Rapproto: An Open Source Platform for Rapid Prototyping of Wearable Medical Devices

Amanda Watson, Hyonyoung Choi, Insup Lee, James Weimer
University of Pennsylvania
Introduction

• It is common to expend a significant amount of time and effort to develop data collection systems

• These systems tend to be customized and highly specific to the task at hand and are not general enough to support other tasks
Challenges

We want to make a rapid prototyping platform that is:

• Open-source
• Easy-to-use by non-coders
• Customizable sensors
• Customizable data collection rates
• Provides visualization of data
• Works with off-the-shelf smartwatches
Outline

1. Introduction
2. Raproto Platform
3. Evaluation
4. Extensibility
5. Conclusion
Rapproto Platform
Smartwatch Application: UI

Main Screen

Raproteo
Watch Name

Device ID
XZF3E3TDSL

ON / OFF

Configuration

Settings

Device ID
2021-02-24 21:53:28
MQTT

ON / OFF

Data Collection

Raproteo
Watch Name

Device ID
R3JkbtzBQb

Sync Now
www.precisea.com
Smartwatch Application: Watch Configuration

• Settings for each watch are configured on the remote server
  • Sensor Selection
  • Sampling Rates
  • Wi-Fi Usage
  • Transmission Rates

• To receive these settings, a configuration button must be pressed in the smartwatch
Smartwatch Application: Data Collection

• Supported Sensors:
  • Accelerometer
  • Gyroscope
  • Gravity sensor
  • Heart rate monitor
  • PPG
  • Battery level

• Sensor data is stored in JSON format
Smartwatch Application: Data Storage

- Sensor data is not always immediately transmitted to the remote server
- 40 megabytes of buffer storage
- Data is packaged in 10 KB messages
- Once a wireless connection is established, data is sent out
Smartwatch Application: Battery Management

The largest drains on the battery life:

• Display
  • One-time configuration
  • Less than 5 minutes

• Wi-Fi Radio Settings
  • Configure time spent between data transmission events

• Volume of Sensor Data
  • Select sensors
  • Configure data sampling rates
Communication Protocol

• MQTT is a publish/subscribe messaging protocol
  • Telemetry: Smartwatches to the Server
  • Commands: Server to the Smartwatches

• MQTT supports three levels of quality of service
  • Level 0: telemetry is sent without acknowledgment that the server has received it.
  • Level 1: guarantees that the server receives the telemetry by sending an acknowledgment back. If the acknowledgment is lost, then telemetry is resent until it receives an acknowledgment.
  • Level 2: guarantees the telemetry will be received exactly one time by completing a "handshake" to confirm that the telemetry has been sent and that the acknowledgment has been received.
Remote Server: Device Administration

Device Name
Device details

DETAILS ATTRIBUTES LATEST TELEMETRY

MANAGE CREDENTIALS DELETE DEVICE

COPY DEVICE ID COPY ACCESS TOKEN

Name
Samsung Watch - Test

Device type
SamsungGalaxyWatchActive
Remote Server: Data Storage

• TimescaleDB
  • Open-Source
  • Optimized for time-series data
  • Fast storage of new entries
  • Quick processing of complex analysis
• SQL queries are used to access the data
• Not directly accessible for end-users
Remote Server: Data Visualization

Rappto Dashboard

Battery Life
Realtime - last 30 minutes

Watch #1

Watch #2

avg

87.66666666666667
99.61538461538461
Remote Server: Data Processing

1. **Input**
2. **script IsSamsungWatch?**
   - **True**
     - **script BatteryLevel(<20%)?**
       - **True**
         - create alarm LowBatteryAlarm
       - **False**
         - clear alarm LowBatteryAlarm
   - **False**
     - create alarm LowBatteryAlarm
Evaluation – Battery Life

Expected Battery Life with Accelerometers at Various Sampling Rates

<table>
<thead>
<tr>
<th>Sampling Rate (Milliseconds)</th>
<th>Expected Battery Life (Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>28.58</td>
</tr>
<tr>
<td>10</td>
<td>18.18</td>
</tr>
<tr>
<td>5</td>
<td>12.5</td>
</tr>
<tr>
<td>1</td>
<td>7.14</td>
</tr>
</tbody>
</table>
# Evaluation – Battery Life

<table>
<thead>
<tr>
<th>Sensor Combinations</th>
<th>Expected Battery Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>33.3 hrs</td>
</tr>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>28.6 hrs</td>
</tr>
<tr>
<td>x</td>
<td>25.0 hrs</td>
</tr>
<tr>
<td>x</td>
<td>22.2 hrs</td>
</tr>
<tr>
<td>x</td>
<td>22.2 hrs</td>
</tr>
</tbody>
</table>
Evaluation – Data Loss

Data Loss and Duplication

<table>
<thead>
<tr>
<th>MQTT Service Level</th>
<th>Total</th>
<th>Lost</th>
<th>Duplicated</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8,965</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3,879</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>47,550</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Evaluation – Data Latency

• Time in Storage
  • Configurable
  • Changes based on Wi-Fi Availability

• Transit Time
  • Dependent on MQTT Level of Service
  • Less than 1 Second with MQTT LoS-1
Extensibility

Android OS Support

Application Store Availability

Cellular Enabled Smartwatches
Conclusion

Rapproto is an open-source, easy-to-use rapid prototyping platform that facilitates data collection from sensors on commercially available off-the-shelf smartwatches.

This platform provides researchers, especially in remote health monitoring and ubiquitous computing, a quick, simple to use, and customizable solution for developing data collection systems.

We evaluated our platform and observed that a smartwatch with the Rapproto application running lasted for over 24 hours on a single charge, has almost no data loss, and experienced less that one second of data latency.
THANK YOU!

http://precise.seas.upenn.edu

https://github.com/weimerj/Rapproto-Tizen