boarded XINU.

**Folder Structure:**

Client: Code for client running on a node.js webserver

DDLParsing-Code Gen: Code for parsing DDL validating it and generating code

Edge\_Cloud: Code for edge server and cloud is present in this folder.

Xinu-code-BeagleBoneBlack: This folder has complete code with new Makefile ,parsing/code-generation code, device driver code. This is complete code in itself. You will need to run cloud, edge server and client from above mentioned folders to run it.

Steps to run the project:

1) Download the code

2)Install golang,java1.8,node.js

3)start cloud > go run dos\_cloud.go

4)start edge server > go run edge\_server.go

5)setup temperature sensor (beagleboard1), (sound sensor and actuator LED(beagleboard2)) on a breadboard and connect to two different Beagle Bone Black running XINU.

6)Boot up two beagle boards to which sensors are connected.

7)Start the webserver on which client will run.

8)Open home.html and fetch cloud.

9)In order to run Temperature sensor change GPIO\_WRITE to “1” and in order to use sound sensor change GPIO\_WRITE in main to “2”.

Description of Project:

**IOT Device Implementation using Beaglebone Black & Xinu OS**

In order to implement the project the following components have been used :-

* Sensor
* Client
* Edge Server
* Cloud
* DDL Parser/Code-generator
* Sensor

We have used the following devices as sensors for the project :-

* Analog Sensor : TMP-36 Temperature Sensor
* Digital Sensor : Phantom Sound Sensor
* Actuator : LEDs

Communication between sensors and Beaglebone handled using device drivers.

**CODE : Xinu-code-BeagleBoneBlack.tar.gz**

2. Client

Client is a HTML and javascript implemented component. When it is up it calls cloud which returns all device information in JSON. This information is represented by client for users. When a device becomes available user enters which operation and device on which that operation needs to be performed. After this data is taken it is validated by getting data from cloud, once validated a request is sent to edge server which in turn calls the sensors and returns data to cloud in JSON format. This is shown by client in tabular format.

**CODE: home.html**

3. Edge Server

Edge server is a Golang application running on PC. It communicates with Beaglebone black devices using **UDP protocol .** It also communicates with **Cloud application** and **Client website** application using HTTP Web-Client-Server model.

**CODE : edge\_server.go**

4. Cloud Server

Cloud is a web-server application written using Golang. It uses the device DDL json in order to update the device information repository. It communicates with client and edge for exchanging device information.

**CODE : dos\_cloud.go INPUT DDL : devices.json**

5. DDL Parser/Code Generator: This component provides an abstraction to user who wants to be agnostic of platform for which they have to write code. Code generation/ XML validation and parsing is done using Java. Code generation and parsing is done when XINU complies.

1: DDL Validation

2:DDL Parsing

3:Driver Code Generation

**Code: ADCDOMParser.java GPIODOMParser.java, Input.xml,ValidationCode.java and ADCInput.xml**