Tutorial:

How to Getting Started Macchina.io for Internet of Things Platforms

Macchina.io merupakan perangkat lunak terbuka untuk membangun aplikasi tertanam (*embedded*) pada *Internet of Things* yang berjalan di perangkat berbasis Linux seperti *Raspberry Pi, Beaglebone, RED Brick* atau *Galileo/Edison*. *Macchina.io* menerapkan *web-enabled*, modular dan dapat dikembangkan dengan *JavaScript* dan C ++ pada lingkungan *runtime*, selain itu macchina.io mudah digunakan untuk membangun block yang memungkinkan aplikasi untuk berbicara dengan berbagai sensor dan perangkat, serta layanan *cloud*. *Macchina.io* dapat diinstal pada Linux dan OS X sistem. Pada Linux, cross-kompilasi untuk target Linux *embedded* didukung.

Untuk langkah pertama, anda dianjurkan membangun dan menginstal *macchina.io* pada desktop Linux atau OS X sistem. Anda juga dapat membangun *macchina.io* pada sistem seperti Raspberry Pi atau *Tinkerforge RED brick*. Atau, anda bisa *cross-compile macchina.io* untuk target Linux *embedded*.

➢ *Getting macchina.io* (Mendapatkan *macchina.io*)

Macchina.io dapat diperoleh dari *Github*, dengan kloning repositori *macchina.io*, atau anda dapat men-*download* arsip rilis format .zip atau .tar.gz dari *macchina.io GitHub Releases page.*

Untuk klon *macchina.io GitHub repository* :

```
$ sudo apt-get install git
$ git clone https://github.com/macchina-io/macchina.io.git
```

Untuk rilis terbaru, anda mungkin ingin memeriksa *master branch* :

```
$ cd macchina.io
$ git checkout master
```

➢ *Building macchina.io* (Membangun *macchina.io*)

***Prerequisites* (Prasyarat) :**

Macchina.io menggunakan sistem build dari *POCO C++ libraries*, yang didasarkan pada GNU. Anda akan membutuhkan versi GNU C++ *compiler* (setidaknya versi 4.4) atau *clang*. Pada sistem Linux,pastikan anda menginstall *OpenSSL Headers* dan *libraries.*

***Compiling macchina.io* (Kompilasi *macchina.io*) :**

Untuk membangun *macchina.io,* GNU Make pada the top-level macchina.io source directory. Pada sistem multicore dengan memori yang memadai, anda mungkin ingin menjalankan *parallel build*, dengan menspesifikasi opsi j.

```
$ make -s -j8
```

Perintah ini akan benar-benar membangun *both debug* dan *release build*. Untuk mempercepat, anda mungkin ingin membangun konfigurasi rilis saja, maka gunakan perintah :

```
$ make -s -j8 DEFAULT_TARGET=shared_release
```

Pada mesin dekstop atau notebook, membangun *macchina.io* akan membutuhkan waktu sekitar 10 sampai 15 menit. Jika anda membangun *macchina.io* pada sistem seperti *Raspberry Pi* atau *Tinkerforge RED Brick,* sistem tersebut akan membutuhkan waktu beberapa jam. Pada sistem *Raspberry Pi* atau *Tinkerforge RED Brick* anda jangan mencoba membangun *parallel build* (-j8), karena hal tersebut akan menyebabkan anda kehabisan memori selama membangun V8 *libraries*. Cobalah untuk me-*restart build* dan pastikan memori tersedia sebanyak mungkin, dengan menghentikan semua proses yang tidak perlu. Ini mungkin lebih baik untuk *cross-compile* pada sistem *desktop* dan mentransfer binari untuk sistem sasaran.

➢ *Cross Compiling macchina.io*

#*note*: jika anda ingin menjalankan *macchina.io* pada sistem desktop saja, anda dapat dengan aman melewati bagian ini dan melanjutkan langsung ke tahap ***Running macchina.io.***

Macchina.io dapat di *cross-compile* untuk target Linux *embedded. Cross-compile* membangun konfigurasi dari target harus dibuat atau memilih konfigurasi yang dipilih. Beberapa konfigurasi untuk target Linux *embedded* dapat ditemukan pada *platform/build/config directory*. Membangun konfigurasi menentukan *toolchain/compiler* yang akan digunakan serta setiap *compiler* atau *linker* khusus argumen diperlukan untuk target. Hal terbaik untuk membuat konfigurasi baru yaitu dengan menyalin yang sudah ada dan menyalin ke *platform/build/config directory*. Poin yang baik untuk target berbasis ARM yang *DigiEL-A8, Angstrom, BeagleBoard, yocto* atau *ELDK.*

Kamu dapat menemukan lebih banyak informasi tentang membangun konfigurasi dan membangun sistem secara umum pada *POCO C++ Libraries GNU Make Build* sistem dokumen.

Sebelum cross-kompilasi untuk target Linux *embedded*, anda harus mengkompilasi sumber untuk beberapa *libraries* dan *tools* untuk sistem *host*. Anda dapat melakukannya dengan perintah berikut:

```
$ make -s -j8 hosttools
```

#note: anda tidak harus membangun seluruh *macchina.io* untuk kedua host dan target dalam direktori yang sama, kecuali untuk alat yang diperlukan *host* dan *libraries*. Setelah anda telah membangun *macchina.io* untuk sistem target, file *bundle* akan tidak lagi dapat digunakan pada host karena mengandung binari target saja.

Jika anda telah menemukan yang ada, atau menciptakan *suitable* membangun konfigurasi baru untuk target anda, anda dapat membangun untuk target dengan mengikuti perintah berikut (dengan asumsi *Yocto* membangun konfigurasi, diganti dengan milik anda sendiri):

```
$ POCO_CONFIG=Yocto LINKMODE=SHARED make -s -j8 DEFAULT_TARGET=shared_
release
```

#note: kecuali anda berniat untuk *debug* pada target, hal itu sudah cukup untuk membangun konfigurasi rilis saja.

Setelah membangun selesai, anda harus menyalin file-file berikut untuk target:

- Berikut *shared library* (libPoco * .so *.) dari *platform/lib/Linux/<target_arch>*: Foundation, XML, JSON, Util, Net, Zip, Crypto, NetSSL, Geo, WebTunnel, RemotingNG dan OSP.
- Semua file bundel (* .bndl) dari *platform/OSP/bundles, devices/bundles, protocols/bundles, services/bundles, webui/bundles.*
- *Macchina.io Server executable*: *Server/bin/Linux/<target_arch>/macchina.*
- *Macchina.io* file konfigurasi server: *Server/macchina.properties*.

Semua bundel harus disalin ke dalam direktori terpisah pada target. Sebuah layout filesystem direkomendasikan pada target akan menjadi:

```
<macchina_root>/
    lib/
        bundles/
    bin/
    etc/
```

Semua *shared library* harus dibuat kedalam direktori lib dan semua bundel dibuat ke lib/bundles directory.

Anda dapat menggunakan *install-runtime* Makefile target untuk disalin semua file kedalam direktori hirarki teratas.

```
$ POCO_CONFIG=Yocto make -s install_runtime INSTALLDIR=/path/to/macchina
-install-dir
```

Kemudian anda dapat mengumpulkan direktori ke dalam arsip (misalnya: tar.gz) file dan ditransfer ke target.

File konfigurasi untuk server *macchina.io, macchina.properties* (terletak di direktori server) harus disalin ke dalam direktori */etc* pada target. File konfigurasi perlu dimodifikasi untuk target. Buka *macchina.properties* di editor teks dan mengubah baris :

```
osp.bundleRepository = ${application.configDir}../platform/OSP/
bundles;...
```

Menjadi :

```
osp.bundleRepository = /path/to/lib/bundles
```

➢ *Running macchina.io* (Menjalankan *macchina.io*)

Setelah *macchina.io build completes*, anda dapat langsung menjalankan *macchina.io* tanpa harus menginstall langkah tambahan. Untuk menjalankan *macchina.io server* anda harus :

- Mengatur sistem anda bersama jalur *library search* untuk menyertakan *macchina.io libraries.*
- Menjalankan program server *macchina.io.*

Berikut adalah perintah untuk menjalankan *macchina.io* pada sistem Linux :

```
$ export LD_LIBRARY_PATH=/path/to/macchina.io/platform/lib/Linux/x86_64
$ cd /path/to/macchina.io/server
$ bin/Linux/x86_64/macchina
```

Sedangkan. berikut ini adalah perintah untuk menjalankan *macchina.io* di sistem OS X :

```
$ export DYLD_LIBRARY_PATH=/path/to/macchina.io/platform/lib/Darwin/x86
_64
$ cd /path/to/macchina.io/server
$ bin/Darwin/x86_64/macchina
```

#note: jika *macchina.io* anda file konfigurasi *macchina.properties* tidak terletak pada direktori yang sama sebagai *executable server* atau dalam direktori induk, anda perlu menentukan jalur ke file konfigurasi ketika memulai server:

```
$ bin/Darwin/x86_64/macchina --config=/path/to/macchina.properties
```

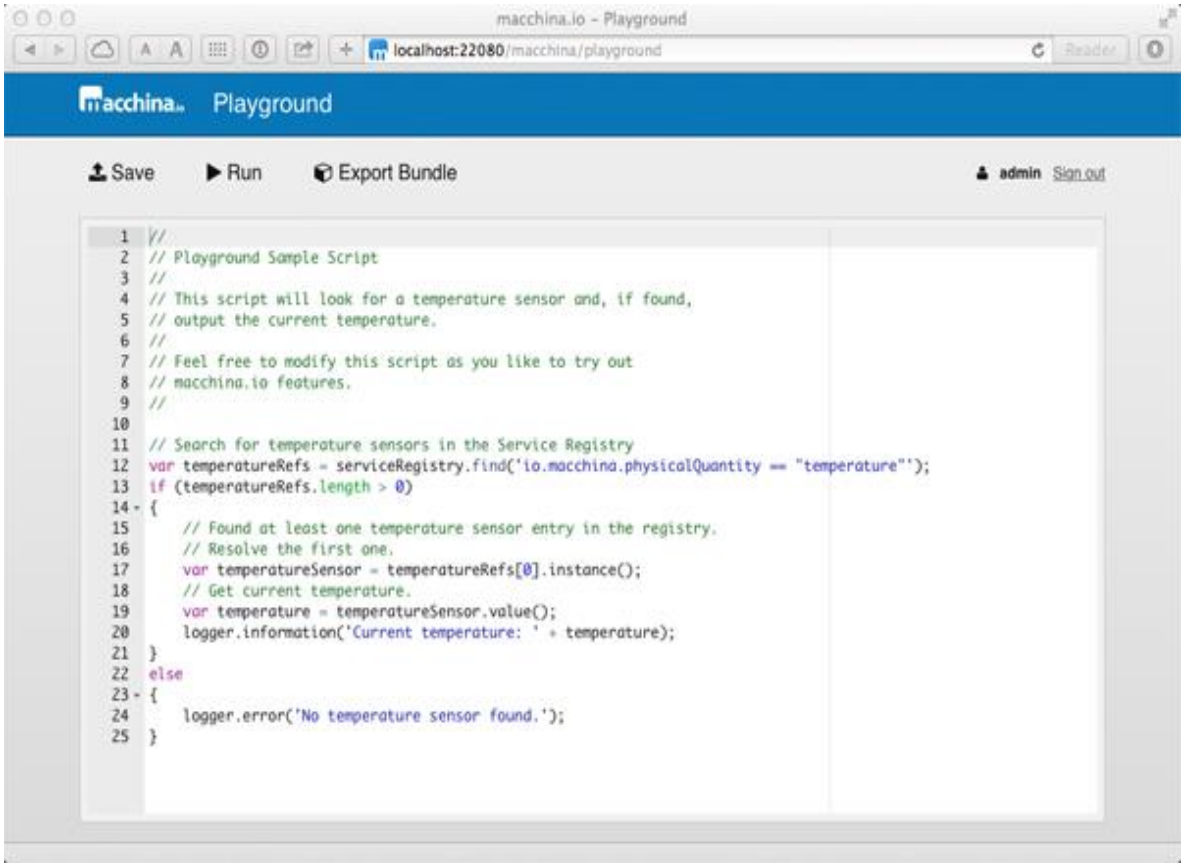
Setelah server *macchina.io* telah dimulai, silahkan anda mengarahkan browser web anda menuju http://localhost:22080. Anda akan melihat layar login *macchina.io*. Untuk login gunakan pengguna username "admin" dan password "admin". Setelah berhasil login anda akan melihat halaman Launcher dari antarmuka web.

Konfigurasi *default* dari *macchina.io* memungkinkan simulasi berbagai sensor, termasuk suhu dan sensor cahaya. Anda dapat menggunakan sensor simulasi dengan cara yang sama seperti anda akan menggunakan sensor yang terhubung nyata dalam kode Anda. Sensor yang tersedia dan pengukuran saat ini dapat dilihat pada "Sensor & Devices" app.

Untuk mendapatkan hasil maksimal dari *macchina.io*, anda mungkin ingin mendapatkan beberapa sensor yang nyata. Dalam rilis pengembang *preview macchina.io*, hanya beberapa sensor Tinkerforge didukung. Selebihnya sensor dan perangkat akan didukung dalam rilis mendatang. Anda tentu saja juga dapat menerapkan jenis sensor, setelah anda terbiasa dengan *macchina.io Software Development Kit* (SDK).

- ***Writing Your First JavaScript Program* (Menulis Program Pertama *JavaScript* Anda)**

Untuk dapat menulis program *JavaScript* yang berjalan di macchina.io, silahkan anda buka *"Playground"* app. *Playground* memungkinkan anda untuk membuat dan mengedit program *JavaScript* secara langsung di browser, menggunakan editor teks.



tampilan macchina.io playground, JavaScript editor teks

- ***Getting Sensor Data*** **(Mendapatkan Sensor Data)**

Playground dilengkapi dengan *pre-loaded* varian *macchina.io* dari "Hello, world!". Program-program kecil yang menemukan sensor suhu dan memperoleh suhu saat ini dari sensor. Berikut adalah contoh pogram untuk referensi.

```
//
// Playground Sample Script
//
// This script will look for a temperature sensor and, if found,
// output the current temperature.
//
// Feel free to modify this script as you like to try out
// macchina.io features.
//

// Search for temperature sensors in the Service Registry
var temperatureRefs =
serviceRegistry.find('io.macchina.physicalQuantity == "temperature"');
if (temperatureRefs.length > 0)
{
    // Found at least one temperature sensor entry in the registry.
    // Resolve the first one.
    var temperatureSensor = temperatureRefs[0].instance();
    // Get current temperature.
    var temperature = temperatureSensor.value();
    logger.information('Current temperature: ' + temperature);
}
else
{
    logger.error('No temperature sensor found.');
}
```

Program diatas menunjukkan dua hal, yaitu :

1. Cara mendapatkan objek layanan sensor dari *Registry Service*.
2. Bagaimana menulis log output.

Untuk menemukan sensor yang tersedia dalam sistem, anda menggunakan *Registry Service macchina.io*. Semua sensor dan perangkat yang didukung oleh *macchina.io* akan selalu tersedia sebagai layanan objek melalui *registry* itu. Dalam rangka untuk mencari objek tertentu, *registry* layanan mendukung bahasa ekspresi query sederhana memungkinkan anda untuk menemukan layanan berdasarkan sifat mereka. Dalam contoh program di atas, *macchino.io* secara khusus mencari sensor suhu. Dalam *macchina.io*, sensor selalu memiliki properti bernama *io.macchina.physicalQuantity* yang dapat digunakan untuk mencari sensor yang mengukur kuantitas fisik tertentu. Metode *find* akan mengembalikan *array* dari referensi layanan. Referensi layanan berbeda dari objek layanan yang sebenarnya. Tujuan mereka adalah untuk menyimpan sifat layanan (seperti *io.macchina.physicalQuantity*), serta referensi ke objek layanan yang sebenarnya. Objek layanan yang sebenarnya dapat diperoleh dengan memanggil fungsi misalnya (), seperti yang kita lakukan dalam sampel.

Setelah kita memiliki objek sensor suhu, kita dapat menggunakannya untuk mendapatkan nilai saat ini dengan memanggil fungsi nilai (). Dalam aplikasi *Playground*, anda dapat menjalankan program dengan mengklik "*Run*" tombol di atas editor. Melakukannya dan mengamati output logging dari server *macchina*. Ini harus berisi sesuatu seperti:

```
2015-03-07 13:12:31.630 [Information] osp.core.BundleLoader<2>: Bundle
io.macchina.webui.playground.sandbox started
2015-03-07 13:12:31.630 [Information] osp.bundle.com.appinf.osp.js<2>:
Starting script sandbox.js from bundle
io.macchina.webui.playground.sandbox.
2015-03-07 13:12:31.631 [Information] osp.web.access<2>: POST
/macchina/playground/run.json HTTP/1.1
2015-03-07 13:12:31.688 [Information]
osp.bundle.io.macchina.webui.playground.sandbox<28>: Current
temperature: 26.999999999999975
```

➢ *Periodically Querying Sensor Data*

Sebagai langkah berikutnya, kita akan memodifikasi program sehingga akan secara berkala untuk output suhu saat ini. Kami melakukan ini dengan menyiapkan *timer*. Berikut kode yang diubah:

```
var temperatureRefs =
serviceRegistry.find('io.macchina.physicalQuantity == "temperature"');
if (temperatureRefs.length > 0)
{
    var temperatureSensor = temperatureRefs[0].instance();
    logger.information('Found temperature sensor.');
    setInterval(
        function() {
            logger.information('Current temperature: ' +
temperatureSensor.value());
        },
        1000
    );
}
else
{
    logger.error('No temperature sensor found.');
}
```

Mengganti kode di editor dengan kode di atas, kemudian klik tombol *Restart*. Anda akan melihat suhu yang dicetak setiap detik.

➢ *Handling Sensor Events*

Sebagai langkah terakhir, kita akan mencetak suhu hanya ketika perubahan. Untuk itu, benda-benda sensor memberikan suatu peristiwa agar kita dapat mengikat fungsi *callback*. Setiap kali sensor berubahan nilai, fungsi *callback* akan dipanggil dan suhu saat akan ditulis ke log. Berikut adalah contoh programnya:

```
var temperatureRefs =
serviceRegistry.find('io.macchina.physicalQuantity == "temperature"');
if (temperatureRefs.length > 0)
{
    var temperatureSensor = temperatureRefs[0].instance();
```

```
    temperatureSensor.on('valueChanged',
        function(event) {
            logger.information('Temperature changed: ' + event.data);
        }
    );
}
else
{
    logger.error('No temperature sensor found.');
}
```

- ***Sending Sensor Data Using MQTT***

Sekarang mari kita memperluas sampel untuk mengirim data sensor melalui internet ke server, menggunakan protokol MQTT. Kita akan menggunakan *MQTT sandbox server* yang disediakan oleh *Eclipse Foundation*, yang merupakan *pre-configured* dalam file konfigurasi *macchina.io*. Dibawah ini adalah contoh program dengan menggunakan timer dan memodifikasinya untuk mengirim suhu saat setiap 10 detik.

```
var temperatureSensor;
var temperatureRefs =
serviceRegistry.find('io.macchina.physicalQuantity == "temperature"');
if (temperatureRefs.length > 0)
{
    logger.information('Temperature sensor found.');
    temperatureSensor = temperatureRefs[0].instance();
}
else
{
    logger.error('No temperature sensor found.');
}

var mqttClientRefs = serviceRegistry.find('io.macchina.mqtt.serverURI
!= ""');
if (mqttClientRefs.length > 0)
{
    logger.information("MQTT Client found.");
```

```
    var mqttClient;
    mqttClient = mqttClientRefs[0].instance();

    setInterval(
        function () {
            var payload = JSON.stringify(temperatureSensor.value());
            logger.information('Sending temperature: ' + payload);
            // Send payload using QoS 0
            mqttClient.publish('macchina-io/samples/temperature',
payload, 0);
        },
        10000
    );
}
else
{
    logger.error('No MQTT Client found.');
}
```

MQTT klien merupakan layanan lain yang tersedia dari registri layanan. Anda dapat menemukan dengan mencari properti di *io.macchina.mqtt.serverURI*. Berdasarkan nilai properti ini, anda juga bisa mencari klien tertentu, misalnya "Tcp: //iot.eclipse.org: 1883"

- ***HTTP Web Services***

Sebagai contoh terakhir dalam tutorial ini, anda akan ditunjukkan bagaimana untuk memohon layanan web berbasis HTTP dari *JavaScript*. Jika suhu melebihi batas tertentu, sistem akan mengirimkan pesan SMS menggunakan Twilio. Perhatikan, agar contoh ini dapat bekerja, anda harus memiliki akun Twilio dengan kemampuan untuk mengirim pesan SMS.

```
function sendSMS(message)
{
    var username = application.config.getString('twilio.username');
    var password = application.config.getString('twilio.password');
    var from = application.config.getString('twilio.from');
    var to = application.config.getString('twilio.to');
```

```
    var twilioHttpRequest = new HTTPRequest('POST',
'https://api.twilio.com/2010-04-01/Accounts/' + username +
'/SMS/Messages');
    twilioHttpRequest.authenticate(username, password);
    twilioHttpRequest.contentType = 'application/x-www-form-
urlencoded';
    twilioHttpRequest.content =
        'From=' + encodeURIComponent(from) +
        '&To=' + encodeURIComponent(to) +
        '&Body=' + encodeURIComponent(message);

    var response = twilioHttpRequest.send(function(result) {
        logger.information('SMS sent with HTTP status: ' +
result.response.status);
        logger.information(result.response.content);
    });
}

var smsSent = false;

var temperatureRefs =
serviceRegistry.find('io.macchina.physicalQuantity == "temperature"');
if (temperatureRefs.length > 0)
{
    var temperatureSensor = temperatureRefs[0].instance();
    temperatureSensor.on('valueChanged',
        function(event) {
            logger.information('Temperature changed: ' + event.data);
            if (event.data > 30 && !smsSent)
            {
                logger.warning('Temperature limit exceeded. Sending
SMS.');
                sendSMS('Temperature limit exceeded!');
                smsSent = true;
            }
        }
    );
}
```

```
else
{
    logger.error('No temperature sensor found.');
}
```

Semua kode baru dalam fungsi sendSMS(). Tutorial ini menggunakan objek application.config untuk mengakses pengaturan konfigurasi dari file konfigurasi macchina.properties. Untuk contoh pada bagian ini agar dapat dijalankan, anda harus menambahkan baris berikut ke file konfigurasi:

```
twilio.username = <your_twilio_account_sid>
twilio.password = <your_twilio_auth_token>
twilio.from = <your_twilio_outgoing_phone_number>
twilio.to = <recipient_phone_number>
```

Tentu saja anda harus mengganti penampung (<your_twilio_account_sid>, etc.) dengan nilai-nilai yang tepat dari akun Twillio anda. Pada tutorial ini digunakan kelas HTTPRequest untuk mengirimkan permintaan HTTPS POST ke API Twillio. Permintaan HTTPS dikirim asynchronous. Kemudian akan diberitahu hasilnya melalui fungsi callback, kami sediakan untuk fungsi send().