PowerEmb - Power Aware System for Small Embedded Systems

João M.P. Gonçalves
Instituto Superior Técnico
IST - UTL
Lisboa, Portugal
joao.m.goncalves@ist.utl.pt

Rui Manuel Rocha
Instituto de Telecomunicações
IT – Lisboa
Av. Rovisco Pais, 1
Lisboa 1049-001 , PORTUGAL
rui.rocha@lx.it.pt

Carlos Manuel Almeida
Department of Electrical
and Computer Engineering
DEEC - AC Computadores
Avenida Rovisco Pais
1049-001 Lisboa, PORTUGAL
carlos.r.almeida@ist.utl.pt

***Abstract*—The energy resources in a Wireless Sensor Network (WSN) are scarce on all nodes typically running on batteries. Hence, energy management is a priority. In this work, this concern is approached knowing that there are no single hardware or single software solutions. There must be an effort at both levels to reduce the nodes energetic consumption and thus prolong their life time. One power saving method that involves both levels is Dynamic Voltage Scaling (DVS). It has been proved to be effective in reducing the energetic consumption in high performance systems. However its implementation in small embedded systems with scarce resources, is still unknown.**

This work exploits the usability of DVS in small embedded systems using a WSN node as a test platform. The Group of Embedded networked Systems and Heterogeneous Networks at Instituto Superior Técnico, has developed an energy efficient sensor node platform (MoteIST++). The sensor node is also modular in terms of communication modes. It implements at the hardware level solutions for a more efficient energy management. This work presents the redesign of this platform with a more powerful new microcontroller and solutions for energy measurement. Also, the new platform design allows for the DVS exploitation. The sensor node runs the open-source TinyOS Operating System (OS), which implied for this work to also develop the port of the OS to the new sensor node's microcontroller.

In this project, a contribution is made to both the university's working Group of Embedded networked Systems and Heterogeneous Networks and to the TinyOS community.

Keywords: Energy Management; Dynamic Voltage Scaling; Wireless Sensor Networks; TinyOS

I. Introduction

The quick growth of intelligent portable devices has been in part motivated by the use and utility that people find in devices such as smart phones, iPods, photographic and digital cameras, etc. Processing speed and autonomy are decisive factors for the choice of a device by a consumer. However, the continued miniaturization and increased computation power of these embedded systems can compromise the device's autonomy if no measures of energy management are taken. Therefore, to meet the consumers needs, progress must be made in the direction of rationalizing the available energy in batteries.

In this context, an active research area for the last few years, has been the Wireless Sensor Networks (WSN). A WSN consists of spatially distributed autonomous nodes, also called as motes, which are used to monitor physical or environmental conditions, such as temperature, sound, vibration, etc. A critical constraint on sensors networks is that sensor nodes employ batteries and are usually deployed unattended and in large numbers, which makes it difficult to change or recharge the batteries of the motes. Typically these motes are very scarce of resources but are still expected to have a long life span, ranging from days to years [1].

When any form of energy harvesting (e.g solar cells) is present in these platforms, it can minimize the impact of energy consumption in the motes batteries. Work has been done towards this approach [2]. Also, there are approaches such as the one in [3], that claim that energy management strategies should include policies for an efficient use of sensors. Other approaches however, taking that the main source of energy consumption in a WSN is comunication [4], focus on minimizing the energy spent by the motes radio [5]. There is no optimal solution to solve the problem of energy consumption. All of these factors should be taken into account when approaching this problem, and continuous improvements at both the Hardware (HW) and Software (SW) level should be pursued.

In high performance systems, significant improvements in the overall energetic consumption of a system can be achieved using a method like Dynamic Voltage Scaling DVS [8], [9]. DVS is a low-power technique in which the necessary supply voltage of the processor is adapted according to its operating frequency value. Currently, several variable-voltage microprocessors [10], [11], [12] have been developed and many DVS algorithms proposed [13], [14], [6], [15], [16], some of which are to be implemented in systems with real-time requirements [6], [15], [16]. The DVS is a solution that comprises both the hardware and software levels of energy management. Such technique has not yet been tested in devices with scarce resources such as the WSN motes. Even though communication and sensing may be of the highest importance, all the solutions should be considered in the pursue of the optimal energy efficient system.

In the Group of Embedded networked Systems and Heterogeneous Networks there has been the tradition of giving special care to the problem of energy consumption in WSN motes.

The group maintains a project/test-bed called the Tagus-SensorNet [21] and recently, the MoteIST++ platform has been developed in GEMS. The produced mote, is modular in terms of communication modes and implements solutions at the hardware level, for a more efficient energy management.

Its MCU is form the 2nd family of the TI MSP430 microcontrollers. Prototypes of the mote with a more recent MCU from the 5th family of the MSP430, have been designed but did not pass to a production stage. This prototype was named MoteIST++s5 due to its 5th family MSP430 microcontroller.

GEMS develops the applications for the motes, mainly with the TinyOS [18] OS. However, the current main release of the OS does not support the 5th family of MSP430 which caused the prototype MoteIST++s5 to be put aside. Recently the TinyOS community's work of porting the newer families of the MSP430 microcontrollers has been very active. This means that the port of the OS to the 5th family of MSP430 might be one step away and the previous work with MoteIST++s5 can be continued.

The purpose of this work is to exploit a combined solution of both software and hardware for a better energy management in a WSN mote. This solution is the DVS method. It requires at the hardware level the possibility to dynamically change the applied supply voltage and at software level the processes of decision and choice for the CPU's operating frequency. Given the previous work with MoteIST++s5 containing a 5th family MSP430 microcontroller, a start point is established to the exploiting of the DVS method that will be presented in this thesis.

The rest of the paper is organized as follows: Section II presents related work on Dynamic Voltage Scaling; Section III describes the upgrades performed in the new MoteIST platform to make it more energy efficient and to support DVS; Section IV presents the MoteIST-Testboard that will serve as test platform for DVS implementation; Section V presents the TinyOS port to the platform's MCU and the developed application to validate DVS; Section VI describes the tests performed to validate the DVS method; Section VII considers the possibility of the MCU entering low-power modes and estimates the energy savings for an application that uses DVS and low-power modes simultaneously; Section VIII concludes this paper with an overall view of the performed work.

II. Related Work

Pillai and K. G. Shin [6] developed real-time dynamic voltage scaling (RT-DVS) algorithms that fit under the InterDVS designation. The InterDVS algorithms commonly consist of two parts: the slack time estimation of a task, and the slack time distribution to the following tasks. The slack times that can be identified statically are the extra times available for the next task (static slack times), while run-time variations of the task execution can only be identified during run-time (dynamic slack time). The most commonly used method for estimating a static slack time is the maximum constant speed, where the lowest possible clock speed that guarantees the possibility of executing the task in time is defined.

The prototype is implemented on the PC architecture and the platform is a Hewlett-Packard N3350 notebook computer with an AMD K6-2+ processor that has PowerNow! technology [10] and maximum operating frequency of 550 MHz. As the voltage and frequency are independently set, a mapping of frequencies and their associated voltage values were determined experimentally. The algorithms are implemented as extension modules to the Linux 2.2.16 kernel. The Periodic RT Task module provides the support for periodic real-time tasks in Linux and the RT Scheduler w/RT-DVS module defines the scheduling policies and the algorithm. PowerNow module handles the access to PowerNow!. The module provides a high-level interface for changing the bits in the processor's register that changes the frequency and voltage. The measurements of the energy consumption are made for the whole system that includes the CPU, memory and peripherals. Only the LCD back light was disabled. A current probe was used and an oscilloscope provided the product of the current and voltage supplied. The oscilloscope was used in transient mode and the power measurements averaged over 15 to 30 second intervals. The algorithms tested were the Earliest Deadline First (EDF), Static Rate Monotonic (RM), Cycle Conserving EDF and Look-Ahead EDF with a set of 5 tasks that always consume 90% of their worst case computation allocated for each invocation. Although the measurements made were not specific of the processor but instead of the total system, the results were significant, resulting in 20% to 40% reduction in power consumption.

III. MoteIST++

The MoteIST++ [7] is a platform designed for use in GEMS [17] and is designed to be modular in terms of communication modes. This mote is equiped with the the Texas Instruments MCU MSP430F2419 but for this work there was a need to improve this platform to a more recent MSP430 MCU, to support the study of the DVS method.
Figure 1 shows the existing prototypes of MoteIST++.

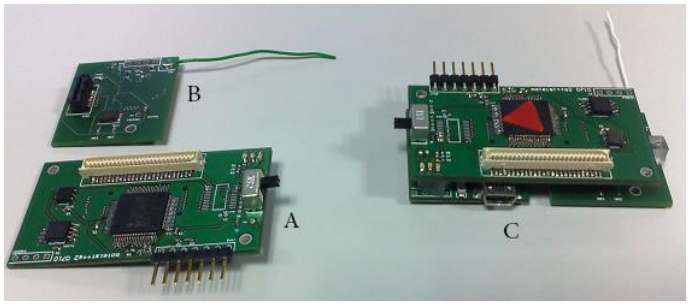

Figure 1. Prototypes of the existing MoteIST++. A) MoteIST++; B) ZigBee CB; C) MoteIST++ with ZigBee Communication Board (CB) and USB CB attached. [7]

A. MoteISTx5 platform

The main purpose of a new platform was to change its MCU. However, other capabilities have been added that further justify its fabrication. The block diagram in Figure 2, adds to the design of the existing prototype a microSD card socket, a current sensor circuit for energy measurement and a new MCU, the TI MSP430F5438A. The USB CB remains unaltered and the ZigBee CB adds a new designed inverted F antenna that is printed on the circuit board. Presently, the platform has been sent to be fabricated and assembled, therefore there are no photos available.

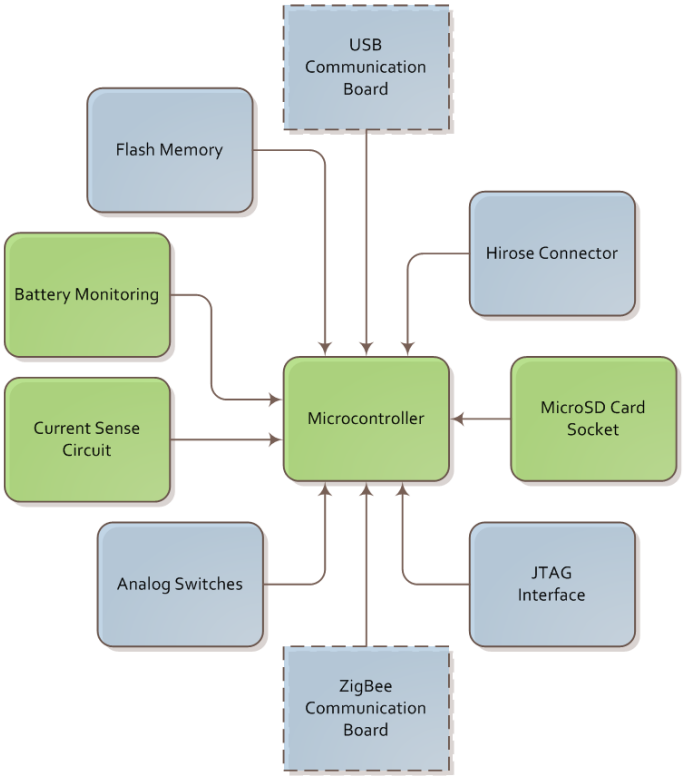

Figure 2. MoteISTx5 Block Diagram.

The focus on the x5 family of this MCU aims at its Power Management Module (PMM). The PMM has an integrated Low Drop Out (LDO) voltage regulator to produce a secondary core voltage (VCORE) from the primary one applied to the device (DVCC). This VCORE supplies the CPU, memories(flash/RAM), and the digital modules, while DVCC supplies the I/Os and all analog modules. The PMM is a key feature to implement the DVS method because it avoids having external voltage regulators on the platform.

Besides the MCU, the new MoteISTx5 platform has an additional microSD card memory socket. The microSD card memory extends the possibility of higher data acquisition and gives portability to that data. The previous flash memory is kept because in normal applications, that do not need that much of storage space, it has a reduced current consumption when compared to the microSD card.

One important component is the current sense that is performed by a current sense amplifier. This allows the platform to know how much current it is consuming at a given time. This gives to the application developer a wide range of software implemented decisions to be taken, i.e. choices can be taken to reduce the current consumption in a case where the estimated life time of the batteries is low.

The analog switches are an energy saving solution already implemented in previous MoteIST++. They are a bundle of digitally controlled analog switches that wire all the devices in the platform and those connected to the CBs to the battery supply. They are to be kept, because it's the most basic, yet very effective way of controlling which devices receive energy.

B. Current sensing and battery monitoring

The current sense circuit designed for the MoteISTx5 is a precision difference amplifier. The OPAMP chosen is ADA4051 form Analog Devices. It is a Rail-to-Rail Input/Output amplifier to allow for the high common mode voltage (V_{DD}) applied in the OPAMPs positive terminal. It has a very low offset voltage of $15\mu V$ and the supply current is $13\mu A$.

The current supplied to the platform goes through a precision sense resistance of 4.99Ω. The voltage difference in the sense resistance (R_S) is amplified and applied to an MCUs ADC input channel (Figure 3).

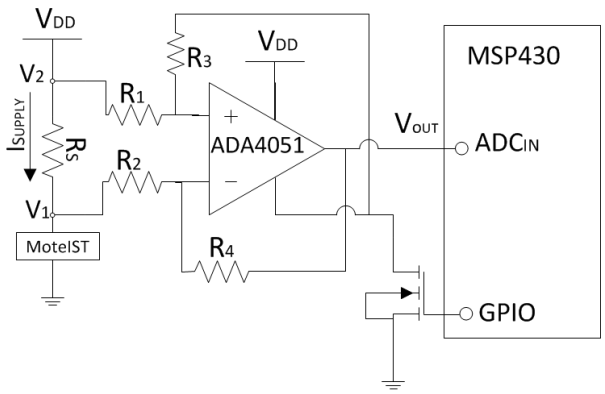

Figure 3. Sensing circuit with ADA4051.

Because there will always be current flowing from V_{DD} to ground passing through R_3, a MOS transistor is used as a switch to open the circuit whenever the circuit sense is not being used. The MOS gate is driven by a dedicated GPIO from the MCU. This is illustrated in Figure 3.

To sense the batteries voltage level a resistive divider was used at the ADCs input channel. This is necessary because if the batteries are new, they supply 3V to the platform, and due to the ADCs saturation voltage of $2.5V$ the voltage needs to be lowered. It can then be adjusted by software to obtain the real value.

IV. The test platform

The work on TinyOS [18] port started with the MSP430F5438 prototype board in MoteIST++ [7]. However this prototype had several problems that made the development impossible and a new prototype board had to be designed.

The new test board was designed not to replicate the old one, but rather to allow the continuation of the work in TinyOS. It was designed to be easily inserted in a breadboard, allowing for expansion with the routed GPIO pins to the connector pinouts. The board allows for an easier hardware debug.

The board was fabricated in IST using a photografic process in a dual layer PCB. The size is not of a big concern because this is to be used in the breadboard where it can be further expanded. Figure 4 presents the fabricated test board mounted on the breadboard with the additional current sensing and a conversion from unipolar signals to RS232 bipolar levels conversion needed for serial communication with a computer and also, the prototype of the 802.15.4 radio board with the antenna printed on the PCB.

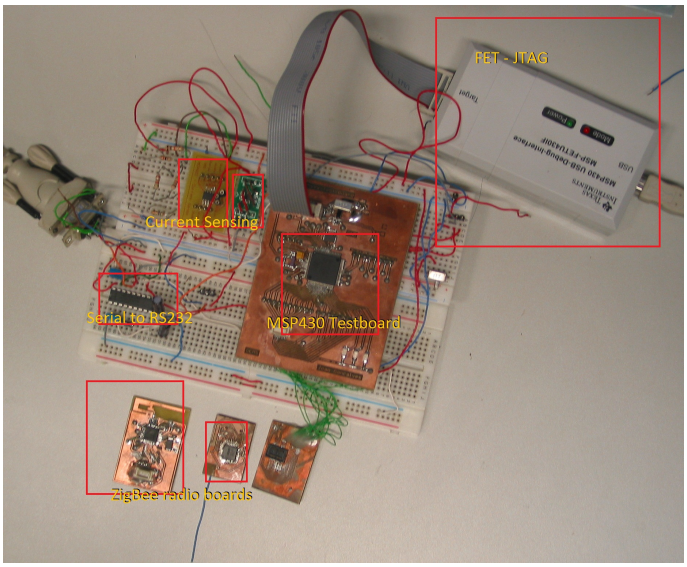

Figure 4. The Testboard with MSP430 and the zigbee boards.

V. TinyOS in MSP430x54xA

To get TinyOS functional for the Testboard containing the MSP430F5438A MCU it was necessary to upgrade components to support the x5 family. For the desired DVS tests, components had to be created and tested with applications designed for this purpose. The port to the fifth family of MSP430 started with resource to the existing port for the cc430f5137. This port is hosted in the TinyProd [20] repository, that is a GIT based TinyOS repository maintained by the academic TinyOS community. The repository has a development branch for MSP430, called msp430-int-pu (msp430 integration proposed updates). It is a work in progress that has already several developed code for the msp430x5xx family. It was created a sub-branch of the msp430-int-pu branch where the code developed to support the MSP430F5438A is maintained and part of it has already been merged to the main development branch. There were two different levels to the port: chip specific and platform specific. The chip specific comprehends the MSP430 dependent components that drive the various parts of the MCU. The platform specific is related to the platform and provides another level of abstraction above the the chip specific code. There was also the application level of development that was needed to create the scenario in which the DVS was tested.

A. Chip specific level

The msp430 chip specific directory implements drivers for the general msp430 MCUs. Some families of this microcontroller have additional features and even the architecture has differences. Figure 5 shows the msp430 directory from the TinyProds MSP430 development branch. Each subdirectory implements a driver for each hardware component in the chip. The specifics of each family are in the subdirectories with the family range name (x1x2, x1xxx, x2xxx and x5xxx).

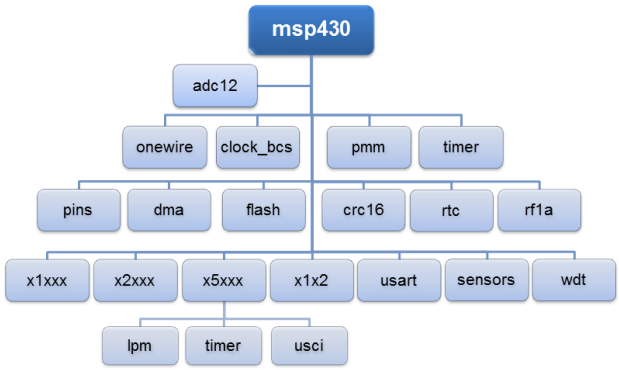

Figure 5. The msp430 chip specific directory.

The first contribution to the development branch was the mapping of pins that had to be made for the specific MSP430F5438A architecture. Secondly, the PMM driver was completely redesigned, because the existing driver had only a simple interface to set the internal core voltage. This was redesigned to allow for a level of abstraction that takes the CPU frequency into account whenever there is an attempt to change the internal voltage.

There are four operating modes each corresponding to an internal core voltage (VCORE). For each frequency demanded, the developed component associates that frequency to one of the four operating modes and then checks the actual VCORE level. If the frequency is in the range of frequencies supported for that operating mode, the PMM remains in the same mode. When the frequency demanded is higher and outside the range of frequencies supported by the actual operating mode, the VCC is checked first, by the supervisory circuitry of the PMM, for the possibility of a VCORE increase.

Figure 6 shows the functional block diagram of the driver.

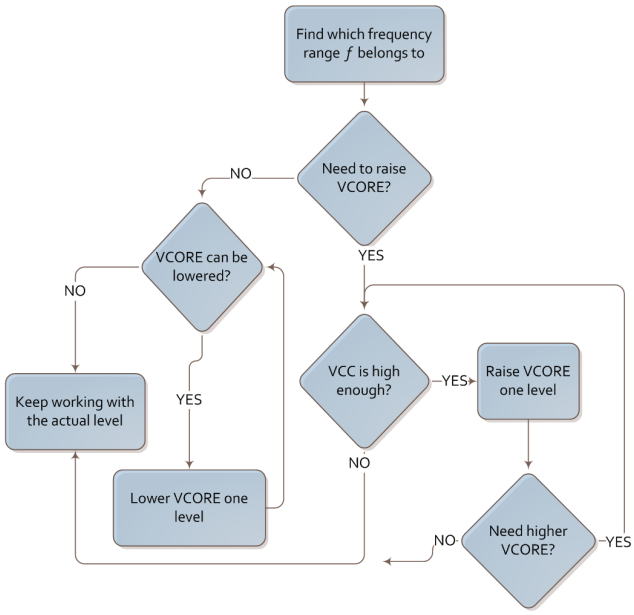

Figure 6. Block diagram for the implemented PMM driver.

The actual timer drivers for MSP430 in TinyOS are not designed to handle frequency changes when a PMM is present on the MCU. The level of the PMM is not taken into account and this lead to the creation of a new component in *msp430/x5xxx/timer*, the *Msp430FreqControl*. The component was needed to give a simple abstraction for the developer so it is possible to easily change the CPUs running frequency at demand. This component uses the redesigned PMM driver to automatically scale the internal MCUs voltage to its minimum allowed for a given frequency. When setting a determined frequency, the *Msp430FreqControl* traces back the timer system configuration to find the source of MCLK that drives the CPU. It checks the actual frequency of the CPU and then decides if there is the need to increase VCORE before raising the frequency. If the frequency to be set is lower than the actual, only after the frequency is changed VCORE is decreased. The component provides a simple interface that receives only the desired frequency. The clock system and PMM configurations is automatically set.

B. DVS test application

At the application level the DVS test application was developed. This application needs the implemented *Msp430FreqControl* and another component created specifically to post heavy work for the CPU.

The DVS test application uses the internal MSP430 ADCs to measure the total current consumed by the platform, the battery voltage and the VCORE pin that outputs the internal voltage supplied by the LDO voltage regulator. In order to verify the variation of current consumption when changing frequencies the MCU must not be idle. To accomplish this need, a component was created to calculate *num_iterations* numbers of the Fibonacci sequence, where *num_iterations* is a 32 bit integer passed by the application.

The task signals an event for each time it concludes an iteration. The event decides if the task is to be reposted (calculating the next sequence number) or if it has reached the last iteration. At the last iteration an event is signalled to the application. This event indicates that the sequence has finished processing, and passes the start and end time for the application.

The designed application puts much effort in guaranteeing that the ADC readings are done in the period that the Fibonacci sequence is calculated. It puts the CPU busy obtaining the Fibonacci numbers and it performs the same calculation *num_iterations* times in a sweep for the frequencies of 24MHz to 1MHz with a step of 1MHz.

The readings values are sent over the UART to the computer. The values sent are the start and stop timings of each ADC reading and each Fibonacci calculation. Information on current consumed, battery voltage and VCORE pin is also sent. Every string sent over the UART, corresponds to a frequency step reading in the sweep application. All the ADC values are sent as raw, in a 12 bit format. There are 3 samples for each of the 3 channels being read in the ADC. Only a mean is performed to the samples, the calculations needed to interpret the values are done after, by the computer. The user has to configure the correct baud rate and serial port in the computer and save the strings to a *.csv* file. The readings can then be imported by a proper software program to evaluate the results.

VI. DVS TESTS

To test the DVS in MSP430, and because the frequency is limited by the supply voltage, the test was performed backwards from $24MHz$ to $1MHz$ with a $1MHz$ step between frequencies. The presented results were obtained with the application, with static maximum supply voltage *versus* running with the dynamic reduction of supply voltage as the frequency is decreased.

It was necessary to know how much work load should be given to the CPU, in order for it be be kept busy during an ADC conversion and for all frequencies it was submitted. Tests were performed with the application calculating 5000 Fibonacci numbers, for each of the frequencies. The ADC performs 3 conversions for each of the 3 channels and calculates the mean result for each channel. The results are presented in Figure 7. The reason for such high number of calculations is because the total ADC sampling and conversion time is in the order of $30.5\mu s$, which is needed for precise conversions. This way the CPU is always kept busy during the ADC conversions. The current measurements with the ADC have been correctly measured during the active CPU processing time. In Figure 7 the current is shown as a line decreasing with the frequency. The blue bars corresponding to the ADC conversion time, are always below the red bars of the processing time.

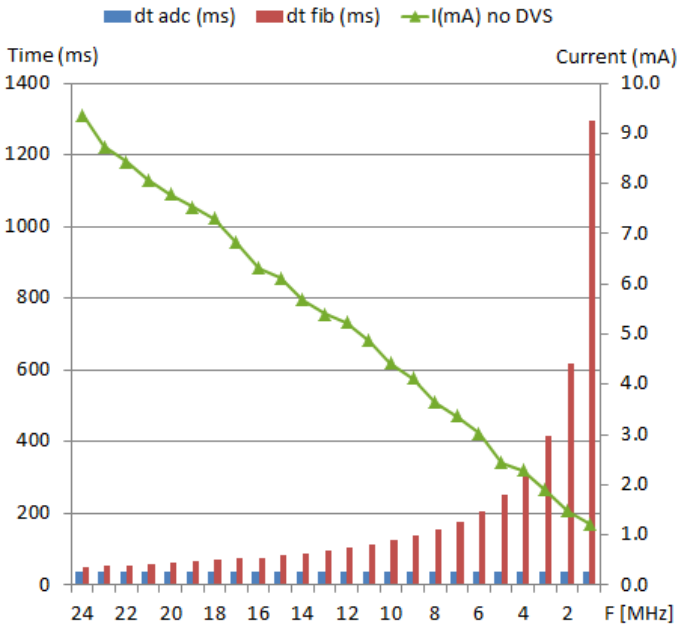

Figure 7. The frequency sweep calculating 5000 numbers of the Fibonacci sequence at each step.

A. Sweep frequency measurements

After the amount of processing required was calculated, the behaviour of the DVS current consumption was studied. Measurements were taken with the PMM in its higher mode, which allows to work at the higher frequencies. Also measurements were taken while dynamically reducing the PMM mode at the time a maximum frequency is set, allowing the change for the mode below. Figure 8 shows the two measurements: one with DVS and another with constant $1.9V$, that is the value for the higher level supply of the PMM.

In this measurements the analog part of the current was subtracted in order to compare it to the supplied voltage for the digital parts of the MCU. The total analog current was calculated with data sheet values for the external crystals, ADC, internal reference generator, PMM, and the internal 32768Hz-REFO clock. The analog current sums up to only about 350μA. Its reasonably fair to say that implementing DVS with the PMM of the MSP430 does not save as much as would be desirable. The current curves are very similar with and without DVS.

B. Energy consumption

The current measurements tests have shown that the current does not vary significantly when the core voltage supplying all the digital modules of the MCU is changed. However, there are conclusions that can be taken form the energy consumed. To measure the energy consumed, the total current consumed and the applied battery voltage must be considered. For comparison, also the digital modules consumed energy is represented. Figures 9 and 10 represent the energy relative to the digital modules of the MCU and the total energy consumed by the platform, respectively. The energy is calculated assuming the current is constant within the time the CPU is actively processing a Fibonacci sequence. The energy spent to calculate 5000 numbers of the Fibonacci sequence at the frequency f is:

$$E = dt_{F=f} * P_{F=f} \Leftrightarrow \quad (1)$$

$$E = dt_{F=f} * I_{F=f} * V_{F=f} \quad (2)$$

Where $dt_{F=f}$ is the time duration of the sequence at the frequency f and $P_{F=f}$ is power consumed at the same f. I is the current being consumed and V the voltage being supplied. From Figure 9 we notice that the processing time

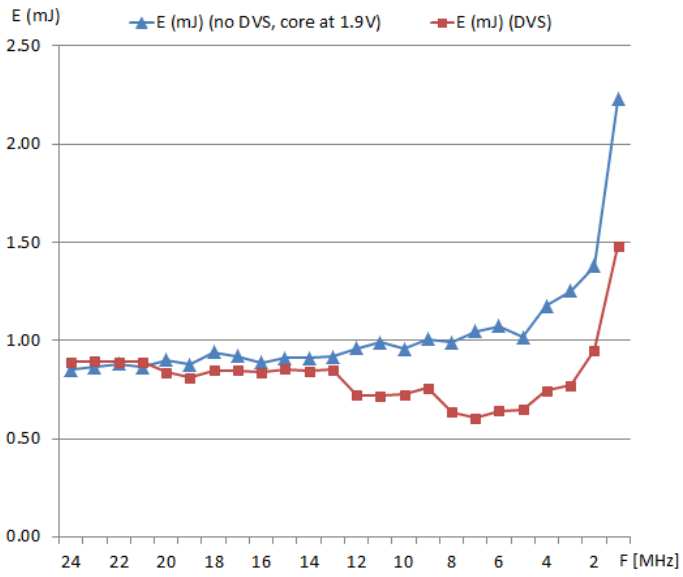

Figure 9. Energy consumption of the digital modules.

of the application is of great importance in the energetic consumption.

In the upper line with triangles on Figure 9, when no DVS is used, for the same amount of work load, the energy increases with the decrease of frequency. This happens because when decreasing the CPU frequency it is actively processing for more time. In this case, although the current increases linearly with the frequency, as it was concluded from Figure 8, it might be preferable to process the work at a higher frequency and spend less time processing. For the same processing work and processing with the CPU at 1MHz, the energy consumed is about $1.5mJ$ higher than when the CPU is at 24MHz.

With DVS there is a different behaviour, since between the frequencies of 2MHz to 12MHz there is a significant difference in the energetic consumption. Also, the energy is not increasing linearly as the frequency decreases. This is opposed to the scenario with no DVS, where the internal voltage is at its maximum value of 1.9V and when DVS is used, the voltage supplied is 1.4 V from 1MHz to 8MHz and 1.6V from 8MHz to 12MHz. This greatly influences the energetic consumption

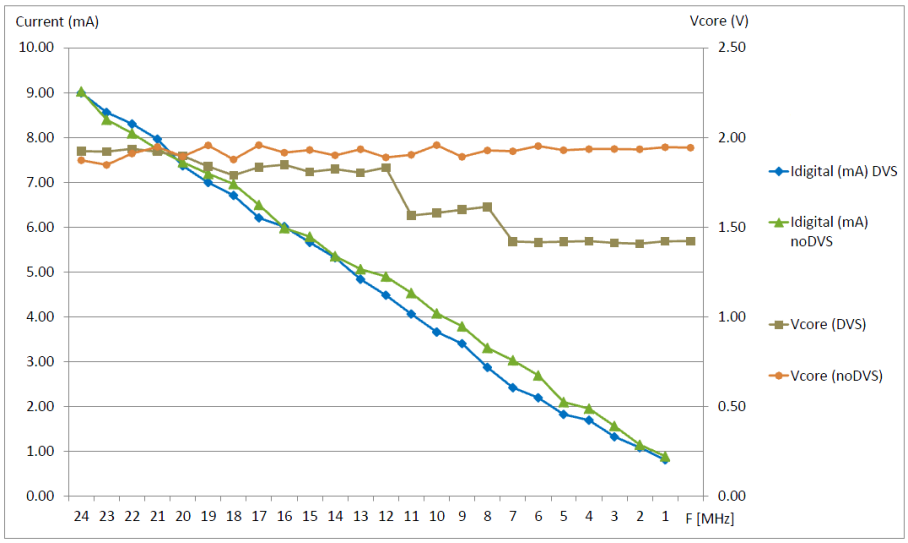

Figure 8. Frequency sweep with: DVS vs No DVS

by the relation of $E \propto V^2$, but it is not the only parameter that influences the energy.

Still in Figure 9 and analysing the line marked with squares, it can be noticed that the energy consumed only decreases significantly at each step when the internal voltage is decreased: at 12MHz and at 8MHz. For the frequency range where the internal voltage is constant, the energy is increasing due to the increasing time that the processes take to run. For this scenario, DVS would clearly save more energy processing at 7MHz with the 1.4V of internal voltage instead of using 1.9V of internal voltage and therefore running at the highest frequency.

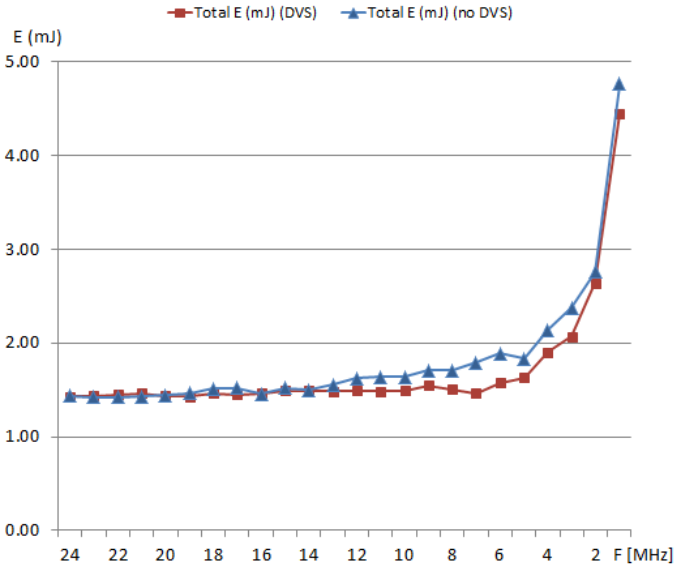

Figure 10. Total energy consumption of the platform.

In order to get the whole system's energy, the total current and the battery supply voltage was used. The energy points in Figure 10 are calculated with equation 1.

The change in the supply voltage is not so evident as compared to Figure 9. At the frequency steps where the voltage changes more significantly (at 8MHz and 12MHz), the total energy appears to be more dependent on the processing time of the CPU than on it's core voltage.

It would be expected that the total energy would be very similar to the digital energy in Figure 9. This happens because the majority of the running modules of the MCU are digital modules supplied by the internal voltage. The remaining analog parts of the MCU are supplied directly by the VCC (that is 3V from the batteries) and are assumed to be consuming constant energy while the application is being run. The assumption comes from the fact that the CPU is the most energy consuming part of the MCU in this test, and that it is the only variable that changes its energy across the CPU frequency sweep.

Observing the total energy in Figure 10 and by comparison to the digital part of the energy in Figure 9, the only difference is in the shape of the energy function of the measurements with DVS.

With no DVS the shape of the energy function in both Figures 9 and 10 is the same and has only a shift in the energy axis. This is justified by Figure 10 not contemplating the analog current consumption, that is assumed constant, and also the applied battery voltage differs to 3V instead of the internal 1.9V.

With DVS the analog current remains constant but the core goes through 4 voltage steps along the whole frequency sweep. The energy function shape should also be similar in both Figures but, in Figure 10 this is not verified. This happens because that the applied voltage is constant in the total energy measurement and therefore, for the energy function to have the same shape as in the digital part of the energy, the total current should be significantly inferior, specially in the 2MHz

to 12MHz frequency range.

To conclude these statements, it is important to understand that when considering the batteries voltage instead of the internal core voltage, it would be expected that the total current consumption should be significantly changing at the frequency steps where the core voltage is changed. Since this doesn't happen, the total energy function is very similar in both scenarios, with DVS and without DVS.

The conclusion is that the internal LDO, aside from supplying only the energy needed for the core of the MCU to function, is far from ideal. This can be seen by the increased energy in the DVS measurements, specifically in the 2MHz to 12MHz frequency range seen from Figure 9 to Figure 10. This increased energy is not useful energy used and so, although DVS may be a good energy saving solution, the fact that the MSP430s LDO dissipates part of the energy supplied, hides its benefits.

VII. DVS AND LOW-POWER MODES

The DVS method should be used altogether with the MCUs low-power modes. As proved in the previous section, time is an important factor when saving energy is a priority. The longer the processor is running, more energy it will consume.

It is concluded from Figure 10 that running at the lowest frequency does not mean one will spend less energy, rather the opposite. In the presented case, where the CPU had to calculate 5000 numbers of the Fibonacci sequence, if the frequency is lowered to its minimum the processing time will raise significantly and the energy consumed is much bigger than at faster frequencies.

Supposing that there are real-time constraints for the calculation of the Fibonacci sequence, a trade-off between energy and time is required. To understand this relation an estimation was made for the consumption of the system.

Considering an application with a deadline of 0.5 seconds to calculate the 5000 Fibonacci numbers, the energy is estimated for 1 period. The period is the time at which a new sequence needs to be calculated after the first has been requested. For simplification, in this estimate the period is the same as the deadline.

The energy values while actively processing the 5000 Fibonacci sequence numbers are already known from Figure 10, and the energy that is spent after the numbers are calculated, depends on the state of the MCU. The maximum wake up time from a low-power mode in MSP430 is less than 5μs and considering the estimated deadline of 0.5s, the use of the low-power mode 0 (LMP0) was considered. Lower modes deactivate the external crystals that can take up to 0.5s to stabilize after being shut down. The frequency sweep application required the use of both crystals, therefore only the first low-power mode is a realistic estimation.

Table I shows the estimated energy for 1 period of execution (0.5s). The Sleep Mode (SM) and Active Mode (AM) Energy (E) is shown as well as the time the application spends in active mode. In the remaining time, the MCU can either be idle (no low-power mode) or enter the LPM0. Only some frequencies are shown and these are centred in the 7MHz where the lowest AM power consumption was verified.

The LPM0 current used for calculations is the maximum 100μA given by the manufacturer, and the idle current of 1mA was measured experimentally. The voltage supply is a constant 3V from the batteries.

It can be verified that it is preferable to enter the low-power mode instead of leaving the CPU idle. Also there is not significant gain in power savings, when processing at the frequency that provides less power consumption (7MHz) and still meets the deadline. Running at the maximum frequency spends almost the same energy, however for that frequency the remaining time before the next period is significantly higher. This adds an increased consumption when in low-power mode as compared to the time spent in low-power mode after processing at 7MHz. For this estimation, it would still be better to use the intermediate frequency although the difference is inferior to mili Joules of energy.

For optimized power savings the active mode energy values and duration time of the tasks should be known before choosing an operating frequency and a determined low-power mode.

VIII. CONCLUSIONS

This work has contributed to TinyOS community by developing code for the OS port to the MSP430 series 5 microcontroller. Much effort has been put in the software due to the TinyOS system architecture and its development tools not being trivial and well documented. Since TinyOS is an open-source OS, the existing code is developed by the community and maintained by it. At the hardware level, the documentation is barely existent and much effort is needed to catch the pace of the continuous development of the OS. An important contribution has been made for the development in the fifth series of the MSP430 MCU, both at platform level as well as chip specific level.

A harware platform for WSN was developed based on the previous MoteIST++. It has a new microcontroller that allows for DVS and circuits to measure both the current consumption and the battery voltage. Also, the platform has a connector for a microSD card to allow for flexible and bigger data storage. The 802.15.4 communication board has also been updated and has now an integrated, high performance, PCB antenna.

The DVS method has been tested and works mainly in high performance systems and this work has created the software and hardware conditions to implement the method in a low-power microcontroller. Savings were verified for a small frequency range, however as the voltage is scaled, the variation in the total current consumption of the MCU, leaves more to wish for. It was verified that low-power modes are essential to prolong the life-time of the motes. In a scenario with real-time demands, the energy curve for a given frequency and a given running time must be known in order to make an

Frequency [MHz]	SM/Idle	Active Time [ms]	Deadline	E (SM/Idle) [uJ]	E (AM) [mJ]	Total E [mJ]
1	-	1296	Miss	0	4.45	4.45
3	LPM0	415	Ok	25.5	2.07	2.10
7	LPM0	177	Ok	96.9	1.46	1.56
24	LPM0	51	Ok	134.7	1.43	1.56
3	idle	415	Ok	255	2.07	2.33
7	idle	177	Ok	969	1.46	2.43
24	idle	51	Ok	1347	1.43	2.78

Table I
ENERGY ESTIMATION FOR 1 PERIOD OF EXECUTION USING DVS AND LOW-POWER MODES.

optimized choice for the operating frequency. Also, the low-power modes that can be used during the idle time, should be carefully chosen because there is an inherent start up time.

Regarding the TinyOS port, there is a continuous development happening in GitHub. The organization of the chip specific directory for the MSP430, the update and insertion of new functionalities is a constant.

There is no official driver for the microSD card in TinyOS however there are implementations of a driver that need further work to be released to the main TinyOS tree. This can be developed because the microSD driver is of interest to GEMS, since the new platform has the hardware to store data in this format.

The full platform initialization is needed, to fully work with the mote. For example, having initialization routines that would identify if the communication boards are connected, and initializing the peripherals by sending the I2C commands to the analog switches that supply them.

Changing the scheduling policy when there are real-time demands for the system and tasks with different priorities. With the DVS study made so far, a solution could be created to save the most power possible and still meet the deadlines. The TinyOS scheduler must be studied closely, because changing tasks priorities might influence the system core, that rely tasks to signal system events.

Implementing the method with an external LDO could make a difference. The MSP430 may work with a VCC voltage as low as 1.8V. With the optimized LDO and lowering the total supply voltage the current decrease might be of some significance.

A board can be made with an external LDO. This board would be supplying the mote through the pins of the battery, and could have a GPIO from one of the connectors or JTAG pins to drive it. This would allow to verify if the current consumption could be lowered with the optimized external LDO.

REFERENCES

[1] K. Römer and F. Mattern, "The design space of wireless sensor networks," 2004.

[2] J. M. Gilbert and F. Balouchi, "Comparison of energy harvesting systems for wireless sensor networks," *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 334–347, 2008. [Online]. Available: http://www.springerlink.com/index/10.1007/s11633-008-0334-2

[3] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *Instrumentation Measurement Magazine, IEEE*, vol. 12, no. 2, pp. 16 –23, april 2009.

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *IN ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS*, 2000, pp. 93–104.

[5] N. H. Zamora, J. chun Kao, and R. Marculescu, "Distributed power-management techniques for wireless network video systems," in *in Proc. Design Automation and Test in Europe (DATE*, 2007.

[6] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," 2001, pp. 89–102.

[7] J. Catela, R. Rocha, and M. Piedade, "A modular architecture for energy efficient wireless sensor networks nodes." *International Conference on Renewable Energies and Power Quality (ICREPQ'10) Granada (Spain)*, Jan. 2010.

[8] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. Wilton, "Dynamic voltage scaling for commercial fpgas," in *in ICFPT, 2005*, 2005, pp. 173–180.

[9] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and dvs processors," in *Low Power Electronics and Design, International Symposium on, 2001.*, 2001.

[10] A. Corporation., "Powernow! technology." http://www.amd.com, December 2000.

[11] I. Corporation., "Intel xscale technology." http://developer.intel.com/design/intelxscale/, November 2001.

[12] T. Corporation., "Crusoe processor." http://www.transmeta.com, June 2000.

[13] W. Tuming, Y. Sijia, and W. Hailong, "A dynamic voltage scaling algorithm for wireless sensor networks," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 1, 2010, pp. V1–554 –V1–557.

[14] A. Qadi, S. Goddard, and S. Farritor, "A dynamic voltage scaling algorithm for sporadic tasks," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, 2003, pp. 52 – 62.

[15] W. Kim, J. Kim, and S. Lyul-Min, "Dynamic voltage scaling algorithm for fixed-priority real-time systems using work-demand analysis," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, 2003, pp. 396 – 401.

[16] V. Culver and S. Khatri, "A dynamic voltage scaling algorithm for energy reduction in hard real-time systems," in *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, vol. 2, 2005, pp. 842 – 845 Vol. 2.

[17] G. of Embedded networked Systems and H. Networks. (2011, Nov.) Gems wiki. [Online]. Available: http://gems.leme.org.pt

[18] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *in Ambient Intelligence*. Springer Verlag, 2004.

[19] T. Burd and R. Brodersen, "Energy efficient cmos microprocessor design," in *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, vol. 1, Jan. 1995, pp. 288 –297 vol.1.

[20] E. Decker. (2011, Jun.) Tinyos production repository. [Online]. Available: https://github.com/tinyprod

[21] GEMS. (2011, Nov.) Tagus-sensornet. [Online]. Available: http://gems.leme.org.pt/PmWiki/index.php/Projects/Tagus-SensorNet