Towards Cyber-Physical Systems as Services: the ASIP Protocol

M. Bordoni, M. Bottone, B. Fields, N. Gorogiannis,
M. Margolis, G. Primiero, F. Raimondi

Middlesex Applied Software Engineering Research Group
Department of Computer Science
Middlesex University, London
http://mase.cs.mdx.ac.uk

17 May 2015

The context

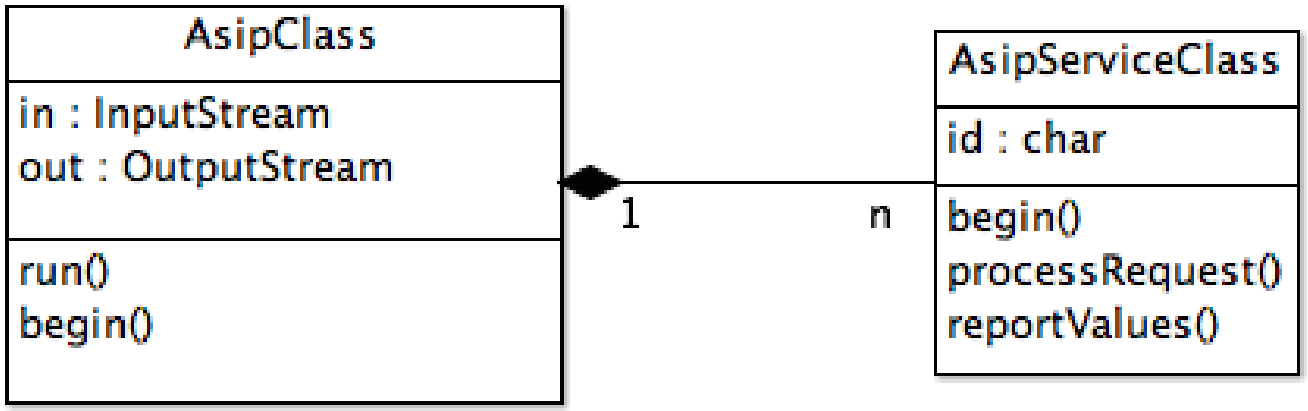
- Cyber-Physical Systems (CPS) are typically built using multiple interacting components (sensors for temperature, light, etc.)
- Components are connected to micro-controllers that often need to be programmed individually to meet a larger application's requirements.
- Focus of this work: expose "components" as services in resource-limited microcontrollers, such as Arduino (2 Kb RAM, approx 30 Kb flash space, no OS, etc.).

Related work

- ROS targets the integration of different platforms with support for a range of programming languages. However, mainly for robotic applications and requires computational resources (at least RPi latest model).
- TinyOS targets motes for WSN, with a typical footprint $>$ 30 Kb, beyond capabilities of most Arduino.
- Firmata is similar in spirit, but ASIP introduces higher level of abstraction and easier extensibility (textual vs 7-bit messages).

The ASIP service model

- A micro-controller should implement the AsipClass.
- An AsipClass is composed of one or more *services*, implementing an AsipServiceClass. Example of services: a distance sensor, servo motor, temperature sensors

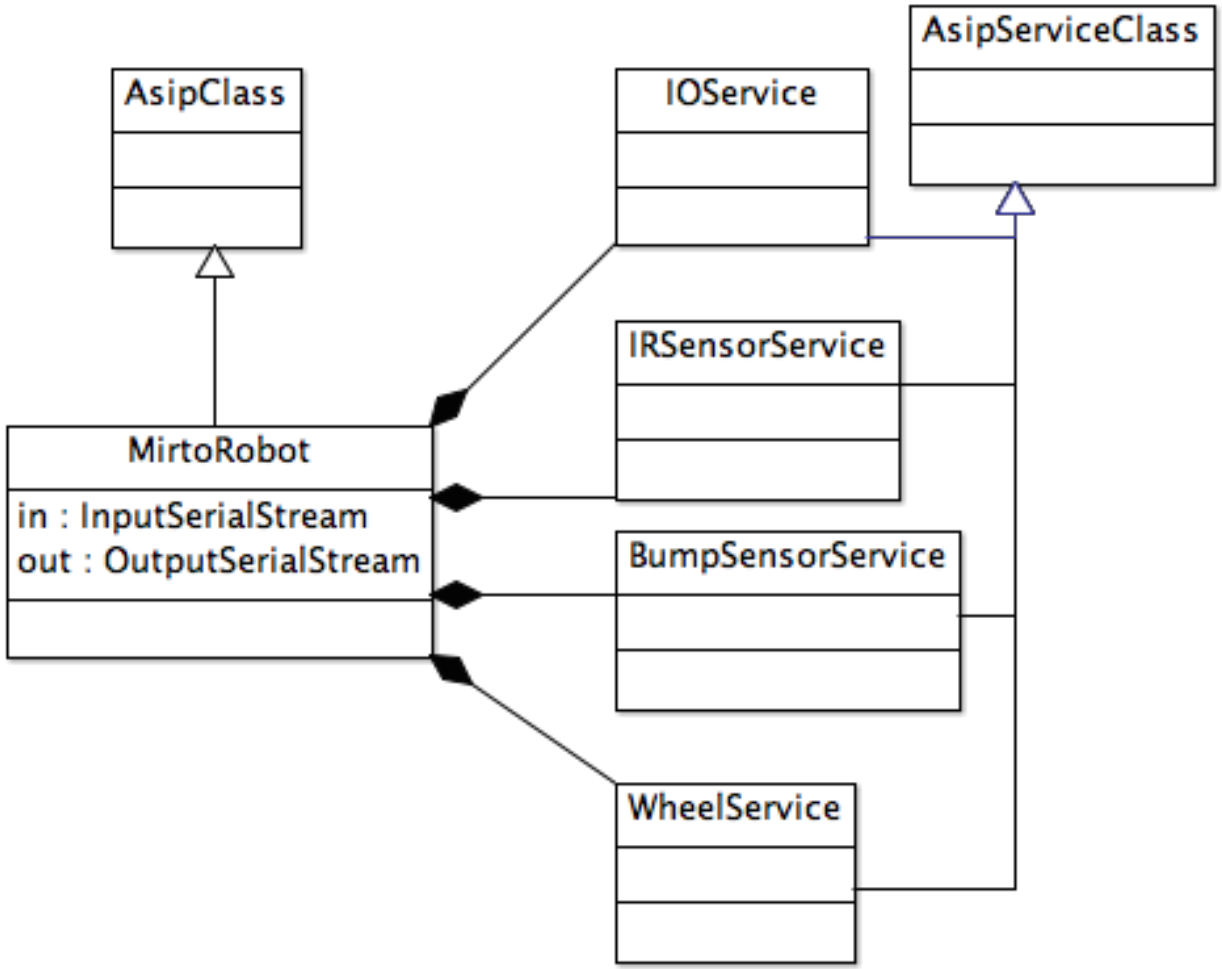


The main AsipClass loop

- The AsipClass on the microcontroller is connected to a stream (serial, recently TCP and MQTT as well).
- Messages are received and sent over this stream.
- The AsipClass dispatches incoming messages to the appropriate service.
- Structure of a message: `serviceID,operation,listOfValues`
  Example: `I,d,13,1`
- See `http://github.com/michaelmargolis/asip` for formal syntax of messages.
- We provide a number of service, but additional ones could be defined by implementing an AsipServiceClass to handle appropriate messages.

Example: uploading ASIP to a microcontroller for a Robot

Example: UML class diagram for the robot



Example part 2: using a client (Java)

```
1  SimpleTCPBoard board1 =
2      new SimpleTCPBoard("192.168.0.2");
3  SimpleTCPBoard board2 =
4      new SimpleTCPBoard("192.168.0.3");
5  // [...] SETUP HERE [...]
6  while (true) {
7    buttonState = board1.digitalRead(buttonPin);
8    if ((buttonState == HIGH) {
9      board2.digitalWrite(ledPin, AsipClient.HIGH);
10   } else) {
11     board2.digitalWrite(ledPin, AsipClient.LOW);
12   }
13 }
```

Example 3: coordination

https://www.youtube.com/watch?v=TVZpqKEnv4I

In summary

- An abstraction for resource-limited microcontrollers that enables faster application of CPS by composing *services*.
- Open source implementation of core ASIP and a number of services.
- Clients in Java, Python, Racket, Erlang.
- Serial communication and experimental TCP sockets and MQTT (pub/sub) messaging.
- Enables model-based and model-to-code development of CPS.
- Allows dynamic reconfiguration of complex applications.
- Currently working on model-to-code transformation to support certification activities.

Thanks! Questions?

- Arduino ASIP available at `https://github.com/michaelmargolis/asip`
- Clients:
  - Java: `https://github.com/fraimondi/java-asip`
  - Python: `https://github.com/gbarbon/python-asip`
  - Racket: `https://github.com/fraimondi/racket-asip`
  - Erlang: `http://github.com/ngorogiannis/erlang-asip`
- Examples and tutorials available at `http://www.rmnd.net`
- Please contact us at `http://mase.cs.mdx.ac.uk` for queries and additional information.