Use of Sensors and SPARQL in Smart Room

Rustam I. Kadirov, Kirill A. Ustimov, Dmitry G. Korzun

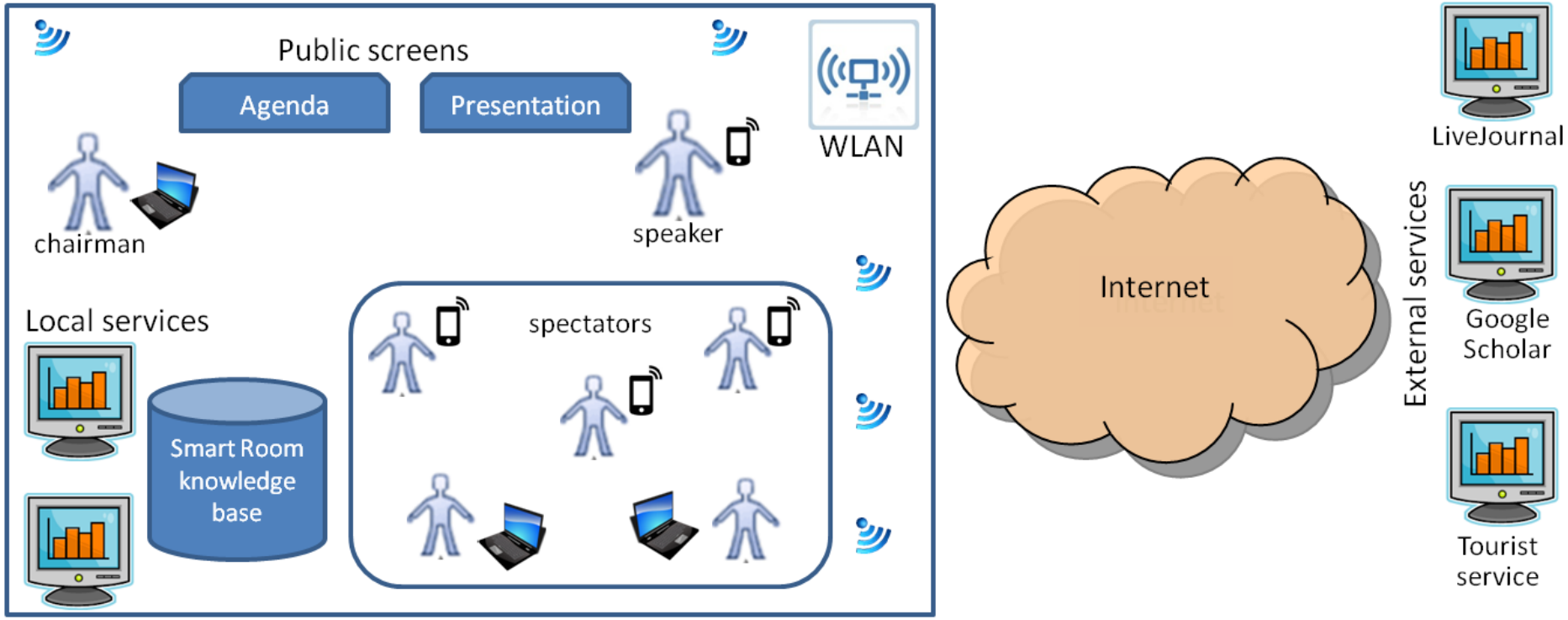
Petrozavodsk State University
Department of Computer Science

This demo was supported by grant KA179 “Complex development of regional cooperation in the field of open ICT innovations” of Karelia ENPI CBC programme 2007–2013 of the European Union, the Russian Federation and the Republic of Finland

13th FRUCT conference

April 25, Petrozavodsk, Russia

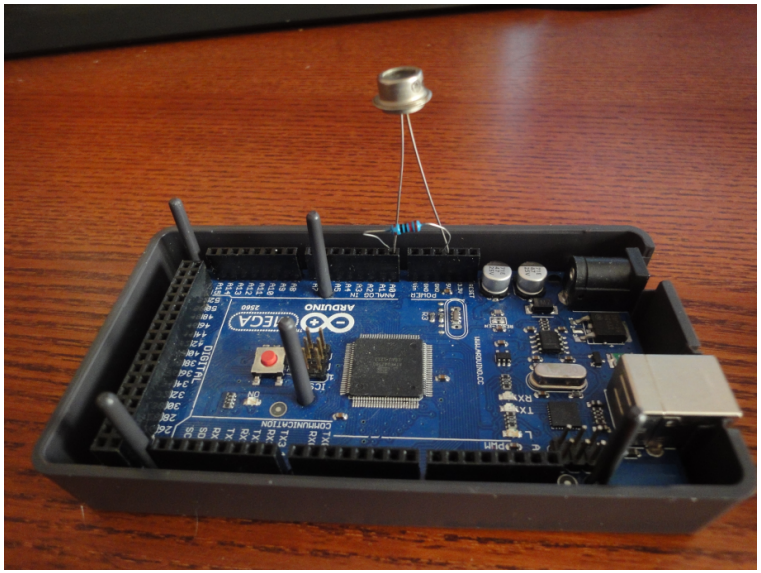
Smart Room description



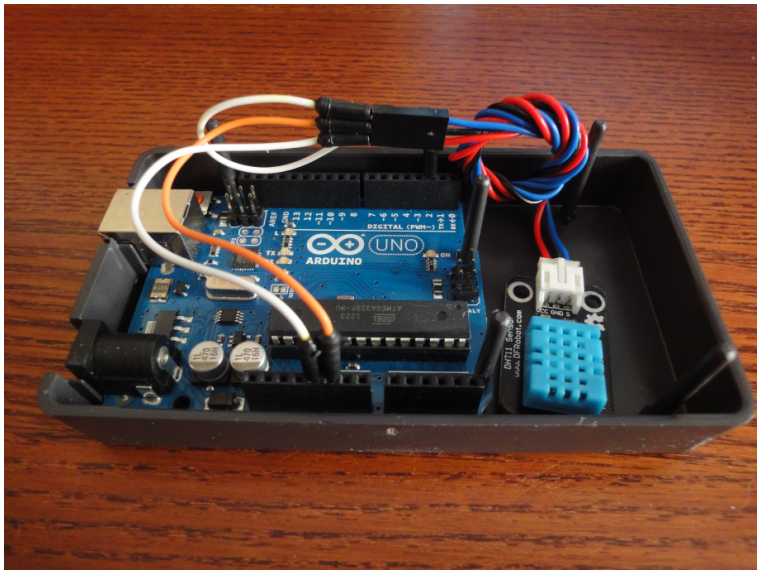
Sensors in Smart Room

- Lighting sensor (FR-764)
- Humidity (DHT-11)
- Temperature (DHT-11)

Lighting sensor:



Temperature and humidity sensor:



Sensor scenarios

- Control of lighting, temperature and humidity in room
- Get the big volume of sensor data and analyse it (detect noise)
- Analyse user activity

RDF data query language SPARQL

- Powerful query language for RDF (triplestore)
- Produced by the W3C RDF Data Access Working Group
- Latest recomendation: 21 March 2013 (SPARQL 1.1)
- Different query forms (select, construct, describe, ask)
- Modifiers (order by) and operators like filter, regex, count and more
- Similar to SQL
- Implemented in Redland SIB

Different types of querying

Triplet templates (with wildcards) in C KPI:

```
ss_add_triple(&req_triple, SUBJECT
    , PREDICATE, SS_RDF_SIB_ANY (
    means any object),
    SS_RDF_TYPE_URI,
    SS_RDF_TYPE_URI);
ss_query(&info, req_triple, &
    result_triples);
```

SPARQL:

```
ss_sparql_construct_query(&info, "
    CONSTRUCT {<SUBJECT> <
    PREDICATE> ?obj} WHERE {<
    SUBJECT> <PREDICATE> ?obj}", &
    results);
or ss_sparql_select_query(&info, "
    SELECT ?obj WHERE {<SUBJECT> <
    PREDICATE> ?obj}", &results, &
    number_of_bindings);
```

Advanced SPARQL query

Triplet templates (with wildcards)

```
ss_add_triple(&req_triple,
    SS_RDF_SIB_ANY, rdf:type, "http
    ://xmlns.com/foaf/0.1/Person",
    SS_RDF_TYPE_URI,
    SS_RDF_TYPE_URI);
ss_query(&info, req_triple, &
    result_triples);
...clear reg_triple, init k = 0 ...
for each (result_triples[i]) {
ss_add_triple(&req_triple,
    result_triples[i]->subject, "
    http://xmlns.com/foaf/0.1/
    status", SS_RDF_SIB_ANY,
    SS_RDF_TYPE_URI,
    SS_RDF_TYPE_LIT);
ss_query(&info, req_triple, &
    result_triples);
if (strcmp(result_triple[0]->object
    , "online") == 0){k++;}
... clear reg_triple...}
```

SPARQL

```
ss_sparql_select_query(&info, "
    SELECT (COUNT(?status) as ?
    onlinePersonCount)
WHERE { ?person rdf:type <http://
    xmlns.com/foaf/0.1/Person>.
       ?person <http://xmlns.com/
           foaf/0.1/status> ?status
           .
       filter regex((?status), "
           online")}", &results, &
           number_of_bindings);
```

Sensor measuring and publishing

```
['Lighting', ' 73\r\n']
Temperature: 25.00

====================
2013-04-22 08:54:37.317499
Old data has been removed
Lighting: 73

['Lighting', ' 73\r\n']
Humidity: 33.00

====================
2013-04-22 08:54:39.319069
Old data has been removed
Lighting: 73

['Lighting', ' 73\r\n']
Temperature: 25.00

====================
2013-04-22 08:54:41.321111
Old data has been removed
Lighting: 75

['Lighting', ' 75\r\n']
Humidity: 33.00

====================
2013-04-22 08:54:43.324291
Old data has been removed
Lighting: 74

['Lighting', ' 74\r\n']
Temperature: 25.00

====================
2013-04-22 08:54:45.325021
Old data has been removed
Lighting: 55
```

```
====================
2013-04-20 16:36:32.257731
Old data has been removed
Traceback (most recent call last):
  File "test_serial_read.py", line 106, in <module>
    first_usb_data = first_usb_port.readline()
  File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 456, in read
    raise SerialException('device reports readiness to read but returned no data (device disconnect
serial.serialutil.SerialException: device reports readiness to read but returned no data (device di
buti@buti-AOD257:~/repositories/git/smartroom/services/local-services/sensors/lighting-sens
or$ cd
buti@buti-AOD257:~$ ssls X::10010
Smart space list $Id: ssls.py,v 1.39 2010/07/23 17:04:03 vluukkal Exp $
['X'] 0 > ls
ns_1:hasMeasurementProperty,rdf:type,rdf:Property
ns_2:Temperature-sensor,ns_1:hasMeasurementProperty,"Temperature"
ns_2:Temperature-sensor,rdf:type,ns_1:Sensor
ns_2:Lighting-sensor,ns_1:hasMeasurementProperty,"Lighting"
ns_2:Lighting-sensor,rdf:type,ns_1:Sensor
ns_2:Lighting-sensor,ns_1:hasValue,"69"
ns_2:Humidity-sensor,ns_1:hasMeasurementProperty,"Humidity"
ns_2:Humidity-sensor,rdf:type,ns_1:Sensor
ns_2:Humidity-sensor,ns_1:hasValue,"33.00"
ns_1:hasValue,rdf:type,rdf:Property
ns_1:Sensor,rdf:type,rdfs:Class
['X'] 0 > l
```

Sensor data consumer

Sensor information consumer

Sensor emulator

Controller

```
kirill@kirill-VirtualBox:~/HelloSensors$ ./controller
KP join to SS. Press ctrl+C to exit.
Sensor http://example.org/#TemperatureSensor hasValue 87
Sensor http://example.org/#LightingSensor hasValue 87
Sensor http://example.org/#TemperatureSensor has new value 3
Sensor http://example.org/#TemperatureSensor old value 87
Sensor http://example.org/#LightingSensor has new value 3
Sensor http://example.org/#LightingSensor old value 87

Sensor http://example.org/#TemperatureSensor has new value 30
Sensor http://example.org/#TemperatureSensor old value 3
Sensor http://example.org/#LightingSensor has new value 30
Sensor http://example.org/#LightingSensor old value 3

Sensor http://example.org/#TemperatureSensor has new value 83
Sensor http://example.org/#TemperatureSensor old value 30
Sensor http://example.org/#LightingSensor has new value 83
Sensor http://example.org/#LightingSensor old value 30
```

Sensor emulator

```
kirill@kirill-VirtualBox:~/HelloSensors$ ./sensor
KP join to SS. Press ctrl+C to exit.
Publishing values from these sensors:
1.TemperatureSensor (status: on)
2.LightingSensor (status: on)
Enter number of sensor to start or stop publish:
```

Demo showing

- Interacts with Smart Room demo
- Consists of three sensors (humidty, temperature, lighting)
- Uses the Smart Room ontology