Universitatea POLITEHNICA din București
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare

Lucrare de Diplomă

Arhitectură pentru rețele wireless de senzori cu energy harvesting.

Autor

Ioan Deaconu

Coordonator: Sl. Dr. Ing. Dan Ștefan Tudose

București, Iunie 2016

University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Diploma Thesis

Arhitecture for energy harvesting wireless sensor networks

Author

Ioan Deaconu

Supervisor: Sl. Dr. Ing. Dan Ștefan Tudose

Bucharest, June 2016

Contents

Abstract

This thesis proposes a new way of implementing a mobile gateway for Wireless Sensor Networks that simplifies applications running in remote locations, where maintenance is difficult. Wireless Sensor Networks islands need a gateway connection in order to reach the outside world, but this is difficult to provide in all instances. The solution to this problem is to use a gateway mounted on a UAV that can reach those islands and extract data from them. This has been proven to be feasible but adoption is low because of the high cost and the technical background needed to operate it. We propose a simpler, easier to operate and cheaper solution for this problem.

Keywords Wireless Sensor Networks, sparrow, energy harvesting, scheduling

Acknowledgements

Foremost, I would like to express my gratitude to my advisor Dan Tudose for the continuous support that he provided.

Besides my advisor, I would also like to give special thanks to Dan Dragomir for providing corrections to this work.

My sincere thanks also goes to Adriana Drăghici, Alexandru Corneliu Olteanu and Dan Ștefan Tudose for their insightful comments and moral support.

I thank my fellow labmates from Robolab robotics group: Tudor Vișan, Andrei Mușat and Andrei Vasiliu for the sleepless nights we were working together before deadlines, and for all the fun we had in the past four years.

Last but not the least, I would like to thank my family who have shown understanding and have given me the moral support needed.

Chapter 1

Introduction

The main problem faced by Wireless Sensor Networks is autonomy. The location in which some nodes can be installed could be difficult to reach, turning a simple task of changing the batteries of the nodes into an impossible one.

The solution to this problem is powering the devices from alternative sources of energy, process called energy harvesting. In the recent years, energy harvesting has become more and more used in the field of Wireless Sensor Networks. There are plenty of alternative energy sources, such as solar cells, vibration absorption devices, wind mills, thermoelectric generators and others, that can be used to power the nodes or charge their batteries in order to become autonomous.

While the energy source problem has a solution, another problem appears in the form of finding a method to store the generated energy. A battery could be used to store the generate energy, but unfortunately, current technology allows to charge one for a few hundreds of cycles. Considering that a cycle would be used per day, in maximum 2 years, the battery will lose most of its original capacity. An alternative to the battery is using a super capacitor, which has a life of 10 years or 500.000 cycles, but are more expensive and store far less energy than regular rechargeable batteries.

In order to alleviate the amount of stored energy, an scheduling algorithm can be used, algorithm that will dynamically vary the frequency with which the node performs various tasks in order to be able to send data without consuming all the available energy.

In this thesis we will describe the architecture of an energy harvesting wireless sensor network, shortly titled EHWSN. We will present a new node and development platform, the Sparrow R, designed for low power and the problems we encountered when using solar energy and a capacitor as storage. The result of this is an efficient but lightweight algorithm for efficiently using the stored energy.

Chapter 2

Related Work

Energy harvesting has been studied for a long time and many tried to find a solution for their shortcomings. Solar panels are frequently used in EHWSNs because they can generate a lot of power but the power they generate is not consistent and predicting it was on of the challenges to be overcome.

2.1 Prediction generated Energy

Because the solar energy is not constant, in order to predict the generated energy, a history of past-days weather conditions or generated energy must be taken into account. The state-of-the-art algorithm for this is Weather-Conditioned Moving Average (WCMA) [25] which can keep a history of number of days of generated energy in order to predict the generated energy in the next hour. The results of the algorithm are shown to be impressive, with an average error of 9.8% in 45 days of testing.

Because the algorithm requires the generated power to be measured in order to have a precise history of generated solar energy, this algorithm is not feasible in application where the node can be very low power and a small solar panel of just 60mW or less can be used, compared to one used in the paper, which could generate more than 300mW. A simpler solution for hardware as well as software must be found for those applications.

2.2 Using stored Energy

The common approach to optimal packet scheduling is using a water-filling algorithm [31] [?], where the time is divided into slots and given a level of energy to be used in that slot. For better power optimizations, this approach is modified into backward water-filling, directional water-filling, generalized iterative water-filling

[?] for offline (deterministic) scenarios. In order to simulate online (stochastic) scenarios, Gaussian noise is added and the constant water level policy, energy adaptive water-filling[23], time-energy adaptive water-filling[?] can be used. The problem with their approach, in our opinion, and even simpler ones like fuzzy power management [13] is that the complexity of the algorithm is rather high, at least O(N), a constant feedback of consumed energy must be provided. The fact that the results of all the algorithms were simulated and not implemented and tested on a real device, indicated that real conditions, like capacitor leakage or temperature variations are not taken into consideration.

Chapter 3

Sparrow R

Most of the nodes are build using AVR processor [4], an 8-bit architecture. Few attempts have been made using newer Arm Cortex-M3 32-bit architecture. The benefits of higher computing power are presented here [20], where a 4.6 throughput can be obtained compared to an 8-bit cpu over the same wireless network. Unfortunately, the downside is higher power consumption that can create difficulties for power supplies. Even though the same average power consumption can be obtained, due to higher power peaks, the power supplies will be depleted more quickly.

The current nodes, Sparrow V3.2 and SparrowV4, use an AVR 8-bit architecture and run at a speed of 16MHz. For our solar powered node, we wanted to use a newer, more powerful and low power microcontroller. Based on the previous requirements, we have chosen an Arm Cortex-M0+ 32-bit Atmel SamR21.

3.1 Performance

We present bellow the main diferences between the two mcus, SAMR21 and ATMEGA128RFA1.

Being a 32-bit architecture and a new architecture, even though SAMR21 consumes 5.5mA compared to 4.1mA of Atmega128RFA1, for simple 32-bit integer addition, SAMR21 consumes only 49nJ per iteration while the 8-bit microcontroller consumes 274nJ, almost 5 times more. Considering performance figures, SAMR21 was 9 times faster with 403950 iterations per second while Atmega128RFA1 managed only 44890 iterations.

Testing the performance of branch predictor, revealed that the M0+ is only 12% better than the older 8-bit counterpart when running at the same speed, but thanks to the frequency difference, it ends up being 3,36 times faster.

Criteria	Atmega128RFA1	SamR21
CPU Speed	16 MHz	48 MHz
CPU architecture	AVR 8bit	Cortex M0+ 32bit
CPU Power	4.1 mA	6.5 mA
Flash	128 kB	256 kB
Ram	16 kB	32 kB
Flash Endurance	50000	150000
Rx Consumption	12.5 mA	11.8 mA
Tx Consumption	14.5 mA @ 3.5mA	13.8 mA @ 4 dBm
Receiver sensitivity	-100 dBm	-101 dBm
Tx Max Power	3.5 dBm	4 dBm
Package	QFN64	QFN48 or QFN32

Table 3.1: Comparison between Atmega128RFA1 and SamR21

Criteria	Atmega128RFA1	SamR21	Total Advantage	Advantage per MHz
Integer Iterations	44890	403950	8.99	2.99
Branch Iterations	27782	93552	3.36	1.12
While(1) Iterations	191536	6693086	34.94	11.64

Table 3.2: Speed comparison

Criteria	Atmega128RFA1	SamR21
Integer Iteration	274 nJ	49 nJ
Branch Iteration	442 nJ	208 nJ
While(1) Iteration	64 nJ	2.9 nJ

Table 3.3: Energy efficiency comparison

The SAMR21 micro-controller is almost the same micro-controller as SAMD21 [7], which is used in Arduino Zero boards. This allowed use to use exiting code, but unfortunately a not well written one.

Even thought the Arduino software is well designed, it was not designed with low power approach from the beginning. We will describe some of the problems encountered when trying to create the software stack.

The first problem noticed was that the Arduino Zero board had no sleep functionality implemented. The ideal idle current consumption should have been less than 5uA, tested and measured using a project created in Atmel Studio 7.0. The current consumption of the board was around 350uA. Further tests revealed that the USB device was always initialized, which accounted for the extra 200 uA. The rest of 150uA came from a default initializations of the pins as input pins, but this only lowered the current consumption to about 60uA. We kept searching for a cause, and discovered that the clock generators are never disabled at start-up, which accounted for about 30uA.

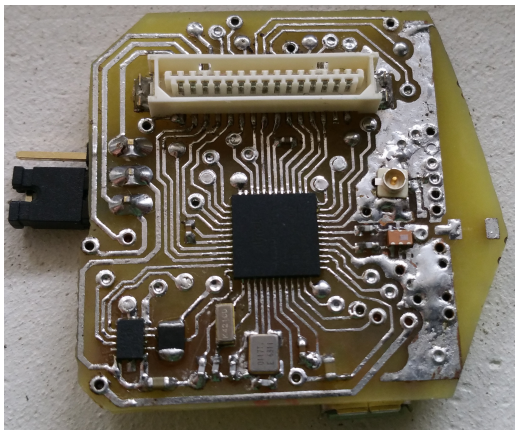

Figure 3.1: Sparrow R node

So far we managed to decrease the idle current consumption for the platform from 350uA to about 30uA @ 3v2, but it is still far from ideal. Surprisingly, lowering the voltage from 3v2 to 1v8 lead to decrease in sleep current consumption down to 3.3uA. When examining the power trace using a digital oscilloscope, we found that a very low frequency clock remains active, which at 3v2 has high spikes in power consumption.

Tough we did not reach the goal of 5uA, we still managed a respectable 30uA @ 3v2 and less than 4 uA @1v8. Due to the time constrains and the need to use the nodes in order to implement and test new features, we decided that for now this is acceptable, and for future revisions, we will come back and find the extra clock source.

Even the run current consumption was not ideal, instead of achieving the promised 70uA/MHz @ 3v2, or around 3.5mA @ 48MHz, the micro-controller consumed 8mA @ 48MHz. We managed to reduce the current consumption to 5.5mA @ 48MHz, due to clock optimizations presented bellow.

The first change was to change the clock of the peripheral interfaces, instead of 48 MHz, we run them at 12 MHz. Also if peripherals are not used, we completely disable them. Due to this, we ran into problems related to SERCOM implementation, a generic module that handle USART, SPI and I2C. It was working on Arduino Zero, because the CPU and the BUS were configured to run the same speed, but due to the previous clock source modifications, the SERCOM did not set the correct speed. Also, there are 6 SERCOMs, and instead of enabling the clock for each one only when it is used, all of them were enabled, which lead to extra power consumption during run time.

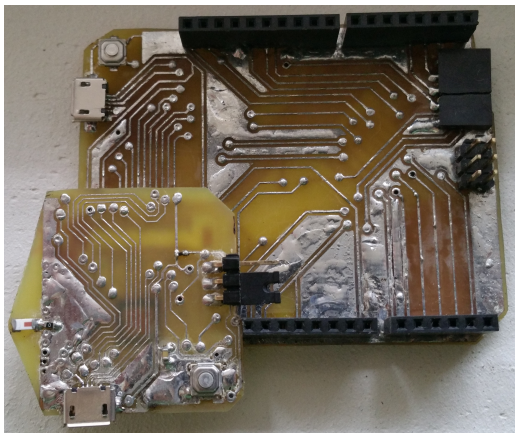

Figure 3.2: Sparrow R node mounted on Aduino compatible base

3.2 Hardware

Because not all sensors are designed to run at 1v8 up to 3v3, we need to be able to dynamically change the operating voltage of the node. We used TPS62742, a step-down switching DC/DC converter with up to 90% efficiency, voltage selectable output from 1v8 to 3v2 with 200mV step, 360nA quiescent current and a special mcu controllable load output, with push-pull transistors. In inactive state the load line is pulled to GND and when active is pulled to VCC. When active it consumes 12uA, but compared to the controlled sensors, this should not be noticeable. Because this allows to completely power down the sensors, the "stand-by" current consumption is 0uA and it also eliminates the previous design problem of floating GND which allowed the sensors to "steal" power from other pins and not be properly disabled.

The node can be connected to an extension daughter board which fully respects the Arduino pinout. The advantage of this approach is that it allows to easily test and prototype new configurations in order to be prepare the project in the shortest time possible. Also existing hardware designed for Arduino can work with this board, which increased the number of compatible hardware. In total, 20 I/O pins are available, pins that can be used for connecting sensors, either on the daughter board, or directly on a specialty designed board.

A jumper can select whether the Node is powered from USB or from other 2v1+ voltage supply. Through the same jumper a power measuring device can be used to monitor the total power consumption. In case the DC/DC converter is not needed the node can use other power source.

3.3 Software

We implemented new modules designed for low power like sleep, power management which can dynamical change the running voltage between 1v8 to 3v2 when requested and enable or disable the LOAD power line. This allows the user to select which voltage is better required for applications. For example, some sensor must be powered at exactly 2v8 while others at 2v5 or lower. This module allows for precise voltage selection with 200mV increments, use the sensor and then switch to the lowest voltage in order to obtain the best power consumption possible.

The RF module is AT86RF233, integrated into the micro-controller. An Arduino library [6] exists for the RF but in order to add new features, we integrated the module for the RF in the core of the platform. This allowed us to let the user focus on what to do with the platform and not how to do something with it. Furthermore, extra futures and bug fixes are easier to be implemented if the module is integrated in the platform. For example, in case the RF is constantly running in receive mode for more than 5 minutes, it is recommended to do Fine Tuning of the PLL clock in order to eliminate possible clock skews. Feature wise, when the module automatically receives a packet, it saves it locally together with RSSI and LQI, which can be later read and used buy the user. The internal buffer is designed for 8 packets of 127KB of data, which amounts for 1 KB of ram. The buffer is cyclic, so in case the buffer is not read, the oldest data is discarded and replaced by a new one. This should not happen very often, because the buffer is large enough to handle all request, even for high bandwidth transfers.

If the user has a need to save the data in case of power failure, the micro-controller has an EEPROM like functionality which allows a 16KB region of flash to emulate EEPROM write endurance. The flash contains pages of 64 bytes, and the EEPROM has an overhead of 4 bytes, which leaves 60 bytes for actual data. Also, for each page, another page must be reserved for further use. The results is that out of 16KB used, the total amount of usable space left is $\frac{16*1024}{2} * \frac{60}{64} = 7680 bytes$. This should be more than enough for normal use because the normal endurance of 25k cycles of flash write and erase are increased to at least 150k, with typical values reaching 600k cycles. If a new software is uploaded, the EEPROM zone is completely erased.

For timekeeping when sleeping, RTC functionality was implemented. Besides keeping the time, RTC provides alarm interrupts for a special date, which can be configured to be triggered every minute, every hour, every day, every month, every year, or only once. Together with another peripheral named EventSys, periodic interrupts are provided and the interrupts interval can range from once every second up to 128 times per second, with increments of power base 2.

Because the software and hardware are never perfect, a watchdog functionality is also implemented, in order to avoid code lock-up or hardware failure due to

extreme environment conditions.

The software can be installed as a new board for Arduino 1.6.x, latest iteration to date, May 2016. It was tested using Windows 8.1, but it should be fully compatible with other operating systems as well.

The node has native USB which allows for code upload and also serial interface over CDC. Because no extra components are needed, the same node can be configured to act as a gateway or as a leaf.

3.4 Power Consumption

Because the node was designed with low power in mind, we can obtain a very low deep sleep power consumption of 5uW. A small solar panel together with a small capacitor can be used as the main power supply.

For example, when a 1F capacitor is used with the voltage ranging from 3v3 to 2v1 the equivalent battery capacity would be

$$\frac{F * (Vi - Vf)}{t} = \frac{1F * (3.3V - 2.1V)}{3600s} = 0.333mAh$$

The node is equipped with a DC/DC converter that can bring substantial power savings, depending on the voltage of the power supply. A LDO wastes a lot off energy as the delta between the voltages increases, were as DC/DC works best when the delta increases, or putting it simple, if an LDO ouputs 1v8 and a chip consumes 5mA, the input current is almost the sam 5mA regardless of the input voltage. So even though the chip consumes 9mW, the total power consumed is actually 25mW. If in the same situation, a DC/DC with a 90% efficiency is used, then the input current would be :

$$Iin = \frac{Iout * Vout}{Vin * Efficiency} = \frac{5mA * 1.8V * 100}{5v * 90} = 2mA$$

we can ignore the quiescent current because it is very small, around 360nA. The total power consumption in this case is 10mW for an output power of 9mW compared with the LDO which consumes 25mW in order to output just 9mW, a 250% difference between them.

In a real world situation, when a battery or a capacitor is used, the difference between them is smaller. We will present two cases, in order to better understand the influence of higher voltage supply.

The minimum voltage will be 2v1 for DC/DC and 1v9 for LDO. The current consumption of the chip is 5.5mA, and a capacitor of 1F.

Case 1: Capacitor charged to 3v3.

$$T_{LDO} = \frac{C * (V_i - V_f)}{I} = \frac{1F * (3.3V - 2.1V)}{5.5mA} = 218.2s$$

$$E_i = \frac{C * V_i^2}{2} = \frac{1f * (3.3V)^2}{2} = 5.445J$$

$$E_f = \frac{C * V_f^2}{2} = \frac{1f * (2.1V)^2}{2} = 2.205J$$

$$E = E_i - E_f = 3.24J$$

$$P = V * I * E_{ffiiciency} = \frac{1.8V * 5.5mA * 90}{100} = 11mW$$

$$T_{DC} = \frac{E}{P} = \frac{3.24J}{11mW} = 294.54s$$

$$Advantage = \frac{T_{DC} - T_{LDO}}{T_{LDO}} = \frac{294.54s - 218.2s}{218.2s} = 35\%$$

Case 2: Capacitor charged to 5v.

$$T_{LDO} = \frac{C * (V_i - V_f)}{I} = \frac{1F * (5V - 2.1V)}{5.5mA} = 527.27s$$

$$E_i = \frac{C * V_i^2}{2} = \frac{1f * (5V)^2}{2} = 12.5J$$

$$E = E_i - E_f = 10.295J$$

$$T_{DC} = \frac{E}{P} = \frac{3.24J}{11mW} = 935.91s$$

$$Advantage = \frac{T_{DC} - T_{LDO}}{T_{LDO}} = \frac{935.91s - 527.27s}{527.27s} = 77.5\%$$

At 3.3V the advantage is not that big, but at 5V we nearly double the battery life. Furthermore, using a LIPO battery which does not discharge as linearly as the capacitor will increase the advantage of the DC/DC.

In real world testing, we charged a 1F capacitor up to 3v3 and let it discharge by transmitting data every second. Even though the software was not optimized for low power, the node managed to run for 8 hours before the capacitor was completely discharged which translates to an average power consumption for almost 112uW.

Chapter 4

Energy Harvesting

This chapter presents the energy harvesting platform.

It is difficult to chose the best solution for solar energy harvesting, due to power requirements of different applications. In this chapter we will present the hardware that we used to generate and store solar energy.

4.1 TI BQ25504

The main component of an energy harvester is the boost converter needed to raise the voltage of the solar cell in order to be able to charge a battery or a super-capacitor. The chip selected by us is TI BQ25504, a very efficient boost converter with battery management for energy harvesting.

The chip can begin charging from a low voltage of only 330mV, has a very small quiescent current, less then 360nA and once started, it can harvest energy until to voltage drops to 80mV or less.

Another advantage is that the output voltage can be selected from 2.5V up to 5.25V, which can be used, as demonstrated in previous chapter, to charge a capacitor to a higher voltage in order to exponentially increase the total amount of stored energy.

Four our experiment, we have selected an output voltage of 3380mV.

4.2 Super-capacitor

The advantages of the super-capacitor over the rechargeable battery are as follows:

- it can charge and discharge almost instantaneously

- it has a very high number of charge/discharge cycles
- it does not suffer from the same aging symptoms as a battery
- it is far less pollutant than a standard battery
- it will allow a longer maintenance-free time than a battery

The disadvantages of the super-capacitor are :

- it is bigger than a battery
- it can store a much smaller amount of energy than similar sized batteries
- it is very expensive
- it operates at low voltages and may require a charge pump to rise the voltage
- much higher current leakage.

The main problem of a capacitor is current leakage. The bigger the capacitor, the bigger the current leakage. Also the technology with which they are built can also have a profound effect on leakage current.

Even though it does not seem much, if a 1F capacitor has self discharge rate of 6uA [?], this is more than the sleep power of the Sparrow R. Even more, if using a 25F capacitor, the leakage current is increased to 45uA, which can be equivalent to the same power consumption of the node sending data every few seconds. Because of this, the total energy that can be stored is greatly affected by the duration needed for the node to use the super-capacitor as power supply.

Charging is also a problem. If for 1F capacitor, a panel that can supply 50uA is more than enough to charge the capacitor and supply the power to the node, for a 25F capacitor that power is barely enough to maintain the stored energy, without even powering the node.

For our experiments we have chosen two 1F capacitors rated at 5V.

4.3 Solar Panel

We can use any solar panel, as long as it is rated up to 3V and has enough power to charge the selected capacitor.

Because we do not need a large amount of energy, a small solar cell of just 60mW is more than enough to fully charge the capacitor in just 5 minutes of sunny weather. This means that regardless of the meteorological condition, we would be able to charge the capacitor daily.

Chapter 5

Software Implementation

Our solution is divided into different modules that run independently but communicate with each other to achieve the main goal. The separation of software modules allows future features to be added easily.

There are multiple applications, running on the SparrowDongle and SparrowV3.2, running on the Parrot AR.Drone and a modified instance of the FreeFlight 2.0 application running an Android phone. The connection between them is presented in figure 5.1.

The SparrowDongle gateway is always in a listen-for-data state and dumps any data it receives on the virtual serial port emulated over USB. When it receives a data packet, it sends back an ACK message to let the SparrowV3.2 nodes know that they can begin sending the entire stored data to the it.

The SparrowV3.2 node is periodically sending a small data packet to check if a gateway is available. It stores the data accumulated during the period when no reply is received. When it receives the ACK message from a mobile gateway it starts sending the previously stored data to the gateway. The data can vary, from sensor readings to debugging information used to check the state of the Wireless Sensor Network.

The data gathered by the gateway is saved into different files in the Parrot AR.Drone's internal memory. The files also contain information such as the node identification tag and time of the transfer. The data can be accessed at any time by any device that connects to the drone's wireless network on port 4242 via FTP.

The drone also processes some of the collected data to provide real time HUD information, such as signal strength, last connection time and number of discovered nodes. This information is sent to the controlling device through a socket connection. The controlling device, PC or Android, gathers that information and displays it to the user.

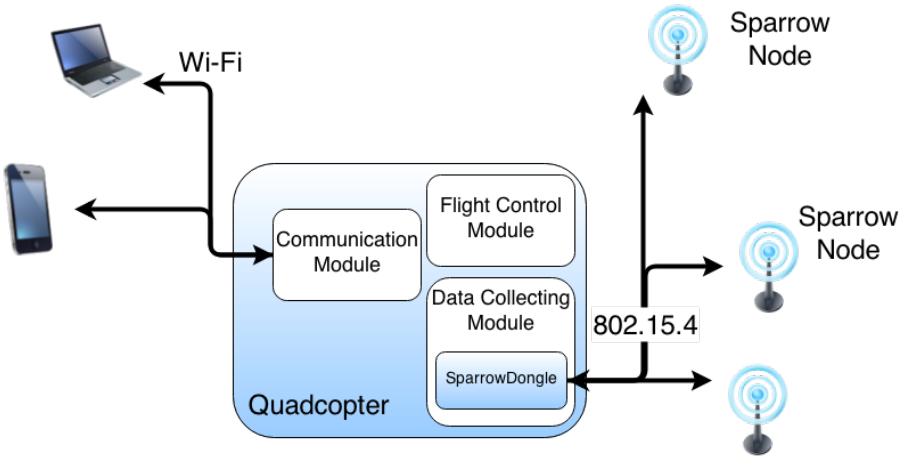

Figure 5.1: *Modules running on the drone and their connections*

5.1 Drone modules

5.1.1 The USB Module

The default software with which the drone is delivered does not implicitly recognize the dongle. This behavior was expected because the drone runs a stripped down version of Linux. In order for the drone to recognize the dongle, a module had to be compiled for the specific CPU architecture and operating system that were installed on the drone.

The required module is called cdc_acm[10]. This module is responsible for emulating serial ports over USB. The module creates a file, named /dev/ttyACM0, linked with the dongle, which can be read just like any other file.

5.1.2 The Debug Module

When performing modification to the existing code, debugging support can greatly speed up the development process. This module allows for displaying control message to the user console even when the process is running in the background. If the messages are always activated, they can slow down the execution speed of the entire application.

In order to see debugging messages, the debug option must be activated and the simple command from listing 5.1 must be run.

Listing 5.1: Simple display message command

```
p=$(pidof read) && strace -p $p
```

Enabling the debug option is just a matter of setting the *DEBUG_ON* constant to the value 1, recompiling and uploading the code to the drone. The constants needed for debugging are shown in listing 5.2.

Listing 5.2: Debug and timing defines

```
/* activates/deactivates printf debug information*/
#define DEBUG_ON 0
/* delay time in microseconds*/
#define DELAY_US 100000

/* time in milliseconds since connecting display the node*/
#define TIME_DELTA 45000
#define DEBUG_PRINT(a...) { if(DEBUG_ON) printf(a); }
```

In certain parts of the modules a delay is needed in order to wait for an action to be executed. The value *DELAY_US* can be changed to any value, but you must be careful in doing so. A small delay sends data more often but it uses a lot of processing power. A big delay could be too slow for the data to be usable. We have chosen a delay of 100 ms, for 10 data updates per second.

Also, the nodes are memorized for the period *TIME_DELTA*. During this period, the node will be saved and informations about it will be sent to the user. If after *TIME_DELTA*, no new data has been received from the node, it is deleted from the list.

5.1.3 The Data Collecting Module

The module saves the collected data into the drone's internal memory and pases the data on to the communication module, which displays on the controller interface certain data items like: the number of nodes currently connected to the dongle, the signal strength, dongle connection status etc.

This module contains some extra features that are designed to make the solution more user friendly and easier to extend in the future.

5.1.4 Modules intercommunication

The memory area in which the information sent to the user is saved is shared between this module and the communication module. The interaction method between these two modules resembles the consumer-producer problem, where the Data Collecting Module is represented by the producer and the Communication Module is represented by the consumer.

A important issue with this approach is synchronization and the avoidance of data races. These are prevented with the use of a mutex construct that only allows one thread at a time to access the data. Example is depicted in listing 5.3.

Listing 5.3: Data Collection use of mutex

```
pthread_mutex_lock(&data_lock);
add_node_data(get_current_timestamp(),read_data + 7);
pthread_mutex_unlock(&data_lock);
```

Similarly, the mutex is used in the Communication Module when it consumes data.

5.1.5 Fault tolerance

Because the Dongle is connected to an USB port on a machine that has a lot of vibrations, it might disconnect / reconnect for a very short period of time. This module has been designed to cope with multiple USB disconnects and reconnects without the need to reset the drone. This information is vital, because you can check if the dongle is still connected to the drone without the need to inspect it visually or to connect to a debug terminal.

Besides the possible USB dongle disconnects, an out of range signal may be experienced. If this happens, the drone will hover until the connection is reestablished.

5.1.6 The Communication Module

All of the information gathered by the Data Collecting Module would be useless if it cannot be accessed easily.

This module, as the name suggests, handles the communication of this crucial information back to the user.

Being a different module, with different attributions than the Data Collecting Module, it has an entire Linux process dedicated to it for 3 important reasons:

1. The approach of having a process per module allows the modules to run independently of each other.

2. The Data Collecting Module can collect the data from the Dongle as soon as this is available.

3. If the Communication Module stops working, the Data Collecting Module can keep collecting data, so complete failure of the system is avoided.

4. System processes can be restarted in case of failure.

5.1.7 Socket with connection reset

The communication is done through socket connections listening on port 8888. The server running on the drone accepts only one connection at a time.

If a connected client decides to disconnect before or while a write operation is in progress, a $SIGPIPE$ error signal will be thrown, stopping all the modules. This is prevented by ignoring the signal, forcing the write action to return a $EPIPE$, and exiting gracefully.

The main process will use the callback accept_socket_connection to reestablish a new connection. Once a connection is established, it will send information once every $DELAY\_US$ microseconds. The program was configured and tested with a 100ms wait period that translates to ten updates per second.

This delay is required because if there was no delay the communication would occupy too much processor time both on the drone and on the controlling device.

5.1.8 JSON Encoding of Data

JSON [3] is an open-standard that uses text to encode data. It is an alternative to XML. Derived from the JavaScript scripting language, it is a language-independent data format available in most programing languages.

JSON is best suitable for this application as a data encode format because it is data oriented, unlike XML which is document oriented. Also, it is very easy to encode because it has a code like structure, the result being smaller than the XML alternative. Another advantage is that all devices can decode it.

The informations encoded by the drone are:

- Dongle connection status
- An array containing node data
  - Node unique ID
  - Last connection time of the node to dongle
  - The power of the received signal

5.2 SparrowV3.2 module

Due to the communication protocol implemented, the SparrowV3.2 node uses very little power when not connected to the drone. The solution is similar to the one described by Cardei et al [17]. The Sparrow node sends a small packet at a fixed interval.If the packet is received by the dongle, it will send back a specific ACK just to the node that sent the packet. When the node receives the ACK, it will try to send all the stored data to the drone, starting from the oldest entry to the newest one. The data is stored in a circular linked list of a fixed size. If the list is full, the oldest entity is always replaced by new data.

The drone saves the received data, locally in files that have the name comprised of the timestamp of the current session and the unique id of the node.

5.3 Android application modules

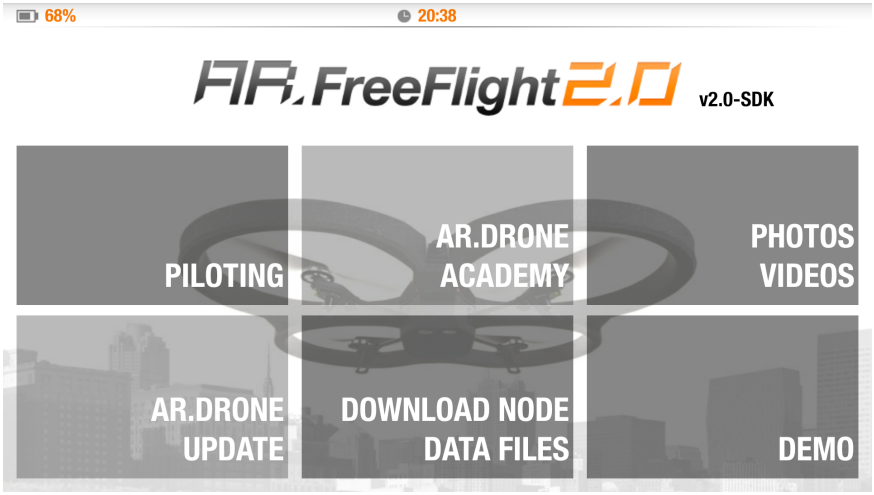

Figure 5.2: *AR Freeflight modified application*

Being an open-source platform we modified the AR Freeflight 2.0 Android application to communicate with the new modules installed on the drone and added a new feature on the main screen, seen in figure 5.2.

Android imposes the use of the background process class AsyncTask when using communication protocols such as http, ftp or sockets. This prevents the UI process from remaining stuck in communication and not responding to user actions.

The class offers 5 very important methods that can be overwritten, 4 running on the main UI process, that prepare data before and after execution, publish progress or simply cancel at any step, and 1 running on the background process.

5.3.1 Display information module

The Piloting screen of the application has been modified to display the received data from the drone.

The information displayed is comprised of the state of the dongle, the number of connected nodes and for each node the unique id, last connection time and the signal strength. Up to 9 nodes sorted in descending order after their signal strength are displayed. Figure 5.3 is an example of how the informations are displayed.

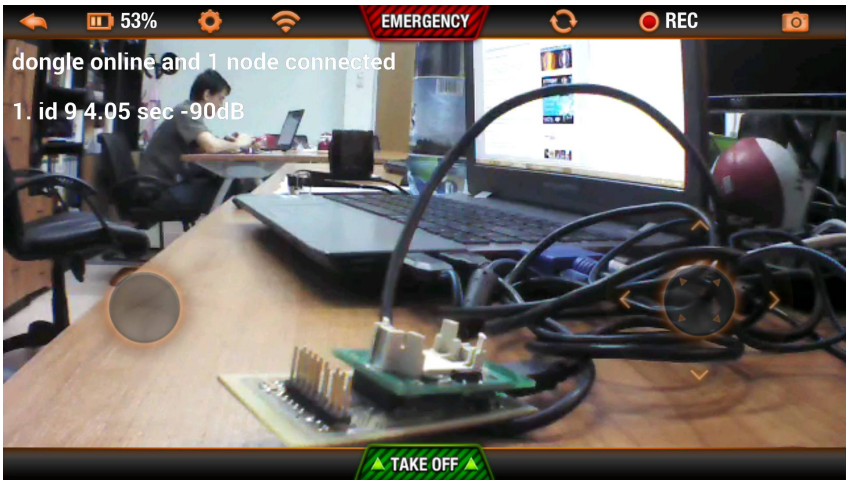

Figure 5.3: *AR Freeflight modified Piloting Screen*

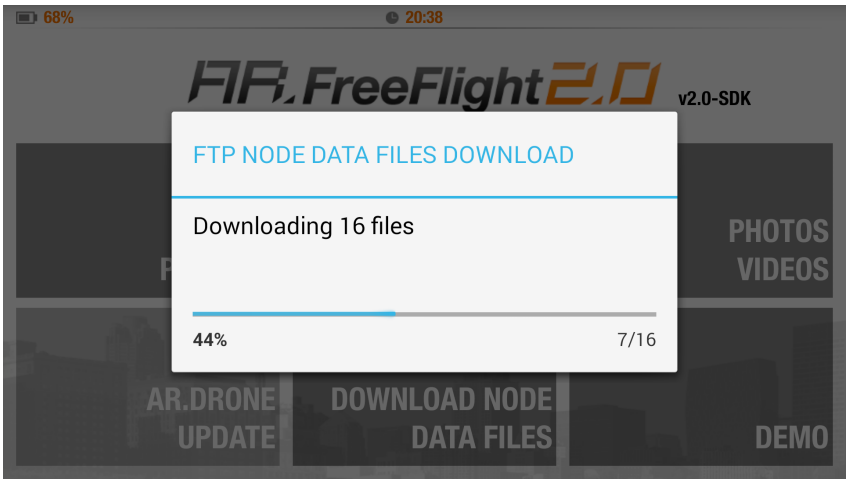

Figure 5.4: *AR Freeflight FTP downloading files*

5.3.2 FTP communication module

The drone has a built-in FTP server that can be configured to allow access to any folders/files on the drone. We have configured the drone so that the folder in which the data is saved can be accessed at any time using port 4242 by any device that has FTP client capabilities.

This feature was added to the Android application as well, to have a better out of the box experience.

The application will download all the files from the drone to the folder *ftp_node_data* from the local storage of the user's Android device and display the progress as in figure 5.4.

Chapter 6

Evaluation

The tests that we have conducted highlight the strengths of the platform but also reveal some of its weaknesses.

The characteristics that we considered to be most important are the maneuverability of the drone with the added components, the maximum range of the dongle and the number of packets sent by the nodes and not received.

6.1 Scenario description

The test scenario is relatively simple. Using the drone, we try to connect and collect the data from different nodes while measuring the identified characteristics.

We conducted multiple tests using a dongle with an 8 dBi antenna, two SparrowV3.2 nodes, one with a standard antenna and one with a 8 dBi antenna, were placed on a hill with the antenna directed upwards for a maximum range test. Another three nodes with 3 dBi external antennas were placed inside of a building. The nodes were configured to send a packet every second. The controlling device of the drone was a Samsung Galaxy S4.

Because both the drone and nodes use the 2.4 GHz band, the drone was configured to use channel 11 and the nodes were configured to use channel 1 in order to prevent any signal interference.

The packet loss test was performed using 4 scenarios. In all scenarios two nodes were used, one with an 8 dBi antenna and the other with -1 dBi antenna.

- Scenario 1: The drone hovered at a distance between 0 and 50 meters from the nodes
- Scenario 2: The drone hovered at a distance between 50 and 100 meters from the nodes

- Scenario 3: The drone hovered at a distance between 100 and 150 meters from the nodes
- Scenario 4: The drone hovered at a distance between 150 and 200 meters from the nodes

6.2 Results

6.2.1 Packet loss

The results of the test based on the 4 scenarios can be seen in figure 6.1. The high gain antenna performs excellently. For up to 100 meters distance there is no packet loss and at almost 200 meters distance, the packet loss is just 14 %. On the other hand, even though the -1 dBi could send packets at almost 200 meters, the loss is 75 %. The acceptable distance up until a -1 dBi antenna should be used is for a 100 meters connection because of a much smaller packet loss of 24%.

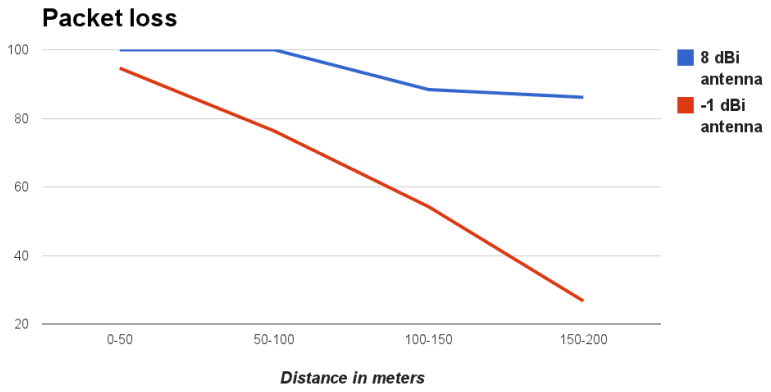

Figure 6.1: *The results for the packet loss test*

6.2.2 Signal range

The drone was able to find the nodes but, as depicted in figure 6.3, the obstructed nodes had a much smaller communication range than the ones placed on the hill. The range test results proved that the drone could receive a signal at a maximum distance of 325 meters from the node with the 8 dBi external antenna located on the hill and a clear line of sight, highlighted in figure 6.2. The other node on hill did manage to send data up to 100 meters. In the case of the obstructed nodes, even though they had a 3 dBi antenna, the distance did not exceeded the 65 meters mark. After this distance, the number of lost packets was to big to be

Figure 6.2: *Measured signal distance*

able to properly communicate with the nodes, even though, occasionally, at 90 meters, some packets were received.

The top mounted antenna worked, but for a better signal quality, the antenna should always be positioned on the bottom of the drone. In this way, the antenna will have a clear path to send and receive signals and not be blocked by the drone's body and its avionics. The problem with this particular drone model is that on its bottom there are sensors that help it determine the ground speed and ground altitude. If we place the antenna on the bottom it would obstruct some of the sensors.The other possibility, a vertical antenna, will increase wind resistance and make the drone too unstable.

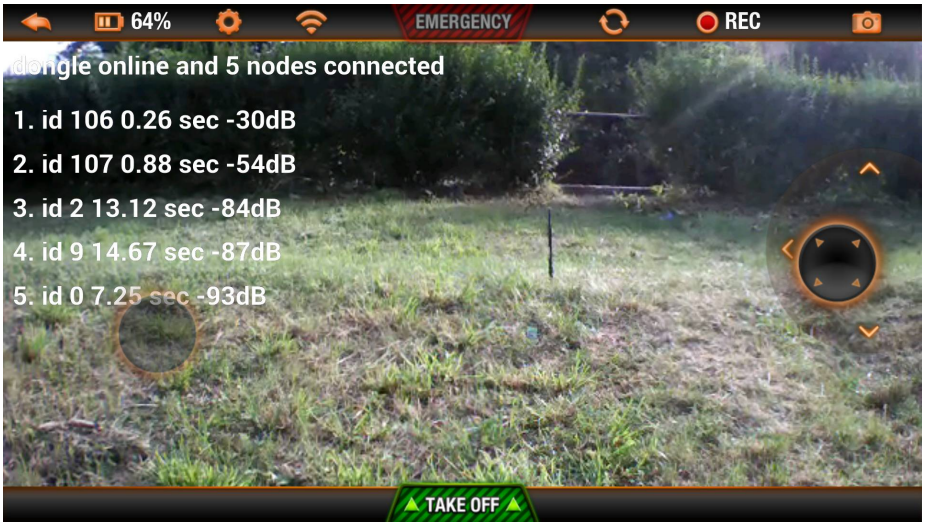

Figure 6.3: *Parrot discovering new nodes before takeoff*

6.2.3 Drone stability

Even though the antenna was mounted on top of the body, because it is a high gain antenna the signal received was very strong.

The dongle is mounted on the side of the drone due to the position of the USB connector. Because the dongle is not centered on the drone, a counterweight had to be glued on the opposite side on the outer shell to maintain the balance of the drone. The antenna, shown in figure 6.4, extended the signal range but also added weight.

The drone was relatively stable during tests, but a better stability and flight control could be obtained if the dongle were to be made smaller and a lighter antenna were to be used.

Figure 6.4: *Top mounted antenna for better signal quality*

6.2.4 Maximum height and maneuverability

The total added weight is 75 grams. Even though it does not sound that much, it does have a substantial effect on the drone. The maximum height it can reach is 50 meters versus the 75 meters it can reach without the added weight. The maneuverability is also affected, the drone response not being as sharp as before, but it is still good.

6.2.5 Problems

The kernel module needed for the dongle does not recognize the dongle if it is plugged in the drone when its powering up. The fix is to power up the drone and then plug in the dongle, but this is more difficult than it might appear because every time this action is performed the hull must be repositioned.

Chapter 7

Conclusions

Wireless Sensor Networks are expanding more and more because they help make our lives easier by giving us information about our surroundings. However, the standard way of creating wireless sensor network islands is not always feasible.

The main goal of this thesis was to bring an alternative solution to the problem of how data can be obtained from Wireless Sensor Network islands. We have proven that a slightly modified low-cost drone offers a good range, being able to communicate with the nodes from a big distance. The communication range is greatly affected by the antenna gain and the objects that obstruct the line of sight from the drone to the node.

The solution, a truly mobile gateway is a viable one and can be implemented with a relatively low cost. The technical knowledge necessary to operate the system is not more complicated than knowing how to use a smartphone.

7.1 Future Work

The proposed system can be improved in a number of ways. A feature that can be used in conventional wireless sensor networks is to determine the source of a communication failure. If the gateway detects that the network has a communication problem and not all of the previous nodes can be reached, a drone can use this information to search and find exactly which nodes are working properly and which nodes are not.

The Parrot AR.Drone 2.0 can perform autonomous flight with a GPS module, but only while it is still in the range of the Wi-Fi connection. This module can allow the drone to fly without the need to still be connected through Wi-Fi to a controlling device. A different autonomous flight mode can be implemented without a GPS module if the signal strength of the nodes is used to perform flight correction and determine the speed and direction of the drone.

Bibliography

[1] Arduino. https://www.arduino.cc. Accessed: 2016-06-05. [cited at p. -]

[2] Atmega128rfa1 datasheet. http://www.atmel.com/Images/doc8266.pdf. [cited at p. -]

[3] Json Standard. http://www.json.org/xml.html. [cited at p. 19]

[4] List of wireless nodes. https://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes. Accessed: 2016-06-20. [cited at p. 6]

[5] Parrot Drone. http://img.clubic.com/04862120-photo-parrot-ar-drone-2-0.jpg. [cited at p. -]

[6] Arduino-at86rf233. https://github.com/msolters/arduino-at86rf233. Accessed: 2016-06-05. [cited at p. 10]

[7] Atmel samd21 datasheet. http://www.atmel.com/images/atmel-42181-sam-d21_datasheet.pdf, . Accessed: 2016-06-05. [cited at p. 7]

[8] Atmel samr21 datasheet. www.atmel.com/Images/Atmel-42223SAM-R21_Datasheet.pdf, . Accessed: 2016-06-05. [cited at p. -]

[9] Ultra-capacitors. http://www.maxwell.com/images/documents/Ultracapacitors_Overview_Flyer_3000615-2EN.pdf. Accessed: 2016-06-20. [cited at p. -]

[10] USB ACM. http://wiki.openmoko.org/wiki/CDC_ACM, March 2010. [cited at p. 16]

[11] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002. [cited at p. -]

[12] Musat Andrei-Alexandru, Tudose Dan, and Deaconu Ioan. Geodynamics monitorization using wireless sensor networks. *CSCS*, 2015. [cited at p. -]

[13] Faycal Ait Aoudia, Matthieu Gautier, and Olivier Berder. Fuzzy power management for energy harvesting wireless sensor nodes. In *IEEE International Conference on Communications (ICC16)*, 2016. [cited at p. 5]

[14] Th Arampatzis, John Lygeros, and S Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724. IEEE, 2005. [cited at p. -]

[15] Aline Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden*, 2005. [cited at p. -]

[16] ALEX BRACETTI. The 10 Best Drones You Can Buy Right Now. http://www.complex.com/pop-culture/2013/03/10-cool-drones-you-can-buy-right-now/parrot-ardrone-20, March 2013. [cited at p. -]

[17] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005. [cited at p. 19]

[18] Aysegül Tüysüz Erman, LV Hoesel, Paul Havinga, and Jian Wu. Enabling mobility in heterogeneous wireless sensor networks cooperating with uavs for mission-critical management. *Wireless Communications, IEEE*, 15(6):38–46, 2008. [cited at p. -]

[19] Longbo Huang and Michael J Neely. Utility optimal scheduling in energy-harvesting networks. *IEEE/ACM Transactions on Networking (TON)*, 21(4):1117–1130, 2013. [cited at p. -]

[20] Raja Jurdak, Kevin Klues, Brano Kusy, Christian Richter, Koen Langendoen, and Michael Brünig. Opal: A multiradio platform for high throughput wireless sensor networks. *Embedded Systems Letters, IEEE*, 3(4):121–124, 2011. [cited at p. 6]

[21] Anibal Ollero, Markus Bernard, Marco La Civita, Lodewijk van Hoesel, Pedro J Marron, Jason Lepley, and Eduardo de Andres. Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1–6. IEEE, 2007. [cited at p. -]

[22] Omur Ozel, Kaya Tutuncuoglu, Jing Yang, Sennur Ulukus, and Aylin Yener. Transmission with energy harvesting nodes in fading wireless channels: Optimal policies. *Selected Areas in Communications, IEEE Journal on*, 29(8):1732–1743, 2011. [cited at p. -]

[23] Omur Ozel, Jing Yang, and Sennur Ulukus. Optimal broadcast scheduling for an energy harvesting rechargeable transmitter with a finite capacity battery. *Wireless Communications, IEEE Transactions on*, 11(6):2193–2203, 2012. [cited at p. 5]

[24] AR Parrot. Drone. *Available: ardrone.parrot.com*, 75, 2012. [cited at p. -]

[25] Joaquin Recas Piorno, Carlo Bergonzini, David Atienza, and Tajana Simunic Rosing. Prediction and management in energy harvested wireless sensor nodes. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 6–10. IEEE, 2009. [cited at p. 4]

[26] Chris Savarese, Jan M Rabaey, and Jan Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 4, pages 2037–2040. IEEE, 2001. [cited at p. -]

[27] Steven K Teh, Luis Mejias, Peter Corke, and Wen Hu. Experiments in integrating autonomous uninhabited aerial vehicles (uavs) and wireless sensor networks. 2008. [cited at p. -]

[28] João Valente, David Sanz, Antonio Barrientos, Jaime del Cerro, Ángela Ribeiro, and Claudio Rossi. An air-ground wireless sensor network for crop monitoring. *Sensors*, 11(6):6088–6108, 2011. [cited at p. -]

[29] Andrei Voinescu, Dan Tudose, and Dan Dragomir. A lightweight, versatile gateway platform for wireless sensor networks. In *Networking in Education and Research, 2013 RoEduNet International Conference 12th Edition*, pages 1–4. IEEE, 2013. [cited at p. -]

[30] Zhe Wang, Vaneet Aggarwal, and Xiaodong Wang. Iterative dynamic water-filling for fading multiple-access channels with energy harvesting. *Selected Areas in Communications, IEEE Journal on*, 33(3):382–395, 2015. [cited at p. -]

[31] Jing Yang and Sennur Ulukus. Optimal packet scheduling in a multiple access channel with energy harvesting transmitters. *Communications and Networks, Journal of*, 14(2):140–150, 2012. [cited at p. 4]

List of Figures

Listings