sleepyCAM: Power Management Mechanism for Wireless Video-Surveillance Cameras

Tenager Mekonnen*, Erkki Harjula*, Timo Koskela† and Mika Ylianttila*

*Centre for Wireless Communication, University of Oulu, Finland
{tenager.mekonnen, erkki.harjula, mika.ylianttila}@oulu.fi
†Center for Ubiquitous Computing, University of Oulu, Finland
{timo.koskela}@oulu.fi

***Abstract*—In this paper, we propose an energy efficient motion detection and power management mechanism, called sleepyCAM, for wireless camera sensor nodes that do not otherwise support low-power modes. In the proposed solution, a low-power sensor node accompanied with a Pyroelectric Infrared (PIR) sensor and a relay is used to detect motion and manage the power usage of a high-power and high-resolution camera sensor node. To validate our work, we used two baseline benchmarks for comparison that are commonly used as motion detection mechanisms on wireless surveillance cameras: (a) hardware based motion detection using a PIR sensor and (b) software based motion detection using video frame comparison. The main contributions of this paper are the prototype implementation of the sleepyCAM, the surveillance application and the comparison of power consumption between the proposed and the baseline methods. The measurement results indicate that the power consumption of a surveillance camera node can be reduced significantly with the proposed mechanism.**

I. Introduction

Wireless Video-surveillance Networks (WVN) are among the emerging paradigms of the Internet of Things (IoT). WVNs are a subtype of Wireless Multimedia Sensor Networks (WMSN) that, for one, belong to Wireless Sensor Networks (WSN). Recent studies and forecasts show that the global IP-traffic will be significantly dominated by multimedia data originating from visual networking applications [1], [2]. According to Cisco Visual Networking Index, the global IP video traffic will be 82% of all the IP traffic by 2020; of which internet video surveillance contributes 3.9%, up from 1.5% in 2015 [1].

WVNs can provide richer information and wider coverage for several applications that have vital scientific, social, and strategic relevance [3]. Examples of these systems include wildlife monitoring to determine guidelines for human/predator coexistence, monitoring large open areas in airports [3], monitoring crops and farm equipment in agricultural plots [4], monitoring the condition of affected people during an accident in mining industry [5], and monitoring elderly people in assisted living scenarios [6].

The three fundamental issues in a WVN are energy consumption, data latency and data quality [7]. For an off-grid video surveillance deployment, energy consumption will be of paramount concern. In contrast to simpler sensors, camera nodes generate higher amounts of data requiring higher processing power, and thus consume more energy [3]. In video-surveillance applications, the visual sensors spend more than 99% of their time waiting for an incident [8]. Thus, the waiting time energy consumption is the dominant factor affecting the total energy consumption of a WVN.

The energy-efficiency of WSN and IoT in general has been a hot research topic during the past years. Most of this work has focused on reducing the idle power consumption by introducing different levels of sleep modes [9]. Nonetheless, the currently available hardware capable of capturing and streaming high-definition video, such as Raspberry Pi (RPi) [10], do not have appropriate power management to support different levels of sleep modes (sleep, deep sleep, awake). Therefore, the battery life of the available camera sensor nodes is closer to hours than months.

Thus, we developed a power management mechanism for RPi and similar hardware using a low-power sensor node as a controller. The controller uses Pyroelectric Infrared (PIR) sensor to detect motion and relay to turn RPi on for recording video when there is motion in the area under surveillance. Using this technique, we can extend the battery lifetime during the waiting time of RPi based camera sensor nodes (and similar hardware) by more than 108 days using a $6000mAh$ off-the-shelf Lithium-ion battery.

The rest of this paper is organized as follows: Section II summarizes the related work. Section III introduces our sleepyCAM prototype. Section IV presents the different surveillance scenarios as a basis for evaluating the proposed solution. Section V shows the evaluation results. Finally, Section VI discusses the results and highlights how each scenario could further be improved based on our observations.

II. Related Work

Although the power management of wireless sensor nodes has been thoroughly studied during the past years, not much has happened in the area of power management in wireless multimedia sensors. In this section, we introduce some related power management solutions that are applicable to the current WVN and WMSN nodes in general.

Wake-on-Wireless is one of the basic methods for allowing nodes to sleep during inactive periods by using the integrated wireless radio to wake the device up when needed. Shih et al. [11], provide an example of this method using a 802.11b wireless radio to wake-up a PDA device. Using out-of-band

control signaling from a low-power radio device, the system maintains connectivity and wakes up the PDA when needed. This approach can be applied to WMSNs if the camera node hardware platform supports reception of wake-on-lan packets after it has gone to a sleep mode.

Similarly, more recent studies on the area of WVN [12], [13] suggest the use of FM radio in WiFi-enabled multimedia sensor networks as an "always-on" point-to-multipoint control channel used to turn off the WiFi radios in camera nodes. This method can minimize the energy consumption of a WVN node that will be wasted on receiving packets that are destined to other nodes. In addition, by scheduling the transmission windows of the nodes this technique can minimize the number of collisions in a congested network; and hence the energy associated with the re-transmission of packets. The limitation of this method is that it only manages the WiFi radios (which consumes fraction of the device's power in the case of RPi and similar devices) and does not control the power state of the main hardware. In addition, this method requires an "always-on" FM radio which consumes energy, and also becomes problematic to scale up the network in mass deployment of WVNs.

Sorber et al. [14], propose a hierarchical power management for mobile devices, aiming to improve their availability in a distributed system. The paper suggests the integration of different power-level mobile devices, such as PDAs, laptops and sensors, into a single multi-tiered device that can function at power-levels of any of its tiers. The fundamental idea of this model is suitable for WVNs and is similar to our approach. However, PDAs, laptops and sensors provide inherent low-power states compared, for instance, to RPi which does not support low-power modes.

III. Prototype Implementation of sleepyCAM

We developed the sleepyCAM power management mechanism for RPi and similar hardware using low-power sensor node (a controller) as shown in figure 1. In sleepyCAM, the controller uses PIR sensor to detect motion and a relay to turn RPi on when motion is detected. Our goal is to achieve as low power consumption as possible during the waiting time period of the surveillance application. There has been lots of effort to achieve low-power mode of WVN during the waiting time, such as using multi-tier architecture [15], [8]. In multi-tier architecture, the camera nodes are kept in sleep mode during the waiting time and will only listen for a wake-up message from other scalar sensor nodes which perform the motion detection task. This approach is however difficult if the camera platform does not support low-power modes (such as the RPi).

The low price, small size, portability and support for high-resolution camera makes RPi an ideal platform for several IoT projects. Despite all of that, RPi does not support Advanced Configuration and Power Interface (ACPI) to perform power management. Hence, it provides no low-power modes that keep it running on battery for a long period of time. It is possible to use the lowest possible clock setting of the

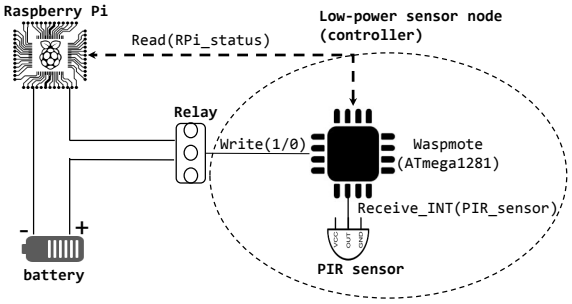

Fig. 1: sleepyCAM: Power management of camera node

CPU (under-clocking), but this does not provide significant power reduction. Thus, we devised a mechanism of controlling the power consumption of RPi using a controller node. The controller basically performs two tasks: (1) motion detection using a PIR sensor and (2) powering-up RPi when motion is detected. In our prototype implementation, RPi is normally powered off, and therefore, it does not consume any energy during the waiting time.

The hardware prototype of the controller node was implemented using Libelium Waspmote sensor platform [16]. The Waspmote main-board has an Atmega1281 microcontroller, running at 8MHz with 8KB SRAM, 128KB Flash, 4KB EEPROM, built-in temperature and accelerometer sensors, and Real Time Clock (RTC). The Atmega1281 consists of two UARTs (UART0 and UART1), SPI port, I2C communication bus and digital/analog input/output (I/O). Using multiplexers, the Waspmote platform extends the UARTs to 6 different ports: USB connector and Socket0 on UART0, and Socket1, UART1 AUX, UART2 AUX and GPS ports on UART1.

To reduce the power consumption of the controller, we use interrupts and put the node into a low-power state by disabling all the unused ports in our surveillance application described in Section IV-C. The built-in accelerometer and RTC are connected on the I2C bus of the microcontroller. Since neither of them are used in our surveillance application, we have turned off the I2C bus to save additional power. Parallax Rev. B PIR sensor and 10A 30VDC Songle relay are used for event detection and switching RPi on when needed. The software for the controller is implemented in C language using the latest Waspmote Pro v040 IDE.

Upon occurrence of an event (motion), the controller switches RPi on to record a video. It is up to RPi to decide when to shutdown itself after completing with the video recording. It can be done so that RPi will do further motion detection and keep on recording video as long as there is activity. When RPi is on, the motion detection can be conducted either using a PIR sensor or by capturing and comparing video frames from the camera stream, whichever is more efficient. Finally, RPi shuts down after completing the surveillance task and makes sure that there are no scheduled tasks (e.g uploading the videos to a remote server).

An important step in the power management process is the detection of the successful shutdown of RPi so that the

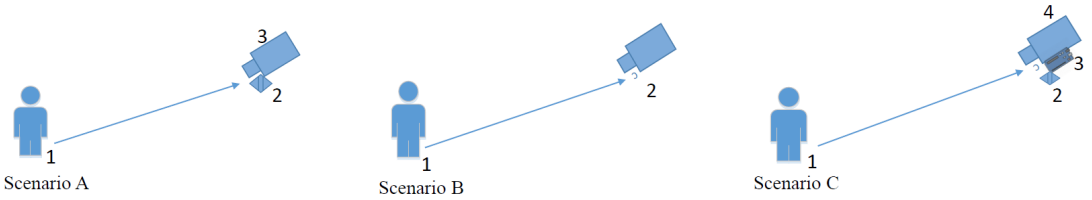

Fig. 2: Scenarios

relay can be safely detached without worrying about memory card corruption. For that, RPi can be configured to signal the completion of a graceful shutdown process over one of its General Purpose Input/Output (GPIO) pins. The state of the selected GPIO pin can be read by the controller node via one of its digital pins, as shown in figure 1.

IV. Scenarios

To demonstrate the benefits of our proposal, we compare the energy consumption of three different scenario setups as shown in figure 2. Scenarios A and B are the baseline methods used for benchmarking the energy efficiency of motion detection using sleepyCAM (scenario C). In each scenarios, when motion is detected, the surveillance applications record video of 10 seconds and go back to waiting mode. In all of the scenarios, the hardware used as the camera node is RPi model B3 with a 5 megapixels Rev. 1.3 camera module, and running the latest Raspbian Jessie Lite OS. RPi model B3, has 1GB RAM, 1.2GHz 64-bit quad-core ARMv8 CPU, 802.11n wifi, Bluetooth 4.1, 4 USB ports, Ethernet port, full HDMI, camera interface and VideoCore IV 3D graphics core. Each setup is particularly distinct to one another by how they detect motion, as explained in the scenario descriptions below.

A. Hardware based Motion detection on a standby RPi

In the first baseline scenario, we have a camera sensor node using a PIR sensor for motion detection. In this setup, both the motion detection and the event capturing functionality are implemented on the camera node (i.e RPi). Thus, this camera node needs to be kept active throughout the run-time of the surveillance application. A PIR sensor is attached to one of the GPIO pins of RPi as shown in figure 3. A surveillance application written in Python is run. When there is an interrupt signal from the PIR sensor, the surveillance application calls a function that starts recording the video. The use of interrupts, instead of pooling the GPIO pin, saves significant amounts of energy during the waiting time.

B. Software based Motion detection on a standby RPi

In the second baseline scenario, the PIR sensor on figure 3 is opted out. Instead, a software algorithm is run to detect motion by analyzing the input from the camera stream. It takes video frames of few seconds as input from the camera stream and computes the absolute difference between the averaged frames. If the difference passes a certain threshold, it is interpreted as motion and the camera starts recording the video. During the recording period, there is no need to compare video frames for motion detection. We used Open Source Computer Vision (OpenCV) libraries and Python programming to develop the surveillance application shown in figure 4. Similar to scenario A, in this setup, both the motion detection and video recording are performed by RPi. Thus, RPi has to be continuously active.

C. Motion detection in sleepyCAM

In this scenario, we take a different approach. Instead of exploiting the high-power camera node during the waiting time, we used our sleepyCAM prototype to implement the motion detection. To realize a low-power WVN, we divide the surveillance application into two modules: (1) motion detection and (2) video recording. The motion detection module takes care of detecting motion in the area under surveillance, whereas video recording module takes care of capturing the event. We deployed the motion detection module on the controller node and video recording on RPi. By implementing the motion detection on a separate, low-power controller node, the main camera node can be switched off during the waiting time and significant amounts of energy can be saved.

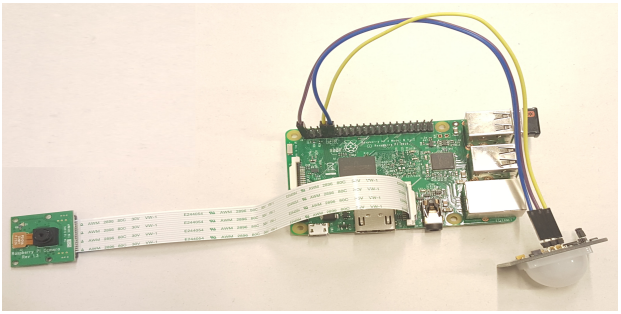

Fig. 3: Baseline camera node setup.

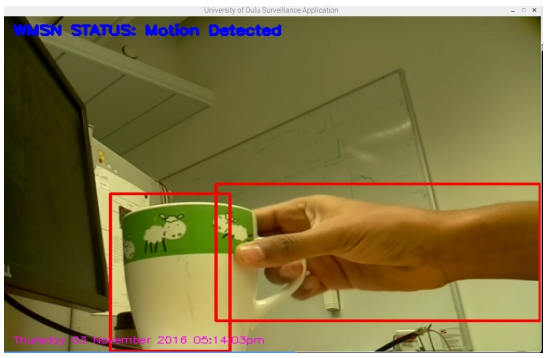

Fig. 4: Software based motion detection from a camera stream

V. Power consumption Analysis

A. Evaluation Setup

The power consumption of both the controller and RPi in all of the scenarios was measured using Monsoon power monitor tool [17]. The measurement data from the tool can be exported as csv file to a workstation. We analyzed the csv data using MATLAB. In order to improve the readability of the figures, we used a moving average filter to smooth the data before generating the graphs.

B. Evaluation Results

In this section, we will present the power consumption measured in all scenarios for different modes of operations: waiting and recording.

Figure 5 illustrates the power consumption transients of the baseline setup in scenario A. The graph shows the power consumption of a RPi camera node in a standby mode before and after the surveillance application launches. The power consumption before the application runs, characterizes the behavior of the camera node (i.e. RPi) when only basic operating system tasks are running. This benchmark can also serve as baseline for standby RPi before the surveillance application is started in scenario B.

Once the surveillance application is run, the camera node starts waiting for some incident detected by the PIR sensor. The PIR sensor becomes active when there is change in passive infrared radiation in the area. The PIR sensor basically consumes $\sim 0.35mW$ when it is inactive and $\sim 8.35mW$ when active. Thus, it is difficult to see the change due to the PIR sensor before and after the application runs. When motion is detected the camera module is turned on and it starts recording video for 10 seconds. The average power consumption of the camera node during the recording jumps to $1854mW$ from $1423.6mW$ of the waiting time (refer table I and table II).

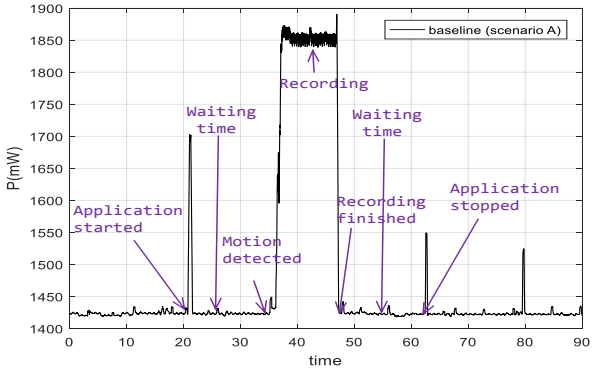

Fig. 5: Power consumption of scenario A

Similarly, figure 6 presents the power consumption transients of the standby camera node in scenario B. When the surveillance application is launched, the camera module turns on and is ready to record/stream video. It is possible to access the live-stream anywhere over the network or by the local-host (RPi). The goal in this scenario is to capture video frames from the camera stream and process them to detect motion. As can be seen in table I, this process consumes significant amount of power ($2162.9mW$), compared to scenario A. An interesting observation in this setup is that the camera node actually consumes more power during the waiting time, compared to the recording time. The application is intentionally stopped processing the video frames while recording as we are not interested in detecting motion while we are actually capturing the event. When motion is detected, the node starts recording the video for 10 seconds and goes back to waiting mode. During the recording, the node consumes $1937.5mW$ power (refer table II).

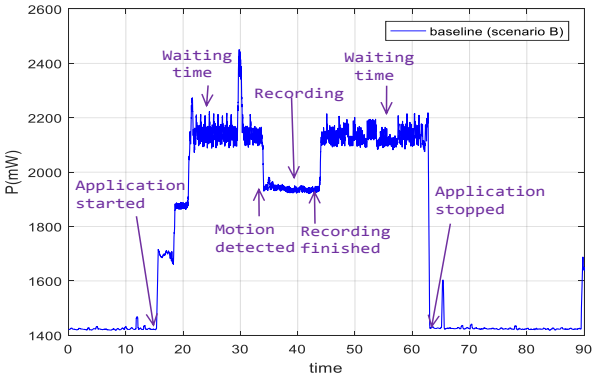

Fig. 6: Power consumption of scenario B

Next, in scenario C, we evaluate the power consumption of motion detection using our sleepyCAM prototype. In sleepyCAM, the controller is running the motion detection module of the surveillance application. The controller will enter the waiting mode soon after it is powered-up and initialization has been completed. During the waiting time, the controller consumes $5.4mW$ and RPi $0mW$. Hence, the total power consumption for the surveillance system in the waiting time is $5.4mW$ (table I). Figure 7 shows the power consumption transients of the controller (purple line), RPi (black line) and the system total (red line). It also presents different power states of the system components. When motion is detected, the controller activates the relay which in turn powers-up RPi. The controller power rises to $212mW$ and will be in this power state as long as RPi is on.

We configured RPi to execute the video recording module of the surveillance application during the boot-up process (even before any user has logged in). The recording module of the surveillance application also commands RPi to shutdown after completing the video recording. We also configured RPi to signal the controller about a graceful completion of the shutdown process so that it can safely deactivate the relay. Figure 8 depicts the optimized system power consumption transients of scenario C (sleepyCAM), including the boot-up and shutdown processes. As it can be seen in table II, the total power consumption of the system during the video recording period is $2140.5mW$.

In figure 9, we present the power consumption comparison between the two baseline scenarios (A and B) and scenario C utilizing our sleepyCAM. The graph regions labeled as (1) and (2) depict the power consumption in each scenario during the waiting time and recording, respectively. The average values of these regions are presented in tables I & II. As can be seen, our proposed mechanism decreases the waiting time power consumption by $\sim 99.6\%$ compared to baseline scenario A and by $\sim 99.75\%$ compared to baseline scenario B. During the video recording, our mechanism consumes $\sim 15.5\%$ more power than baseline scenario A and $\sim 10.5\%$ more than baseline scenario B. The increased power consumption during the recording time is caused by the controller's relay which consumes more power when being active.

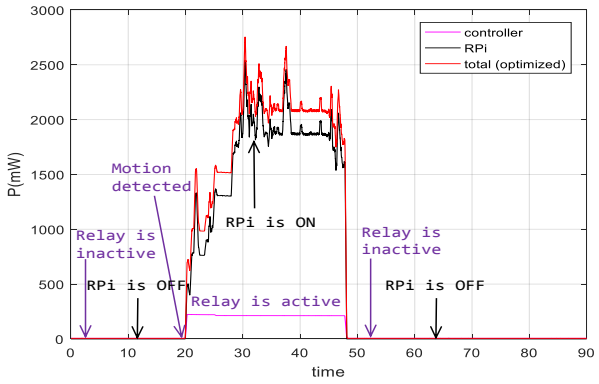

Fig. 7: Power consumption of the nodes in scenario C

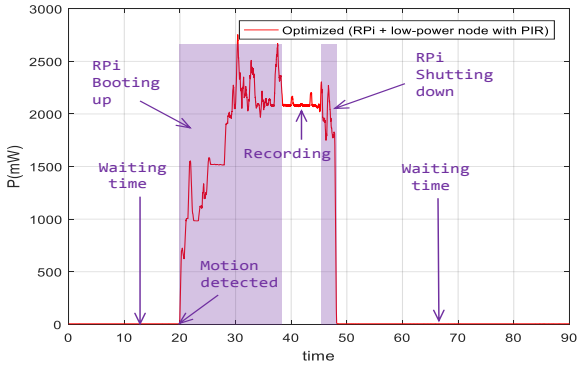

Fig. 8: Total power consumption of scenario C

	Scenario A	*Scenario B*	*Scenario C*
	Power (mW)	*Power (mW)*	*Power (mW)*
RPi	1423.6	2162.9	0
Controller	NA	NA	5.4
Total	**1423.6**	**2162.9**	**5.4**

TABLE I: Waiting time power consumption

We conclude the power consumption analysis by providing battery life estimation of the waiting time in the three scenarios using a $6000mAh$ off-the-shelf Lithium-ion battery. Battery

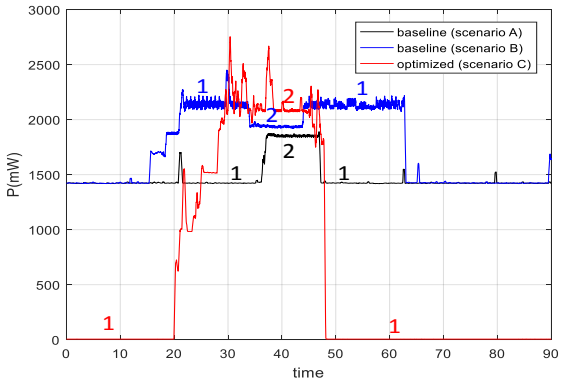

Fig. 9: Overall power consumption comparison between scenario A, B and C

	Scenario A	*Scenario B*	*Scenario C*
	Power (mW)	*Power (mW)*	*Power (mW)*
RPi	1854	1937.5	1928.5
Controller	NA	NA	212
Total	**1854**	**1937.5**	**2140.5**

TABLE II: Recording time power consumption

life can be calculated from the device's current consumption using online tools like provided in [18]. The average current consumption, nominal voltage, and hours of battery life are presented in table III. As can be seen, our sleepyCAM prototype (Scenario C) improves battery life radically, by more than 17600%).

	Voltage (V)	*Current (mA)*	*Battery-life(hrs)*
Scenario A	5.0	284.7	14.8
Scenario B	5.0	387.5	10.8
Scenario C	3.3	1.6	2625

TABLE III: Battery-life of waiting time

VI. Discussion and Future Work

In this paper, we presented an energy efficient motion detection and power management mechanism for a high-resolution video-surveillance camera nodes, called sleepyCAM, and benchmarked its performance against two baseline scenarios. Both baseline scenarios and our proposed sleepyCAM setup consist of RPi single-board computer and a 5MP Rev. 1.3 RPi camera module. In our optimized solution, we proposed an external low-power controller node to conduct the motion detection and manage power of the camera node at the same time. The hardware selected for this purpose is Libelium Waspmote sensor platform with a PIR sensor and a relay. We measured the power consumption using Monsoon power monitoring tool, and analyzed the power consumption transients using MATLAB plots. The results show that our proposed motion detection provides more than 99% power consumption improvement and $\sim$ 2600 hours (108 days) longer battery-life using a $6000mAh$ battery-house. More

importantly, our power management technique can potentially be used for several other embedded Linux based systems, such as Banana Pi, Beaglebone and the likes.

We conclude the paper by providing some remarks on how the baseline methods and our proposed mechanism could be further improved. It is possible to optimize the RPi OS in order to achieve fast boot-up and further decrease power consumption, such as by prioritizing tasks, removing unused services, or using buildroot tools to generate simple, efficient and easy-to-use embedded Linux systems through cross-compilation [19], [20]. Similarly, for a headless RPi (i.e one intended to operate without a display, keyboard and mouse), the active state power can be greatly reduced by disabling all the unused peripherals and turning off the LED lights on the hardware platform and camera module. These could improve all the scenarios discussed in the paper. However, these are beyond the scope of this paper and we did not try to address them here.

In scenario B, the intensive computation during the waiting time can be performed on a remote server. In this case, RPi would open a live-stream to the remote server and the server would do the motion detection by processing the video frames. When motion is detected, the server would start recording the stream. In addition to saving the computational power of the camera node, this would have potential to save the transmission delay incurred by transferring video from the camera node to the server.

In Scenario C, the mechanical relay used by sleepyCAM can be replaced by a solid state relay or even by an optocoupler (photo-isolator) with the exact load-current rating. This could reduce the power consumption of the controller significantly during the run-time of RPi.

To mitigate the limitation of the boot-up time, our future work will focus on extending our solution to a multi-tier WVN architecture. We plan to use scalar sensor nodes as tier-1 devices and our optimized camera node as tier-2 device. The tier-1 devices will send an early alert signal to tier-2 devices; so that tier-2 device will have enough time to wake-up and record/stream the incident on time. Our future work will, thus, consider power consumption analysis of radio transmissions/reception between tier-1 and tier-2. We will also include the power consumption of tier-2 for uploading video to a remote server; or streaming to a terminal over a wireless link.

ACKNOWLEDGMENT

This work is supported by TEKES and the European Celtic-Plus project CONVINcE and was partially funded by Finland, France, Sweden and Turkey.

REFERENCES

[1] Cisco, "White paper: Cisco visual networking index: Forecast and methodology, 20152020," Tech. Rep. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] "Internet of multimedia things: Vision and challenges," *Ad Hoc Networks*, vol. 33, pp. 87 – 111, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870515000876

[3] C. B. Margi, V. Petkov, K. Obraczka, and R. Manduchi, "Characterizing energy consumption in a visual sensor network testbed," in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, 2006, pp. 8 pp.–339.

[4] A.-J. Garcia-Sanchez, F. Garcia-Sanchez, and J. Garcia-Haro, "Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops," *Computers and Electronics in Agriculture*, vol. 75, no. 2, pp. 288 – 303, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169910002553

[5] S. Misra, G. Mali, and A. Mondal, "Distributed topology management for wireless multimedia sensor networks: exploiting connectivity and cooperation," *International Journal of Communication Systems*, vol. 28, no. 7, pp. 1367–1386, 2015. [Online]. Available: http://dx.doi.org/10.1002/dac.2770

[6] M. Hossain and D. Ahmed, "Virtual caregiver: An ambient-aware elderlymonitoring system," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1024–1031, 2012. [Online]. Available: http://dx.doi.org/10.1109/TITB.2012.2203313

[7] C. F. Chiasserini and E. Magli, "Energy consumption and image quality in wireless video-surveillance networks," in *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, vol. 5, Sept 2002, pp. 2357–2361 vol.5.

[8] P. Porambage, A. Heikkinen, E. Harjula, A. Gurtov, and M. Ylianttila, "Quantitative power consumption analysis of a multi-tier wireless multimedia sensor network," in *European Wireless 2016; 22th European Wireless Conference*, May 2016, pp. 1–6.

[9] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104 – 122, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614001418

[10] Raspberry Pi Foundation, *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/

[11] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '02. New York, NY, USA: ACM, 2002, pp. 160–171. [Online]. Available: http://doi.acm.org/10.1145/570645.570666

[12] F. Sousa, R. Campos, and M. Ricardo, "Energy-efficient wireless multimedia sensor networks using fm as a control channel," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, June 2014, pp. 1–7.

[13] J. Dias, F. Sousa, F. Ribeiro, R. Campos, and M. Ricardo, "Green wireless video sensor networks using fm radio system as control channel," in *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Jan 2016, pp. 1–8.

[14] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: Hierarchical power management for mobile devices," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 261–274. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067198

[15] P. Kulkarni, D. Ganesan, and P. Shenoy, "The case for multi–tier camera sensor networks," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '05. New York, NY, USA: ACM, 2005, pp. 141–146. [Online]. Available: http://doi.acm.org/10.1145/1065983.1066016

[16] Waspmote Sensor Boards, http://www.libelium.com/products/waspmote/.

[17] Monsoon Solutions, Inc., *Monsoon Power Monitor*. [Online]. Available: https://www.msoon.com/LabEquipment/PowerMonitor/

[18] Digi-Key Electronics, *Battery Life Calculator*. [Online]. Available: http://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

[19] BuildRoot, *Making Embedded Linux Easy*. [Online]. Available: https://buildroot.org/

[20] Busybox. [Online]. Available: https://git.busybox.net/buildroot/tree/board/raspberrypi/readme.txt