IOT SYSTEM FOR A SMART AND SAFE BIKE-RACK WITH A WEB APPLICATION

Karl Wallentin, Rachaputi Guru Mehar, Asheesh Misra

Uppsala University

***Abstract*—An IoT system prototype with interconnect to a web/mobile application for securing user bikes was designed and implemented using Zolerita-Z1 sensor nodes running contiki-ng OS. Any disturbances in the nodes was captured by sensors and analyzed for any threats to user-bikes and same information was communicated to root-node using TiSCH in the network. This report gives an introduction to the system prototype and succinctly its implementation as well. Also, some analysis is done on RSSI measurements and results are presented based on it. Finally, the report is concluded with its design flaws and future improvements and extensions for feasible eco-friendly commercial but affordable bike-parking system.**

1. INTRODUCTION

The internet of things is no novelty. It has during the last couple of years changed the relationship between humans and physical things. Things we never thought would have internet access or could be "smart" are now using a myriad of sensors and is sending huge amounts of information to the nowadays mundane cloud. The idea of smart bike-rack(SBR), outlines the need of a secure system through use of small IoT devices, to give users a peace of mind while parking their expensive bikes. The prototype that we developed makes use of **contiki-ng**[1] running on small wireless embedded devices with our application. We used **Zolertia-Z1**[2] boards for our design.

2. SYSTEM DESIGN OVERVIEW

The design document consisted of implementing system in three phases or modules. The first: reading sample data from the sensors and doing calculations according to an algorithm, formed the embedded system module. The sensors used are **a)** accelerometer and **b)** IR distance sensors. The second: the wireless communication and scheduling between all sensor nodes using **TiSCH**[3] formed the wireless part. The third: managing, storing sensor and user data remotely is done using implementing a database system and a web application for the user interface as a part of the system prototype.

The main functionality of the system is that an alarm is raised whenever, there is some disturbance (over a pre-defined threshold level) at any of the nodes and user is alerted either via web interface or mobile app.

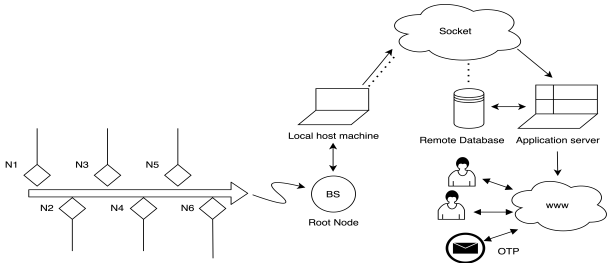

Fig. 1: SBR Block diagram

2.1. On Node Functionality (Embedded System)

All the sensory nodes used are Zolertia Z1 boards. The nodes sends accelerometer and distance data periodically to the root-node and then forwards and pushes it to the remote database. All the sensory nodes are connected to the root-node, which is our base-station. Data from sensor nodes are transported to root node using UDP at transport layer which is implemented on top of RPL in contiki.

2.2. RPL+TiSCH (Wireless Communication)

All the nodes communicate with root-node using TiSCH, which works at MAC layer of the network, for synchronization and scheduling in a time-slotted manner. Due to its low-power consumption, TSCH was chosen for this design implementation using contiki. TSCH communicates with upper layer protocol esp RPL using set of callbacks that notify RPL that TSCH joined or left the network. On the otherside, RPL notifies TSCH of any changes either in parent node or changes in beacon intervals.

2.3. Database and Web client application

Data packets from the base station connected to the local host computer, are pushed into the remote database using a Python script. The main purpose of the script is to parse the base station packets into **JSON** format.

Database and Web application is implemented using MERN[4] stack, which is a combination of four technolo-

gies: [M]ongo DB, [E]xpress JS, [R]eact JS, and [N]ode JS. Setting up of back-end server needed 'Node Package Manager (NPM)', including dependencies such as [E]xpress, for server framework. Back-end database structure consists of various fields for each node such as, Node ID.

3. IMPLEMENTATION

All sensory nodes are running with the node.c code and communicates with the root node which is only receiving data and is meant to relay it to the server/database. The root is implemented as a contiki border-router[5] but is in the final iteration in the scope of this project not completely functional.

The on-board accelerometer sensor was used to collect data and classified as vibrations at the nodes. A Sharp Distance Sensor[6] was used to collect distance data at the nodes. Both of these sensors are continuously read in their own processes and updates a buffer with the sensor data. They also change sensor variables that saves the state of the sensor. The states of the sensors are changed depending on pre-defined ranges and their frequencies to signal a potential threat. A third process is handling the on node functionality that sends the data to the root. It also handles the combined states of the sensors and should alarm to the root if they are in certain states. The process sends the buffered data from the sensors periodically to the root node.

Database/web-client: The information/data-packet from the border-router, gets stored into the database in json format.

```
{
  "SBR_Database": [
    {
      "Node_ID"    : "03",
      "accl_state": "SF",
      "dist_state": "SF",
      "node_state": "UP",
      "alrm_state": "FL",
      "lock_state": "LK",
      "time_stamp": "SS"
    }
  ]
}
```

Listing 1: SBR Data-field example

Data fields are pushed and retrieved into/from the server side of database using 'POST' and 'GET' command at the local host machine connected with border-router. The server and web-client are hosted locally at a remote machine, having access to the outside world using port forwarding mechanism. In this way, the need of hosting database server and web application using paid cloud services was avoided.

4. RSSI MEASUREMENTS FOR PHYSICAL SYSTEM

4.1. Outside Experiments

The system's wireless communication was tested outside on a real bike rack in order to see how the system would perform in a context closer in proximity to the intended end scenario. The bike rack used was a standard ground-mounted bicycle rack with fixed frames [7] with total length 2m and 40Cm between bike slots, and one rack having 5 slots. The factor that will be observed is the RSSI [8] measurement from three nodes sending to one root node (base station) with different distances and topology.

The nodes in these experiments nodes are named '66', '0' and '9e' after the end part of their addresses, and the base station is 'Root'. The 'Empty' in the graphs indicates no node/-root and is used to comprehend distance.

5. RESULT

6. CONCLUSION AND FUTURE WORK

In all of the experiments the RSSI is acceptable. The difference between the results of experiments is that when the base station is moved further away the RSSI increases overall for all of the nodes. Another observation from the experiments is the results between experiment 1 and 2. When the base root is moved more to the center of the topology the RSSI seem to decrease and is varying less. However when the root is then moved further away in experiments 2 and 3 it is also not varying that much compared to experiment 1.

The result of the RSSI measurement is indicating one important aspect of a system depending on wireless communication. The position of a base station can have an impact on the signal and to achieve a robust system and an effective system its placement should be taken into account. Distance can impact the RSSI even though it is in the Z1 communication range of 10m.

The results is a first step in doing a feasibility test of the system and the result is acceptable RSSI in the tested context. Further tests and experiments that could be done in the future is looking at LQI, environmental aspects (shadowing, refraction), and looking at response time from sensor to that the user is alerted.
The final system is an example of a novel IoT solution for a specific problem. Although the system is not a complete product, the final iteration of the system is what was accomplished during the scope of this project. This is probably one of the most efficient way of solving this problem and could perhaps been done in different ways. For example using MQTT [9] for the communication and put the application logic on server level or put it on the users mobile device through MQTTs lightweight subscription of data, maynot be reliable. A further improvement of the existing system is a more advanced accelerometer analyzing at node to take into account for example the winds effect and other false positives of real malicious activity in the system. This project aims to show how a system could be implemented and encourage the creation of more reliable IoT systems for a safer and smarter world.

References

[1] *Contiki-ng documention*. [Online]. Available: https://github.com/contiki-ng/contiki-ng.

[2] *Meet the z1 mote*. [Online]. Available: http://wiki.zolertia.com/wiki/index.php/Main_Page.

[3] *Tsch and 6tisch for contiki*. [Online]. Available: http://www.simonduquennoy.net/papers/duquennoy17tsch.pdf.

[4] *Database and web client application mern stack*. [Online]. Available: https://medium.com/swlh/how-to-create-your-first-mern-mongodb-express-js-react-js-and-node-js-stack-7e8b20463e66.

[5] *Tutorial: Rpl border router*. [Online]. Available: https://github.com/contiki-ng/contiki-ng/wiki/Tutorial:-RPL-border-router.

[6] *Sharp distance sensor (4-30cm)*. [Online]. Available: https://www.phidgets.com/?tier=3&catid=5&pcid=3&prodid=394.

[7] *Standard cykelst[pleaseinsert"prerenderunicode–˜intopreamble]ll*. [Online]. Available: https://nola.se/products/standard-cykelstall/.

[8] *Received signal strength indication*. [Online]. Available: https://en.wikipedia.org/wiki/Received_signal_strength_indication.

[9] *Mqtt - the standard for iot messaging*. [Online]. Available: https://mqtt.org/.