IoT-MAP: پلت فرم اپلیکیشن مشاپ برای اکوسیستم IoT منعطف

چکیده

محیط موبایلی اخیرا انواع گوناگونی از اشیای هوشمند را در خود دارد که به صورت فرصت طلبانه‌ای به محیط پیرامون تلفن‌های همراه نزدیک می‌شوند. بیشتر اپلیکیشن‌های موجود در بازار تنها به مدل خاصی که تولید کننده در فاز توسعه انتخاب کرده است محدود می‌شوند، لذا آن‌ها با اشیای ناهمگن حتی اگر عملکردهای مشابهی داشته باشند، سازگار نیستند. برای بهینه سازی اشیای هوشمندی که در پیرامون هر کاربر هستند، اپلیکیشن مشاپ[1] IoT نیاز است تا به صورت پویا اشیای ناهمگون را کشف کند، ماژول نرم افزاری مورد نیاز را در زمان اجرا دانلود کند، و اپلیکیشن مشاپ را به کاربر ارائه دهد. در این مقاله، پلت فرم اپلیکیشن مشاپ[2] IoT را که از کشف تلفن‌های هوشمند، شناسایی، نصب و راه اندازی، مشاپ و ترکیب اشیای هوشمند پشتیبانی می‌کند را طراحی و اجرا کردیم. علاوه بر این، با جدا کردن نقش اکتورها در اکوسیستم IoT که از اینترفیس اشیای انتزاعی کاربردی استفاده می‌کنند، پلت فرم مزایایی را برای هر اکتور فراهم می‌کند مانند اینکه- تولیدکنندگان اشیا می‌توانند قابلیت محصولات خود را به حداکثر برسانند، توسعه دهندگان اپلیکیشن می‌توانند بر منطق تجاری خود متمرکز باشند، و کاربران نهایی می‌توانند انتخاب کنند که کدام اشیای هوشمند برای ترکیب استفاده شوند.

کلمات کلیدی: اینترنت اشیا؛ترکیب سرویس؛ مشاپ ؛ قابلیت سازگاری و همکاری[3]؛ EPCglobal

[1] mashup application
[2] Mashup Application Platform
[3] Interoperability

1. مقدمه

امروزه، انواع بیشماری از دستگاه‌های هوشمند به اکوسیستم IoT وارد شده اند، و برخی از آن‌ها حتی به بخش‌های جدایی ناپذیر زندگی روزانه ما تبدیل شده‌اند. تحت این شرایط، اگر تعامل منعطف بین این اشیای هوشمند و دستگاه موبایل شخصی امکان پذیر شود، این اشیای هوشمند می‌توانند تجربه بهتر و خدمات جدیدی را در مقایسه با اپلیکیشن‌های موجودی که تنها با دستگاه‌های خاص در تعامل هستند؛ ارائه دهند. برای مثال، می‌توانیم سرویس ساده‌ای را تصور کنیم که دمای بدن انسان و ضربان قلب انسان را با حسگرهای متصل شده به بدن اندازه می‌گیرد، و با کنترل روشنایی و رنگ لامپ هوشمند، دمای بدن را اعلام می‌کند. در محیط توسعه معمولی موبایل، اپلیکیشن‌هایی که این خدمات را ارائه می‌دهند معمولا به مدل خاصی از اشیا محدود می‌شوند که توسط توسعه دهنده در زمان اجرا انتخاب شده‌اند.

در این مقاله، IoT-MAP، یک پلت فرم اپلیکیشن موبایل که به قابلیت همگام سازی منعطف بین دستگاه‌های موبایل و اشیای هوشمند پیرامون آن کمک می‌کند را توضیح می‌دهیم –این اپلیکیشن توسعه یافته بر اساس کتابخانه پلت فرم را، IoT-App نامگذاری می‌کنیم.

شکل ۱: دیاگرام مفهومی سطح بالا

پلت فرم IoT-MAP بهبودهای گوناگونی برای اکتورها در اکوسیستم IoT حاضر فراهم کرده است که در شکل 1 نشان داده شده است. برای توسعه دهندگان اپلیکیشن موبایل، IoT-MAP مجموعه‌ای از APIهایی با درک مستقیم را برای ساخت آسان IoT-App ارائه داده است، مانند کشف شی، ارتباط و بازیابی شی خدماتی انتزاعی که در صورتی که شی حقیقی باشد؛ می‌تواند مستقیما استفاده شود. با موارد پشتیبانی شده در پلت فرم IoT-MAP، آن‌ها می‌توانند منطق تجاری خود را در سبک POJO (شی ساده جاوا[4]) بدون در نظر گرفتن ارتباط حقیقی و اجرای اشیای هوشمند بنویسند. تولیدکنندگان اشیا می‌توانند با داشتن یک سرور نام[5] مبتنی بر ONS و مجموعه‌ای از درایورها، مانع از بوجود آمدن تعارضات مربوط به ID پلت فرم شوند. به دلیل اینکه ONS بر اساس تکنولوژی DNS طراحی شده است، سرور نام تولید کننده می‌تواند در دیگر سرورهای ONS توزیع شده ادغام شود، بنابراین کاربران نهایی می‌توانند مجموعه نرم افزار درایور[6] اشیای خود را که مستقیما توسط تولید کننده ارائه شده است، بدون هیچ دانشی از اشیا (مانند نام تولید کننده، نام شی، شماره سریال و غیره) پیدا کنند. و سرانجام کاربران نهایی، می‌توانند انتخاب کنند که کدام شی با استفاده از اشیای کشف شده در زمان اجرا، می‌تواند در ارائه خدمات مطلوب مشارکت داشته باشد. علاوه بر این، در مواردی که هیچ IoT-App با پیش نیاز کاربر در بازار منطبق نیست، کاربران نهایی می‌توانند از ابزار تالیفی[7] GUI ارائه شده توسط پلت فرم IoT-MAP استفاده کنند. کاربران می‌توانند با استفاده از اشیای کشف شده و ماژول‌های منطقی؛ اپلیکیشنی را بوجود آورند که هر شی را سازگار و قابل تعامل می‌سازد، سپس این ماژول‌ها به صورت پویا دانلود می‌شوند و اپلیکیشن مورد نظر کاربر را تشکیل می‌دهند.

سهم این مقاله به صورت زیر خلاصه شده است. ابتدا، IoT-MAP را، یک پلت فرم اپلیکیشن IoT جدید را که تعامل انعطاف پذیر با اشیای محیط را فراهم می‌کند، پیشنهاد می‌دهیم. دوم، یک رویه برای حل تعارض ID اشیا مبتنی بر استاندارد بین المللی با استفاده از اجرای سرور نام Oliot-ONS- (سرویس نام شی[8]) که مولفه‌ای از پلت فرم زیر ساختی Oliot IoT (http://www.oliot.org) است طراحی می‌کنیم. سوم، دو روش از متدهای توسعه را ارائه

[4] plain old java object
[5] Name Server
[6] Dirver Bundle
[7] authoring tool
[8] Object Name Service

می‌دهیم- کتابخانه IoT-App برای توسعه دهندگان اپلیکیشن برای ایجاد IoT-App، و ابزار نویسندگی GUI برای کاربران نهایی برای اسمبل کردن اپلیکیشن متناسب با هدف خود در زمان اجرا. در آخر، نتایج آزمایشی ارزیابی کلی IoT-MAP را گزارش می‌دهیم و دو سناریو آزمایشی استفاده کننده از پلت فرم برای اعتبارسنجی را ارائه می‌دهیم. باقی این مقاله به این شرح سازمان یافته است. در بخش 2، در مورد مفهوم سطح بالا و ملاحظات طراحی پلت فرم IoT-MAP توضیحاتی ارائه داده‌ایم. در بخش 3 معماری IoT-MAP را برای پشتیبانی از توسعه و عملیات IoT-Apps معرفی کردیم، و بخش 4 پیش نیازهای مهمی را برای توسعه پویا و ترکیب خدمات اشیای هوشمند و اینکه چگونه توسط پلت فرم ما مدیریت می‌شود توضیح دادیم. سپس اجرای آن را در بخش 5 ارائه دادیم، ارزیابی و سناریوهای نسخه نمایشی اپلیکیشن را برای اعتبارسنجی در بخش 6 ارائه دادیم، کار مربوطه در بخش 7 آورده شده است و سرانجام نتیجه گیری مقاله در بخش 8 اورده شده است.

2. ملاحظات طراحی

امروزه دستگاه‌های IoT زیادی به بازارها وارد شده اند، اما قابلیت آن‌ها به دلیل اینکه با یک اپلیکیشن موبایل تجهیز شده اند که اختصاصی تنها برای آن دستگاه طراحی شده است، محدودیت‌هایی دارد. برای مثال، لامپ‌های هوشمند مانند Philips Hue (http://meethue.com) اپلیکیشن موبایلی را برای برای روشن /خاموش کردن یا تغییر رنگ‌ها ارائه داده است. کاربران نهایی نمی‌توانند استفاده از لامپ‌های هوشمند را منطبق با اپلیکیشن، حسگرها یا تعاملات خود سفارشی سازی کنند.

مسئله این است که نقش تولید کننده دستگاه با توسعه دهنده نرم افزار تفاوتی ندارد. توسعه دهنده نرم افزار بر چگونگی استفاده از قابلیت‌های دستگاه‌های ناهمگن با IoT-App API، صرف نظر از پروتکل‌های ارتباطی اصلی تمرکز می‌کند، در عین حال تولیدکنندگان دستگاه بر ارائه درست قابلیت‌های دستگاه با اجرای پروتکل‌های ارتباطی اصلی تمرکز می‌کنند. علاوه بر این، با توجه به انواع الویت‌های کاربران نهایی، کاربران نهایی ممکن است بخواهند که اپلیکیشن‌های

چند منظوره‌ای را در زمان اجرا با استفاده از واسط کاربر گرافیکی[9] (GUI) ایجاد کنند یا ترکیب کنند. این به معنی این است که نقش توسعه دهندگان اپلیکیشن ممکن است به کاربران نهایی بسط یابد. بنابراین، در جایی که توسعه دهندگان اپلیکیشن و توسعه دهندگان دستگاه بر نقش خود برای برآورده کردن انواع پیش نیازهای کاربر متمرکز هستند، پلت فرمی لازم است.

مفهوم معماری مدل محور[10] (MDA) [17] می‌تواند یک معماری لایه لایه را ارائه دهد به طوری که تقسیم نقش بین کاربران نهایی، توسعه دهندگان اپلیکیشن، و تولید کنندگان دستگاه حاصل می‌شود. ایده MDA استخراج مدل مستقل پلت فرم و خاص دامنه (مدل مستقل پلت فرم[11]، PIM) از عناصر خاص پلت فرم مانند زبان برنامه نویسی، سیستم عامل، یا پروتکل‌های ارتباطی (مدل خاص پلت فرم[12]، PSM) برای کسب یک توسعه کارآمد یا قابلیت سازگاری بین سیستم‌های متفاوت است.

کد1: مثالی از واسط انتزاعی و تشریح عملکردی از SensorTag

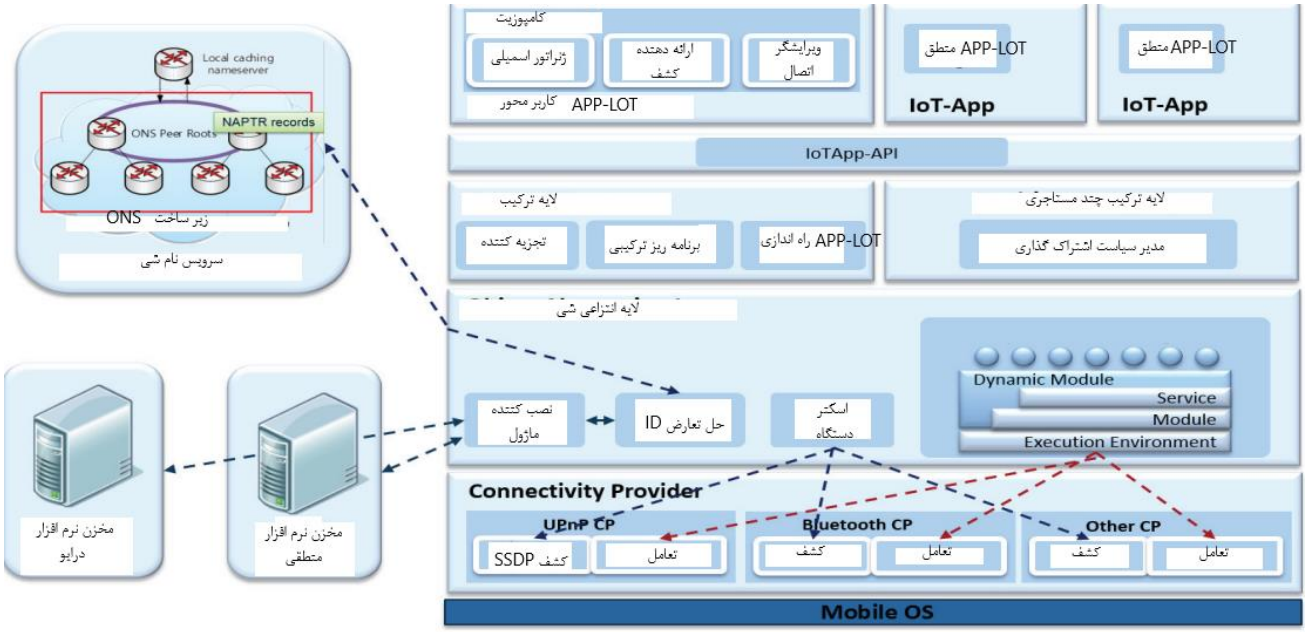
```
public interface ThingService {
    String getThingId();
    String getThingName();
    void setEndpoint(Endpoint endpoint);
    void connect();
    void disconnect();
    boolean isConnected();
}
public interface ButtonService
        extends ThingService {
    int getButtonCount();
    boolean getButtonState(int index);
}
public interface AccelService
        extends ThingService {
    String getAccelRateMeasurement();
}
public interface SensorTagService
        extends ButtonService, AccelService,
                GyroscopeService, ... {
}
```

[9] graphical user interface
[10] model driven architecture
[11] Platform Independent Model
[12] Platform Specific Model

در لایه PIM پلت فرم، توسعه دهندگان اپلیکیشن از طریق اینترفیس به خوبی تعریف شده صرف نظر از پروتکل‌های ارتباطی اصلی به دستگاه‌ها دسترسی پیدا می‌کنند. پلت فرم واسط انتزاعی کاربری تعمیم یافته‌ای را ارائه می‌دهد که به یک مورد خاص فروشنده محدود نمی‌شود، و می‌تواند برای تشریح عملکردهای شی خاص، مانند کد 1، ترکیب شود. ThingService توابع پایه اشیای فیزیکی را تشریح می‌کند مانند بازیابی شناسانه، تنظیمات نقطه نهایی و اتصال/قطعی. ButtonService و AccelService توابع اتمیکی را برای بررسی داده ارسال شده از یک دکمه ساده و داده شتاب دهنده را تعریف می‌کنند. با این واسط‌ها SensorTagService می‌تواند برای تشریح عملکرد یک مدل خاص اشیا؛ تگ حسگر TI نوشته شود. با این انتزاع از عملکردهای دستگاه، پتانسیل IoT-App می‌تواند به صورت چشمگیری افزایش یابد چرا که تولید کنندگان دستگاه، توسعه دهندگان اپلیکیشن، و کاربران نهایی همه با یکدیگر سازگار هستند.

با توجه به حضور لایه PIM، برای کاربران نهایی ایجاد یک اپلیکیشن یا اپلیکیشن‌های چند منظوره در زمان اجرا امکان پذیر خواهد بود. در لایه بالای لایه PIM، یک ابزار GUI می‌تواند توسعه داده شود که لیستی از دستگاه‌های کشف شده و مولفه‌های نرم افزاری موجود از منطق‌های اپلیکیشن را ارائه می‌دهد به طوری که کاربران نهایی آن‌ها را برای ایجاد اپلیکیشن‌ها به صورت پویا اسمبل می‌کنند. این روند به دلیل لایه PIM هم اکنون انتزاعی از پروتکل‌های ارتباطی اصلی و پروتکل‌های ارتباطی دستگاه است.

در لایه PSM در پلت فرم، عملکردهای دستگاه می‌تواند توسط تولیدکنندگان دستگاه ارائه دهنده واسط به خوبی تعریف شده و پروتکل ارتباطی، فراهم شود. توجه داشته باشید که همان قابلیت دستگاه می‌تواند با پروتکل‌های ارتباطی متفاوت اجرا شود. برای مثال، واسط لامپ هوشمند، مانند خاموش/روشن کردن، و رنگ در حال تغییر می‌تواند یا توسط Bluetooth یا UPnP بر WiFi پیاده سازی شود. این انتخاب در کل به تصمیم تولید کننده دستگاه مربوط است، اما اپلیکیشن‌ها می‌توانند هنوز به قابلیت‌های دستگاه با همان واسط، دسترسی داشته باشند.



شکل ۲: معماری پلت فرم IoT-MAP

برای بارگذاری زمان اجرای خاص پلت فرم، مخزن خارجی نیاز است. نه تنها باید تابع بازیابی کدهای اجرا برطبق شناسه‌های دستگاه ارائه شود بلکه باید به صورتی که جستجوی مقیاس پذیر امکان پذیر باشد، توزیع شود. ما زیرساخت سرویس نام شی[۱۳] (ONS) را از معماری GS1 EPCglobal مورد استفاده قرار دادیم. این از سیستم شناسایی به نام کد محصول الکترونیک[۱۴] (EPC) استفاده می‌کنیم، و ساختار سلسله مراتبی سرویس نام دامنه[۱۵] (DNS) را تحت شعاع قرار می‌دهیم، لذا بازیابی کارآمد و توزیع سراسری مقیاس پذیر مدل خاص پلت فرم امکان پذیر است.

3. پلت فرم اپلیکیشن مشاپ IoT

ما معماری IoT-MAP را طراحی کردیم که بر سیستم عامل موبایل کار می‌کند و می‌تواند برای توسعه دهندگان و کاربران استفاده شود. توسعه دهندگان می‌توانند اپلیکیشنی را ایجاد کنند که با اشیای هوشمند گوناگون با استفاده از IoT-App API در تعامل هستند، و اگر شما قابلیت‌های انتزاعی IoT-MAP اشیای هوشمند ناهمگن گوناگونی را در حول تلفن هوشمند کاربر در اختیار داشته باشید، می‌توانید از توابع اشیا به روش یکنواختی استفاده کنید. برای

[13] object name service
[14] electronic product code
[15] domain name service

کاربران، آن‌ها می‌توانند به سادگی از IoT-App اجرا شده توسط توسعه دهندگان با انتخاب اشیای خاص کشف شده توسط پلت فرم در زمان اجرا استفاده کنند یا می‌توانند IoT-App را در زمان اجرا با استفاده از UI ترکیبی (ابزار GUI برای کاربر) تالیف کنند. IoT-App می‌تواند با اشیای گوناگون بدون پیوند استاتیک با فروشنده خاص در زمان اجرا در ارتباط باشد، که به عنوان اپلیکیشن معمولی تنها می‌تواند با اشیای محدودی در زمان اجرا در ارتباط باشد.

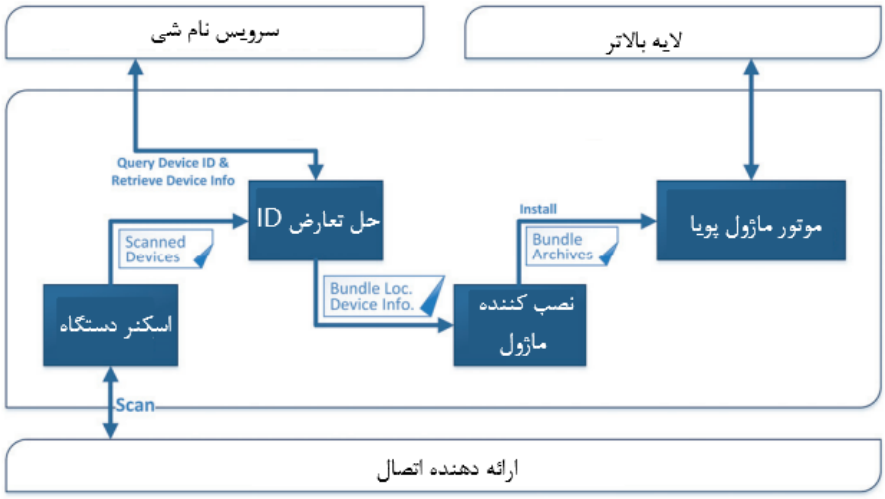
A. معماری سیستم پلت فرم IoT-MAP

پلت فرم انتزاعی از قابلیت‌های پایه OS دستگاه است (مانند ارتباطات، شبکه، و غیره) لذا IoT-Apps می‌تواند از آن‌ها استفاده کند. شکل 2 دیاگرام کاملی از پشته پلت فرم است. بخش زیرین پلت فرم، " ارائه دهنده اتصال[۱۶]" پروتکل‌ها و ارتباطات گوناگونی را ارائه می‌دهد که برای کشف و برقراری ارتباط با اشیای هوشمند پیرامون استفاده می‌شود. و "لایه انتزاعی شکل[۱۷]" مسئول کشف؛ شناسایی، مدیریت شی مجازی، و مدیریت مجموعه نرم افزار، و ترکیب حقیقی از خدمات اشیای هوشمند و منطق تجاری است. "لایه ترکیبی[۱۸]" از عملکردهایی برای اپلیکیشن‌های چندمنظوره[۱۹] پشتیبانی می‌کند. این لایه ترکیباتی از سرویس با اطلاعات تجزیه شده‌ای که توسط کاربر در زمان اجرا با ابزار تالیف تعریف شده است و مجموعه نرم افزار مرجع از لایه انتزاعی برای بازیابی شی، را در خود دارد. و در آخر، یک IoT-APP API وجود دارد، که واسطهای استانداردهای را برای گروه اشیای هوشمند (لامپ ها، حسگرها، دوربین‌ها و غیره) ارائه می‌دهد و همچنین API را برای توابع گوناگونی مانند کشف اشیا، بازیابی اشیای مجازی و غیره ارائه می‌دهد.

1) ارائه دهنده اتصال: ارائه دهنده اتصال پروتکل‌ها و ارتباطات گوناگونی را برای لایه بالاتر پلت فرم، مانند بلوتوث یا بلوتوث کم مصرف[۲۰](BLE) که توسط اندروید پشتیبانی شده است، یا دیگر پروتکل‌ها مانند UPnP که توسط

[16] Connectivity Provider
[17] Object Abstraction Layer
[18] Composition Layer
[19] Versatile App
[20] Bluetooth Low Energy

کتابخانه خارجی پشتیبانی شده است را ارائه می‌دهد. توسعه دهنده می‌تواند اشیای خارجی را با استفاده از API برای کشف اشیا پوشش دهد. Bluetooth CP (ارائه دهنده ارتباط)، UPnP CP، و دیگر CPها، واسط "ارائه دهنده ارتباط" را اجرا می‌کنند، لذا لایه بالاتر می‌تواند از قابلیت‌های پایه مانند کشف و بازیابی شناسه یا مشخصات اشیایی که از هر CP به روش عمومی پشتیبانی می‌کند، استفاده کنند.



شکل ۳: ساختار لایه انتزاعی

2) لایه انتزاعی شی: لایه انتزاعی شی در شکل 3 یک لایه اصلی پلت فرم IoT-MAP است، که دستگاه‌های واقعی را در گروهی از دستگاه‌های انتزاعی وارد می‌کند و ترکیبی از خدمات و منطق تجاری IoT-App را ممکن می‌شود. این لایه شامل اسکنر دستگاه[۲۱] است که اشیای هوشمند پیرامون را کشف می‌کند، حل تعارض ID[22] که اطلاعات داستگاه را شناسایی می‌کند، نصب کننده ماژول[۲۳] که ماژول نرم افزاری دستگاه را دانلود و ثبت می‌کند، و موتور ماژول پویا[۲۴] که به صورت پویایی دستگاه‌ها را به عنوان شی سرویس دهنده انتزاعی با کمک فریم ورک OSGi مدیریت می‌کند (نصب، شروع، توقف، به روزرسانی، لغو نصب). دستگاه‌های کشف شده و موجود از این مولفه‌ها گذر می‌کنند و در موتور ماژول پویا ذخیره می‌شوند تا به عنوان سرویس صادر شده توسط لایه بالاتر استفاده شود.

[21] Device Scanner
[22] ID Resolution
[23] Module Installer
[24] Dynamic Module Engine

• اسکنر دستگاه

اسکنر دستگاه، دستگاه‌های موجود در اطراف تلفن هوشمند را با استفاده از متدهای کشف ارائه دهنده ارتباط (SSDP برای UPnP) اسکن می‌کند. برای پشتیبانی از قابلیت‌های اسکن بر پروتکل‌های و اتصالات ناهمگن، هریک از آن‌ها در الگوی استراتژی اجرا می‌شوند که چند الگوریتم را تعریف می‌کنند، که هر یک از آن‌ها پوشش داده می‌شوند، و آن‌ها با رابط یکنواخت DeviceScanner قابل تعویض هستند. هر دستگاه اسکن شده با شی ScannedDevice تشریح می‌شود که شامل اتصال، پروتکل، شناسه دستگاه، نام، و نقطه نهایی است.

• حل تعارض ID

اشیای هوشمند معمولا ID قابل شناسایی دارند مانند URN، آدرس MAC، UUID یا GS1 ID. مولفه حل تعارض ID سه شناسه را از شی ScannedDevice می‌خواند و آن را از سرور نام دامنه برای قراردادن اطلاعات دستگاه و URL نرم افزار داریور پرس و جو می‌کند. رویه این مورد در بخش 4 آورده شده است.

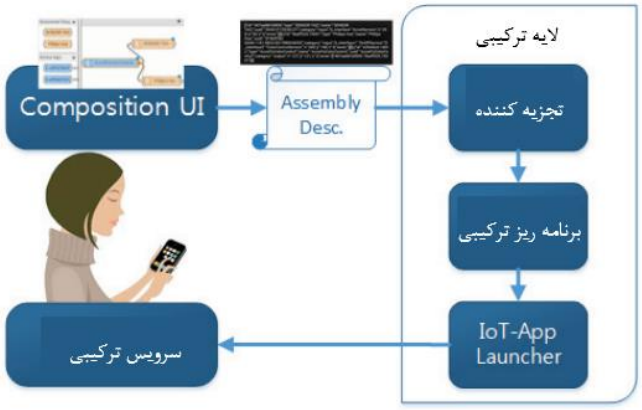
• نصب کننده ماژول

نصب کننده ماژول، محل درایور نرم افزاری را از نتیجه "حل تعارض ID" می‌خواند، و به صورت پویا آن را از مخزن نرم افزاری دانلود می‌کند. سپس نرم افزار در "موتور ماژول پویا" نصب و راه اندازی می‌شود.

• موتور ماژول پویا

موتور ماژول پویا، یک موتور اصلی پلت فرم IoT-MAP است که مسئول بارگذاری پویایی درایورهای دستگاه‌های گوناگون و منطق‌های خدمات است، و این ماژول‌ها از بخش‌هایی از IoT-App ترکیب شده است. لذا این مورد از فریم ورک OSGi [11] برای مصرف اجرای قدرتمند آن مانند محیط اجرایی، مدیریت ماژول، مدیریت چرخه عمر، و مدیریت خدمات استفاده می‌کند. موتور می‌تواند نمونه‌ای از ماژول‌های نرم افزاری را به عنوان سرویس ارائه دهنده؛ و

IoT-App این نمونه‌ها را بارگذاری کند اگر یک اپلیکیشن واحد باشند. پیچیده ترین رویه‌های این عملیات می‌تواند به فریم ورک OSGi منتقل شود.



شکل ۴: ساختار لایه ترکیبی

B. لایه ترکیبی

لایه ترکیبی توسط اپلیکیشن‌های چدمنظوره استفاده شده است، که یک نوع خاص IoT-App است که می‌تواند کشف شود و دستگاه‌ها را با تجزیه اطلاعات رسیده از ابزار تالیف کاربر به هم متصل کند. IoT-Apps عمومی اشیای هوشمند را بر اساس منطق کسب و کار توسعه دهنده اپلیکیشن ادغام می‌کند، اما اپلیکیشن‌های چند منظوره دستگاه‌هایی را از بر مبنای اطلاعات تالیفی جمع آوری می‌کنند، و هر ماژول نرم افزاری را بر اساس آن اطلاعات می‌نویسند. ساختار این لایه در شکل 4 آورده شده است. ترکیب UI لیستی از نرم افزارهای سرویس و همچنین لیستی از نرم افزار منطقی اپلیکیشن موجود را در ابزار تالیف کاربر نشان می‌دهد. این نقش ارائه اطلاعات تالیفی را برای لایه ترکیب بازی می‌کند. مجموعه نرم افزاری سرویس[۲۵] یک ماژول نرم افزاری است که برای استفاده از توابع ارائه شده توسط اشیای هوشمند لازم است در حالی که مجموع نرم افزار منطق اپلیکیشن یک ماژول نرم افزاری است که برای ارائه یک اپلیکیشن ترکیبی از نرم افزار سرویس استفاده می‌کند. زمانی که کاربر ترکیبی را ایجاد کند، این ترکیب توسط این لایه، تجزیه می‌شود، تشکیل می‌شود و راه اندازی می‌شود.

[25] Service bundle

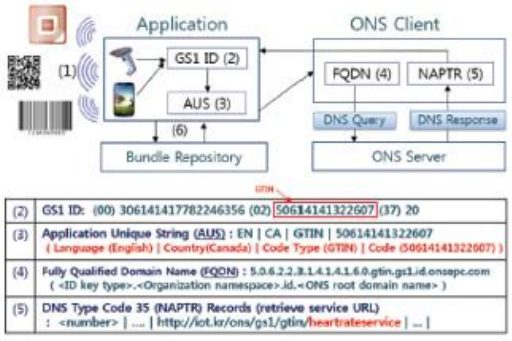
4. استقرار و شناسایی پویای اشیای هوشمند

برای یکپارچه سازی دستگاه‌های ناهمگن به روش یکنواخت، پلت فرم IoT-MAP مکانیزم استقرار و شناسایی واقعی را که از استاندارد EPCglobal [1] استفاده می‌کند را ارائه می‌دهد.

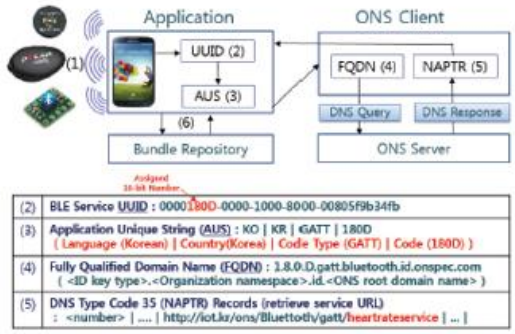
A. سرویس نام شی و EPCglobal

EPCglobal مجموعه از استانداردها برای مدیریت زنجیره تامین است، که توسط سازمان استانداردسازی بین المللی GSI منتشر و حفظ و نگهداری می‌شود. این روش‌هایی را برای ردیابی و شناسایی هر محصول در کل فاکتورهای زنجیره تامین از جمله تولید، توزیع و مصرف ارائه می‌دهد. هسته این استاندارد کد منحصر EPC (کد محصول الکترونیک[۲۶]) است که به هر محصول نسبت داده می‌شود و آن کد محصول واقعی و داده محصول مجازی را به یکدیگر ربط می‌دهد. در میان استانداردهای EPCglobal، پلت فرم IoT-MAP از سرویس نام شی [2] استفاده کرده است، که به ذی نفعان تجاری کمک می‌کند تا اطلاعات محصول را از سرور تولیدکننده بازیابی کنند. ONS بر اساس سرویس نام دامنه (DNS) اجرا شده است، لذا می‌تواند اطلاعات را در سرورهای توزیع شده گسترده که توسط تولیدکنندگان به صورت موثری نگهداری می‌شود، پیدا کند.

[26] Electronic Product Code



(a) رویه شناسایی دستگاه‌هایی که از مشخصه BLE GATT



(b) رویه شناسایی محصولات با EPC واحد

شکل 5: رویه شناسایی دستگاه با ONS

شکل 5a رویه شناسایی استاندارد ONS با EPC را تشریح می‌کند. ابتدا، کاربر اشیا را با اسکنر یا تلفن هوشمند (1،2) اسکن و کشف می‌کند، EPC را از آن می‌خواند (3)، سپس رشته منحصر اپلیکیشن[۲۷] (AUS) را ایجاد می‌کند که شامل اطلاعات محیط کاربر مانند کشور، کد زبان است؛ و آن را به کلاینت ONS ارسال می‌کند (4). کلاینت AUS ONS را به نام دامنه واجد الشرایط[۲۸] (FQDN) بر اساس استاندارد ONS تبدیل می‌کند، و آن را از سرور ONS محلی پرس و جو می‌کند، سپس سرور ONS سرور دیگری را پیدا می‌کند که شامل اطلاعات اشیا در همان راستای DNS باشد. سرور ONS مقیم رکورد اشاره گر نام[۲۹] (NAPTR) را تولید می‌کند که یکی از استانداردهای پاسخ DNS است. سپس آن را به کلاینت ONS کاربر ارسال می‌کند (6). و در نهایت، کلاینت ONS این اطلاعات را به اپلیکیشن کاربر ارسال می‌کند (7). به دلیل اینکه لیست اطلاعات توسط تولیدکننده حفظ و نگهداری می‌شود،

[27] Application Unique String
[28] Fully Qualified Domain Name
[29] Name Authority Pointer

این می‌تواند یک متد خوب برای ارائه محل درایور دستگاه باشد که در سرور تولیدکننده قرار گرفته است. با این محیط، پلت فرم IoT-MAP می‌تواند اشیای هوشمند را شناسایی کند، محل درایور را پیدا کند، برای استقرار اشیا در تلفن هوشند کاربر آن را دانلود و نصب و راه اندازی کند.

مزیت قدرتمند استفاده از EPCglobal این است که یک استاندارد بین المللی برای زنجیره تامین است. همانطور که بار کد GSI استانداردی برای تجارت بین المللی دارد، همه محصولات شامل اشیای هوشمند گوناگونی هستند که می‌توانند EOC واحدی در آینده داشته باشند. و اگر این چیزها EPC داشته باشند، سپس می‌توانیم به سادگی از آن استاندارد برای شناسایی و استقرار اشیای هوشمند ناهمگن به روش یکنواختی استفاده کنیم.

B. کد محصول الکترونیک و جایگزین‌ها

همانطور که در بخش قبلی گفته شد، استاندارد EPCglobal کد منحصری را به هر محصول اختصاص می‌دهد. اما آشکار است که بیشتر اشیای هوشمند EPC ندارند چرا که شناسه آن‌ها و مواردی که در حال حاضر در بازار در دسترس هستند شناسه‌ای مانند آدرس MAC، UUID یا دیگر شماره سریال اختصاصی هستند. لذا به برخی از گزینه‌های EPC یا دیگر تبدیل‌های معقول یا الگوی نگاشت برای استفاده از ONS نیاز داریم. در این پلت فرم، ما الگوی استقرار و توسعه یافته مشخصه صفات عمومی[۳۰] BLE (GATT) را توسعه دادیم. پروفایل GATT یک پروفایل جنریک از پیش تعیین شده از اپلیکیشن اصلی یا رفتاری است که می‌تواند برای ارتباطات ساختار یافته با دستگاه‌های BLE استفاده شود. هر پروفایل GATT کلاس عملکردی دستگاه را نشان می‌دهد و می‌تواند از UUID نسبت داده شده متمایز باشد؛ لذا می‌توانیم از این روند برای پرس و جو از ONS برای بازیابی اطلاعات دستگاه جز برای برخی از اطلاعات یکنواخت مانند شماره سریال نام محصول، استفاده کنیم. همانطور که در شکل 5b تشریح شد، کاربر می‌تواند شناسه سرویس پروفایل GATT را پرس و جو کند و درایور سازگار متناظر را برای استفاده از توابع دستگاه BLE بازیابی کند.

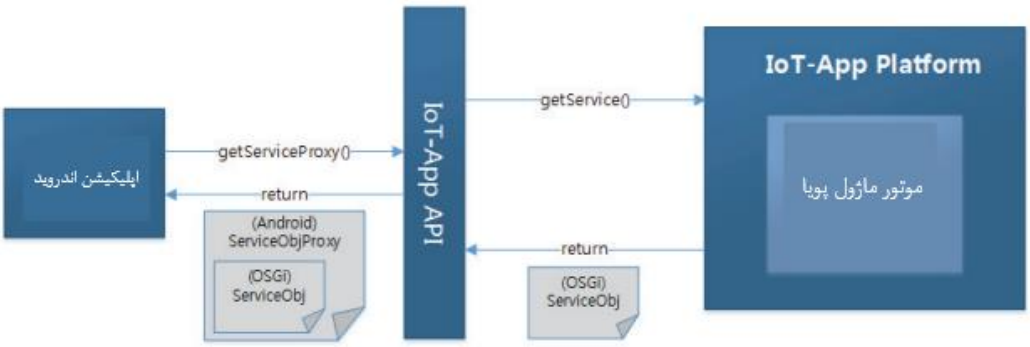
[30] Generic Attribute Profile

5. اجرا

ما نمونه‌ای از پلت فرم IoT-MAP را در اندروید اجرا کردیم. پلت فرم به عنوان یک سرویس در اندروید اجرا می‌شود. پلت فرم IoT-MAP برای اجرای هر سیستم عاملی که از HJVM یا موتور سازگار (مانند Dalvik) پشتیبانی می‌کند، طراحی شده است، و ما این پلت فرم را در اندروید برای اعتبارسنجی قابلیت‌های آن اجرا می‌کنیم.

A. پل زدن فریم ورک OSGi و Android Dalvik VM

اینترفیس‌ها یا رابط‌ها، انتزاعی از توابع یا قابلیت‌های گوناگون اشیای هوشمند هستند که به عنوان پلی بین فریم ورک OSGi و اپلیکیشن اندروید استفاده می‌شود. زمانی که توسعه دهنده از شی سرویس درخواست فریم ورک OSGi تعبیه شده را دارد، شی جاوا[31] را باز می‌گرداند. اما محیط اجرایی فریم ورک OSGi و Android Dalvik VM با classloader متفاوتی ایزوله شده است، لذا زمانی که توسعه دهنده سعی می‌کند که شی را به مورد دیگری تبدیل کند حتی اگر اینترفیس مشابهی برای تبدیل استفاده شود، ClassCastException فراخوانی می‌شود.



شکل ۶: پروکسی پویای شی سرویس OSGi که در محیط اندروید استفاده می‌شود.

اینکه دو محیط نمی‌توانند به درستی ادغام شوند، مسئله‌ای است که نه تنها در Android/OSGi بلکه در Tomcat/OSGi که دارای فریم ورک OSGi تعبیه شده در جاوا است رخ می‌دهد. معمولا این مسئله با رجیستر کردن واسط قبل از اجرای فریم ورک OSGi برطرف می‌شود، اما به دلیل جنبه پویای پلت فرم IoT-MAP ، همچین گزینه‌ای در آن وجود ندارد. دیگر گزینه استفاده از کتابخانه‌هایی است که می‌توانند Java bytecode را به صورت

[31] Java Object type

پویا اصلاح کنند و تغییر دهند، اما این کتابخانه‌ها نمی‌توانند در Android Dalvik VM استفاده شوند. لذا در موتور اجرای پویا پلت فرم IoT-MAP، ما از Reflection و Proxy برای غلبه بر این مسئله استفاده می‌کنیم. همانطور که در شکل 6 گفته شد، پلت فرم شی سرویس را با پروکسی پویای API جاوا با همان واسط ترکیب می‌کنیم. با این بهبود، پلت فرم می‌تواند به صورت موفقیت آمیزی فریم ورک OSGi و اپلیکیشن‌های اندروید را ادغام کند.

6. ارزیابی و نسخه نمایشی اپلیکیشن

A. سناریوی تماس با Bendi

این سناریو ادغام اشیای هوشمند متعدد و منطق تجاری توسعه دهنده اپلیکیشن را نشان می‌دهد. همانطور که در شکل 7c نشان داده شد، Philips Hue و دستگاه تعامل لمسی Bendi [14] در یک تلفن هوشمند برای ارائه خدمات تماس ارتباطی تعاملی ترکیبی ادغام شده‌اند. برای تشریح این سناریو، یک اپلیکیشن تماس VoIP ساده را به عنوان یک IoT-App توسعه دادیم. این اپلیکیشن کاربران را قادر می‌کند که با یکدیگر تماس برقرار کنند، و همچنین تعاملی را از طریق صدا فراهم می‌کنند. همانطور که در شکل 7 نشان داده شده است، هر یک از سرویس‌های کشف دستگاه، استقرار، یکپارچه سازی توسط Hue and Bendi ارائه شده است؛ و سپس ورودی خود را بر سرویس VoIP SMS ارسال کند. این یک تابع اضافی با یک کد مختصر است، چرا که همه جزییات اجرایی برای کشف؛ ارتباط و کنترل به پلت فرم واگذار می‌شود.

B. سناریوی اپلیکیشن‌های چند منظوره

اپلیکیشن‌های چند منظوره [15] برای ارائه آزادی بیشتر به کاربر در انتخاب و ترکیب اشیا در تلفن هوشمند اجرا شده است. GUI برای ادغام اشیای هوشمند بر اساس ترکیب سرویس متن باز Node-RED [12] اجرا شده است. در اپلیکیشن‌های چند منظوره، منطق تجاری ادغام شده در اشیای هوشمند در فرمی از منطق خدمات[32] ارائه می‌شود.

[32] Service Logic

این منطق سرویس در مقایسه با منطق تجاری IoT-App عمومی، منطق اتمیک کوچکتری است، لذا کاربر می‌تواند آنچه که قصد دارد را انتخاب کند. در این سناریو، ما از TI SensorTag [13] استفاده کردیم، که حسگرهای شتاب و لامپ هوشمند Hue را در خود دارد. همانطور که در شکل 8 تشریح شد، دستگاه‌های اسکن شده در GUI نشان داده می‌شوند، کاربر به دستگاه‌ها متصل می‌شود و سپس سرویس راه اندازی می‌شود. مقادیر تولید شده توسط SensorTag توسط منطق سرویس به مقدار RGB تفسیر می‌شود؛ لذا زمانی که کاربر SensorTag را تکان دهد، لامپ‌ها چشمک می‌زنند.

(a) Devices on Scenario

(b) Service Launching

(c) Scenario Outline

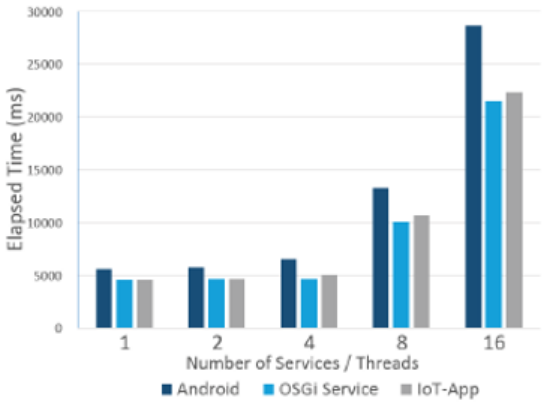
Figure 7: Bendi Call Scenario

C. ارزیابی سر بار پلت فرم IoT-MAP

برای ارزیابی سربار پلت فرم IoT-MAP، از اجرای اشیای هوشمند مجازی و مقایسه تفاوت زمانی با و بدون پشتیبان پلت فرم IoT-MAP استفاده کردیم. این شی مجازی در CPU تلفن هوشمند بارگذاری می‌شود، و IoT-App آن را از 1 تا 16 بار اجرا می‌کند. تست بر Nexus 7 (2013) اجرا شده است که دارای یک پردازنده دو هسته‌ای است و نتایج در شکل 9 نشان داده شده اند.

در نمودار، اندروید به کد شی مجازی در حال اجرا در Android Activity اشاره دارد، OSGi Service به شی اشاره دارد که مستقیما در فریم ورک OSGi اجرا می‌شود، و IoT-App به شی اشاره دارد که بارگذاری شده است و در پلت فرم IoT-MAP اجرا شده است. تفاوت دقیق سربار می‌تواند با سرویس OSGi و IoT-App محاسبه شود. هیچ تفاوت قابل ملاحظه‌ای در زمانی که یک یا دو شی استفاده می‌شوند وجود ندارد، اما زمانی که تعداد بیشتری استفاده می‌شود، حدود 5٪ سربار اندازه گرفته شده است. این به نظر می‌رسد که زمانی که 4 نخ یا بیشتر اجرا می‌شود نخ[۳۳] در IoT-App سربار کمتری دارد. این نتایج شرایط بحرانی را ایجاد می‌کند که در آن اشیا بار کامل را به تلفن هوشمند واگذار می‌کنند؛ لذا سربار حقیقی استفاده معمولی است که اهمیتی هم ندارد. اندروید پرفورمنس کمتری را نسبت به سایرین نشان می‌دهد، و این نشان می‌دهد که خود فریم ورک OSGi می‌تواند شی را کارامدتر از فعالیت نخ در اندروید اجرا کند.

شکل 8: رویه اجرای اپلیکیشن‌های چند منظوره



شکل 9: گراف ارزیابی سربار پلت فرم IoT-MAP

[33] thread

7. کارهای مربوطه

ترکیب اطلاعات موجود و خدمات از منابع متعدد برای ایجاد یک سرویس جدید در حال حاضر در تکنولوژی شبکه سراسری وب تحت نام Web Mashup شهرت یافته است. تلاش برای استفاده از این مفهوم در اشیای هوشمند مطالعات نتیجه شده‌ای مانند مشاپ در وب اشیا [34][4] یا دستگاه‌های ناهمگون گوناگون است. همچنین، بسیاری از مطالعات علمی (e.g. Cilia[7] or Dynamo[8]) و پروژه‌های متن باز (e.g. Eclipse IoT[10] or OpenHAB[9]) بر اینکه چگونه خدمات اشیای هوشمند را ترکیب کنند و ارزش جدیدی را به کاربران ارائه دهند کار کردند. به هر حال، بیشتر این مطالعات و پروژه‌ها معمولا وجود یک سرور اصلی یا عامل دروازه‌ای را برای مدیریت دستگاه‌های اصلی فرض می‌کنند.

مشاپ IoT به عنوان یک سرویس[35] [6] یک مدل سرویس مبتنی بر ابر تعمیم یافته را ارائه داد که شامل مدل نرم افزاری، مدل اشیا، مدل مناسب محاسباتی بود، و یک معماری مبتنی بر ابر را برای کاربران نهایی با ترکیبی از خدمات با اسمبل کردن مولفه‌های نرم افزاری منعطف اپلیکیشن و پروکسی‌های دستگاه‌های ناهمگمن طراحی کردند. در حالی که IoTMaas مدل سرویس مبتنی بر ابر و معماری را هدف قرارداده بود،کار ما بر محیط فراگیر موبایل بحث کرده است، لذا این شانس وجود دارد که با ترکیب پویای دستگاه‌های شخصی و محیطی تجربه بهتری برای کاربر فراهم کند. Ambient Dynamix [16] یک ترکیب عملیاتی از خدمات را برای اپلیکیشن‌های موبایلی که از اشیای هوشمند استفاده می‌کنند، ارائه داده است. این زمینه پیشرفته قابلیت سنجش برای پشتیبانی از تابع ترکیب مبتنی بر جامعه را فراهم می‌کند. به هر حال، تنها می‌تواند دستگاه‌های ثبت شده در سرور پلاگین Dynamix را برای کشف بخواند در حالی که IoT-MAP می‌تواند دستگاه‌های احاطه کننده را به روش عمومی با سرور نام ایجاد شده مبتنی بر متن باز Oliot-ONS کشف کند، و این رویه حل تعارض ID را برای بازیابی لیست خدمات از سرویس نام شی اجرا می‌کند. سرویس نام شی از ساختار معماری سلسله مراتبی DNS تبعیت می‌کند و به صورت سراسری توزیع شده است. لذا با استفاده از زیر ساخت ONS، سیستم مخزن خارجی به شدت مقیاس پذیر است.

---

[34] Web of Things

[35] IoT Mashup as a Service

8. نتیجه گیری

در این مقاله، یک پلت فرم ترکیبی از سرویس اشیای هوشمند تلفن هوشمند محور IoT-MAP را معرفی کردیم که به صورت پویایی دستگاه‌ها را کشف می‌کند، درایورها را مستقر می‌کند، و واسط یکنواختی برای IoT-App، برای آن‌ها فراهم می‌کند. برای کار آینده، متدی برای تحریک واسط عمومی انتزاعی باید با پروتکل‌های مبتنی بر پروفایل موجود برای انتزاعی از بیشتر اشیای هوشمند وجود در بازار مطالعه شود. و، به دلیل اینکه واسط انتزاعی نمی‌تواند همه قابلیت‌های هر شی را پوشش دهد، پلت فرم باید واسط تعمیمی یافته‌ای را برای مدیریت محاوره GUI برای تعامل اضافی بین کاربر و درایور شی هوشمند ارائه دهد.

REFERENCES

[1] Traub, Ken, et al. *The GS1 EPCglobal Architecture Framework*, GS1, 2014.

[2] Dean, Kevin, et al. *GS1 Object Name Service (ONS)*, GS1, 2013.

[3] Murugesan, San. *Understanding Web 2.0*, IT Professional, 2007, 9.4: 34-41.

[4] Guinard, Dominique, et al. *Towards physical mashups in the web of things*, In: Networked Sensing Systems (INSS), 2009 Sixth International Conference on. IEEE, 2009. p. 1-4.

[5] Bandyopadhyay, Debasis; Sen, Jaydip. *Internet of things: Applications and challenges in technology and standardization*, Wireless Personal Communications, 2011, 58.1: 49-69.

[6] Im, Janggwan; Kim, Seonghoon; Kim, Daeyoung. *IoT Mashup as a Service: Cloud-Based Mashup Service for the Internet of Things*, In: Services Computing (SCC), 2013 IEEE International Conference on. IEEE, 2013. p. 462-469.

[7] Lalanda, Philippe; Escoffier, Clment; Hamon, Catherine. *Cilia: An autonomic service bus for pervasive environments*, In: Services Computing (SCC), 2014 IEEE International Conference on. IEEE, 2014. p. 488-495.

[8] Avouac, P.-A.; Lalanda, Philippe; Nigay, Laurence. *Adaptable multimodal interfaces in pervasive environments*, In: Consumer Communications and Networking Conference (CCNC), 2012 IEEE. IEEE, 2012. p. 544-548.

[9] OpenHAB. Open HAB site. http://www.openhab.org/index.html

[10] Eclipse IoT. Eclipse IoT site. http://iot.eclipse.org/index.html

[11] OSGi Alliance, OSGi site http://www.osgi.org/

[12] IBM Emerging Technology, Node-RED site http://nodered.org/

[13] Texas Instrument, CC2541 SensorTag Development Kit site http://www.ti.com/tool/cc2541dk-sensor

[14] Park, Young-Woo; Park, Joohee; Nam, Tek-Jin. *Bendi: Shape-Changing Mobile Device for a Tactile-Visual Phone Conversation*, In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems. ACM, 2015. p. 181-181.

[15] Sungpil Woo, Sehyeon Heo, Janggwan Im, and Daeyoung Kim. *Versatile Internet of Things Application on Mobile Dynamic Service Composition Framework*, In: The 4th International Conference on Internet of Things (IoT 2014), MIT, USA, Oct. 4-8, 2014

[16] Darren Carlson, Andreas Schrader *Dynamix: An Open Plug-and-Play Context Framework for Android*, In: The 3rd International Conference on Internet of Things (IoT 2012)

[17] OMG Architecture Board ORMSC. Model driven architecture (MDA). OMG document number ormsc/2001-07-01, available from www.omg.org, July 2001