Article

Clock-Frequency Switching Technique for Energy Saving of Microcontroller Unit (MCU)-Based Sensor Node

Pumin Duangmanee and Peerapong Uthansakul *

School of Telecommunication Engineering, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand; puminsoft@gmail.com
* Correspondence: uthansakul@sut.ac.th; Tel.: +66-850-865-588

Received: 27 April 2018; Accepted: 7 May 2018; Published: 8 May 2018

Abstract: In this paper, a technique is proposed for reducing the energy consumption of microcontroller-based sensor nodes by switching the operating clock between low and high frequencies. The proposed concept is motivated by the fact that if the application codes of the microcontroller unit (MCU) consist of no-wait state instruction sets, it consumes less energy when it operates with a higher frequency. When the application code of the MCU consists of wait instruction sets, e.g., a wait acknowledge signal, it switches to low clock frequency. The experimental results confirm that the proposed technique can reduce the MCU energy consumption up to 66.9%.

Keywords: energy saving; microcontroller unit (MCU); clock frequency

1. Introduction

The operating clock frequency of most modern microcontroller units (MCUs) used in Wireless Sensor Networks (WSNs), such as Alf and Vegard's reduced instruction set computing (RISC) processor (AVR), peripheral interface controller (PIC), mixed signal processing (MSP), and 8051, is fixed [1,2]. The clock source is an internal/external crystal oscillator/resonator. The energy used by such MCUs can be decreased by reducing the current consumption. Reduced current consumption can be achieved by reducing the operating clock frequency of the MCU. The result of such energy saving is a long sensor node operation lifetime. A sensor node always includes a short instruction to read a sensor value and to complete data transmission. Most of the energy is used for transmitting the data to the target node. For example, the energy required to read the temperature with a 16-bit resolution is very small as compared with that required to send this value to a target node/server every minute. However, if the sensor node is of the image type or has a considerable amount of data that needs to be processed [3], the MCU needs more time to process the data before transmitting them to the other module. Examples of data processing include image compression, data compression, data integrity checking, and data encryption.

The time consumed by the MCU to process data depends on the operating clock frequency, the architecture of the MCU, and the data size. The energy consumption of data processing is currently increasing significantly, and with this increase, may approach or exceed that of data transmission. This means that the data-processing energy is also an important factor and thus should be reduced at the sensor node. Many means of reducing the processing energy exist, e.g., replacing the MCU with an field programmable gate array (FPGA) [4–6], employing a mixed FPGA/MCU [7,8], and improving the MCU algorithm [9,10].

When the sensor node is of the image type, the data size increases hugely. The raw data size of an image sensor may be range from 1 kilobyte to 100 kilobytes, depending on the image resolution.
This amount of raw image data is not suitable for sending via wireless communication and, thus, the amount of raw data must first be reduced by applying an appropriate method. A popular image compression scheme is joint photographic experts group (JPEG) compression [11,12]. JPEG is a lossy compression scheme that can convert a two-dimensional image pixel to frequency components and coefficients. Most of the high-frequency components of the image are lost during the compression process, but this loss is not detected by the human eye.

In JPEG compression, first the data are divided into blocks of 8 × 8 pixels. Then, quantization, zigzag scan, and discrete cosine transform (DCT) are performed. All these operations consume processing time of the MCU/FPGA. In the case of an MCU, the data are processed in the sequence of instruction, block by block of the image. The energy consumption of the MCU depends on its operating frequency and architecture. In the case of the FPGA, the image compression is performed by using logic synthesis and parallel processing, which is much more energy efficient than MCU processing. However, the cost of an FPGA that can support image compression is very high compared to that of an MCU. An additional advantage of an MCU over an FPGA is that the firmware of the system can be upgraded over wireless communication, which provides greater flexibility. Many MCU vendors support in-application programming (IAP) [13,14]. The code together with the IAP concept allow the MCU to program itself, which is an advantage in terms of maintenance.

In many studies in the literature, attempts have been made to improve the instructions used in the JPEG compression process in MCU-based image compression [15–18]. The authors of [15] improved data loading and restoration by replacing the zigzag scan with a lookup table. In [16], the authors improved the processing speed by using parallel instruction. The floating-point in the C program is mapped to integers [17] with the purpose of increasing the precision of the MCU. The results reported in [18–20] show that the energy consumed by FPGA-based image compression is greatly reduced as compared with that of MCU-based image compression. The results include many details of the energy consumed in each task, e.g., image capturing, segmentation, and morphology. In [20], the authors implemented a zonal DCT to reduce the number of alternating current (AC) coefficients. The results indicate a significant reduction in the energy consumption. The authors of [21] reduced the energy consumption by integrating processors, the image processing unit, and wireless communications into a single unit using a very-large-scale integration (VLSI) design. Their paper also presents battery-aware task scheduling to control the current consumption of a task so that it remains within a certain range.

In [22], the authors designed a JPEG image compression scheme for a capsule endoscope based on an ARM Cortex M3 processor with a JPEG compression engine. This device is effective for human body images. The MCU receives a JPEG image from an external JPEG compression engine, the energy consumption of which is lower than that of the MCU.

An additional means of reducing the energy consumption of image compression is to reduce the number of data retransmissions. In [23], the authors presented an architecture and protocol for energy-efficient image transmission on FPGA-based hardware. The image is divided into smaller blocks and effective queue control is used to reduce the packet error rate. The number of retransmissions is reduced and, thus, the energy consumption is also reduced. Data compression is another interesting topic in WSN research areas. A sensor node may collect many data records and send all the records to a target node at a predefined time. The results presented in [24] show that the optimal parameters can be designed for compression on a TI MSP430 series, which runs on a fixed operating clock frequency. The energy efficiency on an MCU-based platform without the energy consumption of the radio frequency (RF) circuit was presented in [25]. The RF part of a sensor node receives energy from the RF harvester circuit and does not consume any of the battery’s energy, while the MCU, which has to consume energy supplied by a battery, is in deep sleep mode and waits for an interrupt signal from the RF circuit.

Thus far, in the literature no studies exist that focus on saving energy by considering the operating clock frequency of the MCU. In some MCUs that contain a phase lock loop (PLL), the user can set the operating clock frequency dynamically to achieve the desired performance. From datasheets
and experiments, we found that an increase in the operating clock frequency leads to an increase in the current consumption. However, the current consumption does not increase at the same rate as the operating clock frequency. This is because an MCU contains many components, the current consumption of which is either dependent on or independent of the operating clock frequency. The relation of the current consumption of no-wait state instruction sets to the operating clock frequency [26] can be estimated by 

$$i_{MCU} = m \cdot f + i_{offset},$$

where \(m\) is a constant value and depends on the MCU architecture, \(f\) is the operating clock frequency, and \(i_{offset}\) is the current consumption of the MCU, which is independent of the operating clock frequency. If, for example, the operating clock frequency is increased two-fold, the increase in current consumption is less than two-fold, whereas, the application code can be completed more than two times faster. This means that if the application code contains no wait-state instruction sets, the total energy consumption of the application code can be reduced if the MCU operating clock frequency is higher.

Normally, the processing of a sensor node can be divided into three steps: (1) retrieve the data from the sensor; (2) process the data; and (3) transmit the data via wireless communication. The application codes in Steps 1 and 3 contain a wait-state instruction set, because of the delay time and waiting time in the data bus. In this processing period, we can set the MCU operating frequency to the lowest value, but the data bus peripheral still operates correctly. In Step 2, the code contains a no-wait instruction set. In this period, we set the MCU operating frequency to the highest value to save energy. In this paper, a technique for switching the MCU operating clock frequency is newly proposed in order to reduce the energy consumption of MCU-based sensor nodes. The experimental results show that the proposed technique can reduce the energy consumed more effectively than a fixed operating clock frequency in many applications, e.g., JPEG compression, cyclic redundancy check (CRC), and the Lempel–Ziv–Oberhumer (LZO) method. The results confirm that this technique can be used to improve the energy saving of MCU-based sensor nodes. It has two major attractive advantages: low cost and the upgraded flexibility of the platform.

Notations: there are many technical terms used as acronyms in this paper. Table 1 is presented in order to improve the readable quality of this paper.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
<tr>
<td>AVR</td>
<td>Alf and Vegard’s RISC processor</td>
</tr>
<tr>
<td>PIC</td>
<td>Peripheral Interface Controller</td>
</tr>
<tr>
<td>MSP</td>
<td>Mixed Signal Processing</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Experts Group</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>IAP</td>
<td>In-Application Programming</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very-Large-Scale Integration</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Lock Loop</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>LZO</td>
<td>Lempel–Ziv–Oberhumer</td>
</tr>
<tr>
<td>ARM</td>
<td>Acorn RISC Machines</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computing</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>RTC</td>
<td>Real-Time Clock</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver-Transmitter</td>
</tr>
</tbody>
</table>
2. Clock-Frequency Switching Technique

A high performance MCU running at high speed always contain a phase lock loop (PLL) unit. The PLL boosts input frequency to a higher frequency, e.g., from 12 MHz to 72 MHz. It is the responsibility of the designer to choose an appropriate operating frequency. For example, the MCU must operate at 48 MHz to provide appropriate universal serial bus (USB) operation, regardless of the input frequency.

Some applications do not need to run at a specific frequency. In the case of sensor nodes, we attempt to keep the clock frequency of the MCU as low as possible to maintain a low-level current consumption. This is because many sensor nodes contain a much shorter time for data acquisition than for data transmission. However, in the case of a long data task, it is possible that the processing time may be close to or greater than the transmission time. The reduction in the energy consumption for such a long data task affects the total energy consumption of the sensor node.

The energy consumption of a fixed frequency can be described as:

$$E_{mcu} = E_{proc} + E_{tx}$$  \hspace{1cm} (1)

where $E_{proc}$ is the energy required for data processing and $E_{tx}$ is the energy required for data transmission.

The energy required by a typical sensor node for processing is very small compared with that required for transmission energy, e.g., reading the analog input from the analog input pin of the MCU takes a few milliseconds. The value of the term $E_{proc}$ is thus very small as compared to that of $E_{tx}$. However, in the case of a long data processing time, such as that required for image compression, data integrity checking, and data compression, it takes more time to complete the processing and thus the term $E_{proc}$ is the dominant term of the total energy consumption. In a fixed frequency platform, such as a low cost 8-bit MCU that does not contain a PLL unit, one means of reducing the processing energy is to improve the instruction code or acquire an off-the-shelf module to provide better energy consumption; however, the cost of the node is also increased.

An MCU that contains a PLL unit cannot exploit the benefits of a frequency-switching technique that allows the MCU to operate at high frequency in the data-processing period and then return to low-frequency operation for the data transmission period. The relation between the execution time and the operating frequency is:

$$t_{exec} = \frac{k}{f}$$  \hspace{1cm} (2)

where $t_{exec}$ is the execution time, $k$ is the constant value depending on the MCU architecture, and $f$ is the operating frequency.

When the operating frequency is increased, the execution time is decreased. However, from many datasheets and experimental results, we found that the current consumption is increased as a function of the operating frequency. The relation is:

$$i(m \times f_{low}) < m \times i(f_{low})$$  \hspace{1cm} (3)

where $i(f)$ is the current consumption of the MCU at the operating frequency, $m$ is the multiplier of the new higher operating frequency and $m > 1$, and $f_{low}$ is a low operating frequency of the MCU.

The processing energy during $t_{exec}$ is:

$$E = v \times i \times t_{exec}$$  \hspace{1cm} (4)

The processing energy at high frequency in terms of low frequency is:

$$E_{proc} = v \times i(m \times f_{low}) \times \frac{k}{m \times f_{low}}$$  \hspace{1cm} (5)
The processing energy at low frequency is:

\[ E_{\text{proc}} = v \times i(f_{\text{low}}) \times \frac{k}{f_{\text{low}}} \]  

(6)

Finally, a comparison of the energy consumed at high frequency (5) and at low frequency (6), on the left and right side of (7), respectively, reveals that less energy is consumed at the higher frequency.

\[ \frac{i(m \times f_{\text{low}})}{m \times f_{\text{low}}} < \frac{i(f_{\text{low}})}{f_{\text{low}}} \]  

(7)

In the transmission part, the transmission energy of the MCU is the energy expended when the MCU transmits a data packet to a target node successfully; that is, the sender receives an acknowledge packet of all the data packets that it sent. The transmission time period may vary depending on the conditions of the wireless networks, e.g., the number of hops, quality of signal, and level of noise. 802.15.4 PHY allows 127 bytes per packet at 2.4 GHz. The RF band rate is 250 Kbps. We assume the condition of the acknowledgement timing is the best case and the address length is 16 bit. The total duration of the transmission period is:

\[ t_{tx} = 0.928 + (0.032 \times b) \]

(8)

where \( t_{tx} \) is the total transmission time in ms and \( b \) is the data size in bytes.

Then, the total energy consumed by the MCU for the best case transmission is:

\[ E_{tx} = i \times v \times t_{tx} \]

(9)

where \( E_{tx} \) is the transmission energy, \( i \) is the average current consumption of the MCU, and \( v \) is the supply voltage of the MCU, which is 3.3 V.

Now, suppose that only the data processing is considered and the data acquisition period is ignored. The energy consumption of data processing and transmission when the operation is fixed at a low frequency is:

\[ E_{\text{mcu, fix}} = E_{\text{pll, low}} + E_{\text{proc}} + E_{tx} \]

(10)

where \( E_{\text{mcu, fix}} \) is the energy consumed by the MCU operating at a fixed low frequency and \( E_{\text{pll, low}} \) is the energy consumed by the PLL locked at a low frequency.

Next, the proposed technique allows the MCU to operate at high frequency during the data-processing period and then return to low-frequency operation during the data-transmission period. This is represented by

\[ E_{\text{mcu, sw}} = E_{\text{pll, high}} + E_{\text{proc}} + E_{\text{pll, low}} + E_{tx} \]

(11)

where \( E_{\text{mcu, sw}} \) is the energy of the MCU when the frequency-switching technique is applied, \( E_{\text{pll, high}} \) is the energy of the PLL locked at high frequency, and \( E_{\text{pll, low}} \) is the energy of the PLL locked at low frequency.

It should be noted that there exists an issue based on (11): the data-processing period must not contain any wait-state instructions; if it does, the frequency-switching technique cannot be used to achieve energy reduction.

Figure 1 shows the difference between operating at a fixed frequency and when the frequency is switched. In the transmission period, the MCU always operates at the lowest frequency, because a wait-state is included in the communication with the transceiver. Some examples of applications that can benefit from the switching technique are cyclic redundancy check (CRC), image compression, and data compression. In this study, all three examples were implemented to validate the benefit of the proposed technique.

CRC is useful for checking data integrity. A sender node appends the CRC code at the end of the data packet. The receiver node receives packets and checks all the data packets that include the CRC code. If the resulting value is zero, this means the packet is valid and then the data packet
is passed to the upper layer for further processing. The implementation of the CRC in this study was based on C programming, that is, a lookup table, because it offers the best MCU performance. There are various CRC polynomials that are suitable for each application [27].

As an image-compression scheme, JPEG compression was selected for our demonstration, because JPEG is a popular image type widely used in many sensor node applications.

Finally, for data-compression, in this study the LZO method was selected because it requires a low memory footprint so that it is suitable for an MCU-based platform. LZO compression is lossless data compression, which allows the receiver to receive data that are exactly the same as the original data transmitted by the sender.

3. Experimental Details

3.1. Hardware

The ARM7 MCU family, LPC2387, was selected for use in all experiments. This MCU has a PLL unit that can boost input crystal frequency up to 72 MHz. The MCU has an Static RAM (SRAM) of 64 K bytes for general purpose memory. This SRAM is used for many data-processing operations. The experimental environment is shown in Figure 2. The direct current (DC) input voltage power supply was stepped down to 3.3 V to supply the MCU and the MAX232. RS-232 communication was employed to debug the MCU and to set the parameters, such as data length, for the MCU operation for each experiment. The current consumption of the MCU was measured by a simple resistor in series with the current path to the ground. A resistor value of 0.5 ohm was selected for all experiments. Note that the resistor value should be kept as low as possible, so that it does not affect the voltage drop across the resistor, which may cause the MCU to stop operating.

![Experimental setup](image1.png)

**Figure 1.** Difference between (a) fixed frequency and (b) frequency-switching technique.

![Experimental setup](image2.png)

**Figure 2.** Experimental setup (a) hardware diagram (b) photograph of operating hardware.
The voltage across the resistor was amplified to the appropriate level so that it could easily be analyzed by the digital oscilloscope. In our experiment, the amplify gain was set to 50. The USB port from the digital oscilloscope was for data transfer/setup, so that the data points could easily be saved on the computer in many file different formats, including bmp and csv.

3.2. Software

The compiler suite used for compiling the source code was ARM with GNU Compiler Collection (GCC). This is freeware that has a command line style. First, the program initializes scripts, which are implemented in assembly language. These initializing sections set up the stack for each operation mode of the Acorn RISC Machines (ARM) processor, clear the memory, and then jump to the main function.

The main function is written using C programming. It contains one-time initialized code and continues with forever loops. In the initial section, it sets a sleep mode, real-time lock (RTC) for wakeup period, memory, and universal asynchronous receiver-transmitter (UART), and then, waits for a command string from the PC. The command string contains parameters, e.g., data size and quality. After receiving a command string, the MCU goes to sleep mode with a periodic wakeup periodically 15 s for energy measurement. The parameters of the MCU are as follows.

- Exp: experimental applications: CRC-8, CRC-32, JPEG, and LZO.
- FsEn: operating modes: either the frequency-switching technique or a fixed frequency.
- Len: data length for CRC-8 and CRC-32.
- Qual: quality of compression for JPEG.
- TxEn: choice of only data processing or including data transmission.

To measure the amount of energy, the MCU is woken up every 15 s and then the PLL is set, as shown in Figure 1.

When the MCU is woken up from the sleep mode, it sets the PLL according to the FsEn parameter. If the value of FsEn is zero, the MCU operating frequency is set to 4 MHz. This is the lowest frequency for this MCU that allows communication to be operated correctly. If the value of FsEn is 1, the MCU operates with the frequency-switching technique and the operating frequency is set to 72 MHz, which is the maximum frequency for this MCU. After the PLL has been locked, the MCU runs the code according to the Exp parameter.

If the Exp parameter is CRC-8 or CRC-32, the code is run using the Len parameter. If Exp is JPEG, the code is run using the Qual parameter. If Exp is LZO, no additional parameters are needed, because the data size is fixed to 500 bytes.

The final parameter is TxEn. If its value is zero, the MCU goes to sleep mode immediately after completing a data processing operation. If the value is 1, the MCU performs a transmission process to send data to the target node. The time spent during this period is according to the parameter Len in (4) for CRC-8 and CRC-32. If Exp is JPEG, Len is assigned a compressed image data length. If Exp is LZO, Len is assigned the compressed data length of sample data records.

4. Results

This section presents the comparative experimental results for operations at fixed frequency and when the frequency-switching technique was applied. As described previously, the amount of energy saving depends on the time period of the data processing. To obtain the exact time period, the assembly code that is generated after the C source code is compiled needs to be examined, which is not convenient. It is better to show the results in parameters that are easily understood, e.g., data length. The experimental results for CRC-8, CRC-32, JPEG, and LZO with different data lengths, fixed frequency/frequency switching, and transmission/no transmission are shown. The current consumption in all experiments was scaled down by a factor of 50, e.g., 1 volt/division is equivalent to 20 mA.
4.1. Cyclic Redundancy Check (CRC-8)

The polynomial of CRC-8 in this experiment was \( x^8 + x^5 + x^4 + 1 \) and it requires a lookup table for the best performance for C code. The parameters of this experiment were as follows:

- Data size: 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000.
- With and without transmission.
- With and without frequency-switching technique.

The current consumption of data-processing and -transmission operations at a fixed frequency and when the frequency-switching technique was applied is shown in Figure 3. The total energy consumption is presented in Figure 4. It can be clearly seen that the energy consumption is at approximately the same level when the data size is ~10 bytes. In addition, when the data size is increased, the difference between fixed frequency and switching frequency is more pronounced. It can be seen that the difference appears to be a linear function of the data size after a data size of 10 bytes.

![Figure 3. Current consumption of cyclic redundancy check (CRC-8) generator code with fixed frequency (black) and switched frequency (blue) for data size 500 bytes.](image)

![Figure 4. Total energy consumption of CRC-8 versus data sizes.](image)

4.2. CRC-32

The polynomial generator used for CRC-32 was
\[
x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1.
\]

In addition, the processing employed a lookup table, as does CRC-8. The parameters of the experiments are also the same as for CRC-8.
The current consumption of data processing and transmission operations at a fixed frequency and when the frequency-switching technique was applied is shown in Figure 5. The total energy consumption is presented in Figure 6. It is clear that the energy consumption is still at approximately the same level when the data size is ~10 bytes. When the data size is increased, the difference between fixed frequency and switched frequency is also increased and the difference appears to be a linear function for a data size >= 100 bytes.

![Figure 5](image1.png)

**Figure 5.** Current consumption of CRC-32 generator of fixed frequency (black) and switched frequency (blue) for data size 500.

![Figure 6](image2.png)

**Figure 6.** Energy consumption of CRC-32 versus data sizes.

4.3. Joint Photographic Experts Group (JPEG) Compression

Image compression is more complicated than CRC generation. The compression process consists of partitioning an image into sub-images of 8 × 8 pixels, forward discrete cosine transform (FDCT), quantization, and Huffman encoding. The DC/AC coefficients of luminance, chrominance, and Huffman encoding are obtained by the JPEG standard [28]. The input parameter of compression is a quality value in the range 1 to 99. This quality is multiplied by all the entries in the luminance and chrominance tables. The total processing time varies according to the quality value. If the quality value is higher, more luminance and chrominance AC coefficients have a non-zero value and the compressed JPEG data size is higher. This consumes more MCU processing time and also increases the transmission time to other nodes. The raw image data was a C constant array located on a flash section of the MCU. A popular Lena image was selected for the experiments. The size of the image was 64 × 64 pixels in YUV422 format.
The current consumption of JPEG compression with transmission operating at a fixed frequency and when the frequency-switching technique was applied is shown in Figure 7. The total energy consumption is presented in Figure 8.

![Graph showing energy consumption of JPEG compression](image1)

**Figure 7.** Current consumption of joint photographic experts group (JPEG) compression at fixed frequency (black) and switched frequency (blue) with a quality value of 50.

![Graph showing energy consumption of JPEG compression versus quality](image2)

**Figure 8.** Energy consumption of JPEG compression versus quality.

### 4.4. Lempel–Ziv–Oberhumer (LZO) Compression

LZO compression is a lossless data compression that is suitable for application on an MCU, because it requires a low memory size. It is a dictionary-based compression. For real-time compression, it should be carefully designed because of the dynamic memory usage of the dictionary. In this experiment, the samples of temperature data for 500 records were compressed using LZO and transmitted to a target node. The efficiency of the compression depends on the repeat pattern of the data. In Figure 9, the samples of temperature in Celsius are shown. The sampling interval between each record is 1 m. The current consumption of the operation at fixed frequency and when the frequency-switching technique was applied, where the operation was completed with the transmission process, is shown in Figure 10. It can be seen in this figure that the energy consumption of LZO when the frequency is fixed is ~3.5 mJ and when the switched frequency technique is applied is ~1.6 mJ.
The energy saving in percentage is summarized in Table 2. The maximum of energy saving is 66.9% when JPEG compression is employed.

The experimental results for CRC-8, CRC-32, JPEG, and LZO show that the frequency-switching technique reduced the energy consumption of the MCU in all the experiments. The energy saving in percentage is summarized in Table 2. The maximum of energy saving is 66.9% when JPEG compression is employed.

Table 2. Comparative summary of energy consumption for fixed frequency and switched frequency.

<table>
<thead>
<tr>
<th>Experiment (with TX.)</th>
<th>Parameters</th>
<th>Energy (mJ)</th>
<th>Energy Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Fixed Frequency</td>
<td>Frequency Switching</td>
</tr>
<tr>
<td>CRC-8</td>
<td>Len = 100</td>
<td>0.24</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>Len = 500</td>
<td>0.82</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>Len = 1000</td>
<td>1.6</td>
<td>1.35</td>
</tr>
<tr>
<td>CRC-32</td>
<td>Len = 100</td>
<td>0.27</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>Len = 500</td>
<td>0.88</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>Len = 1000</td>
<td>1.65</td>
<td>1.37</td>
</tr>
<tr>
<td>JPEG (Lena 64 × 64)</td>
<td>Q = 10</td>
<td>24.5</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td>Q = 50</td>
<td>26.5</td>
<td>9.2</td>
</tr>
<tr>
<td></td>
<td>Q = 99</td>
<td>39.7</td>
<td>17.8</td>
</tr>
<tr>
<td>LZO (Temperature)</td>
<td>Len = 500</td>
<td>3.5</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Figure 9. 500 samples of temperature data.

Figure 10. Current consumption of Lempel–Ziv–Oberhumer (LZO) compression of 500 temperature samples with a fixed frequency (black) and switched frequency (blue).
5. Conclusions

The experimental results showed the advantage of using the frequency-switching technique in terms of reducing the energy consumption of an MCU. The amount of energy saving depends on the data-processing time, the difference between two operating frequencies, and the architecture and the peripherals of the MCU. An additional advantage of the proposed technique is that there is no additional hardware cost and it can be implemented easily in only the software. The proposed technique can be used to improve the energy efficiency of a low-cost MCU-based sensor node when a PLL unit is available. In summary, the image sensor nodes can save energy by using the proposed technique. The experimental results indicate that energy saving up to 66.9% can be achieved when using the frequency-switching technique compared with normal operation.

Author Contributions: P.U. worked on the conceptual modelling and finding solutions. P.D. was responsible for the experiments. Both authors contributed to revising and improving the quality of the paper. The major and the most significant contribution was by the second author P.U.

Acknowledgments: This work was supported by Suranaree University of Technology (SUT) Research and Development Fund. We would like to thank the reviewers and editor for their valuable comments and constructive suggestions, which helped to improve the quality of this article.

Conflicts of Interest: The authors declare no conflict of interest.

References


