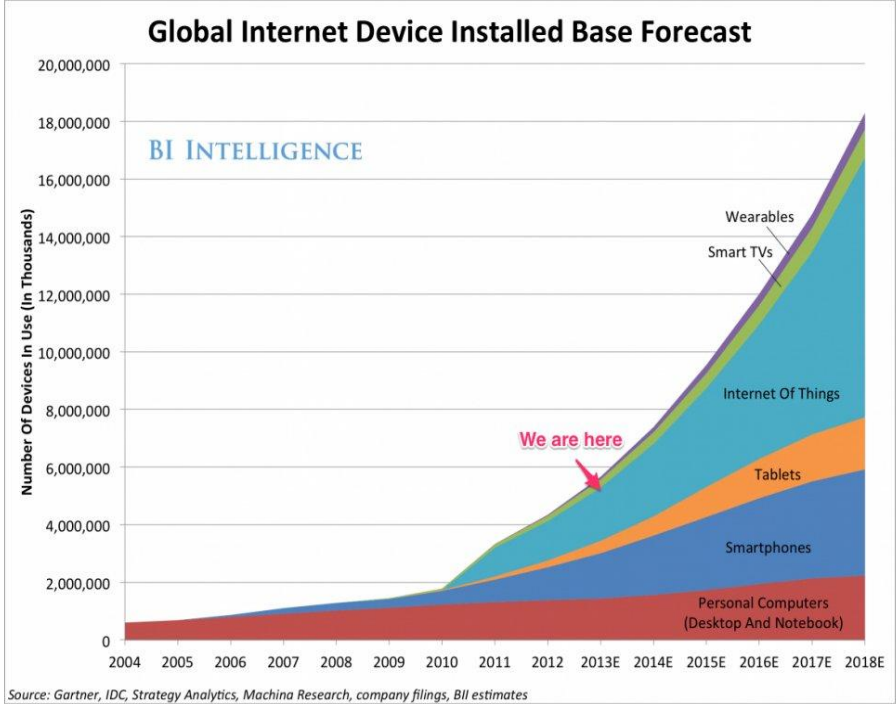
Modern Methods for Communication, Mobility and Portability for the tATAmI Framework

Author: ing. Ionuț Cosmin Mihai
Scientific adviser: Ș.l. dr. ing. Andrei Olaru

Context and motivation

- Ambient Intelligence technologies are currently expanding
- Embedded systems are powerful enough to run higher level of abstraction
- MAS-enabled Ambient Intelligence / Ubiquitous Computing research is in need of MAS Deployment platforms that are flexible and easy to use



What is tATAmI

- Flexible Multi Agent System deployment framework implemented in Java
    - Component-based architecture
    - Several options for the means of inter-agent communication
- tATAmI back in 2014:
    - Local communication
    - JADE based communication
    - PC deployment
- tATAmI now in 2016:
    - Added Websocket communication
    - Added agent mobility
    - Added deployment on Android and Raspbian

Objectives for this research - successfully completed

- Websocket-based communication
    - Modern flexible, easy to use technology
- Agent mobility
    - Enables the MMAS paradigm for implementing ambient services
- Support for Android
- Support for Raspbian
- New components for sensors and actuators
    - Enables context-aware applications
- Integrate the framework on a Raspberry Pi based system
- Architecture changes to support several types of User Interface and specialized logging

State of the art

- Multi Agent System frameworks with support for embedded targets
  - JIAC / microJIAC (Targeted for embedded)
  - JADE / JADE for Android (The most used)
  - Agent Factory / Agent Factory Micro Edition (Large palette of components)
  - MAS C++ (C++ example)
- Current MAS Systems aspects to be improved:
  - **resource consumption adjustment** - now it is affordable to use more resources on embedded systems
  - ignore some MAS unnecessary **features that are not necessarily needed** (e.g. Ontology support)
  - the Micro Editions versions have a **reduced set of features**: no dynamic class loading, precompiled XML configuration file

Websocket communication

- Full Duplex Communication for the Web
- Increased security
  - well known, intense tested
  - no additional ports are required to be opened(only the traditional 80)
  - Encrypted connection due to WSS
- Increased Client-Server efficiency
  - the overhead relative to usual HTTP is reduced up to 1:1000
- SOA oriented architecture
  - simpler to make the server available

Agents mobility

- A paradigm-specific feature specified in the FIPA standard

- Uses Java serialization but is not enough
  - The agent can have transient members
  - Methods for pausing and resuming needed

- Limitation: raise security issues
  - The agent can be corrupted

Portable core extraction

- Extract the functionality that can be used on all target devices
- Add a new component for agent control
- Split the log into agent level log and development level log
- Interfaces for generic HMI (Human Machine Interface)

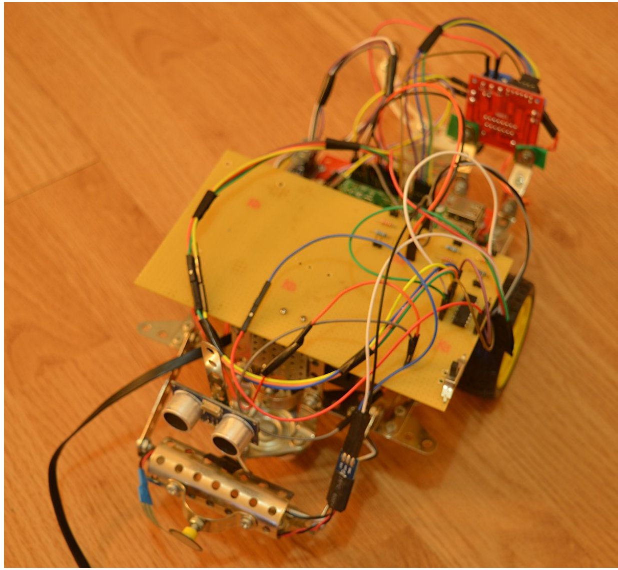
Android Implementation

Several issues:

- the only available IPC method is AIDL, not suitable:
  - Can’t be ported
  - Considerable overhead for marshalling and serialization
  - Solved by directly including the core library in the application
- XML Validation bug on Android - not available for now
- Android restricted policy - the resources are kept differently in the application context instead of a certain path on disk

Raspbian Implementation

- Basic control from command line with two way Java RMI for framework Control
- Used sensors with several types of interfaces:
  - Medium range distance sensor connected to the GPIO pins
  - Accelerometer interfaced through I2C
  - Force analogous sensor interfaced through SPI
  - Electric motors interfaced through GPIO pins
- Used Pi4J library for Raspberry Pi - tATAmI interfacing



Testing

- Individual testing for every hardware component (sensors and motors) using Python scripts and simple Pi4J programs
- Manual checking:
  - The websocket server and client loads correctly
  - The Communication between agents works
  - The Mobility works
  - Sensor sample checking against the samples obtained with the python scripts

Results

- Sensors samples per second

	Designed SPS	Real Maximum SPS	tATAmI SPS
HC-SR04(Distance sensor, GPIO)	N/A	5	4.6
MMA8452Q(Accelerometer, I2C)	800	647	100
Force sensor(SPI)	200	170	90

- Agent transport speed

(seconds)

from \ to	PC	Android	Raspbian
PC	0.34	0.42	4.6
Android	0.42	0.4	0.55

- Memory footprint

Windows	37.8MB
Android	34.7MB
Raspbian	36.9 MB

Conclusion & Further development

- Successfully implemented the project objectives
- Extend the Control component to receive commands
- Implementation of a new ML component (i.e. using TensorFlow)
- Study of the agents behaviour composed of different components

Thank You!