libMiimetiq

:D

Motivation/Problem statement

- IoT devices are scattered all over, and skynet needs to be able to talk to them

- “Talk to them”?

  - send updates – eg. Fuel level, power etc..

  - perform actions - “powerOff()”, “add(1,2)”, “riseUp()”

- Hence this library. A thin abstraction layer on top of RabbitMQ.

Concepts

- **MiimetiqService**: An object representing the connection to the central exchange that facilitates the message passing. A rabbit MQ server in this case.

- **MiimetiqFeed**: A way to publish/subscribe to the signals we are interested in.

- **MiimetiqRPCServer/MiimetiqRPCClient**: A way to perform actions on a remote device

MiimetiqService

Represents a connection to the MiimetiqService

- Needs: (‘host’, ‘model’, ‘instanceName’, ‘username’, ‘password’)

  eg. (‘127.0.0.1’, ‘TX’, ‘100’, ‘jamesconnor’, ‘resistanceIsFutile’)

- Provides:

  connect(function(error){});

  isReady();

  getFeed(signalParams, function(error, feed){});

  getRPCEndPoint(mode, rpcParams, function(error, endPoint){ });

  disconnect()

MiimetiqFeed

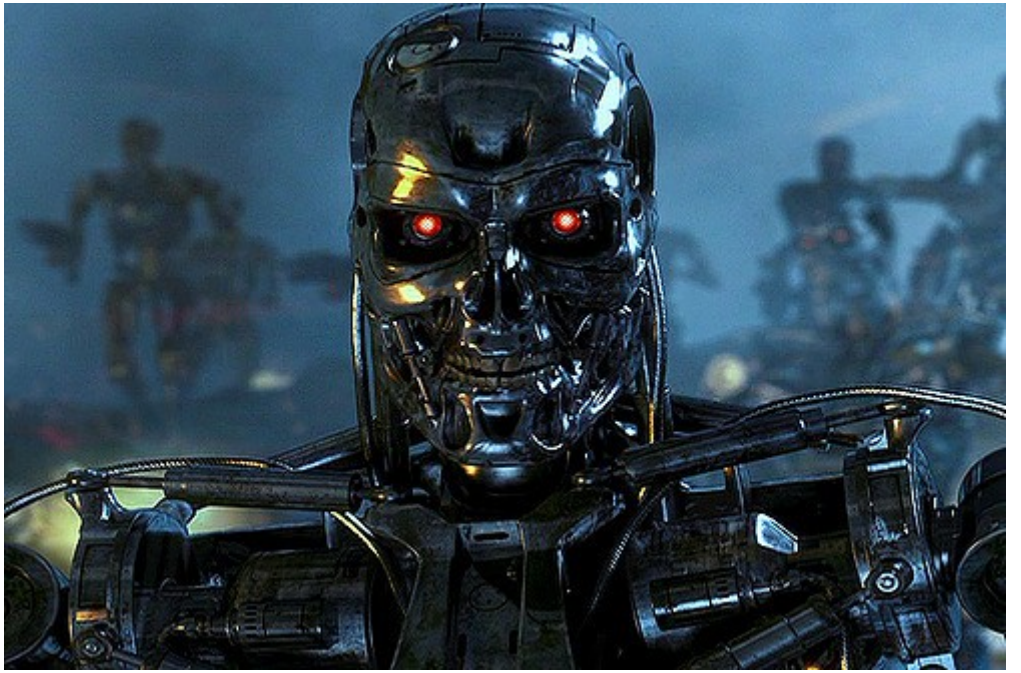
- Needs / Identified by- (type, model, deviceId, instrument, writer)

  eg. (“number”, “T”, “800”, “terminator”, “ammoLevel”)

- Provides

  publish(data)

  subscribe(function(err, rawMessage, data){})



(obligatory poster from the future)

Miimetiq RPC EndPoint

- Needs/Identified by - (model, deviceId, instrument)

  eg. (“TX”, “100”, “terminator”)
- Takes care of serialization, timeouts etc.. and provides:

MiimetiqRPCServer:

```
start({ ‘sayCheese’: function(){ return ‘cheese’; } });

stop();
```

MiimetiqRPCClient

```
connect(function(error, remoteMethods){ });

invoke(‘sayCheese’, [], function(err, result) {});

or

remoteMethods.sayCheese([], function(err, result){});

isReady();

disconnect();
```

Thank You

:D