Homework 5 - Flexible Embedded Architectures

Developing Java ME Embedded Applications by Using a Raspberry Pi: Homework for Lesson 5

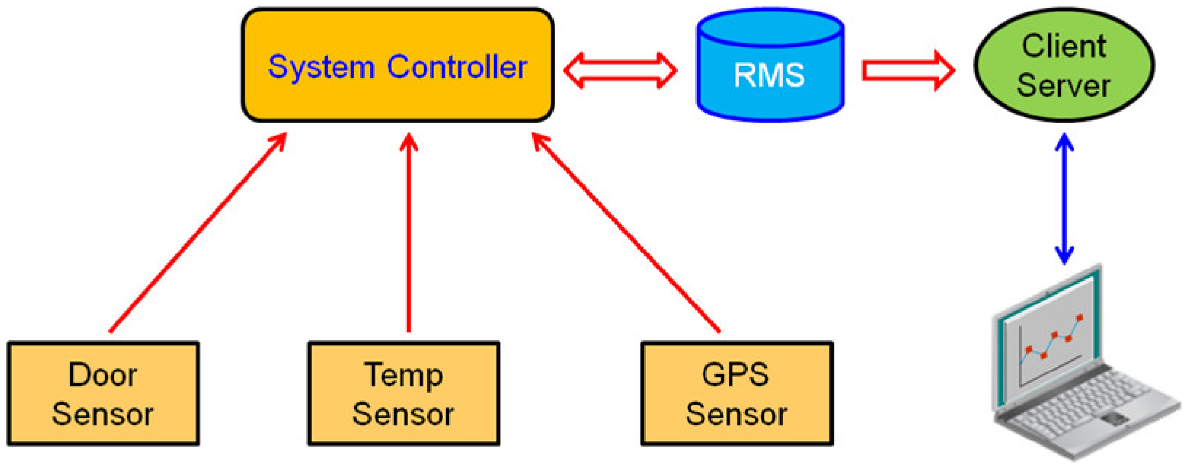
Assumptions

- You have successfully completed the homework in lessons 2 - 4
- You have a wired and functionally complete breadboard as shown in lesson 4.

Part 1 - Run the GPSDataIMC Project

- Download and unzip the GPSDataIMC.zip file
- Open the project GPSDataIMC - this is the project Simon describes in the video.
- Make sure that you have the AMS running on the Raspberry Pi
- Run the project
    - Select GPSReaderMIDlet from the list of IMlets to run
    - Note that the MIDlets RMSPersistMIDlet and FilePersistMIDLet are started automatically.
    - These MIDlets are registered as PushRegistry clients - when the GPSReaderMIDlet connects to each of them in turn, they start.
- Run the ReaderMidlet to provide that the GPS data was in fact saved to the RMS
- Look in the /tmp directory on your Raspberry Pi to see that a file gps-data.txt has also been created.

Part 2 - Using IMC to create distinct sensor MIDlets



Overall Design Concept

- This project illustrates a potential design for the prototype embedded sensor for the container problem. Rather than one monolithic suite, each sensor is a separate MIDlet suite. This design allows for in-the field updates to each sensor as required.
- The SystemController is designed to respond to asynchronously sent "events" from any number of sensors, currently, three - the door, temperature and GPS.
- Data from the senors can come in at any time. Typically, temperature and GPS data are recorded at regular intervals. The actual interval is determined through a property set in the JAD for each project, but defaults to 30 second intervals.
- The sensor for the door only generates an event when the door is opened (button pressed), and only after the door is "opened" for a specific period of time. This approach prevents spurious events - for example, when the container is traveling on a particularly bumpy section of road, or if the door is temporarily opened and then closed. By default, a door open event is logged when the door is open for more than 5 seconds.
- Events are sent via IMC as message strings to the SystemController MIDlet (using the message format discussed in lesson 3 and 4). The SystemController records the messages in the RMS - however, this could be extended to include the file system.The SystemController makes no attempt to process the messages, it simply records them - this pushes some of the intelligence down to the sensor code.
- The SystemController also provides a client server MIDlet, which allows a client to read the data stored in the RMS, and the ReaderMidlet, a MIDlet that simply prints the stored RMS records to the system console.

DoorSensor

- This code is altered from Angela's original code for the DoorProject.
- Instead of having the button pressed indicate a door closed event, the logic has been swapped so that a button press is a door open event.
- When the button is held down for more than 5 seconds, an asynchronous door event is sent to the controller and recorded in the RMS.

TempSensor and GPSSensor

- These projects utilize the code from the previous week's projects.
- The sensor will send an event to the System Controller periodically.
    - The default value for Temperature and GPS position and velocity is every 30 seconds, however, the default can be overriden through an attribute:
        - The TempSensorMIDlet reads the TempReadFreq application property to determine the frequency of temperature reads.
        - The GPSSensorMIDlet reads the GPSReadFreq application property to determine the frequency of GPS reads.

Preparation:

- By default, the AMS on the Raspberry Pi limits the number of MIDlets that can run simultaneously.
- Quit the AMS (if running) on the Pi.
- Edit the `jwc_properties.ini` file on the Pi, and change the value of MAX_ISOLATES to 8.
- Restart the AMS.
- Be sure that you remove ALL other suites/MIDlets from the AMS/Device Emulator before starting this exercise.

Tasks:

- Unzip the four projects from Lesson5Homework.zip
    - Run the projects from NetBeans directly.
    - Run the SystemController suite first
        - Start the SystemController IMlet

    - Run the DoorSensor project
    - Run the TempSensor project
    - Run the GPSSensor project

- Note that the temperature and GPS events are not sent to the SystemController, they are simply written to standard out (System console).
- Write an implementation of GPSSensorClient:
    - This class extends ServiceClient
    - Implement a constructor to take the String service name and invoke ServiceClient's constructor
    - Implement the sendData method to send the GPS position and velocity data to the System Controller. Be sure to use the message format discussed in lesson 4.
    - Hint: Look at DoorSensorClient for ideas on how to implement this code.

- Modify GPSSensorTask to create an instance of GPSSensorClient and invoke the sendData method with the position and velocity strings.
- Write an implementation of TempSensorClient
    - This class extends ServiceClient
    - Implement a constructor to take the String service name and invoke ServiceClient's constructor
    - Implement the sendData method to send temperature data to the System Controller using the proper message format.

- Modify TempSensorTask to create an instance of TempSensorClient and invoke the sendData method with the temperature string

Testing

- Run the SystemController suite first.
    - Start the SystemController IMlet

- Run the DoorSensor
- Run the TempSensor
- Run the GPSSensor
- If implemented properly, you should start seeing messages indicating that records are being stored in the RMS.
- Run the ReaderMidlet to dump the contents of the RMS to the console.
    - Note: You may see a MIDletSuiteLockedException - this is a known bug in EA2.

- Run ServerMidlet.
- Run VisualClient to connect to the Server and get the current temperature and GPS results.

Troubleshooting

- You may find that you cannot remove a Midlet suite. Follow these steps to clear the state of the AMS and the Device Emulator:
    - Quit the AMS on the Pi.
    - Run sudo ./listMidlets.sh to get a list of suites installed on the Pi.
    - Using the number of the suite, for example:

```
Suite: 15
  Name: SystemController
  Version: 1.0
  Vendor: oracle
  MIDlets:
    SystemController: systemcontroller.SystemControlMIDlet
    ReaderMidlet: rmsreader.ReaderMidlet
    ServerMidlet: server.ServerMidlet
```

- Run sudo ./removeMidlet.sh <n> for each installed suite, where <n> is the suite number returned from listMidlets.
- Quit and restart the Device Emulator
- Restart the AMS.

Part 3 - Using the SWM to load and start the MIDlet suites

- Download and unzip SystemController2.zip.
- Open the project in NetBeans
- Look at the Attributes for the suite (Properties > Application Descriptor)
    - Note that there are 3 custom attributes InstallSuite-<n> that contain a file URI pointing to JAD files on the local filesystem on the Pi. The URI could also have been an http server, socket server - the file URI is just the simplest way to get the files loaded by the SWM.
    - The number indicates the order in which the suites will be loaded and started.

- Create a JAD and JAR file for each of the three project: DoorSensor, TempSensor and GPSSensor:
    - Right-click each project and select Clean and Build

- Copy the JAD and JAR file from each project to a temp directory
    - The JAD and JAR files are in the dist directory of each project

- Zip the 6 files (3 JAD file and 3 JAR files) into a single zip archive.
- FTP the zip archive to the /home/pi directory on the Pi (Use psftp to put the zip file onto the Pi).
- On the Pi, unzip the archive.
    - You should have three JAD files and 3 JAR files in /home/pi
    - Be sure the files are in the /home/pi directory! Alternatively, modify the file URI's to point to the location of the JAD files.

- Run SystemController2 Project from NetBeans.
    - Choose SystemController from the Select IMlets to run dialog
    - The SystemController will load and start the DoorSensor, TempSensor and GPSSensor projects (in that order)
    - Again, you will see a com.sun.midp.midletsuite.MIDletSuiteLockedException thrown for each of the MIDlet's loaded and started.

- To stop the application, stop the SystemController MIDlet, which will remove the SystemController suite.
    - Note that the suites loaded by the SystemController are left behind. Select these and click Remove

Part 4 - Auto-restart of applications

- The TaskManager can also detect state changes in a MIDlet (task).
- In another PuTTY window, modify the JAD file for TempSensor to add an attribute:
    - In the /home/pi directory, type:
      sudo nano TempSensor.jad
    - Add a line at the bottom of the file:
      AutoStart: true
    - Save the changes and close the JAD file

- Re-run the SystemController2 application
- After TempSensorMIDlet starts running, attempt to stop it using the Device Emulator (select TempSensorMIDlet and click the Stop button).
    - Note that while the MIDlet stops for a brief moment - it automatically restarts.

- If you wish - change the other two JAD files as well.

Going Further - Additional Ideas (not yet implemented)

- Ideally, a door event should cause the SystemController to begin monitoring the temperature and record GPS locations more frequently, because a door event in transit is likely to create temperature changes (particularly in refrigerated containers.)