Mobile Computing WS-19/20– IoT Smart Home

Child Monitoring

Name : Yash Vyas

Matriculation Number : 1266490

Guidance : Mr. Gregor Frick

Mr. Besfort Shala

Prof. Armin Lehmann

Table of Contents

Acknowledgement

I would like to thank Prof. Lehmann, Mr. Frick , Mr. Shala for the continuous support and guidance provided for this project. I am very much thankful for providing the open platform via moodle to ask question , clearing doubts and sharing valuable suggestions. Furthermore, I would also like to put my sincere gratitude for providing all the necessary information and useful links via moodle which proved to be very helpful for successful completion of this project.

I) INTRODUCTION

.

In modern times with the development of technology many traditional problems like handling house appliances and gadgets resulted with providing flexible environment and secure future . Iot applications are booming topic for present life conditions to get better by integration of house with smart appliances which can be used for taking care of repetitive task which traditionally required. Human presence to complete a task like { appliances: On, Off}

Following advances can be used in smart home to achieve Child Monitoring system which can detect, prevent, notify about the events which can harm one's children health or security which is the aim of following system.

Functionality of present Smart Home Technology available today:

- Controlling Home appliances via smartphone, Laptop,
- Putting scheduler task for system like water system in smart home gardens or heating of coffee
- Grocery reminder.
- Automatic Temperature Control
- Fire Alarm system.
- Humidifier which can auto actuate itself.

Figure 1 Smart Home IOT

II) Use Case Description:

1. Environment condition controlling:
   Consider the scenario in which child room is equipped with heater sensor, temperature sensor, heater actuator. Now in winter as well summer season the temperature fall and rise can make child sick. Following jumps in the temperature can be reduced by smart system which control the temperature of the room within certain range and if the room temperature falls below or rises above certain range then actuator is activated automatically to control the temperature. Scientific studies show that the children in stable environment sleeps better which in turn helps for better growth of children.

2. Child monitoring via camera and microphone.
   Consider the scenario in which parent want to monitor the child in her absence they can have the monitoring facilities available because of camera and microphone sensor. Camera will provide live feed of the room while microphone will provide sound feed of the room. In Iot Gateway by using the neural network the child crying sound can be identified which can be used to actuate the alarm which notifies the user about the present attention required situation. Another situation is when parents are in different room and child is sleeping or playing in the room for keeping eye on child's activity the camera and microphone combo is used for providing security.

3. Child heart beat monitoring:
   In early years of child, heart monitoring is helpful to get the information about child's health and can be used to detect early stages of heart borne diseases. Iot Gateway processes the data from sensor and actuate alarm if the child heartbeat reading are above or below certain ranges.

4. Using room sensor for the betterment in child monitoring:
   Speaker sensor are used to play calm music or night stories which helps in kids mental growth and maintain calm and serene environment for the child
   Light sensor and actuator are used to control light in the room which helps in saving electricity and time.

III) Project Details:

Aim: Use present Iot Technology for Child Monitoring System.

Smart-Home details:

scenario: Children’s room are equipped with following sensors and actuators

Sensor	Actuator
TemperatureSensor	TemperatureActuator
CameraSensor	CameraActuator
MicrophoneSensor	MicrophoneActuator
SpeakerSensor	SpeakerActuator
HeartBeatSensor	HeartBeatActuator
HeaterSensor	HeaterActuator

Description:

TemperatureSensor	**To measure temperature of the room.**
CameraSensor	To provide live feed of the room.
MicrophoneSesnor	To provide the sound feed of the room
SpeakerSensor	To play music in the room
HeartBeatSensor	To measure heartbeat of child.
HeaterSensor	To increase temperature in the room
TemperatureActuator	To change powerstatus of temperaturesensor.
CameraActuator	To change powerstatus of camerasensor
MicrophoneActuator	To change powerstatus of Microphonesensor
SpeakerActuator	To change powerstatus of Speaker and to playmusic.
HeartBeatActuator	To change powerstatus of heartbeat sensor.
HeaterActuator	To change powerstatus of Heater Senor

3.1) Project Architecture:

Child monitoring system architecture is based on web server and client which exchanges data via http protocol on the network which communicates with sensor and actuators of the smart home via Coap protocol

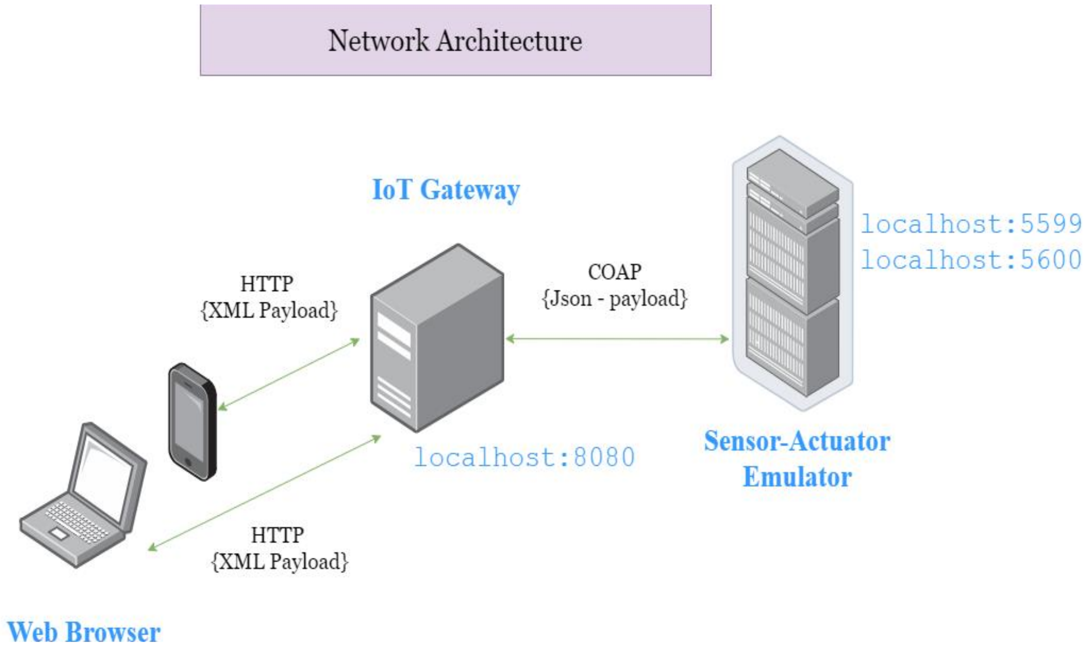


Figure 2 Network basic architecture

3.2) Protocols used for communication:

- COAP {CONSTRAINED APPLICATION PROTOCOL}
- HTTP {HYPER TEXT TRANSFER PROTOCOL}

a) COAP Protocol:

Constrained Application Protocol is a specialized Internet Application Protocol for constrained devices, as defined in RFC 7252. It enables those constrained devices called "nodes" to communicate with the wider Internet using similar protocols.

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine- to-machine (M2M) applications such as smart energy and building automation. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

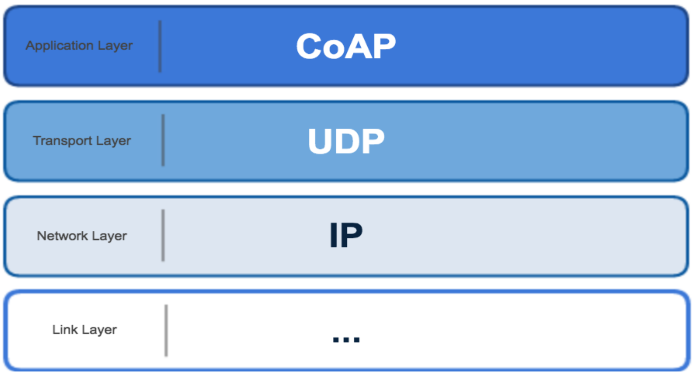


Figure 3 COAP over UDP protocol stack

The sensors and actuators in the emulation environment communicate to the gateway via CoAP over UDP on their respective L4 port addresses. The CoAP interface of the gateway and the sensors and actuators in the emulator environment together constitute a Constrained ReSTful Environment In Coap network sensors and actuator are sending Json object as payload CoAP is one of the latest application layer protocol developed by IETF for smart devices to connect to Internet. Thus lightweight protocol CoAP is intended to be used and considered as a replacement of HTTP for being an IoT application layer protocol [2]. Unlike HTTP based protocols, CoAP operates over UDP and employs a simple retransmission mechanism instead of using complex congestion control as used in standard TCP. It uses a unique Transaction ID to identify each GET request for retransmission purposes to keep reliability.

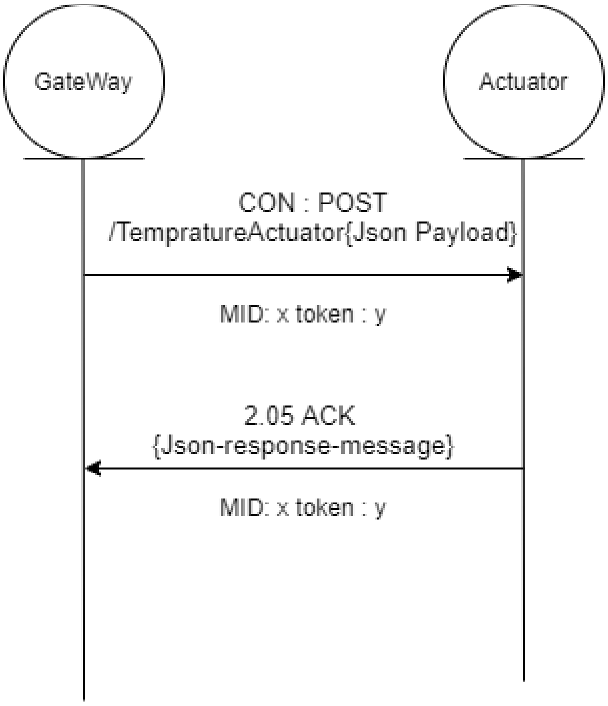


Figure 4 Coap POST for resource at /TemperatureActuator

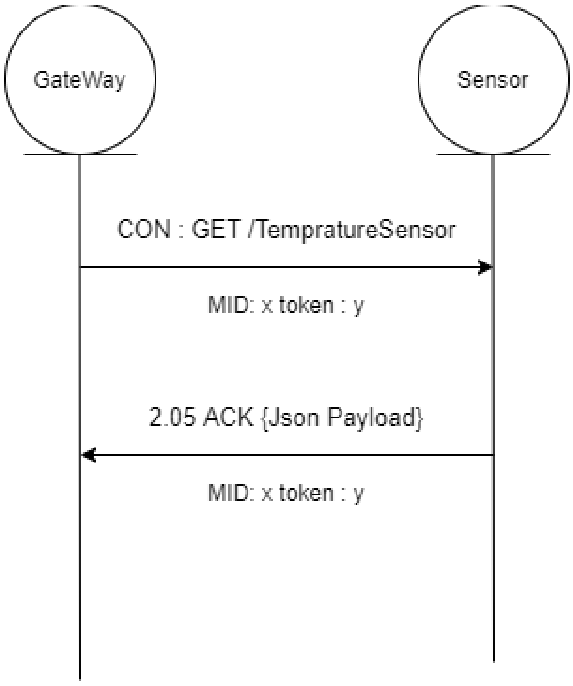


Figure 5 Coap GET for resource at /TemperatureSensor

In the similar manner message sequence chart can be generated for the following calls:

- Get /TemperatureSensor
- POST /TemperatureActuator
- GET /HeartBeatSensor
- POST /HeartBeatActuator
- GET /LightSensor
- POST /LightActuator
- GET /MicrophoneSensor
- POST /MicrophoneActuator
- GET /SpeakerSensor
- POST /SpeakerActuator
- GET /HeaterSensor
- POST /HeaterActutator
- GET /CameraSensor
- POST /CameraActautor

b) Http : HyperText Transfer Protocol

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over connection with a server. The server responds with a status line, including the message's protocol version and a success or error code followed by a MIME-like message containing server information, entity metain formation, and possible entity-body content [2].HTTP is a communication protocol which is employed for delivering data usually HTML files, multimedia files, etc.) on the World Wide Web through its default TCP port 80. However, there are other ports also which can be implemented for this function. HTTP has two different versions, HTTP/1.0, which is the old one and the newest HTTP/1.1. In its older version, a separate connection was required. In the case of a new version, the same connection can be recycled several times [2] The Web browser communicates with the gateway via HTTP ReST messages [4]. The application utilizes HTTP GET and POST methods in this case. All of the HTTP resources of the gateway sends and receives payload in XML content format.

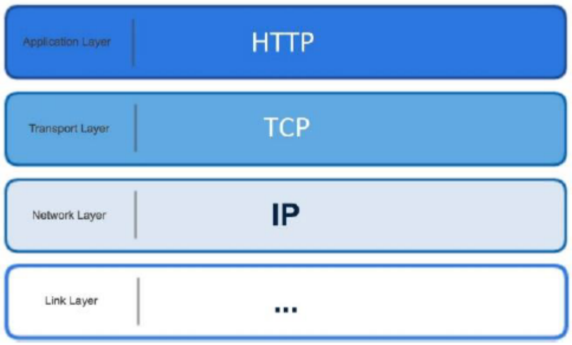


Figure 6 Http over TCP Stack

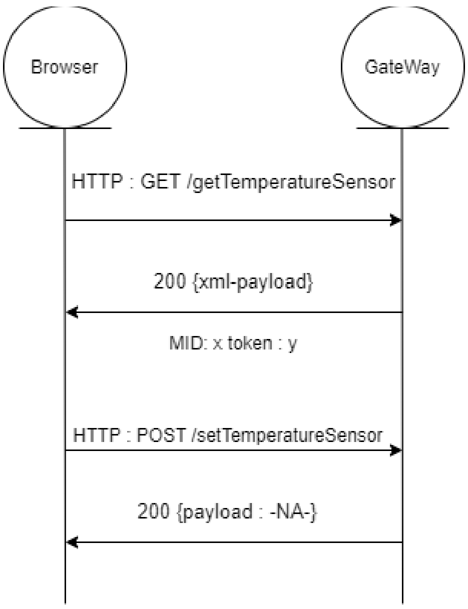


Figure 7 Http GET POST TemperatureSesnor

IV) Implementation Strategy

The project has two major components – Emulator for sensor-actuator environment and IoT gateway.

a) *Emulator*

The emulator has all the sensors and actuators related to the use-case residing in localhost.Two main servers are running on localhost 5599 and 5600 which are hosting Actuator Server and Sensor Server. The emulator nodes all use coap over UDP scheme. Each sensor and actuator runs a CoAP resource on a CoAP server(Datagram Server) on a single port performing its respective function(reading temperature, etc.). The payload that is sent or received by the hosts in the emulator is of JSON content format.

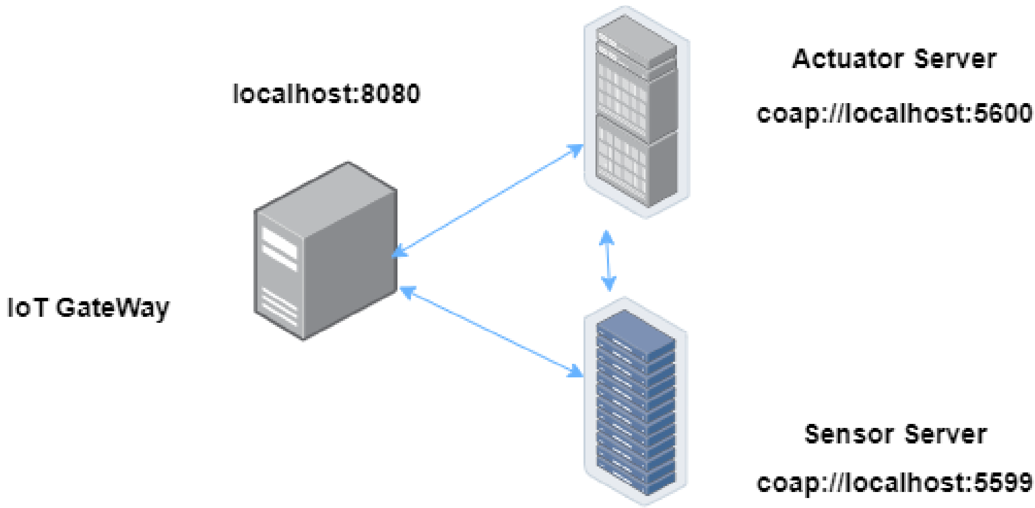


Figure 8 IoT Gateway and Coap Servers

b) Iot GateWay

The IoT gateway has two interfaces . As shown in the figure the interface communicating via CoAP over UDP, resides on localhost:62235. It communicates to the sensors and actuators residing in the emulator environment. It hosts multiple CoAP client services and subscription listeners on a single endpoint at localhost:62235.It communicates with the emulator environment with JSON payload content format. The gateway also has a web-console which it hosts at the address localhost:8080. The web-console resides under the URL- *http://localhost:8080/console*. The gateway communicates with the browser over this interface via HTTP/1.1 ReSTful services (the descriptions of these HTTP service endpoints are provided in HTTP URI summary section). The services produce and consume payload of XML content format.

c) Technology Stack:

IDE: Eclipse 2019-09 R (4.13.0)
Framework: Spring-Boot -version 2.2.2.Release
Backend Technologies: Core JAVA, J2EE, Inbuilt support for Tomcat webserver, XML, JSON
Frontend Technologies: HTML5, Bootstrap , JavaScript, Jquery
Protocol Implemented: HTTP/1.1 and CoAP(IOT)→Eclipse californium
Build Tool:- Maven 4.0.0 (Pom.xml)

d) SNapShot of SensorDataFile:t

```
LightSensor : ON,
MicrophoneSensor : ON,
HeaterSensor : OFF,
SpeakerSensor : ON,
HeartBeatSensor : ON,
CameraSensor : ON,
TemperatureSensor : ON,
TemperatureSensorReading : 22,

```

V) Application Logic

- **Case 1 : Auto Temperature Controlling:**
  Temperature in the room can be maintained by setting the range in the application. By default room temperature is maintained between18-22 .

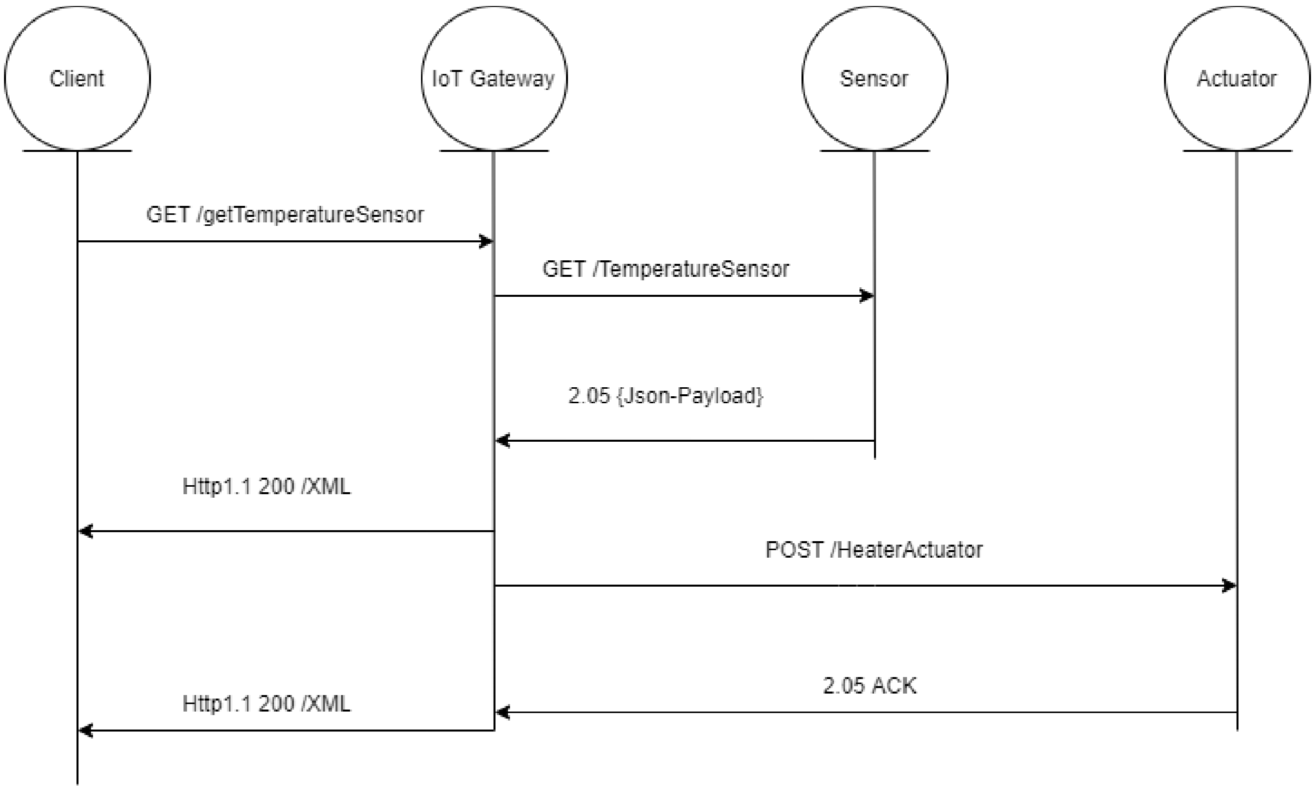


Figure 9 Message sequence chart for case1

Client send http Get /getTemperatureSensor for temperature reading of the room, Iot Gateway processes the data, if the data is not in the defined range 18-23 then POST /HeaterAcutator is sent to Actuator which turns on the Heater in the room

- ## Case 2: HeartBeat Monitor:

HeartBeat sensor reads child's heartbeat and send it to IoT Gateway which processes the data and if the heartbeat is not in range of 80-120bpm then alarm actuation is done and notification Is sent to the Client.

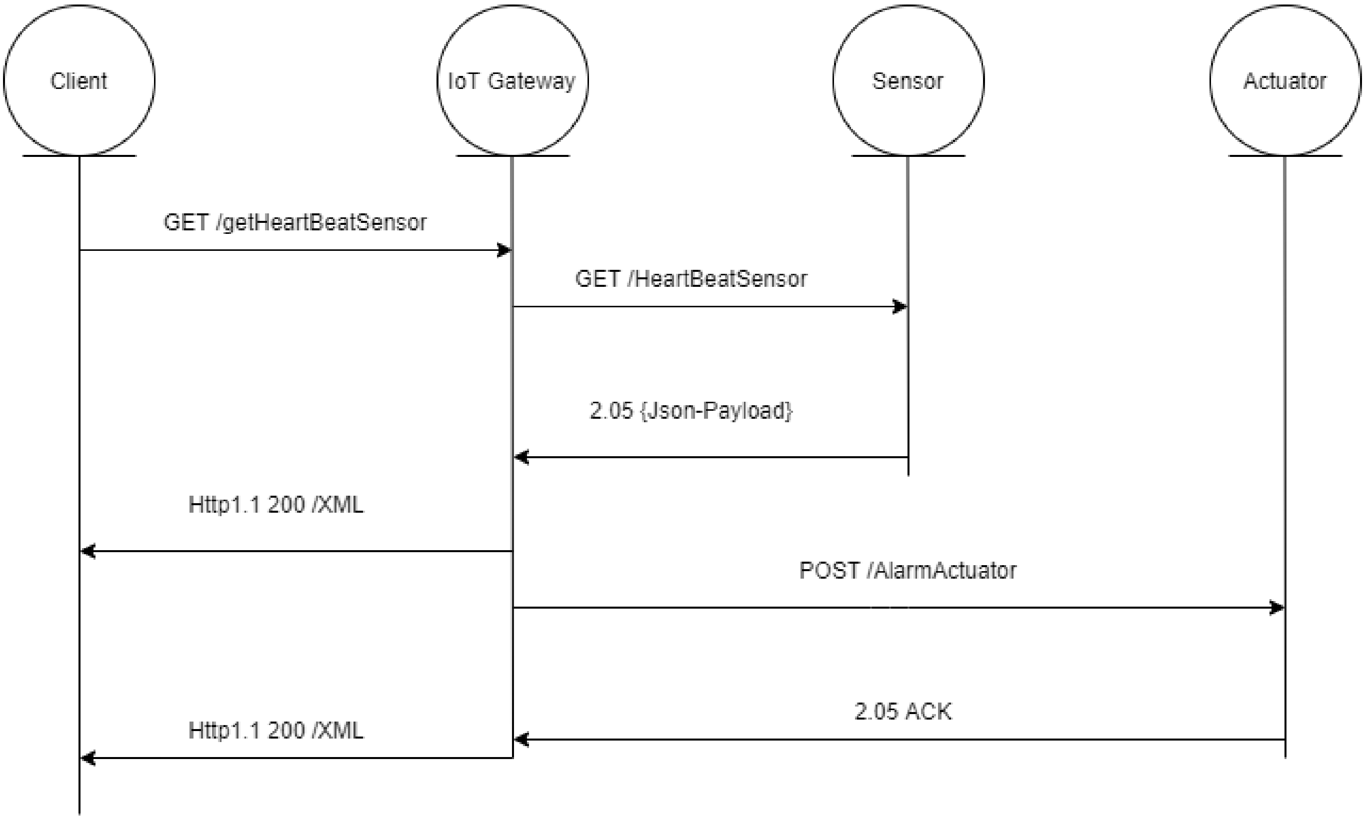


Figure 10 Message Sequence Chart for case2

Client send http Get /getHeartBeatSensor for heartbeat reading of the child, Iot Gateway processes the data, if the data is not in the defined range then POST /HeaterAcutator is sent to Actuator which turns on the Heater in the room 80-130 then POST /AlarmAcutator is sent to Actuator which sends the notification to the parent

- ### Case 3: Microphone Monitor:

Microphone sensor gets the sound data from the child's room Iot Gateway processes the data if the data is in the range of child's crying frequency alarm notification is actuated which sends the notification to the parent.

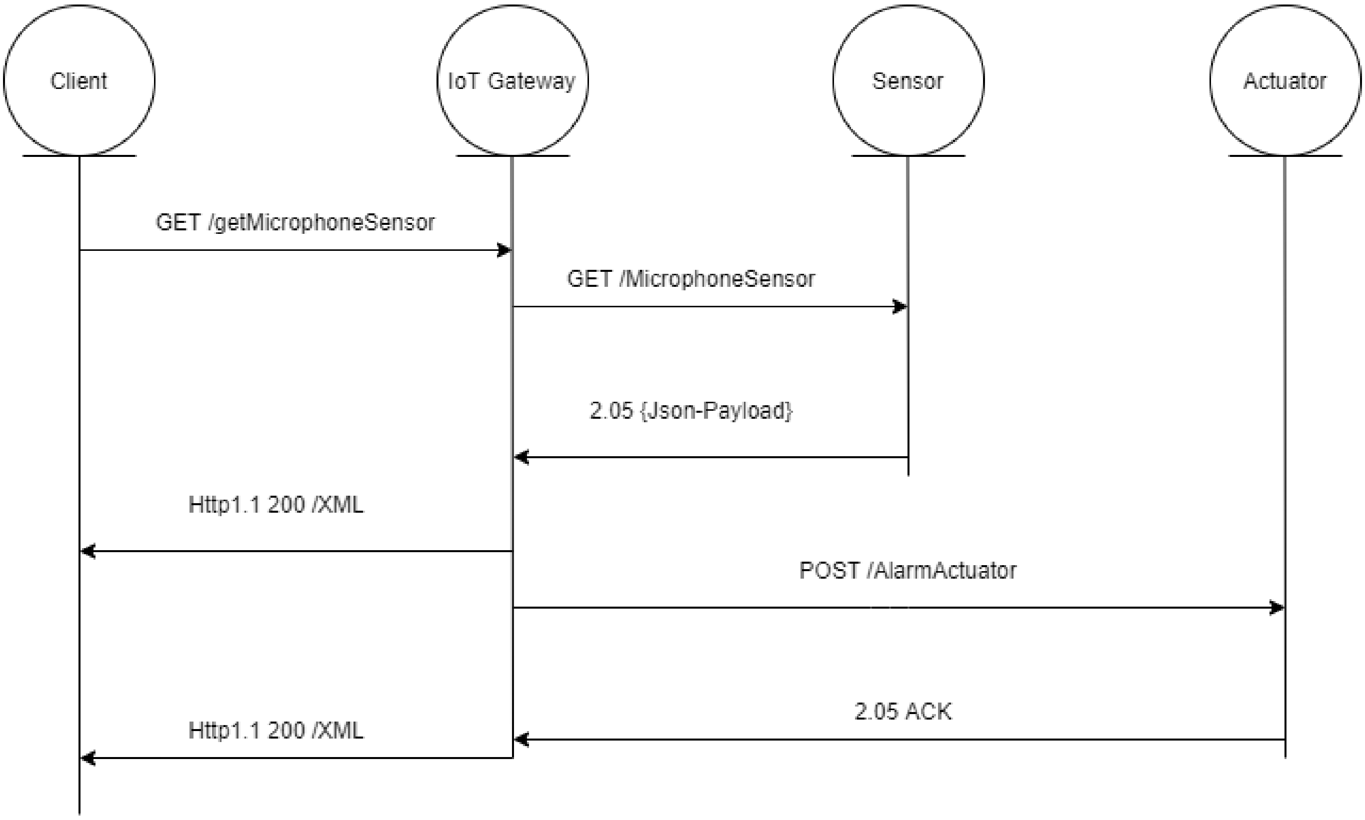


Figure 11 Message Sequence chart for case3

VI) Project Structure

```
├───.mvn
│   └───wrapper
├───.settings
├───.sts4-cache
├───src
│   ├───main
│   │   ├───java
│   │   │   └───com
│   │   │       └───example
│   │   │           └───demo
│   │   └───resources
│   │       ├───static
│   │       │   └───image
│   │       └───templates
│   └───test
│       └───java
│           └───com
│               └───example
│                   └───demo
└───target
    ├───classes
    │   ├───com
    │   │   └───example
    │   │       └───demo
    │   ├───META-INF
    │   │   └───maven
    │   │       └───com.example
    │   │           └───demo
    │   └───static
    ├───generated-sources
    │   └───annotations
    ├───generated-test-sources
    │   └───test-annotations
    ├───maven-archiver
    ├───maven-status
    │   └───maven-compiler-plugin
    │       ├───compile
    │       │   └───default-compile
    │       └───testCompile
    │           └───default-testCompile
    ├───surefire-reports
    └───test-classes
        └───com
            └───example
                └───demo
```

Figure 12 Project Structure

VII) WIRESHARK CAPTURES

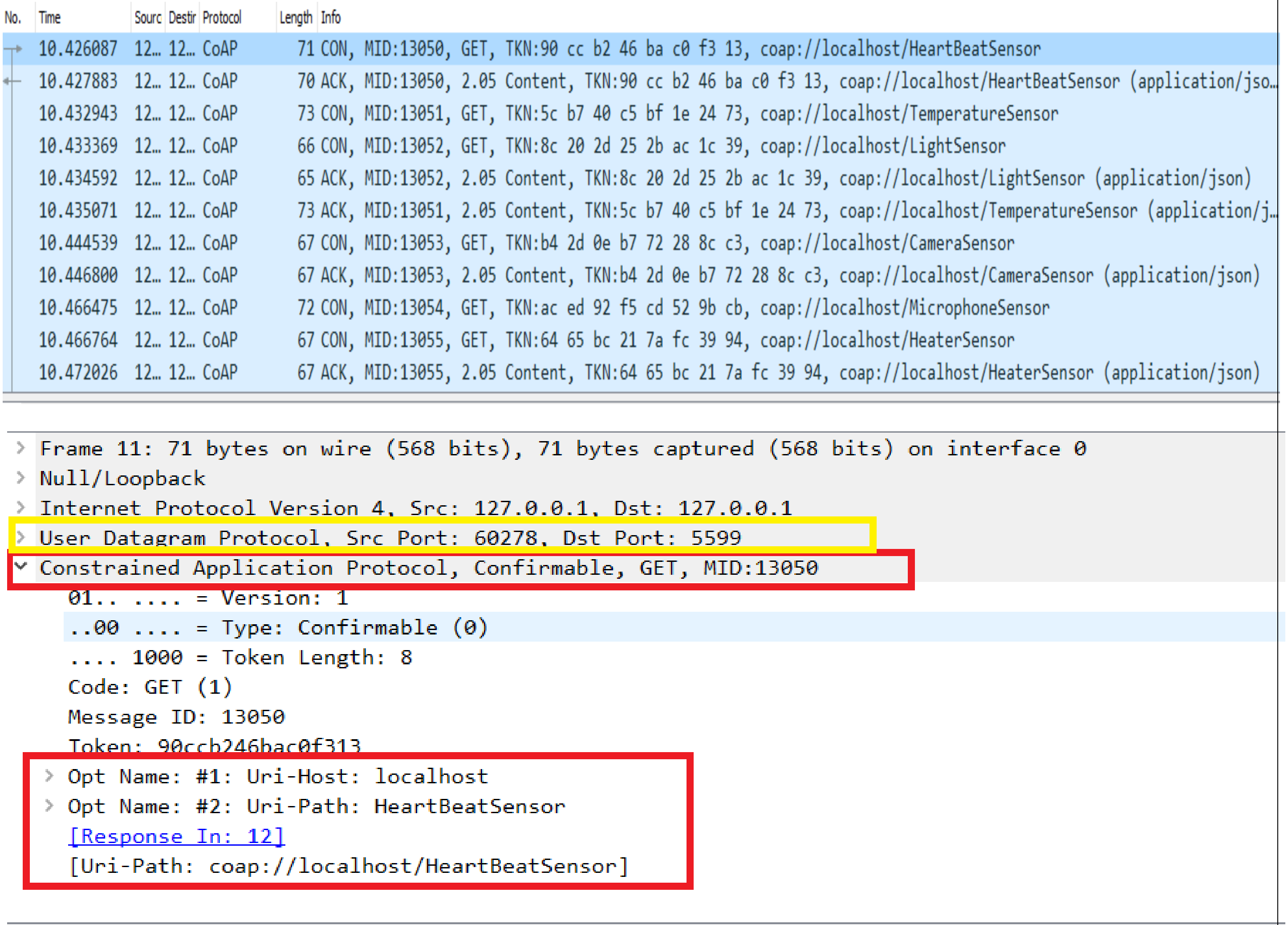


Figure 13 Wireshark Capture for HeartBeat sensor

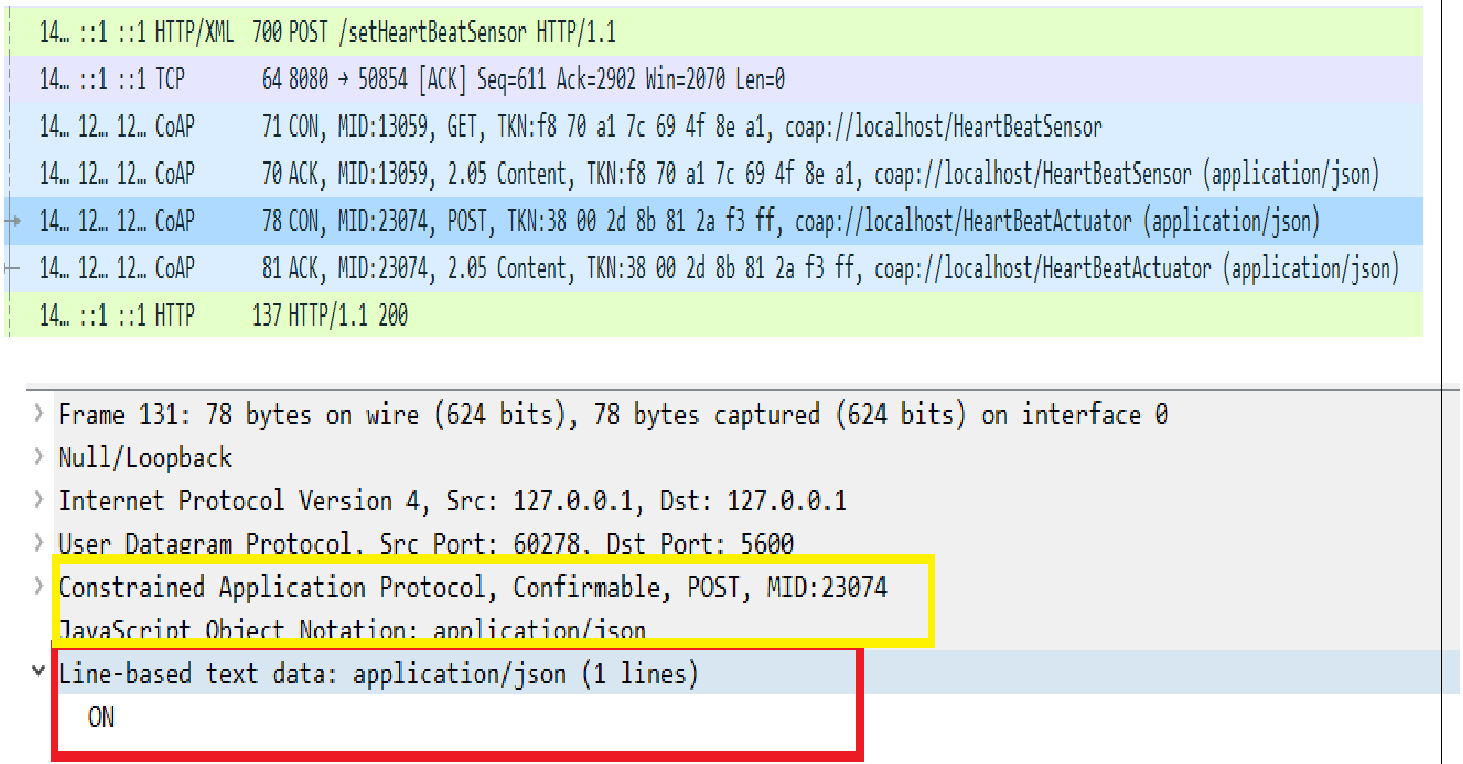
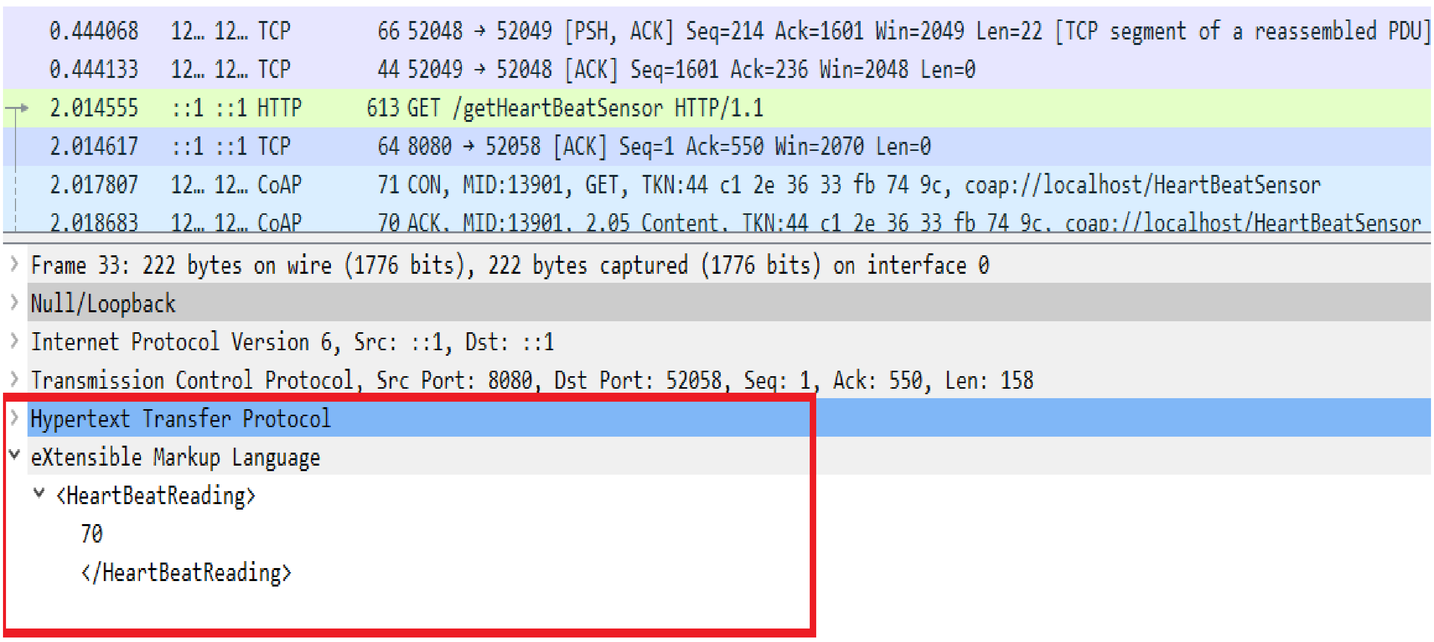


Figure 14 Wireshark capture for heartbeat actuator



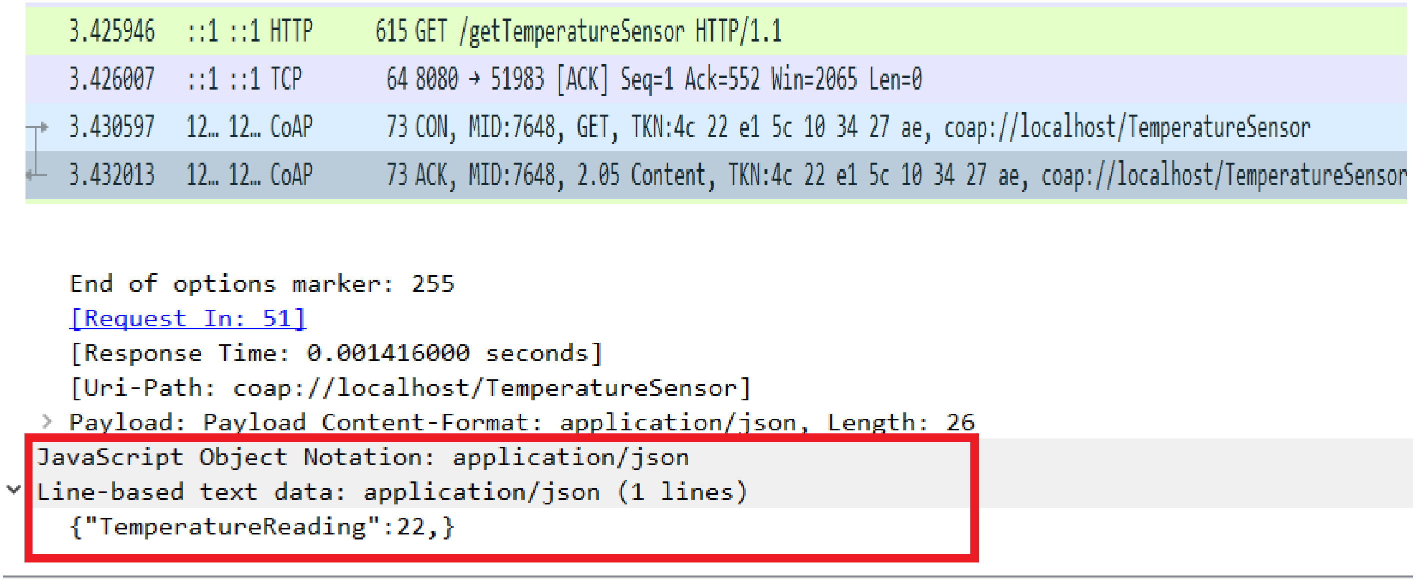


Figure 15 Wireshark capture for Temperature Sensor

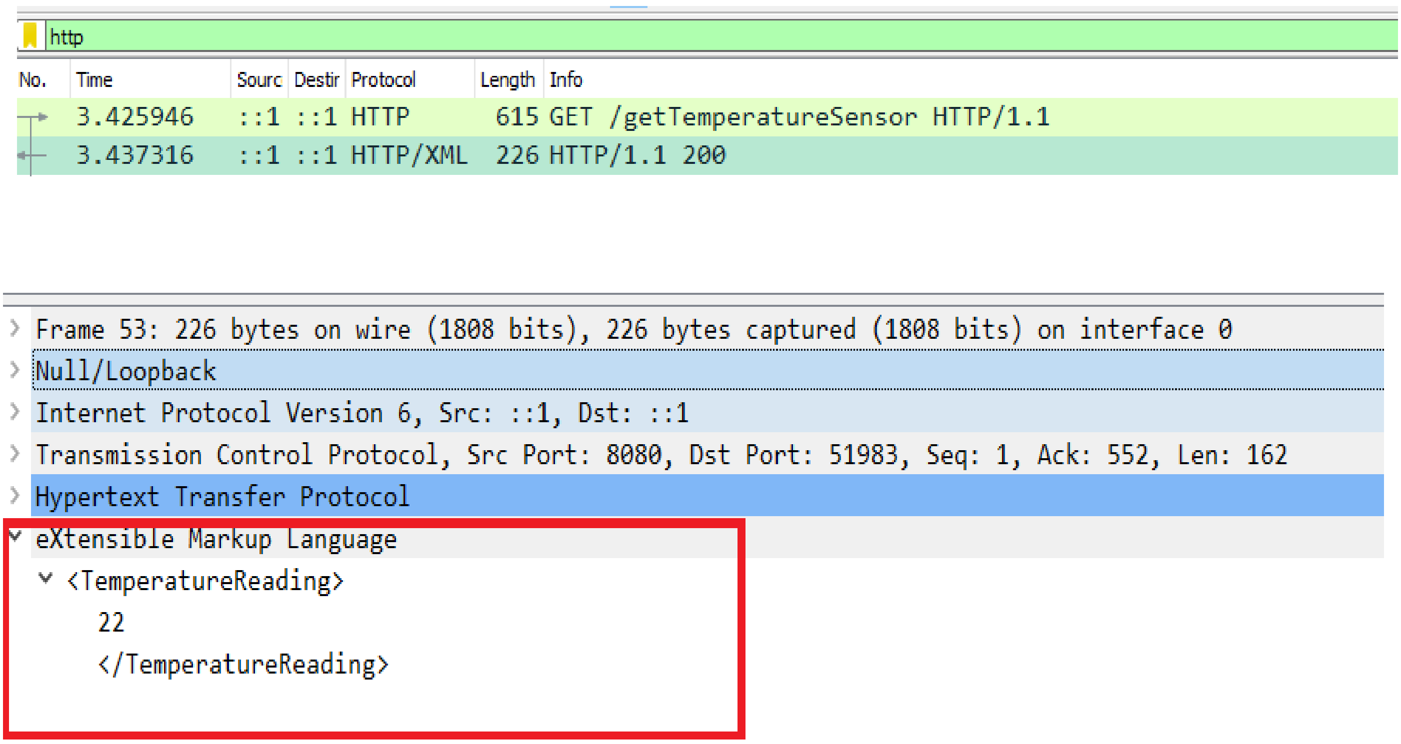


Figure 16 Wireshark capture Temperature reading

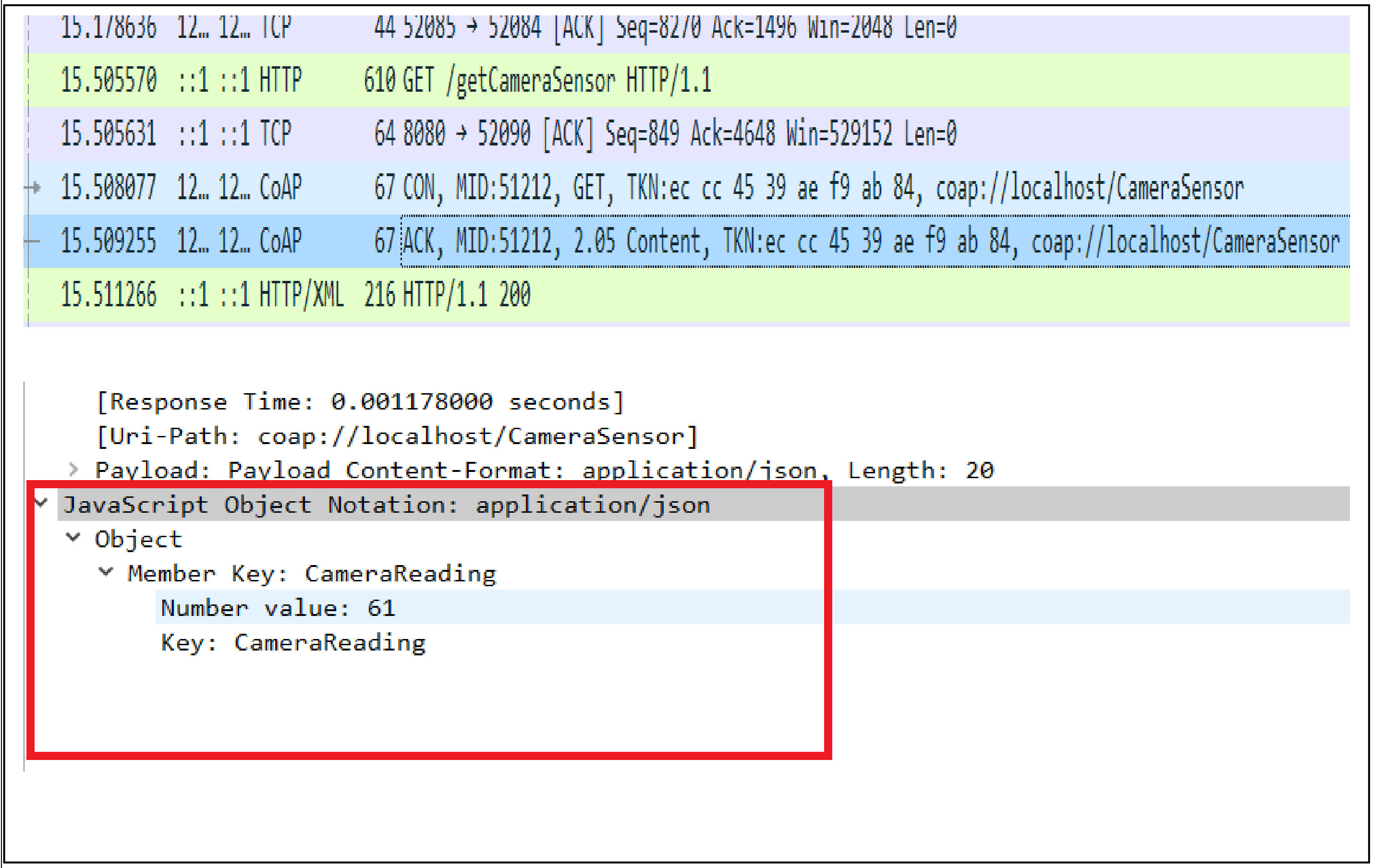


Figure 17 Wireshark capture of Camera Reading

VIII) URI SUMMARY

Coap URI:	Resources
coap://localhost:5599/TemperatureSensor	Temperature Sensor
coap://localhost:5599/CameraSensor	Camera Sensor
coap://localhost:5599/HeaterSensor	Heater Sensor
coap://localhost:5599/MicrophoneSensor	Microphone Sensor
coap://localhost:5599/LightSensor	Light Sensor
coap://localhost:5599/HeartBeatSensor	HeartBeat Sensor
coap://localhost:5599/SpeakerSensor	Speaker Sensor
coap://localhost:5600/TemperatureActuator	Temperature Actuator
coap://localhost:5600/CameraActuator	Camera Actuator
coap://localhost:5600/HeaterActuator	Heater Actuator
coap://localhost:5600/MicrophoneActuator	Microphone Actuator
coap://localhost:5600/LightActuator	Light Actuator
coap://localhost:5600/HeartBeatActuator	HeartBeat Actuator
coap://localhost:5600/SpeakerActuator	Speaker Actuator

HTTP URI:	Description
http://localhost:8080/getTemperatureSensor	Temperature Sensor
http://localhost:8080/getCameraSensor	Camera Sensor
http://localhost:8080/getHeaterSensor	Heater Sensor
http://localhost:8080/getLightSensor	Light Sensor
http://localhost:8080/getHeartBeatSensor	HeartBeat Sensor
http://localhost:8080/getMicrophoneSensor	Microphone Sensor
http://localhost:8080/getAlarmSensor	Alarm Sensor
http://localhost:8080/setTemperatureSensor	Temperature Actuator
http://localhost:8080/setCameraSensor	Camera Actuator
http://localhost:8080/setHeaterSensor	Heater Actuator
http://localhost:8080/setLightSensor	Light Actuator
http://localhost:8080/setHeartBeatSensor	HeartBeat Actuator
http://localhost:8080/setMicrophoneSensor	Microphone Actuator
http://localhost:8080/setAlarmSensor	Alarm Actuator
http://localhost:8080/setHeaterSensor	Heater Actuator

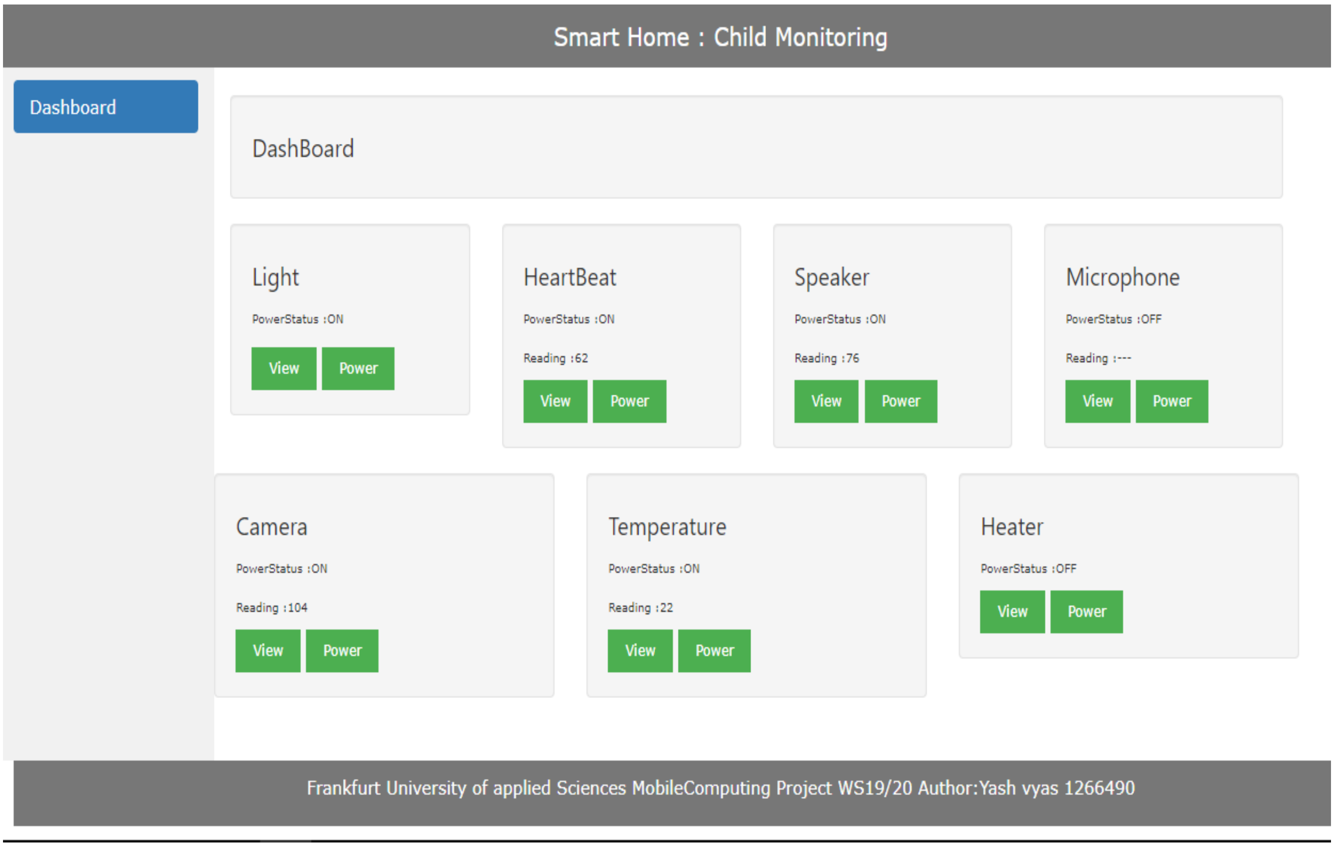


Figure 18 Dashboard Screenshot

IX) Project Setup Instructions

Ubuntu

1. Move the jar *iotChildMonitoring-1.0.0.jar* to the folder where you wish to run the application.
2. The application will create a *Californium.properties* file (properties file containing CoAP stack configuration values) and *SensorData.txt* file, so make sure the required permissions are given to the application to enable creation of necessary setup files and folders.
3. Open shell terminal.
4. Make sure that none of the ports used by the project are used by some other application running on the machine. Execute *lsof -i -P* and check with the list of ports used by this smart home application mentioned in Appendix A. If there are any application running on the ports used by this project , close that application by executing *kill -9 {pid-of-the-application}*.
5. Navigate to the folder where you have moved the application jar file.
6. Execute *java -jar iotChildMonitoring-1.0.0.jar* for running the application server.
7. After you see the application has successfully started running ,open a new browser tab and navigate to the web dashboard of the smart-home use-case at *http://localhost:8080/*.
8. Respond to the requested prompts to observe the use case operation.

Windows

1. Move the jar *iotChildMonitoring-1.0.0* to the folder where you wish to run the application.
2. Open a command shell prompt and navigate to the folder where you have kept the jar file.
3. Execute *java -jar iotChildMonitoring-1.0.0.jar* for running the application.
4. After you see the application has successfully started running , open a new browser tab and navigate to the web dashboard of the smart-home use-case at *http://localhost:8080/*.
5. Respond to the requested prompts to observe the use case operation.

X) REFERENCES

[1] https://tools.ietf.org/html/rfc7252
[2] http://hinrg.cs.jhu.edu/joomla/images/stories/coap-ipsn.pdf
[3] https://www.w3schools.in/http-tutorial/intro/
[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
[5] [Shelby11] Zach Shelby, "CoRE Link Format," draft-ietf-core-link- format-07 https://tools.ietf.org/html/draft-ietf-core-link-format-01
[6] [Z. Shelby13] Z. Shelby, Sensinode, K. Hartke, "Constrained Application Protocol (CoAP)," draft-ietf-core- coap-18. http://tools.ietf.org/html/draft-ietf-core-coap-18