Ubiquitous and Secure Networks and Services

Redes y Servicios Ubicuos y Seguros

Unit 6: SunSpot Development Platform

Pedro Castillejo Parrilla
pcastillejo@diatel.upm.es

UNIT 6: SunSpot Development Platform

SUNSPOT TECHNICAL SPECIFICATIONS

Pedro Castillejo Parrilla

SunSpot

Sun

Small
Programmable
Object
Technology

SunSpot Hardware

Hardware

- **Small size package**
- **Modular architecture**
  - Stackable boards
  - Up to 3 floors
- **Power**
  - Li-Ion Battery (nodes)
  - USB power (gateway/sink)

SunSpot Hardware

- **Processor**
  - ARM 920T CPU (180MHz 32-bit)
- **Memory**
  - 512Kb RAM, 4Mb FLASH
- **Network**
  - Chipcon 2420 radio with integrated antenna
  - IEEE 802.15.4 @ 2.4GHz
- **Data**
  - USB interface— mini-b connector
- **Power supply**
  - 3.6V rechargeable 750 mAh Li-Ion battery
  - Normal power consumption: 40-100mA
  - Deep sleep mode consumption: 36 µA

SunSpot Hardware

eDemo Sensor Board

- 2G/6G 3-axis accelerometer
- Light and temperature sensors
- 8 RGB LEDs
- 6 analog inputs (0-3Volts)
- 2 switches
- 5 general purpose I/O pins
- 4 high current output pins

SunSpot Hardware

External Interfaces

- Digital Lines
  - 4x Input/Output
- Analog
  - 10 bit ADC
  - 6 input lines
  - Range: 0-3 Volts

SunSpot Hardware

Add-ons

- Gyroscope (1x for 2D and 2x for 3D resolutions)
- Data Glove (gaming, Virtual-Reality, …)
- Game-Pad
- Compass
- Servo motors/controllers
- Voice Synth.

SunSpot Software

Squawk VM

- No underlying OS
- Base code written in Java
- Interpreter and low level I/O code written in C
- Application development in a Java ME (CLDC 1.1) environment
- Libraries to manage basic elements (sensors, leds, switches, …) already codified

SunSpot Software

Isolates

- Single node-Single VM multiple apps running allowed
- Isolated from each other
- Asynchronous
- No system down if one app crash
- Ideal for security applications: several isolates for individual application needs (secured or unsecured).

SunSpot Software

Sun SPOT SDK Libraries

- Both SunSpot nodes and desktop apps run over Squawk Java VM
- Several Libraries already implemented
  - Java ME CLDC 1.1 libraries
  - Desktop libraries (Basestation, Host Apps)
  - Hardware management
    - Demo sensor board library
    - Radio and network libraries

SunSpot Software

Versions

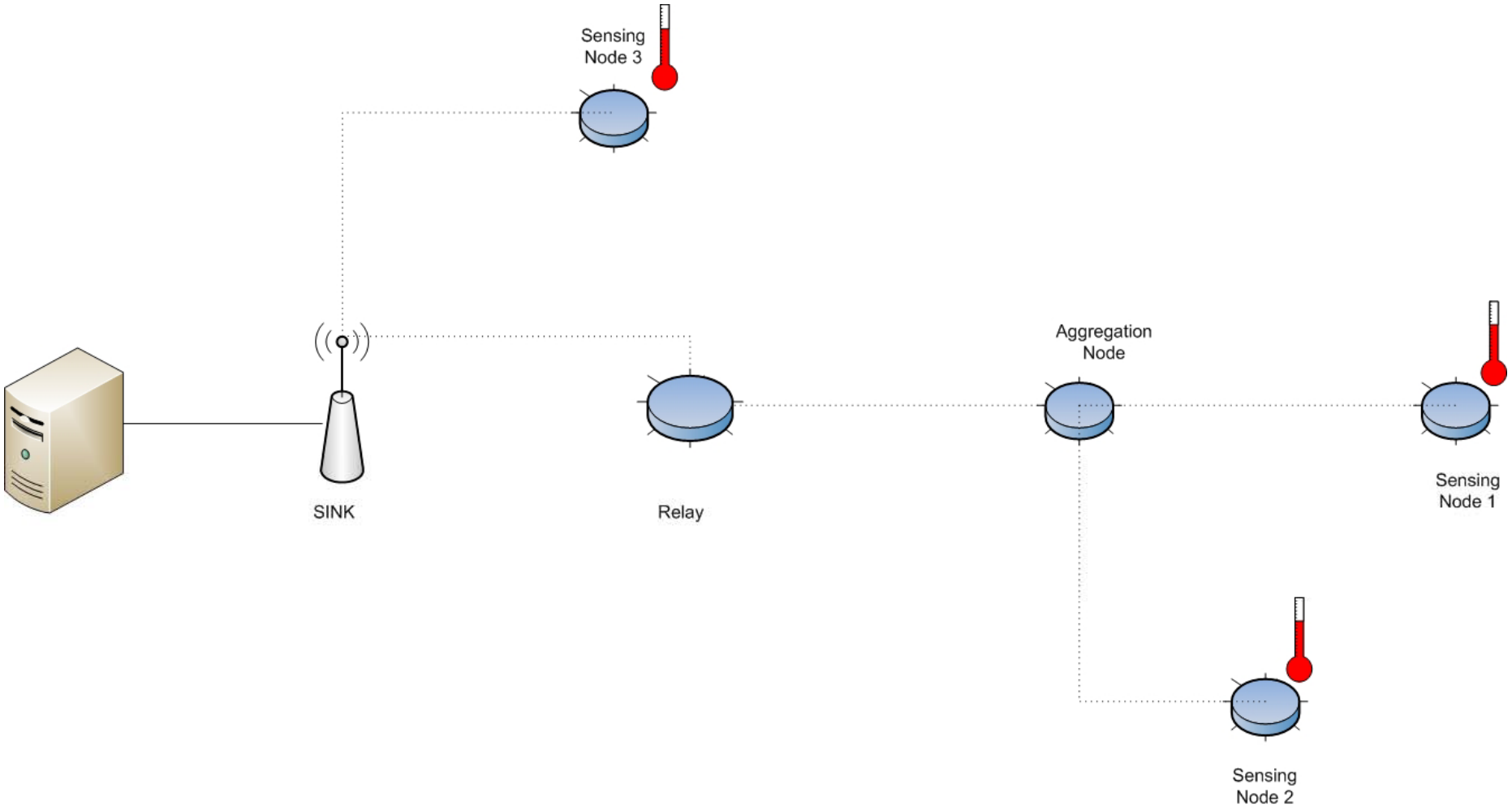
- v1.0 (Green)
- v2.0 (Orange)
- v3.0 (Purple)
- v4.0 (Blue)
- v5.0 (Red)
- V6.0 (Yellow)

UNIT 6: SunSpot Development Platform

INTRODUCTION TO SUNSPOT NETWORKS

Pedro Castillejo Parrilla

Network Topologies



Routing

- **Link Quality Routing Protocol (LQRP)** algorithm is used by default to determine the best route, sending RREQs when necessary:
  - RREQ: requests for a route to a particular target SPOT that are broadcast by a requester, and then re-broadcast by each Sun SPOT that receives them. Each Sun SPOT that knows how to route to the requested target sends a reply back to the requester. The route that will be used is the one with the best link.
- Routing Policies available:
  - ALWAYS: node will respond to and route RREQ and pass packets for other nodes. In order to guarantee that node will be always available for routing, deep sleep is disabled
  - IFAWAKE: similar to ALWAYS, but deep sleep is not specifically handled, so if a deep sleep is performed by the applications, the node may stop participating in the routing algorithm
  - ENDNODE: the node will not repeat RREQs to others or process packets for other nodes unless it is either the ultimate sender or destination
- Ad-hoc On-Demand Distance Vector (AODV) algorithm can also be used, instead of LQRP.

Addressing

- Every node is identified by its unique MAC Address
- IEEE 802.15.4 MAC layer is used, with 64 bits addresses
  - Binding→ address:port Example: 0012.2CB4.A331.1DE9:77
- Ports management:
  - 0-31 reserved
  - 32-255 available
  - Each node can manage several connections on the same port, to different destinations
- IPv6 addressing is also implemented

Radio Protocols

- Two protocols available, implemented on top of the 802.15.4 MAC layer
- Radiostream protocol
  - Stream-based communication
  - Reliable and buffered
- Radiogram protocol
  - Datagram-based communication
  - Sequence of packages and delivery and not repetition guarantee is NOT provided

- Examples:

```
RadiostreamConnection
    conn=(RadiostreamConnection)Connector.open("radiostream://<destAddr>:<portNo>”)
RadiogramConnection conn=(RadiogramConnection)Connector.open("radiogram://<destAddr>:<portNo>”)
```

Radio Protocols

Radiostream Example

- **Node A**

```
RadiostreamConnection conn =
   (RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.0006:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();
try {
    dos.writeUTF("Hello up there");
    dos.flush();
    System.out.println ("Answer was: " + dis.readUTF());
} catch (NoRouteException e) {
    System.out.println ("No route to 0014.4F01.0000.0006");
} finally {
    dis.close();
    dos.close();
    conn.close();
}
```

Radio Protocols

Radiostream Example

- **Node B**

```
RadiostreamConnection conn =
    (RadiostreamConnection)Connector.open("radiostream://0014.4F01.0000.0007:100");
DataInputStream dis = conn.openDataInputStream();
DataOutputStream dos = conn.openDataOutputStream();
try {
     String question = dis.readUTF();
     if (question.equals("Hello up there")) {
     dos.writeUTF("Hello down there");
} else {
    dos.writeUTF("What???");
}
    dos.flush();
} catch (NoRouteException e) {
    System.out.println ("No route to 0014.4F01.0000.0007");
} finally {
     dis.close();
     dos.close();
     conn.close();
}
```

Radio Protocols

Radiostream Example (Server End)

```
RadiogramConnection conn = (RadiogramConnection)Connector.open("radiogram://:100");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
Datagram dgreply = conn.newDatagram(conn.getMaximumLength());
try {
    conn.receive(dg);
    String question = dg.readUTF();
    dgreply.reset(); // reset stream pointer
    dgreply.setAddress(dg); // copy reply address from input
    if (question.equals("Hello up there")) {
    dgreply.writeUTF("Hello down there");
    } else {
        dgreply.writeUTF("What???");}
   conn.send(dgreply);
} catch (NoRouteException e) {
   System.out.println ("No route to " + dgreply.getAddress());
} finally {
   conn.close();
}
```

Radio Protocols

Radiostream Example (Client End)

```
RadiogramConnection conn =
   (RadiogramConnection)Connector.open("radiogram://0014.4F01.000
   0.0006:100");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
try {
    dg.writeUTF("Hello up there");
    conn.send(dg);
    conn.receive(dg);
    System.out.println ("Received: " + dg.readUTF());
} catch (NoRouteException e) {
   System.out.println ("No route to 0014.4F01.0000.0006");
} finally {
   conn.close();
}
```

Broadcasting

- Broadcasting is allowed, with some restrictions:
  - By default, broadcasts are transmitted over two hops (inside the PAN).
    - Can be changed to *n* hops using ((RadiogramConnection)conn).setMaxBroadcastHops(n);
  - Broadcast is not inter-PAN.
  - Broadcasted datagrams might not be delivered.
  - Broadcast connections cannot be used to receive. Open a server connection for receiving replies to a broadcast.
- Opening a broadcasting radiogram connection:

```
DatagramConnection conn =
   (DatagramConnection)Connector.open("radiogram://broadcast:<portnum>");
```

Signal Strength Measures

- At the reception side of a datagram, using RADIOGRAM connections, several measures regarding radio signal quality can be obtained.
  - RSSI (received signal strength indicator) measures the strength of the signal for the packet received, in a range between +60 (strong) to -60 (weak). Use getRssi() method.
  - CORR measures the average correlation value of the first 4 bytes of the packet header, in a range between 110 (maximum quality packet) to 50 (lowest quality packet). Use getCorr() method.
  - Link Quality Indication (LQI) is a characterization of the quality of a received packet, calculated using the correlation value. The LQI ranges from 0 (bad) to 255 (good). Use getLinkQuality() method.
- Radio signal measures can be used to a wide variety of applications: localization, tracking, monitoring, …

HTTP Protocol Support

- Any SPOT can open a http connection to any remote host or web service (via an Internet connected host computer), using the implemented http protocol stack.
- Opening connections:

```
HttpConnection connection
   =(HttpConnection)Connector.open("http://host:[port]/filepath");
```

HTTP example:

```
HttpConnection connection =
   (HttpConnection)Connector.open("http://www.sunspotworld.com/");
connection.setRequestProperty("Connection", "close");
InputStream in = connection.openInputStream();
StringBuffer buf = new StringBuffer();
int ch;
while ((ch = in.read()) > 0) {
   buf.append((char)ch);
}
System.out.println(buf.toString());
in.close();
connection.close();
```

SunSpot Programmer´s manual.
http://www.sunspotworld.com/docs/Yellow/SunSPOT-Programmers-Manual.pdf

References

- [Spw] www.sunspotworld.com
- [Squawk] www.squawk.dev.java.net
- [Sapiy] SunSpot API, Yellow Version available at:
  - www.sunspotworld.com/docs/Yellow/javadoc/index.html
- [Susp] SunSpot Programmer´s manual:
  - http://www.sunspotworld.com/docs/Yellow/SunSPOT-Programmers-Manual.pdf
- [Sapi] SunSpot API available at:
  - www.sunspotworld.com/docs/