Autonomous Agents on the Edge of Things

Timotheus Kampik[1] Andres Gomez[2] Andrei Ciortea[2,3] Simon Mayer[2]

1: Department of Computing Science, Umeå University, Sweden
2: University of St. Gallen, Switzerland
3: Inria, Université Côte d'Azur, CNRS, France

Motivation

- Semi-autonomous systems are increasingly prevalent across devices/run-time environments
- Can Agent-Oriented Programming (AOP) abstractions further facilitate autonomy?
- To answer this question, we need to move AOP towards cutting-edge industry-scale technology ecosystems

Architecture and Software Stack

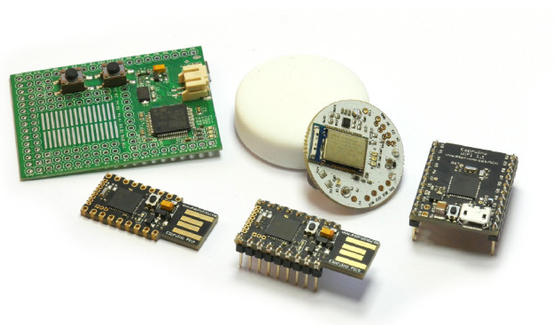

Figure 1:Espruino devices (microcontrollers).

- Starting point: W3C Web of Things Specification and bleeding-edge W3C WoT Scripting API specification draft
- Integrate JavaScript-based *JS-son* agents with Scripting API reference implementation
- Fully JavaScript-based implementation allows deployment across broad range of devices and run-time environments (*write once, run anywhere*)
- Example *edge* devices: Espruino (`http://www.espruino.com/`) devices: JavaScript-enabled microcontrollers with different I/O hardware interface options (Figure 1)

Core Contribution

Cognitive Agents and Web of Things Standards

Video: `https://youtu.be/MUhUuqd2jt0`

- We integrate cognitive agents with Web of Things (WoT) standards and technologies
- In particular, we provide a bridge between the JS-son JavaScript library for agent-oriented programming and the Node.JS reference implementation of the W3C WoT Scripting API
- The agents can run on a broad range of devices (see, for example, below)

Towards Cognitive Agents on Constrained Devices

- Many WoT use-cases involve constrained devices (limited hardware resources)
- We provide a minimalistic version of JS-son that can run on Espruino devices (microcontrollers + I/O + limited JavaScript run-time environment)
- Work can be seen as a point of departure for efficient approaches to run agents on constrained devices

Implementation: `http://s.cs.umu.se/qqza1j`

Application Scenario

- Simplified assembly scenario, several JS-son agents and devices (Figure 2)
  - High-level robot-arm autonomous controller agent (server-side)
  - Browser-based agent for human monitoring and intervention support
  - Espruino Pixl.js interface (micro-controller + simple display) agent for human monitoring support
- Additional non-agent *things* (sensors)
- W3C WoT Scripting API reference implementation serves as middleware

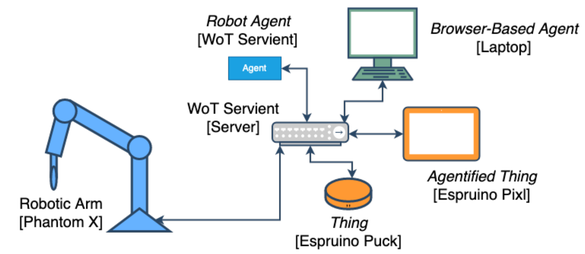

Figure 2:Hardware devices in the application scenario.

Acknowledgments

This work was partially supported by the Wallenberg Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, as well as by a GFF-IPF Grant of the Basic Research Fund of the University of St.Gallen.

Contact Information

- tkampik@cs.umu.se
- andres.gomez@unisg.ch
- andrei.ciortea@unisg.ch
- simon.mayer@unisg.ch

WASP UMEÅ UNIVERSITY University of St.Gallen