TUGAS KAPITA SELEKTA

MOSDEN

DISUSUN OLEH:

NAMA : YOGA YOLANDA

NIM : 09011181320041

JURUSAN SISTEM KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJAYA

2016

Referensi : **MOSDEN: An Internet of Things Middleware for Resource Constrained Mobile Devices**

Author : **Charith Perera, Prem Prakash Jayaraman, Arkady Zaslavsky, Dimitrios Georgakopoulos** (CSIRO ICT Center, Canberra, ACT 2601, Australia) dan **Peter Christen** (Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia)

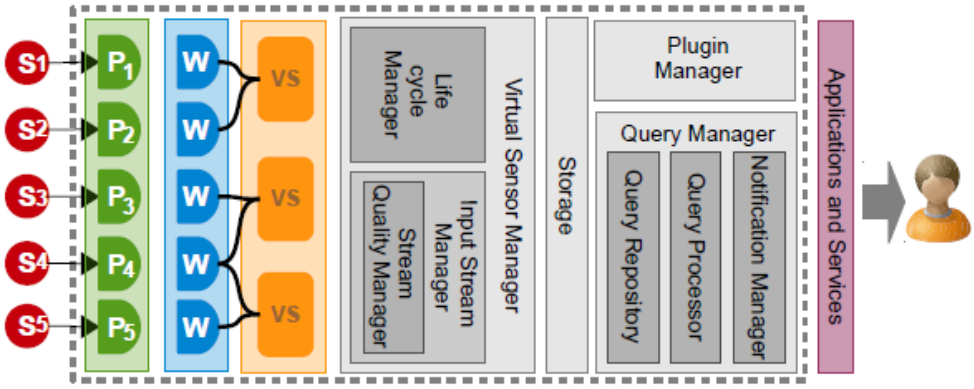
Mobile Sensor Data Processing Engine (MOSDEN) merupakan sebuah *plug-in* berbasis IOT *middleware* untuk perangkat *mobile* yang memungkinkan untuk mengumpulkan dan memproses data sensor tanpa melakukan pemprograman. MOSDEN dirancang untuk mendukung penginderaan (*sensing*) sebagai layanan model *native*. MOSDEN merupakan *true zero programming middleware* dimana *user* tidak perlu menulis kode *program* atau spesifikasi lainnya menggunakan bahasa deklaratif. MOSDEN dapat memproses data sensor berdasarkan SQL seperti *query* yang biasanya dapat menurunkan jaringan komunikasi karena penggabungan pada data sensor. MOSDEN dapat digunakan di semua aplikasi untuk meningkatkan infrastruktur IOT secara berkala dengan menggunakan energi yang tersedia secara optimal dan mengurangi komunikasi data yang tidak perlu. Khususnya pada penggunakan perangkat penginderaan (*sensing*) luar ruangan yang membutuhkan pengisian tenaga pada baterai secara intensif.

- **Arsitektur Desain MOSDEN**

1. Arsitektur *Plug-in*

Dalam MOSDEN, arsitektur *plug-in* dibentuk untuk mendukung tiga persyaratan utama: **skalabilitas, kegunaan, dan pembangunan berbasis komunitas**. Sebuah *plug-in* adalah komponen perangkat lunak yang independen, dengan menambahkan fitur khusus ke dalam aplikasi perangkat lunak yang sudah ada dalam MOSDEN. Setiap *plug-in* menterjemahkan pesan

umum dari komunikasi ke sensor dengan perintah tertentu yang memungkinkan komunikasi antara MOSDEN dan sensor tertentu. Ketika sebuah aplikasi mendukung *plug-in*, maka kustomisasi dapat dijalankan. Selanjutnya, MOSDEN *plug-in* dapat di *install* dan dikonfigurasi pada saat dijalankan.

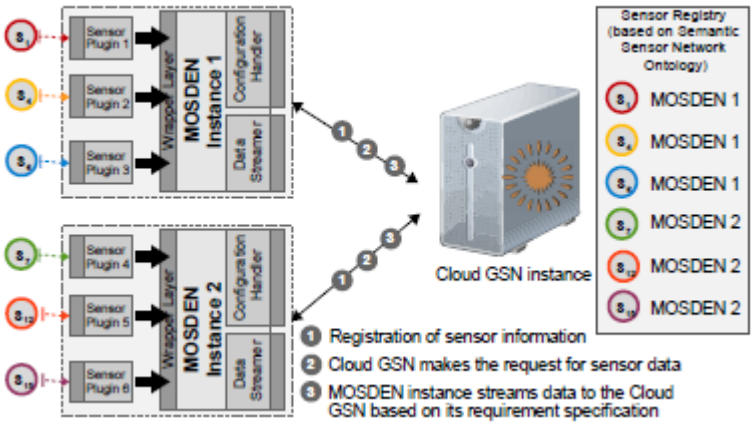


Gambar 1. Arsitektur Desain dari MOSDEN. *Legend* : Sensor (S), *Plug-in* (P), *Wrapper* (W), *Virtual* Sensor (VS). Komunikasi *plug-in* dengan sensor dan penerimaan data. Setiap *plug-in* harus kompatibel dengan sensor yang ingin melakukan komunikasi dengannya.

Arsitektur MOSDEN didasarkan pada arsitektur GSN (*Global Sensor Networks*). Perubahan utama terdapat pada penambahan *plug-in manager* dan lapisan *plug-in* untuk mendukung dan memanipulasi *plug-in*. GSN membutuhkan *wrappers* yang berbeda untuk terhubung ke sensor yang berbeda. Dalam MOSDEN, *wrappers* tidak langsung berkomunikasi dengan sensor. Sebaliknya, komunikasi *generic wrappers* dengan *plug-in* dan *plug-in* yang berkomunikasi dengan sensor (yaitu *wrapper* → *plug-in* (Pi) → Sensor (S_i). Karena pengenalan *generic wrappers*, menyebabkan panduan re-kompilasi dari MOSDEN tidak diperlukan ketika sensor baru ditambahkan.

2. Interaksi Dengan Cloud dan Peers

MOSDEN adalah desain yang akan digunakan sebagai bagian dari penginderaan (*sensing)* model layanan.

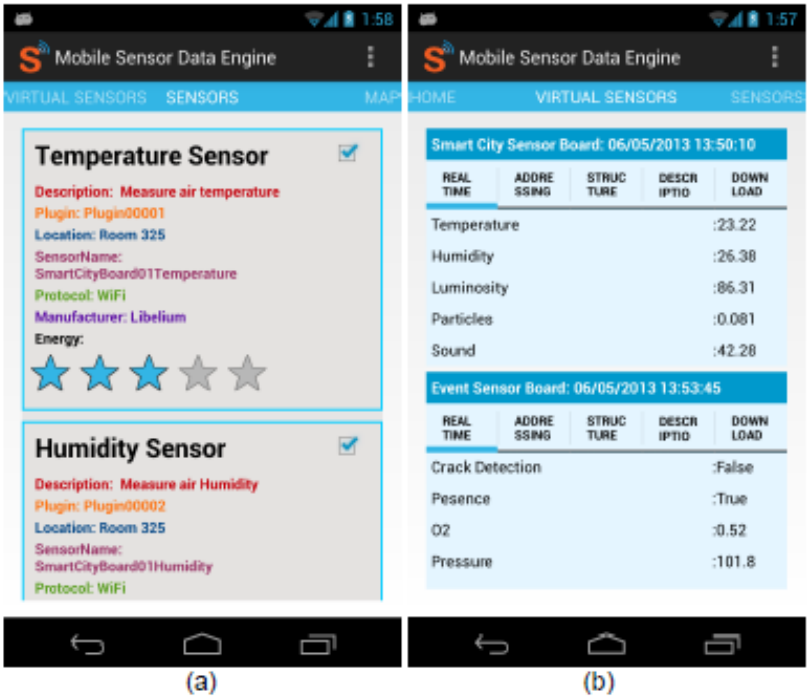


Gambar 2. Interaksi antara MOSDEN dengan GSN (*Global Sensor Networks*)

Ketika *Cloud* GSN (*Global Sensor Networks*) menerima permintaan dari pengguna, maka registri dari sensor deskripsi akan meminta untuk menemukan sensor yang relevan yang sesuai dengan kebutuhan pengguna. Setelah menemukan sebuah MOSDEN yang mampu memenuhi permintaan pengguna (yaitu apakah MOSDEN yang diberikan mampu mengumpulkan data dari sensor yang diperlukan oleh pengguna). Selanjutnya, GSN mengirimkan permintaan untuk MOSDEN tersebut. Kemudian, masing-masing MOSDEN register melakukan permintaan. Akhirnya, MOSDEN mulai melakukan *streaming* data yang diminta ke *cloud* GSN. *Cloud* GSN dapat membuat permintaan baik pada kondisi *pull* dan *push* dalam mekanisme. Pada metode *pull*, GSN membuat permintaan setiap kali GNS menginginkan data dari MOSDEN. Sedangkan pada metode push, *cloud* GSN akan mengirimkan permintaan dan MOSDEN mengirimkan data kembali sampai permintaan berakhir.

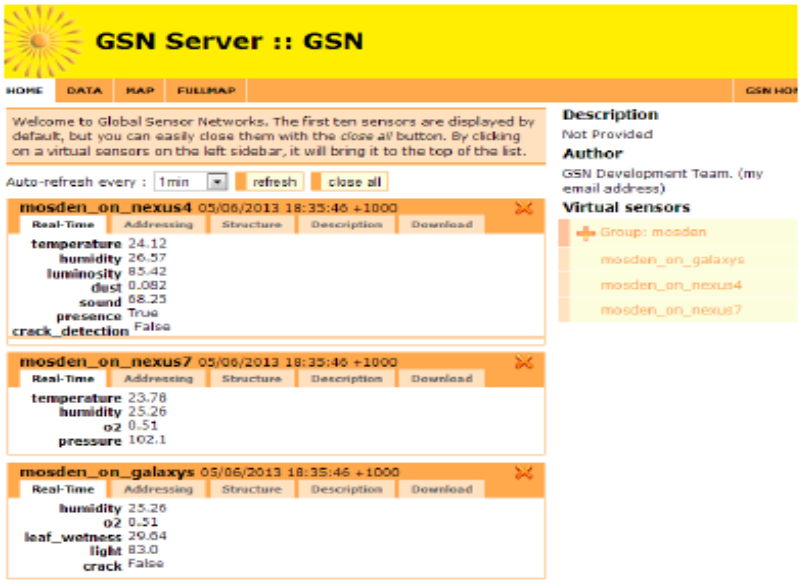
3. Implementasi

Tujuan dalam mengembangkan MOSDEN tidak hanya untuk mendukung *platform* ponsel tetapi juga untuk mendukung perangkat seperti Raspberry Pi. MOSDEN didasarkan pada *middleware* populer disebut *Global Sensor Networks* (GSN).



Gambar 3. MOSDEN *screenshot* : (a) Daftar sensor yang terkoneksi ke MOSDEN ; (b) Daftar *virtual* sensor yang sedang berjalan di MOSDEN dan rincian dari sensor.

MOSDEN tidak hanya mendukung ponsel tetapi juga mungkin dapat mendukung perangkat yang memiliki keterbatasan yang hampir serupa, seperti perangkat yang memiliki layar atau tidak memiliki layar.



Gambar 4. *Screenshot* dari sebuah *cloud* GSN yang menunjukan tiga contoh dari MOSDEN yang terdaftar.

Semua fitur yang tersedia di GSN juga tersedia di MOSDEN termasuk pengolahan data dan REST-*base* komunikasi *peer-to-peer* melalui HTTP. Dibandingkan dengan GSN, struktur *wrappers* telah berubah dan *generic wrappers* lebih dikembangkan. Selanjutnya, konsep *plug-in* mulai ditampilkan dan penambahan pada lapisan *plug-in* serta *plug-in manager*. Serta penggantian pada *user interface* berbasis *web* dengan aplikasi *native adroid.*

MOSDEN *platform* bisa sangat membantu dalam pengembangan layanan data yang inovatif tergantung pada perangkat IOT sebagai sumber data. Salah satu contohnya adalah aplikasi *crow-source* dimana data sensor (misalnya tingkat kebisingan di lingkungan luar) bisa dikumpulkan dari perangkat seluler pengguna yang menjalankan MOSDEN. Data yang dikumpulkan dapat digunakan oleh aplikasi di *cloud* dalam proses pengambilan keputusan (misalnya menentukan tingkat *noise* di sebuah persimpangan di kota dengan menggabungkan data dari beberapa contoh MOSDEN). Contoh lain adalah untuk menentukan kondisi *real-time* lalu lintas menggunakan data yang diperoleh dari MOSDEN yang aktif pada perangkat *mobile* pengguna.

- **Kesimpulan** :

1. MOSDEN berfokus pada pengumpulan dan pengolahan data sensor baik secara internal dan eksternal sensor.
2. MOSDEN memberikan kemudahan dan cara yang tepat dalam menghubungkan sensor untuk perangkat *mobile* dengan upaya *zero programming*.
3. MOSDEN mampu mengumpulkan data dari beberapa sensor yang berbeda dan memprosesnya secara bersamaan.
4. MOSDEN 100% kompatibel dengan *Global Sensor Network* (GSN) *middleware* yang berjalan di *cloud*.