THE FACULTY OF AUTOMATIC CONTROL AND COMPUTER SCIENCE

Designing a low power node for wireless sensor networks

student: Ioan Deaconu
coordinator: Dan Tudose
January 27, 2016

1 INTRODUCTION

The wireless sensor networks are developed and implemented in order to ease our life by offering feedback regarding the surrounding environment. In order to achieve this, the technology must be able to help us, ease of use being the most important feature.

The biggest problem of a wireless device is the battery life. Because they are small, they can be placed in areas where a constant source of power is difficult to provide, batteries being the only solution.

Due to ever developing technologies, newer faster, low power and at a lower price technologies are released each year. In this paper we will present an existing node and try to find ways to improve the power consumption. In architecture we describe the new cpu and compare it to the current one. In the results section we can discover the power benefits brought by the new design and try to come with new ideas that will improve the power consumption even further.

2 RELATED WORK

In the previous research, we have used sparrow v4 in order to gather data from the accelerometer and send it to the base station for further processing. We have tried to obtain the lowest power consumption possible, but due to hardware limitations, the average power consumption was around 2.1 mW , where 1 mW was the power needed to keep the accelerometer powered on. [1]

Due to the small battery, a CR2032, which has a capacity of approximatively 220 mAh, the life of the node was of approximately 15 days. This small autonomy indicated us that is was time to design a new wireless node, one that can be small and at the same time, last more than 2 months on this battery.

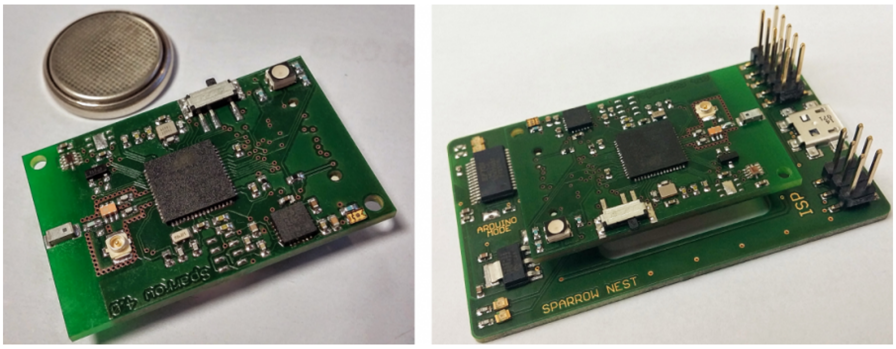

Figure 2.1: Sparrow V4 node

3 Architecture

3.1 Hardware

When designing a new device, backwards compatibility is necessary in order to be able to utilize existing technology. Because older sparrow sensors are already being used [4], wireless compatibility is a very important feature needed to be kept.

We also wanted to be able to design a smaller node if possible, so single chip approach must be taken. After doing some research, we have found the best possible candidate to be Atmel SAMR21[2]. The new cpu has many advantages over the Atmega128RFA1[3], including power consumption, speed and a smaller capsule which can allow a smaller design for the nodes. But most important, the radio is fully compatible with ZigBee protocol.

In the table 3.1 we present the main differences between the cpu of the Sparrow V4 and the proposed cpu.

The accelerometer is a big cause of the high power consumption, so we are testing a lower power accelerometer, LSM330DLC, which draws just 11 uA for 50 samples per second instead of 350 uA for the previous accelerometer LSM9DS0. A draw back is represented by the fact that the precision is just 12 bit instead of 16 bit for the previous one, but in order to reduce RF power consumption, over the network we transmit just 8 bit for each axis. The sampler rate is kept at 25 samples per second, which means that every second, the RF has to send a frame with a PSDU of at least 75 bytes.

Due to the fact that we are testing new components, we have used development kits for both SamR21 and LSM330DLC. For SAMR21 we use Atmel SAMR21 Xplained pro and for LSM330DLC we use STEVAL-MKI122V1. SAMR21 Xplained Pro kit allows the power consumption of the cpu with RF to be measured using an external device power, for example an

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

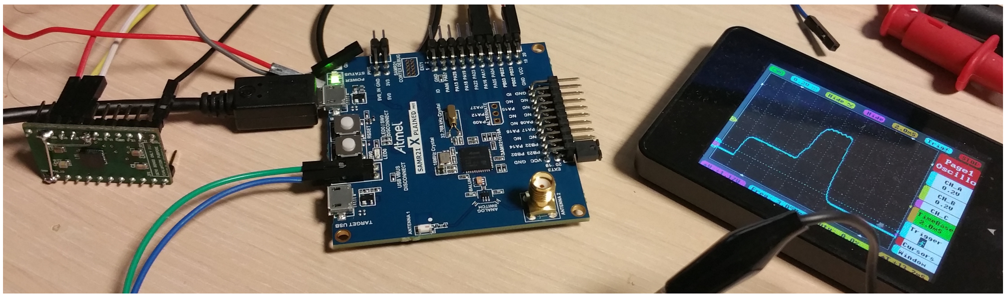

Figure 3.1: Development kits connected together

oscilloscope. The accelerometer has been measured separately and confirmed to draw 11 uA at 50 samples per second.

3.2 Software

The accelerometer has a FIFO buffer of 32 samples which allows to request data only once a second for a 25 sample rate. Data can be gathered by using I2C or SPI interface. In this experiment we are using I2C interface clocked at 400kHz. The node will send 8 bit raw accelerometer data every second.

In order to better determine the power efficiency between the two cpus, we have created two small programs, one that test the integer performance and other that tests the branch predictor. In this way we can determine if the cpu is better suited for computational or conditional algorithms.

```
static volatile int a, b, c;
void iterate() {
        a++;
        b = a + 1;
```

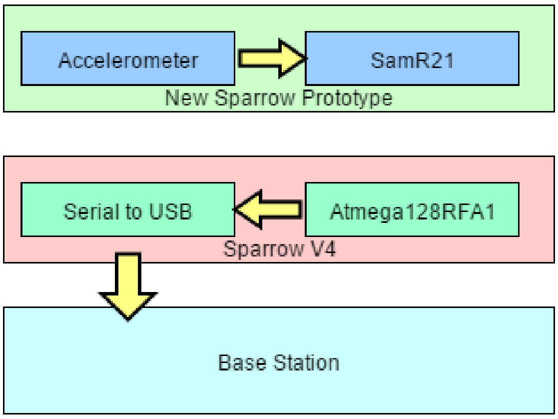

Figure 3.2: Software Architecture

```
        c = 4 * b / a;
}

void branch() {
        if ( a % 3 == 0) {
                a += 5;
                b += 3;
        } else {
                if( b % 5 == 0) {
                        b++;
                        a++;
                } else {
                        iterate();
                }
        }
}
```

The data is packet in a frame that contains the total content length, the node id, the accelerometers samples and the crc for verifying data integrity. The total size of the sent frame is 80 bytes.

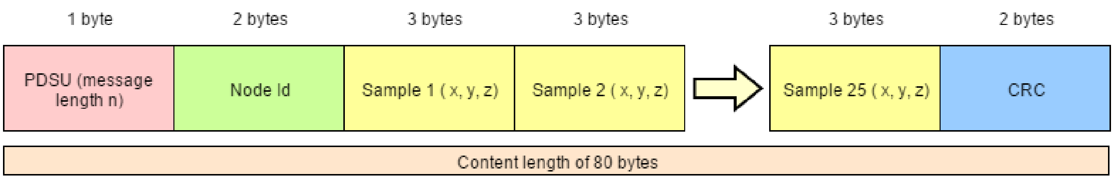

Figure 3.3: Frame content

4 Results

The average power consumption obtained with an unoptimized software version is 750 uW. The software can be further optimized by staying as much as possible in sleep and waiting for peripherals to trigger an interrupt when an event has occurred, instead of doing busy waiting. Another optimization is to replace the I2C with a SPI connection at 5 MHz or higher between the CPU and accelerometer. This would reduce the transfer time from 9 ms to less 1 ms. When sending the data to transceiver, DMA transfer would be preferable, during which the cpu can sleep and be waken when the transfer is done.

If this optimizations will be implemented, the power could be reduced from 750uW to 450 uW and the estimated battery life for a CR2032 battery will increase from 36 days to 61 days. Because the components use a linear voltage regulator and the fact that the datasheet mentions a smaller current draw at 1.8V over 3V, a dc-dc step-down converter can be used to further reduce the power consumption. With an efficiency of 85%, the new power consumption should decrease to a value lower or equal to 320 uW, increassing the battery life to 87 days. Sacrificing the size for a bigger battery, of 1000 mAh for example, would allow the node to run for almost 395 days, or just a little over an year, with data being send at every second.

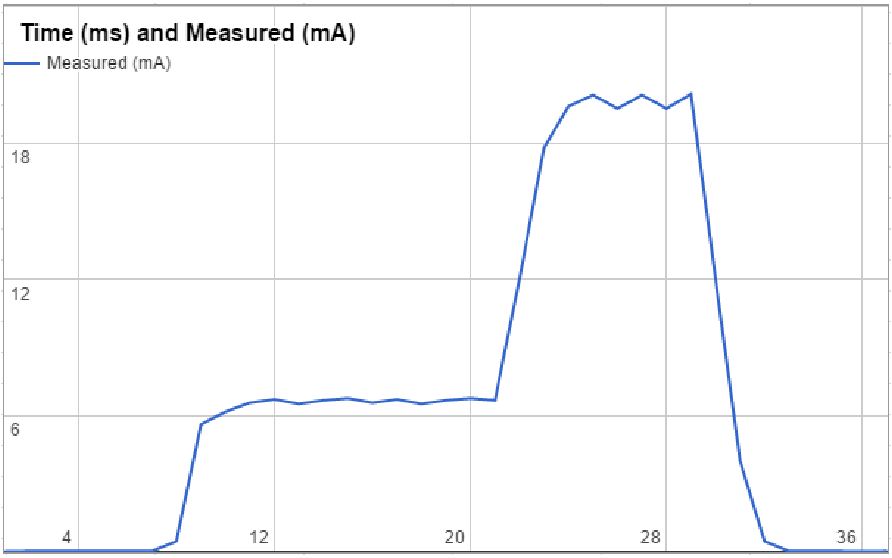

Figure 4.1: Measured power consumption

The power advantage of the SAMR21 can be expressed in the efficiency of running integer operations. For the same integer operations, Atmega128RFA1 consumes 274nJ, while SAMR21 is able to consume 49 nJ, less than 5 times. The worst case scenario is represented by a code with little computation and a lot of jumps to test the branch predictor. Even in this case, the power advantage is substantial, almost two and a half times better then Atmega128RFA1. This power advantage could lead to computational offloading, which would allow to process the data directly on the node and send the result directly to the base station with less power than to the send large raw data.

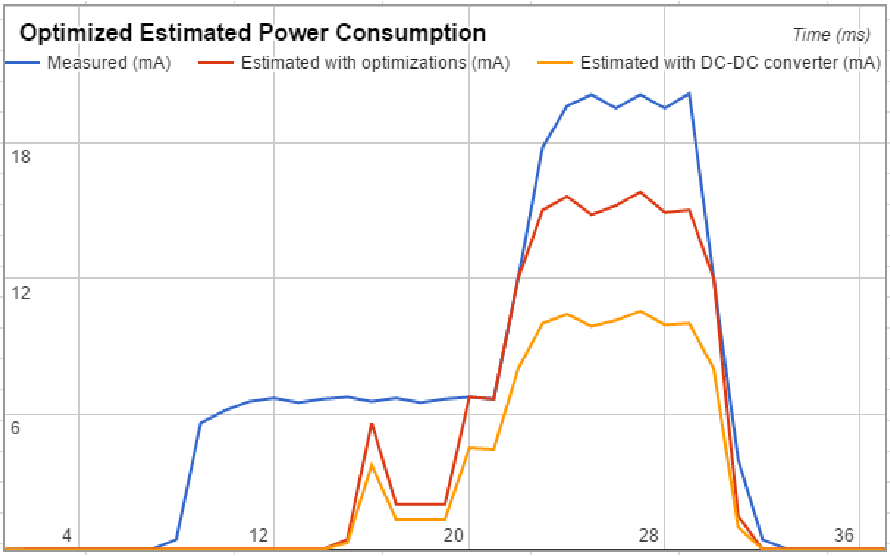

Figure 4.2: Estimated gain brought by optimizations

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 4.1: Speed comparison

5 Conclusions and Future Work

We have proven in this paper that a lower power node that sends large amount of data every second is feasible, with increased in computational power too.

As a continuation of this research, we intend to build the final node that takes into considerations the results of this paper. The node will contain an accelerometer, light sensor, pressure sensor, temperature sensor and if possible an air quality sensor. Further sensors can be added, for better understanding of our surrounding environment.

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

Table 4.2: Energy efficiency comparison

References

[1] D. T. Andrei Musat, Ioan Deaconu. Geo-dynamic monitoring using wireless sensor networks. 2015.

[2] Atmel. *Atmel SAM R21 Datasheet*, 2015, accesed 20 january 2016.

[3] Atmel. *Atmel Atmega128RFA1 Datasheet*, 2015, accesed 21 january 2016.

[4] A. Voinescu. Wireless sensor networks, energy considerations for protocol and application design. 2016.