Data replication in mobile computing

Bachelor’s Thesis in Electrical Engineering

Rodrigo Christovam Pamplona
Details

Name: Rodrigo Christovam Pamplona
University: Halmstad University, Sweden
Degree Program: Electrical Engineering
Title of Thesis: Data replication in mobile computing.
Academic Supervisors: Edison Pignaton de Freitas
               Wagner Ourique de Morais
ABSTRACT

With the advances of technology and the popularization of mobile devices, the need of researching and discussing subjects related to mobile devices has raised. One of the subjects that needs to be further analyzed is data replication.

This study investigates data replication on mobile devices focusing on power consumption. It presents four different scenarios that propose, describe, apply and evaluate data replication mechanisms, with the purpose of finding the best scenario that presents less energy consumption.

In order to make the experiments, Sun SPOT was chosen as a mobile device. This device is fully programmed in a java environment. A different software was created in each scenario in order to verify the performance of the mobile devices regarding energy saving.

The results found did not meet the expectations. While trying to find the best scenario a hardware limitation was found. Although software can be easily changed to fix errors, hardware cannot be changed as easily. The implications for the hardware limitation found in this study prevented the results to be optimal. The results found also imply that new hardware should be used in further experimentation. As this study proved to be limited, it suggests that additional studies should be carried out applying the new version of the hardware used in this study.
LIST OF FIGURES

Figure 1: Simple scenario schematic .......................................................... 2
Figure 2: Sun SPOT .................................................................................. 6
Figure 3: Solarium – Sun SPOT emulator .................................................. 8
Figure 4: Always on .................................................................................. 14
Figure 5: Timer activated ........................................................................ 15
Figure 6: Motion activated ....................................................................... 16
Figure 7: Motion activated with timer ....................................................... 17
Figure 8: Base station software ............................................................... 19
Figure 9: Battery drain for scenario 1 first implementation ...................... 21
Figure 10: Battery drain for scenario 2 first implementation ................. 22
Figure 11: Battery drain for scenario 3 first implementation ................... 23
Figure 12: Battery drain for scenario 4 first implementation ................... 24
Figure 13: Battery drain for scenario comparison at first implementation .... 25
Figure 14: Battery drain for scenario 1 second implementation ............. 26
Figure 15: Battery drain for scenario 2 second implementation ............ 27
Figure 16: Battery drain for scenario 3 second implementation .......... 28
Figure 17: Battery drain for scenario 4 second implementation .......... 29
Figure 18: Battery drain for scenario comparison at second implementation 30
LIST OF LISTINGS

Listings 1: Accelerometer classes ............................................................................. 8
Listings 2: Measure acceleration ................................................................................ 9
Listings 3: Radio class ................................................................................................. 9
Listings 4: Radio receive packages ............................................................................ 9
Listings 5: Radio send packages .................................................................................. 9
Listings 6: Turn off radio ............................................................................................. 10
Listings 7: Battery class ............................................................................................... 10
Listings 8: Battery reading .......................................................................................... 10
Listings 9: Read and write file on base station ............................................................. 11
Listings 10: Files classes ............................................................................................ 11
Listings 11: Reading and writing file on Sun SPOT .................................................... 11
Listings 12: Get timestamp and battery on SPOT ....................................................... 11
LIST OF TABLES

Table 1: Data example ................................................................................................. 12
TABLE OF CONTENT

ABSTRACT .................................................................................................................................................. III
LIST OF FIGURES......................................................................................................................................... IV
LIST OF LISTINGS......................................................................................................................................... V
LIST OF TABLES.......................................................................................................................................... VI
TABLE OF CONTENT ................................................................................................................................. VII
1 INTRODUCTION ......................................................................................................................................... 1
   1.1 CONTEXT DESCRIPTION .................................................................................................................... 1
   1.2 GOALS .................................................................................................................................................. 3
   1.3 WORK OUTLINE ................................................................................................................................. 3
2 RELATED WORK ......................................................................................................................................... 4
3 METHODS AND TOOLS............................................................................................................................. 5
   3.1 EXPERIMENTATION ............................................................................................................................ 5
   3.2 SUN SPOT ............................................................................................................................................ 5
   3.3 SUN SPOT DEVELOPMENT KIT .......................................................................................................... 6
      3.3.1 Configuration ................................................................................................................................ 6
      3.3.2 Environment ................................................................................................................................... 7
      3.3.3 API .................................................................................................................................................. 8
   3.4 DATA ACQUISITION ............................................................................................................................ 11
4 SCENARIOS .............................................................................................................................................. 13
   4.1 SCENARIO 1 – ALWAYS ON .............................................................................................................. 13
      4.1.1 Specification ................................................................................................................................. 13
   4.2 SCENARIO 2 – TIMER ACTIVATED ...................................................................................................... 14
      4.2.1 Specification ................................................................................................................................. 14
   4.3 SCENARIO 3 – MOTION ACTIVATED .................................................................................................... 15
      4.3.1 Specification ................................................................................................................................. 15
4.4 SCENARIO 4 – MOTION ACTIVATED WITH TIMER .................................................. 16

4.4.1 Specification ........................................................................................................ 16

5 IMPLEMENTATION AND RESULT ....................................................................... 18

5.1 BASE STATION ........................................................................................................ 18

5.1.1 Implementation .................................................................................................... 18

5.2 FIRST SUN SPOT IMPLEMENTATION ................................................................. 20

5.2.1 Scenario 1 – Always on ..................................................................................... 20

5.2.2 Scenario 2 – Timer activated .............................................................................. 21

5.2.3 Scenario 3 – Motion activated ............................................................................ 22

5.2.4 Scenario 4 – Motion activated with timer ......................................................... 23

5.2.5 Scenario comparison ......................................................................................... 24

5.3 SECOND SUN SPOT IMPLEMENTATION ............................................................ 25

5.3.1 Scenario 1 – Always on ..................................................................................... 26

5.3.2 Scenario 2 – Timer activated .............................................................................. 26

5.3.3 Scenario 3 – Motion activated ............................................................................ 27

5.3.4 Scenario 4 – Motion activated with timer ......................................................... 28

5.3.5 Scenario comparison ......................................................................................... 29

6 DISCUSSION ............................................................................................................. 31

6.1 PROBLEMS ............................................................................................................ 31

6.1.1 Development software problems ...................................................................... 31

6.1.2 Software problems ............................................................................................ 31

6.1.3 Hardware limitations ......................................................................................... 32

7 CONCLUSION .......................................................................................................... 33

REFERENCES ............................................................................................................ 34
1 INTRODUCTION

Mobile computing devices allow users to have access to information at anytime and anywhere. Mobile devices can also be used as routers in Mobile Ad hoc NETworks (MANET). The challenge then is how to update information carried by the mobile device while maximizing the system lifetime.

This project presents different scenarios in the mobile computing domain. It also applies and evaluates distinct data replication mechanisms in each scenario. At the end, it points out the best technique regarding energy saving is.

In this project, the chosen mobile computing devices are SunSpots. The first step is to carry out a study of these devices and their required development environment. The second step involves the description of possible application scenarios and an investigation of data replication methods used in mobile computing. In the next step, models for data replication are proposed and implemented for the described scenarios. Finally, the results are reported and discussed.

1.1 CONTEXT DESCRIPTION

In past years, the number of mobile devices such as mobile phones, PDA’s, laptop’s, among others, have had such rapid growth that they can be easily found everywhere. With the popularization of these devices, new applications that try to find alternative uses for them have emerged.

A new application for mobile devices is that the users could have access to important data anytime which would be carried by their mobile devices. The issue is finding out how this data will be updated. The main challenge today is regarding how long a mobile device can be used before it needs to be recharged.

In order to study the aspects discussed above, a scenario in which voice recognition software is used in a user’s computer is proposed. This software needs to be trained by each user in every computer that will be utilized by the user. The user wants to have a mobile device that can carry his updated profile from one computer to another.

Since almost everyone carries a mobile phone these days, the logical choice for the mobile device to be used in this context would be to choose a mobile phone. But instead of choosing a mobile phone the choice made was to use a mobile device called Sun SPOT, created by Sun Microsystems.

There are many reasons for choosing the Sun SPOT. One of them is that the development process is easier and it works in a java environment, which is very similar to many of the modern mobile phones, so the solution can be ported to a mobile phone if necessary.

It is important to say that the Sun SPOT should be used only as an experimental platform and not as a commercial product due to its value and the difficulty of finding one for sale. Having chosen the mobile device, one way of sending the information to this device is through data replication.

Data replication is a way of sharing the same information in many devices. In order to do this, the data is copied, which means it is replicated, from one device to another. The definition of data replication comes from databases which are discussed by many authors [5] who believe that there are differences between data replication and data synchronization.
Besides these differences, when working with mobile devices the term synchronization can be used to refer to radio synchronization. To avoid mistakes with the terms, it is assumed that there are no significant differences between data replication and data synchronization, and only the term data replication is used here.

A simple example to illustrate these differences is what happens with the contact information in mobile phones. For instance, if there are 10 contacts saved in a mobile phone’s memory and they are copied onto the phone card, the data is replicated. If there are 10 contacts saved in a mobile phone’s memory, and 5 on the phone card, there will be 15 contacts on the phone memory and 15 on the card when the data is synchronized.

There are different data replication techniques in databases. One of them is database replication, in which a device releases a copy of part of a database or the entire database to another device. Another one is disk storage replication in which two disks save the same information. And there is also the distributed shared memory, in which one device shares a portion of its memory with another, so both can access the same information.

When using data replication in mobile devices, some different techniques can also be used. In some more complex systems, using distributed shared memory can be a good solution. In this study, a technique similar to database replication will be used. Instead of an entire database, only a small portion of the data will be replicated.

When using mobile devices to replicate data, a mobile ad hoc network (MANET) is created, and as every system that uses wireless network, it should be designed in a way to consume as little energy as possible, extending the battery lifetime.

Different scenarios measuring power consumption will be tested in this study. An optimal scenario in which the device replicates the data efficiently in a battery-saving way, allowing the software to be run until the device needs to be recharged, is expected to be found.

A main scenario is used to exemplify the use of the mobile devices. In this scenario, a user moves from one computer to another with the mobile device, and the data should be replicated.

This scenario is presented in Figure 1.

![Figure 1: Simple scenario schematic](image)

The computers illustrated above could be the user’s home desktop, office desktop or laptop computers. Thus, the user carries the device from one place to another and the data replicated.
To study this mechanism of data replication, more detailed scenarios are proposed. All scenarios have detailed specifications on how the mobile device connects to the computer and how they replicate the data.

The proposed scenarios are the following:

In scenario 1, which is called “Always on”, the device is always connected, replicating the data;

Regarding scenario 2, which is called “Timer activated”, the device connects from time to time, and replicates the data;

In scenario 3, referred as “Motion activated”, the device turns on when moving, and replicates the data;

Concerning scenario 4, called “Motion activated with timer”, the device turns on when moving or stops moving, and makes use of a timer to save the battery until the data is replicated.

The first scenario is expected to be the worst case possible. However, it is essential because it is the base for implementing the other scenarios and it is also the reference for the scenario comparison.

In order to run the tests in the scenarios described above, some variables must be fixed, so that the results can be properly evaluated. The chosen variables in this study are:

- Time - the time during which the mobile device works;
- Information size - the same amount of data should be sent in each scenario;
- Battery level - the same battery level must be used every time;
- Periodicity of replication - the data should be replicated at the same time in each scenario.

### 1.2 GOALS

This project aims at proposing different scenarios on data replication in mobile computing and evaluating their performance in order to indicate the best technique considering energy saving, probably those or one that makes less use of the radio.

### 1.3 WORK OUTLINE

This study is organized in seven different chapters. Firstly, in chapter 2 previous studies related to this subject are discussed.

In chapter 3, the description of the methodology used to build and test the scenarios is described.

In chapter 4, the different chosen scenarios are presented.

In chapter 5, the results of this study can be seen.

In chapter 6, the discussion of the results can be found.

Finally, in chapter 7, the conclusions reached throughout the experiment are given.
2 RELATED WORK

Data replication is a widely discussed subject, but most of the efforts on the area are focused on databases [3][4]. Due to the fact that the number of mobile devices has increased greatly in the past recent years; with mobile phones, the popularization of Bluetooth technology, new devices like tablets; the discussion of data replication in mobile devices is increasing, even though there are only a few studies on it.

There are some studies with a context similar to the one described in this study. One of them [5] deals with a mobile device which connects to a computer in order to replicate data, is run by software developed in Java. It works with a database and uses PDAs instead of Sun SPOTs, and its aim is to synchronize data, not replicate it. One issue raised in this study is the use of more than one database. A simple solution is proposed which is to set a “dirty flag” each time the data changes. This can be useful in systems with only one base station and one mobile device, but not with more devices.

Another study which uses a mobile phone [6] suggests that a mobile phone which is connected to a wireless internet uses very complicated algorithms to determine which data should be replicated, which are based on the current date and global position.

What is interesting about this study is the conclusion. It suggests that it is very complicated to handle large amounts of data on mobile devices; therefore, this conclusion should be taken into consideration when designing a mobile system for data replication.

In another study [7], the replication mechanism is broadly discussed. However, this study takes into consideration only the software point of view, thus it should be considered a complementary study.

In another study [8], some protocols for synchronization of PDAs are discussed. It can be considered complementary for this experiment due to its conclusions, which reveal that the important aspects of synchronization would be the amount of data exchanged between two devices during synchronization, the amount of data stored in the mobile device and how complex the synchronization is. There is no best choice for protocol, which indicates that it would be necessary to study each case to determine what the best protocol is.

Data replication is being widely discussed nowadays, but the discussions are mostly about protocols, synchronization, algorithms, etc. Since no other study discussing the system lifetime was found, it is impossible to compare the results obtained here with other studies, making the present study complementary to all the other studies carried out before.
3 METHODS AND TOOLS

3.1 EXPERIMENTATION

In order to achieve the goals of this paper, experimentation was carried out based on the study of the four scenarios previously discussed. These scenarios attempted to reproduce some of the possible situations encountered in data replication in mobile systems. The scenarios were modeled to evaluate power efficiency.

The first scenario proposed in this study was designed to be the worst possible in order to serve as a reference for the other scenarios. This comparison was made analyzing the remaining battery charge of a mobile device after the first test. The scenario with higher battery charge at the end of the test was supposed to be able to remain working for more time, consequently, being the one with the best power efficiency.

Regarding the processes used in order to perform the experiments, some steps were followed. Firstly, the Sun SPOT was chosen to be used as a mobile device and then its hardware was studied. Having acquired basic knowledge about the hardware, the software was studied and subsequently the writing of the codes for each desired scenarios started.

3.2 SUN SPOT

The Sun SPOT is a wireless sensor network developed by Sun Microsystems, which can be used in a wide range of applications, from hobbyists to professional applications. It has a main processor running Java VM “Squawk” that serves as an IEEE 802.15.4 wireless network node. As it can be seen in the documents from Sun Microsystems [11], “The Sun SPOT is designed to be a flexible development platform, capable of hosting widely differing application modules. The Sun SPOT development kit, as supplied, contains two different configurations. One of the configurations includes a demonstration application module, the eDemo board.” [11].

The hardware has two boards, the eSPOT main board, also known as processor board, and the eDemo or sensor board. Besides the boards, the device needs power supply, and the most common are a USB cable and battery, even though other sources can be used.
The processor board contains a main processor, memory, power management circuit, 802.15.4 radio transceiver and antenna, battery connector and daughterboard connector.

The sensor board contains an Atmega88 processor, flash memory, a light sensor, a temperature sensor, an accelerometer, eight tri-color LEDs, two switches and I/O pins.

Each Sun SPOT kit is composed by one base station and two mobile sunspot devices. The base station consists of one main board powered by one USB cable, and the mobile devices consist of one main board connected to one sensor board powered by one battery.

3.3 SUN SPOT DEVELOPMENT KIT

The software that controls the Sun SPOT runs in Java. Therefore, the development of this software requires some setting-up on the computer first.

3.3.1 Configuration

In order to use the Sun SPOT, it is necessary to install the Sun SPOT Software Development Kit (SDK). This SDK is composed by several programs which are needed for communication with the Sun SPOT and to develop its software.

The installation can be done from the CD-Rom that comes with the Sun SPOT kit, or on the Sun SPOT website [14].

The first step is to install the Sun Java Runtime Environment (JRE), which allows the computer to run software in Java.

After this, the SPOT Manager starts. The SPOT Manager is the software that verifies if the computer has all the required software to use the Sun SPOT, installs them if needed, and puts
all the information together, so that all the Sun SPOT documentation and software can be found in a specific place.

The first action of the SPOT Manager is to verify if the computer has the Sun Java Development Kit (Sun JDK). In case it is not installed, the SPOT Manager installs the Sun JDK.

Secondly, it verifies if the NetBeans IDE (Integrated Development Environment) is installed. This installation is not obligatory since other IDE software, such as Eclipse, can be used with the Sun SPOT if the correct modules are installed. As it is necessary to have a least one IDE software, and the CD-Rom comes with NetBeans, it is the choice for IDE.

After checking the IDE, the Spot Manager checks for Apache Ant, software used to build Java projects. It then checks for the Sun SPOT NetBeans modules, which makes NetBeans work with the Sun SPOT. Finally, after all the software is checked, the Sun SPOTs SDK is installed.

After all the installation, it is required that the computer is restarted and the Sun SPOT will be ready to be used. More information on the Sun SPOTs SDK installation can be found in the Sun SPOTs documentation [13].

3.3.2 Environment

All the software is developed in NetBeans and for every scenario two pieces of software are necessary: one for the SPOT and one for the base station. The software for the base station runs directly on NetBeans. The fact that the base station runs directly from the computer makes it much more powerful than the SPOTs, due to the computer’s processing capacity. The base stations run the Java SE (Java Platform, Standard Edition), the most common java, while the SPOTs run the Java ME (Java Platform, Micro Edition), a special version of java developed for small devices, such as cell phones, PDA’s, and in which the SPOTs are based.

The SPOT Manager comes with a SPOT emulator, called Solarium. In this emulator, all the software developed for the SPOT can be tested.

Since this study requires the measurement of the battery level, the emulator was used only in the beginning, after which the real devices were used in testing the scenarios.
3.3.3 API

The Java API (Application Programming Interface) is an interface that contains all the java packages, classes and interfaces that can be used. The API for the current SDK can be found at Spot Manager, in the section called Docs, in the subtopic JavaDoc, and then in the subtopic index.html.

The most important packages used in this study are listed there, with their classes.

3.3.3.1 Accelerometer

In order to read the accelerometer, two classes were imported (Listings 1):

```java
1 com.sun.spot.sensorboard.peripheral.IAccelerometer3D;
2 com.sun.spot.sensorboard.peripheral.LIS3L02AQAAccelerometer;
```

Listings 1: Accelerometer classes

With these classes, readings from the accelerometer were obtained and measurements of the device’s movement were taken. Listings 2 shows how this reading was done.
Listings 2: Measure acceleration

The acctotal variable (Listings 2, line 4) obtained the total acceleration in all three axes. This acceleration should be equal to 1 when the device is resting, 1 meaning that the device has 1 G force applied to it. When the device moves, this total acceleration should be bigger than 1.

3.3.3.2 Radio

To use the radio, the following class was imported (Listings 3).

Listings 3: Radio class

The radio function can be divided in two, one for receiving packages, and one to send packages.

To receive the packages, the following code was used (Listings 4):

Listings 4: Radio receive packages

The variable command (Listings 4, line 6) would get the read information. To send packages, the following code was used (Listings 5):

Listings 5: Radio send packages

```java
private
LIS3L02AQAcelerometer accel = (LIS3L02AQAcelerometer)
demo.getAccelerometer();
accel.setScale(LIS3L02AQAcelerometer.SCALE_6G);
acctotal = Math.abs((accel.getAccelZ() + accel.getAccelY() +
accel.getAccelX()));

1 com.sun.spot.io.j2me.radiogram.*;

1 Private RadiogramConnection rCon = (RadiogramConnection)
Connector.open("radiogram://:123");
2 Private Radiogram dg = (Radiogram) rCon.newDatagram(50);
3 dg.reset();
4 rCon.setTimeout(100);
5 rCon.receive(dg);
6 command = dg.readUTF();

1 Private RadiogramConnection tx = (RadiogramConnection)
Connector.open("radiogram://broadcast:124");
2 Private Radiogram xdg = (Radiogram) tx.newDatagram(50);
3 xdg.reset();
4 xdg.writeUTF(times);
5 tx.send(xdg);
```
In some moments it was necessary to turn the radio off. In order to do that, the connections needed to be closed and the following code was used (Listings 6):

```java
1 tx.close();
2 rCon.close();
```

Listings 6: Turn off radio

3.3.3.3 Battery

To read the battery, the following class was imported (Listings 7):

```java
1 com.sun.spot.peripheral.IBattery;
```

Listings 7: Battery class

After doing that, the battery could be read and its status was printed out, or saved into a variable (Listings 8).

```java
1 IBattery battery =
2 Spot.getInstance().getPowerController().getBattery();
3 System.out.println("Battery Level after write " +
4 battery.getBatteryLevel() +"%");
5 System.out.println("Available Capacity " +
6 battery.getAvailableCapacity());
7 System.out.println("Max Capacity " + battery.getMaximumCapacity());
```

Listings 8: Battery reading

3.3.3.4 Files

Reading and writing files work in different ways on the base station and on the Sun SPOT. On the base station the following code is used (Listings 9).
To work with the files on the Sun SPOT, the following classes were imported (Listings 10):

```java
1 com.sun.spot.flashmanagement.FlashFile;
2 com.sun.spot.flashmanagement.FlashFileInputStream;
3 com.sun.spot.flashmanagement.FlashFileOutputStream;
```

The following code was used to work with the files (Listings 11):

```java
1 FlashFile dataf = new FlashFile("datafile");
2 FlashFileInputStream ffidata = new FlashFileInputStream(dataf);
3 DataInputStream indata = new DataInputStream(ffidata);
4 dataf.createNewFile(1020);
5 dataf.createNewFile(1020);
6 dataf.close();
7 FlashFileOutputStream ffoodata = new FlashFileOutputStream(dataf);
8 DataOutputStream outdata = new DataOutputStream(foodata);
9 outdata.writeUTF(file);
10 outdata.close();
```

Listings 11: Reading and writing file on Sun SPOT

### 3.4 DATA ACQUISITION

The Sun SPOT has the capacity of printing out information to the console. In this case the following line was added to the code (Listings 12):

```java
1 System.out.println("minutes," + c.get(Calendar.MINUTE) + ",seconds," + c.get(Calendar.SECOND) + ",battery," + battery.getAvailableCapacity());
```

Listings 12: Get timestamp and battery on SPOT
With this code, one timestamp (minutes and seconds only) and the current battery charge in milliamps was printed in the system. One example of data collected (Table 1):

<table>
<thead>
<tr>
<th>Time stamp</th>
<th>Battery Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes</td>
<td>Seconds</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Data example

As the tables collected during the experiment are far too big, containing more than 500 points each, they will not be presented in this study. Instead, a graphic made with these points will be shown. For each table collected, the point where the data will cross the x axis was calculated, using the Excel intercept function. This function uses one interpolation method with all the collected points, and its result is important as it shows the expected lifetime of the device, until the battery remains out of charge.

To read this data on the computer the Sun SPOT needs to communicate with the computer. There are two options available in order to do that. One is using an USB cable and the other is by using the base station and enabling the OTA (over the air) function. As the Sun SPOT is charged by the USB cable, there is no use in measuring battery consumption while charging. Thus, the OTA function was enabled on the Sun SPOT and used to read the print outs. It was later understood that this was not the best solution. The problems with the OTA will be discussed in the results.
4 SCENARIOS

Four scenarios were used for the testing in order to find out which one would be the most effective considering the battery lifetime as the main issue. Details such as which kind of data was used were irrelevant to this study.

The basis for all the scenarios is a mobile Sun SPOT device, which was programmed in such a way that when it came close to a base station a connection was established between them. If necessary, the data contained in one of them (the most updated data), would be replicated to the other device.

The I/O pins, light sensor, and switches were not used at all in the scenarios, and were turned off all the time. For this study, other parts of the SPOT are implied as being off, unless it is specifically mentioned as being on.

4.1 SCENARIO 1 – ALWAYS ON

In this scenario, the mobile device was always looking for the base station and if connected to it, kept trying to replicate the data all the time.

This scenario was not expected to be the most effective one, but it was the first to be implemented for its simplicity. It was also used as a model to compare to the other scenarios.

4.1.1 Specification

The software for this scenario runs on an infinite loop. After starting, it keeps trying to find a connection. When connected, the base station asks the remote station which version of the data the remote station has, and two answers are allowed. One with the version of the data, and the other with a null version, which means that the remote station does not have any data. When the base station receives the remote station’s version, it compares it with its own version, and takes the action to “synchronize” the versions, replicating the newest version. A diagram for this scenario is shown in Figure 4. As can be seen in this diagram, after the beginning of this scenario, the software goes into an infinite loop. In this loop the software is always trying to connect to another device. When one connection is established, the device requests the version of the other device, and then updates the oldest one. After this, it goes back to the beginning and tries to establish a new connection.
4.2 SCENARIO 2 – TIMER ACTIVATED

The connection was periodically activated, and if the base station was found, the data was synchronized. This scenario was a small modification from the first one: with a timer now added. Different values for the period of time could be tested. The battery lifetime was expected to be much higher than in the first scenario, but might not be the optimal.

The device was on, trying to update the data from time to time, and during the waiting time the device was on deep sleep mode.

4.2.1 Specification

After starting the software, the on timer is set. The system tries to make a connection and replicate the data exactly the same way as in the first scenario. Once the timer ends its counting, the system goes into sleep time. The diagram for this scenario is shown in Figure 5. Analyzing the diagram, it can be seen that there is a timer in the scenario that is started just in the beginning of the software. This timer puts the Sun SPOT in a sleep mode from time to time. When the Sun SPOT is not sleeping, it works exactly in the same way as the first scenario, establishes a connection, verifies which device has the newest data, and updates the oldest data, in a continuous loop.
4.3 SCENARIO 3 – MOTION ACTIVATED

Using the Sun SPOT platform’s built-in accelerometer, it was possible to determine when the device was moving, and trying to make the connection with the base station. As the device turned on only when moving, a significant amount of battery could be saved during the time it was not moving.

4.3.1 Specification

Once the software starts, it reads the accelerometer. If the device moves, it follows the same reading procedure as in the first scenario. When still, the device only reads the accelerometer, waiting for the device to move. The diagram is shown in Figure 6. Instead of having one timer in the beginning of the software, like in the previous scenario, this scenario starts reading the Sun SPOT’S accelerometer. When it senses motion, it starts the routine to connect, requests data version and updates the data.
4.4 SCENARIO 4 – MOTION ACTIVATED WITH TIMER

This was the most complete scenario; it used the accelerometer and the timer between the readings.

4.4.1 Specification

This scenario starts by reading the accelerometer in the same way as in the third scenario, but after moving, it starts a timer in the same way as in the second scenario, and then reads the data as in the first scenario. After sleeping, it goes back to reading the accelerometer. The diagram is shown on Figure 7. This diagram is a mix of the two previous ones, it starts reading the accelerometer, and then the timer. When the device is moving, and it’s not sleep time, it tries to establish a connection, requests data version, and updates the data.
Figure 7: Motion activated with timer
5 IMPLEMENTATION AND RESULT

Two different types of software were needed in order to implement the scenarios, one for the base station and one for the Sun SPOTs.

5.1 BASE STATION

Most of the work and logic is done on the base station software. A choice was made to develop all the hard work on a regular PC because the base station runs on a regular PC. From there, commands that the Sun SPOT would recognize and respond to were sent.

All the work on the base station began when the basic software for the Sun SPOT was done. After it run successfully, there was almost no change.

5.1.1 Implementation

In order to make the tests reliable, some changes were made in the base station software. One of the changes was to control some of the environment variables so that the software worked in a specific way. A diagram of the base station software can be seen on Figure 8.

After getting a reply from one Sun SPOT, the software for the tests sends 10 bytes to the device, then sleeps for 60 seconds; sends 100 bytes, sleeps for 30 seconds; sends 1 byte, sleeps for 180 seconds; sends 50 bytes, sleeps for 90 seconds; send 500 bytes.
This software was used with no modification when changing the scenario in all the tests performed.

To compare each scenario, the Sun SPOT battery was fully charged and then the software run for 600 seconds. At that time the battery level was measured.
5.2 FIRST SUN SPOT IMPLEMENTATION

At first, the creation of one generic scenario software was done, on which the other scenarios were implemented. In the first version this scenario was identical to scenario 1.

5.2.1 Scenario 1 – Always on

5.2.1.1 Implementation

In this scenario, two basic pieces of software were made, one for the base station and one for the SPOT. The first version of the software sent only one value from the base station to the SPOT, and turned a led on in the SPOT when the value was received. Even though the led was not used in the scenarios, it helped during the implementation, due to the fact that is made it possible to verify if the device was working. As the basic send and receive structure was working, it was time to make the SPOT send the information to the base station. When it received the correct information, it replied to the base station.

Having the base station and the SPOT sending and receiving the data (Listings 9, Listings 11), the part that verified who had the newest data was accomplished. In the end, the file was read with the time stamp. After all this was done, this scenario’s specifications were fulfilled.

5.2.1.2 Scenario 1 – First result

In this scenario, the system was always on, with the software running, so this was the worst case for the system power. During 600 seconds, the battery level went from 720 mA to 703.62 mA.

Based on the collected points table from this experiment, a graphic was made. The graphic can be seen in Figure 9 which shows the battery level in mA.
The result for battery lifetime (7h 12min 28s) should be used to compare with the other scenarios. This value was calculated using excel intercept function: 25948.61s = 7h 12min 28s. This graphic shows an almost linear behavior of the battery level.

5.2.2 Scenario 2 – Timer activated

5.2.2.1 Implementation

Using the first scenario as a model, it was just a matter of adding a timer to the code, so that after the device turned on, a timer was set, and after files were updated or if the device did not find a connection in 1 minute, the device slept for 1 minute.

5.2.2.2 Scenario 2 – First result

Even though there was some improvement when comparing the battery lifetime with the first scenario, the difference was minimal, in 600 seconds the battery went to 704mA. Figure 10 shows the graphic of the battery level in mA.
Battery time, calculated using excel intercept function: $27943.17 \text{s} = 7\text{h 45min 43s}$.

5.2.3 Scenario 3 – Motion activated

5.2.3.1 Implementation

Using the first scenario as a model once again, the accelerometer reading was implemented, the acceleration at the three axes (Listings 2) was read, and the absolute value was taken with “total acceleration = abs (accelx+accely+accelz)”. By doing this, when the device was not moving the total acceleration had to be equal to 1. When the device moved, its value rose and after some testing the value of 1.5 was defined as a good value to ensure if the device was moving. Therefore, if a reading for accelerations of more than 1.5 G occurred, the same software of the first scenario started running.

5.2.3.2 Scenario 3 – First result

This result was even worse than in the first case. The reason is that the system kept working in the same way as in the first scenario, with the addition of the built-in accelerometer that was now on all the time. In 600 seconds the battery went to 701.65 mA.

Figure 11 shows the graphic of the battery level in mA.
5.2.4 Scenario 4 – Motion activated with timer

5.2.4.1 Implementation

This scenario was a mix of the previous scenarios. After reading a move with the accelerometer, the device started a timer, and then looked for a file to update.

5.2.4.2 Scenario 4 – First result

This result was very close to the first one, balancing the gain of lifetime from the timer and the loss of lifetime from the accelerometer. In 600 seconds the battery went to 703.75 mA. Figure 12 shows the graphic of the battery level in mA.
Battery time, calculated using excel intercept function: 25815.64s = 7h 10min 15s.

5.2.5 Scenario comparison

By analyzing the data, it could be observed that the battery run time is very similar in all scenarios, which is an indication of an error in the software. In the first Always On scenario the battery level was 703.625 mA. In the Timer Activated scenario, the battery level was 704 mA. In the Motion Activated scenario the battery level was 701.65 mA. In the Motion with timer scenario the battery level was 703.75 mA. Figure 13 shows a graphical comparison of all scenarios.
Figure 13: Battery drain for scenario comparison at first implementation

Battery time: 25948.61s = 7h 12min 28s. Always on.
Battery time: 27943.17s = 7h 45min 43s. Timer activated.
Battery time: 23515.42s = 6h 31min 55s. Motion activated.
Battery time: 25815.64s = 7h 10min 15s. Motion with timer.

5.3 SECOND SUN SPOT IMPLEMENTATION

After the first implementation’s bad results, an analysis of the software was conducted. The sleep function was first checked. The Sun SPOT has two ways of sleeping: regular sleep mode and deep sleep mode. Both modes are started the same way, but when some requirements are fulfilled, the Sun SPOT enters the deep sleep mode, otherwise it just sleeps.

After some research in the official forum, it was discovered that there are two common mistakes that do not allow the Sun SPOT to enter in the deep sleep mode: one is to leave a secondary thread running, and the other is to leave the radio on. As one thread was running while trying to read the radio, it was determined that the problem was with this thread.

First the thread was stopped, and the radio was turned off inside this thread. Even then, the Sun SPOT did not enter the deep sleep mode. After researching in the official Sun SPOT forum more extensively, some discussion about using threads was found. Other people had experienced the exact same problem with Sun SPOT and threads, so a decision to rewrite the whole software without the use of threads was made.

With the new software all in one thread, the radio was turned off, and the Sun SPOT entered the deep sleep mode.

Instead of creating generic software with the basic functions and then adding the additional needed functions, the opposite was done. Thus, software that runs scenario 4 was created and it served as the base for the other scenarios.
5.3.1 Scenario 1 – Always on

5.3.1.1 Implementation

For the base software to remain running all the time without interruptions, the accelerometer and the timers were disabled.

5.3.1.2 Scenario 1 – Second implementation result

This result was 5% worse than the same scenario in the previous implementation. However, as the experiment was conducted two weeks after the first implementation, no comparison can be made, because even though the same device was used, the battery can behave differently on different days. Therefore this result was used in comparison with the other results of this implementation. In 600 seconds the battery dropped from 720 mA to 702.64 mA. Figure 14 shows the graphic of the battery level in mA.

![Battery Drain Graph](image)

Figure 14: Battery drain for scenario 1 second implementation

Battery time: 24888.31s = 6h 54min 48s. Always on.

5.3.2 Scenario 2 – Timer activated

5.3.2.1 Implementation

In order to fulfill the requirements of the scenario 2, the base software was used with the timer enabled and with the accelerometer disabled. It run for 60 seconds, and entered the deep sleep mode for 60 seconds.
5.3.2.2 Scenario 2 – Second implementation result

With the deep sleep mode working, an expressive result with 202% more battery lifetime than in the first scenario was achieved. The battery charge after 600 seconds is 714.23 mA. Figure 15 shows the graphic of the battery level in mA.

![Battery level graphic](image)

Figure 15: Battery drain for scenario 2 second implementation

Battery time, calculated using excel intercept function: 75366.95s = 20h 56min 6s.

5.3.3 Scenario 3 – Motion activated

5.3.3.1 Implementation

Using the base model with the accelerometer enabled, and the timer disabled, the specifications for scenario 3 were completed. When the device moves, it turns on and keeps on until the file is replicated.

5.3.3.2 Scenario 3 – Second implementation result

This result was not very expressive, with only 15% improvement. When the device was waiting for movement the radio was off, which caused some improvement. However, the device was still running, which did not occur when it entered in deep sleep mode. In 600 seconds the battery level was 705.05 mA.

Figure 16 shows the graphic of the battery level in mA.
Battery time, calculated using excel intercept function: 28698.83s = 7h 58min 18s.

5.3.4 Scenario 4 – Motion activated with timer

5.3.4.1 Implementation

This software was the full version of the base software which was used in the second implementation having all the functions enabled.

5.3.4.2 Scenario 4 – Second implementation result

In this scenario the result was once again a mix of the improvement from the timer, and the average result from the accelerometer. The battery level in 600 seconds was 712.47mA. Figure 17 shows the graphic of the battery level in mA.
5.3.5 Scenario comparison

Since the best results were obtained with the timer only, the motion activated scenario must be studied, since it should offer a better result than the ones found. The battery level for scenario Always On was 702.64 mA. In the Timer Activated scenario, the battery level was 714.23 mA. For the Motion Activated scenario, the battery level was 705.055 mA. In the Motion with timer scenario, the battery level was 712.47 mA.

Figure 18 shows a graphical comparison of all scenarios.
Battery time: 24888.31s = 6h 54min 48s. Always on.
Battery time: 75366.95s = 20h 56min 6s. Timer activated.
Battery time: 28698.83s = 7h 58min 18s. Motion activated.
Battery time: 62066.54772s = 17h 14min 26s. Motion activated with timer.

Figure 18: Battery drain for scenario comparison at second implementation
6 DISCUSSION

When implementing a mobile device, both software and hardware should be analyzed to achieve the best results. The software can be easily changed to fix errors or follow a better result whereas the hardware cannot be changed as easily.

In this case, the hardware used in the tests was the Sun SPOT. Because it is a well-known device, it was thought that it would be the perfect hardware for the tests due to the fact that it contained the qualities required for the experiment. These qualities can be described as: First, it is a mobile device. Secondly, its software is implemented in JAVA. Also, it offers radio communication and has an accelerometer to measure whether it is moving or not.

6.1 PROBLEMS

Some problems were found when the development of the software and the experimentation began. These problems are discussed below.

6.1.1 Development software problems

There are many versions of the Sun SPOTs SDK. The version changes depending on the hardware version, so it is recommended that the software from the CD-Rom is installed if the version to be used in not known. In this study, the SDK V4.0 Blue was used first (the version that comes with the CD-Rom), but after some problems with the SDK, an update to the SDK V5.01 Red was done, and each Sun SPOT firmware was updated as well.

With the SDK working, some time was spent trying to read the battery level from the emulator. After failing on doing it, some research was carried out and the following answer was obtained from the official Sun SPOT forum:

“Sorry, it is an issue with Solarium. When the Emulator was written SPOTs did not support the ability to read the available capacity of the battery. All that was available was the current battery voltage & the charge/discharge rate, so that's why the Sensor Panel in the Emulator has sliders for those two values. I'll add it to the list of requested features, but do not expect that it will be done anytime soon.”

As it was impossible to work with the emulator, the experimentation continued with the real Sun SPOT.

6.1.2 Software problems

The first implementation did not work well, because the software did not achieve the hardware requirements to enter the deep sleep mode.

Researching the official Sun SPOT forum, much divergence about how much current the device uses in the deep sleep mode was found, so the ideal case would be to measure the consumption physically, not only by means of software as occurred in these experiments. Unfortunately, due to lack of time, this measurement was not done, and the only conclusion that can be possibly extracted from the discussions in the forum was that when the device is active or
in regular sleep, the system drains a current in mA from the battery, and when in deep sleep mode, this current falls to μA.

Even without knowing the exact value of the currents, the difference between the regular sleep mode and the deep sleep mode could be observed when comparing the first and second implementations. Thus, it can be said that this part of the experiment was successful, as the system lifetime was drastically improved only by using a timer and the deep sleep function.

6.1.3 Hardware limitations

The results with the motion activated scenario were not as good as expected, so an explanation had to be found. After searching the Sun SPOT forum, some discussions about the accelerometer and one of the Sun SPOT limitations were found: One explanation is that it cannot read the accelerometer while in deep sleep mode, so when the device was waiting for some movement, it was draining current from the battery in mA, not in μA like when in deep sleep. There were some discussions about how to make some modifications to the Sun SPOT to allow it to read the accelerometer in deep sleep mode, but the only solution found was to use another kind of sensor, instead of the accelerometer, connected to the I/O pins of the Sun SPOT to measure the movement.

Thus, it can be assumed that the bad results of the motion activated scenario were caused by a hardware limitation. And then no more experiments were conducted due to time and resources constraints.

After the experimentation part finished a new version of the Sun SPOT was released, which confirms the limitations found in this study. Consulting the release notes of this version [23], the following lines can be found:

“The SPOT sensor board has been redesigned. New features include:
New MMA7455L accelerometer replaces LIS3L02AQ accelerometer. New accelerometer has three scale ranges: 2/4/8G.
By default sensor board ATmega microcontroller now stays awake when main processor board is powered down for deep sleep, and can generate interrupts to wake SPOT up on pin changes, switch presses, etc.”

Had the same experiment been done with the new hardware, results even better than those obtained with the timer could be expected, as now it is possible to read the accelerometer while the main board is in deep sleep.

However, it is believed that the results obtained are valid and may be used by anyone when designing a mobile device or choosing a hardware for a system.
7 CONCLUSION

Mobile devices are becoming everyday use devices for people. With the recent growth and development of devices such as mobile phones, PDAs, PADs among others, it is natural that more research about these tools is carried out.

New uses and new applications created for these devices are found every day, and data replication is one subject that is being more and more discussed. The majority of data replication discussions focus exclusively on the software, discussing protocols of communication, algorithms and other software specific details about data replication. In addition to this software discussion, it is fundamental to discuss the hardware, especially in mobile devices that have a limited system lifetime before their battery needs to be recharged. Some software and hardware limitations were found in this study which led to some unexpected results.

The chosen hardware in this study was the Sun SPOT. This device uses free software, and seems perfect for testing purposes only, as it is expensive for large scale uses.

At first, the importance of the software in a system was demonstrated. The first implementation of the software was not useful. Due to bad software development, it was not possible to utilize the full hardware potential, creating a very limited system. Even in these simple tests, there was a great difference in the results with the improved software.

The second result occurred when the hardware did not meet the software expectations. The system was expected to be able to wait for a long time for a movement, draining just a few μA, but this did not happen.

From the experiments results with each scenario, the best result was achieved using a simple timer, and putting the device in a sleep state that drains some μA while in wait. Taking into consideration the results obtained in the scenery tests with the software and hardware limitations, it is safe to say that both can contribute significantly to the results.

When developing a system, the person who creates the software must know every detail of the hardware used, otherwise some hardware resources that could improve the system might not be used. In the same way, when choosing or creating hardware, all the required resources must be evaluated, or the system may not work as expected.

Although the best scenario for these software and hardware was found, it is believed that a better result could have been achieved if a different hardware had been used. Another factor is that there is very little discussion about hardware in mobile devices compared to what can be found about software. It is much easier for an independent researcher to run experiments only with software, using limited hardware, but as it was seen in this study, when finding a hardware limitation the researcher does not have the means to fix the limitation and continue running the experiments.

As a suggestion for future studies, an experiment can be conducted with the new Sun SPOT hardware, or with another hardware that solves the limitation that was found. Another suggestion is to create simple replication software that could work on mobile phones.
REFERENCES


