Jurnal Teknologi, 54 (Sains & Kej.) Keluaran Khas, Jan. 2011: 231–254

WIRELESS MULTIMEDIA SENSOR NETWORK PLATFORM FOR LOW RATE IMAGE/VIDEO STREAMING

ROZEHA A. RASHID[1*], NORSHEILA FISAL[2] & ABDUL HADI ABDUL HAMID[3]

Abstract. Conventional Wireless Sensor Network (WSN) application mainly deals with scalar data such as temperature, humidity, pressure and light, which are very suitable for low rate and low power networking technology such as IEEE 802.15.4 standard. The availability of commercially off the shelf (COTS) complementary metal-oxide semiconductor (CMOS) camera has made a single chip solution possible and consequently fostered researchers to push WSN a step further. Multimedia data delivery unique properties posed new challenges for resource-constrained sensor networks. Transferring raw data is very expensive while sensor nodes processing power put a serious limitation on it for any sophisticated multimedia processing. This project proposed a new platform for wireless multimedia sensor network (WMSN) namely TelG mote and WiseOS operating system to support the mote operation. The mote design for WMSN consists of ATmega644PV microcontroller from Atmel Co. as its processing unit, XBee module as its communication unit and C328R CMOS camera as its sensing unit. To hide low-level details of TelG motes such as processor management, memory management, device management, scheduling policies and multitasking, the developed WiseOS provides a clear application programming interface (API) to the application developer. WiseOS is designed to be monolithic, event-driven and using first-in first-out (FIFO) scheduler policy. The low rate video/image streaming application that was developed shows that multi-hop communication for multimedia content in WMSN using TelG mote supported by WiseOS proved to be practical.

Keywords: Wireless sensor network; wireless multimedia sensor network; CMOS camera; embedded operating system; IEEE 802.15.4

Abstrak. Aplikasi *Wireless Sensor Network* (WSN) konvensional secara umumnya mengendalikan data yang bersifat skala seperti suhu, kelembapan, tekanan dan cahaya yang mana ianya sesuai untuk teknologi perhubungan rangkaian yang mempunyai kadar data dan penggunaan kuasa yang rendah seperti piawaian IEEE 802.15.4. Kebolehsediaan kamera *complementary metal-oxide semiconductor* (CMOS) di pasaran yang memungkinkan selesaian serpih tunggal telah menarik minat pengkaji untuk memajukan WSN. Keunikan penghantaran

[1-4] Telematics Research Group, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor Darul Ta'azim, Malaysia

* Corresponding author: Email: rozeha@fke.utm.my

data multimedia mewujudkan suatu cabaran baru kepada nod penderia yang mempunyai kekangan sumber. Proses penghantaran data secara mentah adalah sangat mahal manakala keupayaan pemprosesan nod penderia meletakkan had yang serius untuk sebarang pemprosesan data multimedia yang sofistikated. Projek ini mencadangkan suatu platform baru untuk aplikasi *Wireless Multimedia Sensor Network* (WMSN) yang dinamakan nod penderia TelG dan sistem pengendalian WiseOS untuk menyokong operasi nod penderia tersebut. Reka bentuk nod penderia untuk aplikasi WMSN terdiri daripada mikro pengawal model ATmega644PV dari Atmel Co. sebagai unit pemproses, modul XBee sebagai unit komunikasi dan kamera CMOS C328R sebagai unit penderia. Untuk menyembunyikan selok-belok aras rendah nod penderia TelG seperti pengurusan pemproses, ingatan, peranti, polisi penjadualan dan tugas berbilang, WiseOS yang dibangunkan menyediakan *application programming interface* (API) yang jelas kepada pengembang aplikasi. WiseOS direka bentuk dengan seni bina monolit, terpacu peristiwa dan menggunakan polisi penjadualan *first-in first-out* (FIFO). Aplikasi kadar data rendah penjurusan video/imej yang dibangunkan menunjukkan bahawa komunikasi berbilang hop untuk kandungan multimedia didalam WMSN menggunakan nod penderia TelG yang disokong oleh WiseOS praktikal untuk digunakan.

Kata kunci: Rangkaian penderia tanpa wayar; rangkaian penderia multimedia tanpa wayar; kamera CMOS; sistem pengendalian operasi terbenam; IEEE 802.15.4

1.0 INTRODUCTION

Communication technologies available today are so sophisticated and pervasive ranging from wired to wireless technologies. WSN is perceived as a new technology that would change the notion of "personal computing" into "pervasive and embedded computing" [1] [2]. With the advances of technologies in electronics, low cost small devices consuming low power capable of ubiquitously capturing multimedia content such as video, still images, audio and scalar data from the environment has influenced the development of WMSN [3]. By incorporating multimedia data into networks of wirelessly interconnected devices, not only will enhance the existing WSN applications such as tracking, home automation and environmental monitoring, but would forge the way for new applications as well.

Multimedia content plays a major role in various fields such as fine arts, entertainment, commercial, industrial process control, engineering, education and military. Society today, demanded that multimedia content to be accessed anywhere and anytime. As categorized in [4], the application can be classified into five fields, which are surveillance, traffic monitoring and enforcement, personal and health care, gaming, environmental and industrial.

WMSN has several unique characteristics that must be considered in designing WMSN applications and test beds. Communication, signal processing and embedded computing are the main fields of disciplines related to WMSN. Cross-

disciplinary research on these fields is very important to enable wirelessly interconnected embedded devices to interact with the environment, sense and control the physical environment. These research fields lay out the fundamental key factors in designing WMSN such as application specific requirements, high bandwidth demand, source coding technique, multimedia in-network processing, power consumption and integration with other wireless technologies.

In this paper, a new hardware platform for WMSN is presented complete with its operating system. The rest of this paper is organized as follows; Section 2 discusses the existing hardware platform used for WMSN. Section 3 discusses the proposed hardware platform. The proposed operating system is presented in section 4 while section 5 and 6 concludes this paper and the suggestion for future works respectively.

2.0 WMSN MOTE AND OPERATING SYSTEM

WMSN mote in general is the same as any WSN motes, which consists of sensing unit, processing unit and communication unit as shown in Figure 1.

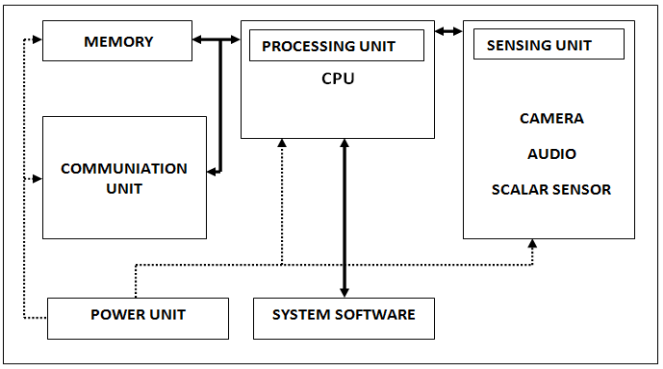

Figure 1 Basic building blocks of WMSN mote

As shown in Figure 1, the three important building blocks are discussed in this section by emphasizing on the imaging devices

2.1 Imaging and Sensing Device

CMOS camera low power profile, small form factor and highly integrated single chip solution in general is targeted to be interfaced with computationally powerful host devices such as personal digital assistant (PDA) and cell phones. In order to

interface CMOS camera to low-end host devices such as an 8-bit microcontroller, a hardware interface is required. Cyclops modules are an example of such design [5]. Figure 2 shows the picture of Cyclops module.

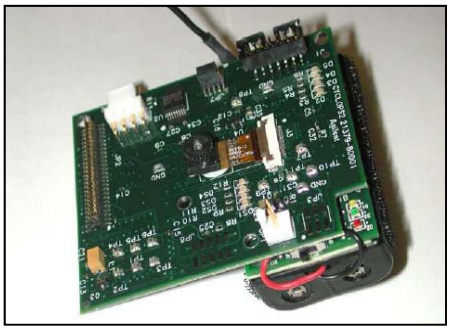

Figure 2 Cyclops module attached to Mica2 mote

Cyclops module consists of CMOS camera, programmable logic and external memory for high-speed data communication. The CMOS camera used by the Cyclops is Agilent ADCM-1700 CIF camera attached on the board together with ATMEL ATmega128L 8-bit microcontroller (MCU) to controls the camera i.e. configuring its parameters and performing local image processing. Complex programmable logic device (CPLD) is used to enable high speed data transfer since the MCU speed is limited to 4MHz which is lower than the camera address generation for capturing images. Cyclops uses nesC language [6] to write its firmware based on TinyOS libraries. The firmware hides the complexity of the module operations and provides a high-level interface to present the image data to the host.

Another approach of enabling CMOS camera sensor to be used in WMSN platforms is by Carnegie Mellon University researchers which is a complete embedded computer vision platform namely CMUcam3 [7]. Figure 3 shows the picture CMUcam3.

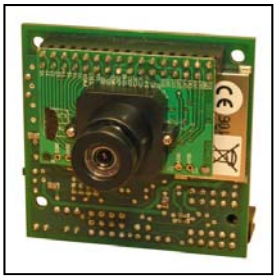

Figure 3 CMUcam3 embedded computer vision platform

CMUcam3 is developed on low cost components consisting ARM7TDMI MCU, Averlogic AL4V8M440 FIFO frame buffer and Omnivision CMOS sensor. The ARM MCU used is chosen because of its capabilities to execute a broad set of algorithm while the frame buffer makes it possible for the camera to operate at full speed of 26 frames per seconds. CMUcam3 comes with open source software for JPEG compression and basic images libraries. It claims to be able to perform several vision tasks as in [8], [9], [10], [11] and [12]. The platform also provides a set of communication interface such as I^2C, serial peripheral interface (SPI), and RS232 which enabling it to be interfaced with other motes such as TeloB.

A medium-resolution imaging motes has been built by Intel that can be used for WMSN development. Stargate platform [13] is designed by Intel and manufactured by Crossbow. Figure 4 shows a photograph of Stargate board.

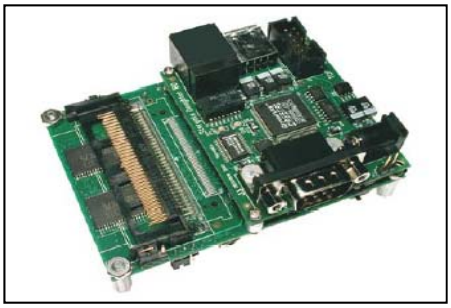

Figure 4 Stargate board from Crossbow

Stargate board is designed for various applications such as signal processing, control, robotics and sensor networking applications. The processor is based on Intel PXA255 XScale 400MHz RISC processor. The processor has 32Mbyte of Flash memory, 64 Mbyte of SDRAM and on board interface connector for TelosB, MicaZ, personal computer memory card international association (PCMCIA) Bluetooth and IEEE 802.11 cards. As reported in [14], Stargate platform, consumes considerably high power.

2.2 Operating System for WSN

Low-level details for a sensor node such as processor management, memory management, device management, multitasking and scheduling policies are the core functions of an operating system (OS). OS handles these low-level details in such a way that it is hidden from the application developers by providing a clear API to access the underlying hardware. This section discusses the existing OS for sensor node in terms of its design challenges, execution model and scheduling.

2.2.1 Operating System Design Challenges

A typical sensor node as evaluated in [15] is constrained by its resources such as limited power, processing capabilities, memory and bandwidth. As opposed to conventional sensor node in WSN, which consumed more power in communication activity [16], WMSN consumed more power in processing activity [17]. As a result, power optimization in WMSN is even more important than WSN. To conserve energy, OS is responsible to strategize a mechanism for it such as periodic sleep as in [18] where the OS provides a clean and concise API for its power management.

A long running task can starve other task from processing time and miss its deadline. In real-time applications such as WSN, deadline is very important. Task with priority should properly be scheduled [19] [20] [21] by the operating system effectively. TinyOS for example, employs run to completion and two-level scheduler model [16].

2.2.2 Operating System Classifications

In terms of architecture, OS can be classified into three categories which are monolithic such as TinyOS [16] and MagnetOS [22], modular such as MantisOS [18], Contiki [23], Sensor Operating System (SOS) [24], Bertha [25] and CORMOS [26] and virtual machine such as Mate [27]. The tradeoff between which architecture to be used is based on the OS flexibility and its performance.

Execution model of an OS directly influence its performance. The most two popular model used in WSN motes are event-based [28] such as TinyOS [18] and EYES/PEEROS [29] and thread based such as MantisOS [18]. Table 1 shows the comparison between these two models.

Table 1 Events vs. Thread [30]

Events	Threads
Computation is handled by event handlers	Computation is divided between thread
Mostly run to completion event handlers	Can be preempted
No stack overhead as there can be only one event handler running at one time	Context switch overhead
Used when applications requires efficiency	Used when applications require flexibility
Allows high concurrency	Not for concurrency intensive operations

3.0 TELG MOTE DEVELOPMENT

A typical WSN platform limits its data acquisition mechanism to low bandwidth data types such as humidity, temperature, light and pressure. For a higher bandwidth requirements i.e. data types such as audio and visual, a more robust sensing mechanism, processing capability and communication mechanism must be considered in designing the platform. In this section, a new platform for WMSN namely TelG mote is proposed.

3.1 System Architecture

The proposed WMSN platform composes of several basic components such as sensing unit, processing unit, communication unit and power unit as shown in Figure 5.

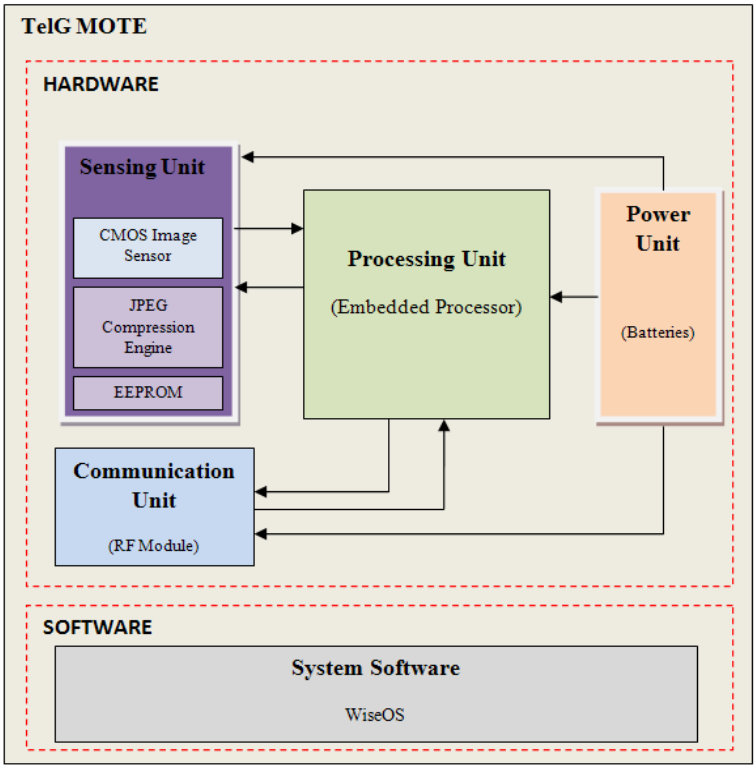

Figure 5 TelG mote system architecture

TelG mote has been built with goals to minimize power consumption, ensure flexibility and provides software and hardware robustness. In addition to that, it allows applications to be easily developed and employs standard communication interface.

3.2 Sensing Unit

CoMedia C328R camera is a mobile imaging devices that has a lens, built in RAM, and JPEG compression engine integrated in a single chip. The dimension of the camera is 20x28mm. This small form factor is desirable for miniaturization of mote. Figure 6 shows the photograph of the camera.

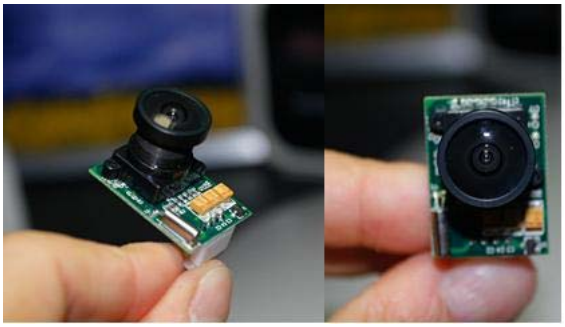

Figure 6 C328R CMOS camera from CoMedia

C328R can capture VGA image and down sample it to quarter video graphic array (QVGA) format or common intermediate format (CIF). C328R offers a range of resolution to choose. The smallest resolution is 80x86 pixels and the largest resolution is up to 640x480 pixels. The embedded joint photographic experts group (JPEG) codec enabled the host to select the color conversion such as JPEG and grayscale. By having hardware implementation of image compression instead of software implementation, significant power consumption is reduced. C328R uses serial universal asynchronous receive transmit (UART) interface to communicate or issue command to send or receive the image data.

The camera module operates at 3.3V with low power consumption at 60mA in active state. To issue a command to the camera, the camera provides a built in serial type program as its application programming interface (API) with a set of user-friendly command for interfacing with the external host. Figure 7 shows the functional block diagram of C328R camera.

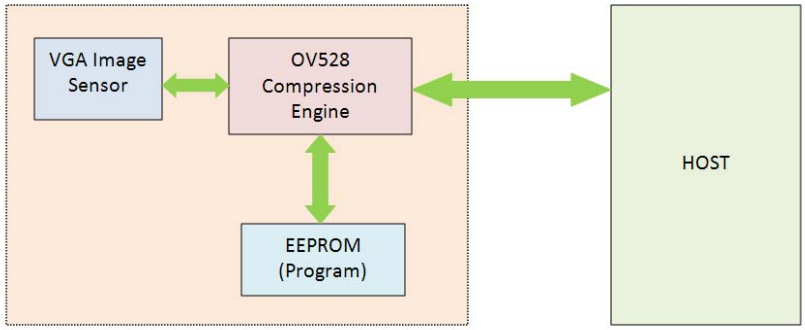

Figure 7 C328R System block diagram

The electrically erasable programmable read only memory (EEPROM) is used to store the camera API to interact with the host. The host must issue a set of command in order to setup the camera such as synchronization, image resolution setting, and color setting before capturing images from the camera. The user-friendly command to communicate with the camera provides the host a certain degree of flexibility in capturing images from the environment. Multimedia data is well known to be huge and when deployed in a network with multiple sources of such data, can easily congest the network. This flexibility allows the host to adjust the images generated by the camera according to the network condition.

3.3 Processing Unit

There is a wide variety of microcontroller available in the market. New microcontroller keeps offering new features such as low power consumption, various on-chip peripherals and a range of random access memory (RAM) and flash sizes. Current 8-bit microcontroller is capable of executing a few million instructions per seconds (MIPS).

ATmega644PV microcontroller from Atmel Corporation has been chosen to be used in TelG mote. The power consumption of TelG mote is kept at minimal value without compromising its processing capability. It is chosen after evaluating existing microcontroller from various manufacturers such as Microchip, Motorola and Atmel.

ATmega644PV has a very low power consumption in active, power-down and power save mode. Table 2 shows the power consumption of ATmega644PV.

Table 2 ATmega644PV power profile

Mode	Active	Power Down	Power Save
Power Consumption	0.4 mA	0.1 μA	0.6 μA

Table 2 values are given under the condition that the microcontroller operates at 1 MHz clock, 1.8V voltage supply, and 25 °C environment temperature. TelG mote is typically designed to operate at 3.3 V with 7.3728 MHz external clock at ambient temperature of 25°C. These settings yield approximately 4.9 mA active current, 1.8 mA idle current, 0.25 μA power-down current and 0.62 μA power-save current consumption [33].

RAM and flash sizes are the most common features of microcontrollers that need to be considered. Microcontrollers RAM are important in developing an application. Larger RAM size is better since it enables processing of applications that are more complex and sophisticated. Atmega644PV has a 4KB RAM for application development. Flash storage is used to store the application program. Larger program would need a larger flash. Up to date, flash storage is not the limiting factor in developing WMSN applications. Atmega644PV provides flash storage of 64KB which is adequate for most embedded applications.

Atmega644PV in particular provides two serial UART interface. Atmel family microcontrollers have several programming methods to choose from in order to program its flash. TelG mote is designed to be programmed using in-circuit programming (ISP) mode which uses the standard SPI port for ease of use.

3.4 Communication Unit

One of the components that needs to be carefully selected when designing a WSN is the radio part. Communication in WSN is reckoned to be one of the most power consuming activities. When dealing with multimedia data, even though visual processing power consumption is more pronounced, radio communication still proven to be one of the main cause of rapid energy depletion. TelG mote is developed to conform to the typical WMSN platform specification, which is low power and cost.

The most suitable radio for typical WMSN platform is low rate and low power. Since it is foreseeable that TelG mote might need to communicate with other devices from different vendor, it is desirable to choose a well known standard for communication. Based on this assumption, IEEE802.15.4 compliant radio has been selected. Any devices sharing the same physical layer that conform to this standard will be able to communicate with each other.

The wireless device for TelG mote is XBee module from MaxStream Inc. XBee module is an IEEE 802.15.4 compliant radio based on carrier sense multiple accesses (CSMA) which can provide point to point, point to multipoint and also peer to peer communication. It is designed for low-latency and predictable communication timing applications. Figure 8 shows the photograph of XBee module.

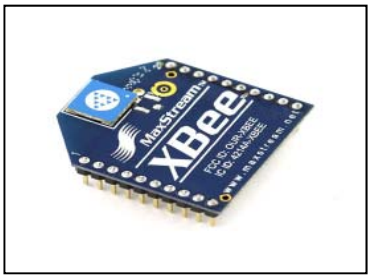

Figure 8 XBee modules photograph from MaxStream

XBee module RF data rates can go up to 250Kbps that operates at 2.4GHz Industrial Scientific and Medical (ISM) frequency band. XBee module has a small form factor (2.438cm x 3.294cm). It has the power output of 1mW (+0dB) capable of transmitting up to 30m indoor and 100m outdoor with the receiver sensitivity of -92dBm [34].

With sixteen 5MHz channels ranging from 2.405 to 2.480 GHz, it has 65,000 addressable network addresses for each channel. Since it employs IEEE 802.15.4 standards, the data transmitted is in packets where the maximum transmission unit (MTU) for a packet is 127 Bytes. Each packet is acknowledged at the link layer in unicast mode providing best-effort-delivery except for broadcast mode. It is interesting to note that, the link layer standard requires a coordinator in the network but XBee is designed to work even without coordinator.

XBee used serial UART for its interface. The host can be interfaced with XBee module at the speed up to 115.4Kbps. Since the UART interface operates at 3V and ATmega644PV (host) also operates at the same voltage level, XBee module can be connected directly without any voltage leveler. One of the most appealing features of XBee module for application developers is its API mode of operation. The API provides an easy way for developers to issue a command to the module.

3.5 TelG Mote Design

To increase the mote robustness, TelG mote is designed to be integrated with programming, computation and communication onto a single device. Sensing unit is designed to be a separate pluggable module.

TelG mote is programmable using ISP type programmer. The robustness of the mote is that it can be reprogrammed on the board. Every integrated development environment (IDE) for Atmel MCU such as AVRstudio and Programmer Notepad support ISP programming. ISP is chosen as it is one of the most popular programmers for Atmel microcontroller and also due to its simplicity.

Both the sensing unit and communication unit use serial UART interface to communicate with the host. Atmega644PV has two serial UART interface for this purpose. TelG mote is designed to support independent applications from the sensing unit. For this reason, sensing module is to be a separate entity from the mote. UART communication port is very precious and limited while there is a wide range of sensors that can use it. By designing it to be pluggable, various sensors can be attached to the mote without compromising its input/output (I/O) pins.

TelG mote provides four light emitting diodes (LEDs) for the application to use. LEDs are indispensable to application developer because of the lack of visual information in embedded system to determine the system status. The LEDs are useful indicators to monitor the hardware and software state during program debugging and deployment. Two push buttons are provided by TelG mote; one for reset button and the other one for user button. The user and reset button provide means for the user to issue an interrupt to the mote to execute any user defined instructions and to reset the system respectively.

TelG mote design does not include any voltage regulator since it is battery powered. The voltage and current is assumed to be at a certain maximum value initially and gradually decreasing. The maximum value must not exceed the safe operating region of any devices on the mote. Although this looks like a serious drawback, the advantages of this design is to fully utilize the battery lifetime. ATmega644PV has an operating voltage as low as 1.8V while a single AA battery cutoff voltage is 0.9V. TelG mote is expected to operate using two AA batteries in series. While a single AA battery has the cut off voltage of 0.9V, two AA batteries in series will result in a cutoff voltage of 1.8V. 1.8V cutoff voltage is ideal for TelG mote given the operating voltage is the same value as the battery cutoff voltage. If a voltage regulator of 3.3V is placed in the design, the cutoff voltage would depend on the voltage regulator cutoff voltage which is approximately 2.7 V, wasting almost 50% of the battery lifetime

4.0 WISEOS OPERATING SYSTEM

Almost every mote available to date has its own lightweight operating system either it is being ported from other mote or specifically designed for it. TelG mote is developed to be a complete platform with its own operating system. For this purpose, WiseOS is developed base on the popular operating system for sensor mote, TinyOS which is written in nesC language. It supports modularity and concurrency based on event driven approach.

There are several factors that must be considered when designing an operating system for embedded system such as resource constraint (in term of memory, processor and power), modularity and concurrency. WiseOS design goals are to implement TinyOS basic kernel structure fully written in C language and have a small footprint for TelG mote. Additionally, it must also enable a flexible and rapid prototyping of WSN applications. This chapter provides the descriptions of the WiseOS architecture, its API and the design implementation of WiseOS.

4.1 WiseOS Software Architecture

The architecture of an operating system intended for WSN is inherently different with the legacy operating system. The constraints imposed on WSN proved that legacy operating system is impractical to be used. The intended software is required to be efficient in term of power, memory and processing capabilities while it must support multiple applications to be run simultaneously using the same system resources. In event-driven systems, it is conceivable that multiple event might occur simultaneously and must be handled concurrently. Based on these observations, concurrency and modularity plays a major role in determining the architectural design of the operating system. The software architecture must also enable an efficient integration of sensors, processing and communication.

While the bottlenecks of TelG mote capabilities are limited by its hardware design, to fully utilize the resources, efficient system software is of utmost important. To tackle the concurrency and modularity problem, WiseOS use the same technique employed by TinyOS which is two level scheduling structures where hardware interrupt can be processed almost immediately by interrupting long running tasks. This type of scheduling is also known as cooperative scheduling as oppose to preemptive scheduling.

4.1.1 WiseOS Design

WiseOS architecture is based on state machine programming model where the system is composed of state machines. In this model, the transitions of one state to another must be quick, have a low overhead and non-blocking. The non-blocking element is very important in designing an event-driven operating system such as WiseOS. Figure 9 shows an example of WiseOS operation in capturing and transmitting JPEG images.

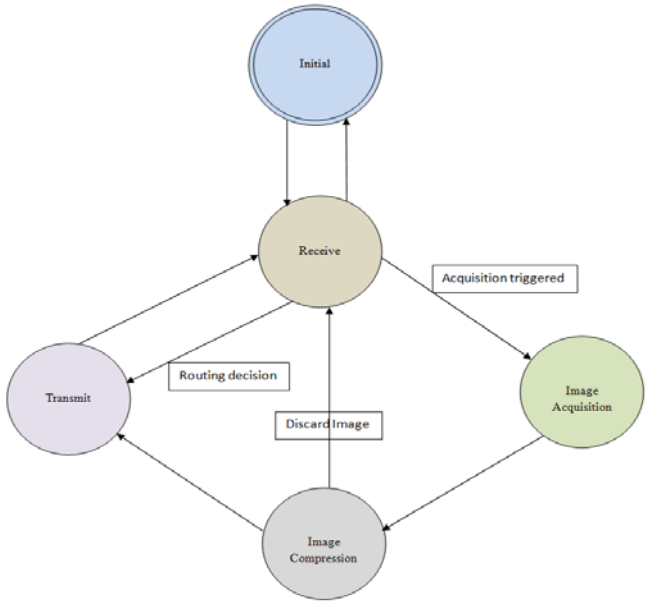

Figure 9 State transition diagram for capturing and transmitting JPEG images

The initial state is invoked when the central processing unit (CPU) detects power-on-reset, brown out reset or watchdog reset. In this state, the CPU executes the initial process in order to get the system up and running. The CPU can save up power consumption by turning on only the necessary peripherals such as UART and timer which is necessary in order to configure radio and sensing module. CPU enters idle mode once the camera and radio has been initialized and transit to the receiving state waiting for the UART interrupt.

If the radio module receives a message to acquire the image, the state will transit from receive state to the image acquisition state. In other scenario, the state transition can be triggered by other interrupt such as time out or from other sensors such as motion detection sensor. In this state, the sensor will start acquiring images to be processed. As mention before, C328R camera is capable

of processing the images on board or passing the image data to the CPU or any image processing unit available before transiting to the transmit state.

The compression state will produced a single JPEG file ready to be transmitted. In this state, a decision can be made whether to transmit the file or not based on the network condition or any other constrains imposed by the application. If the image has been decided to be transmitted, the transmit state will be invoked. Transmit state is responsible to pack the data into the necessary package format in order to transmit it over the network. The package is transmitted by the Xbee module to other nodes. Since each nodes in WMSN are capable of routing other nodes messages, the system only need to switch between receive and transmit state. The switching between these two states can be strategized in such a way that the power usage can be optimized.

In general, WiseOS architecture components can be described into several layers such as the hardware layer (HL), hardware abstraction layer (HAL), hardware presentation layer (HPL), network layer (NL) and application layer (AL). Figure 10 depict the WiseOS design.

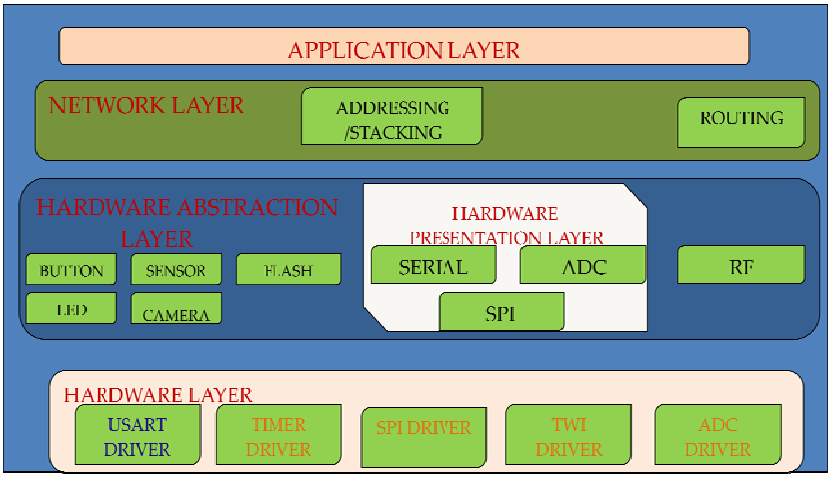

Figure 10 WiseOS design

Resource constraint mote such as TelG require the operating system to share the same memory space as the application (monolithic). As shown in Figure 10, the HL components such as USART, TIMER, SPI, two wire interface (TWI) and ADC, deals with the setting and initializing the hardware and this section normally have to be rewritten when porting WiseOS to another platform. The HPL is used to abstracting the HL by providing a set of standard API commonly used for that particular HL component. For example, USART driver only provides byte per byte basis data transmission on the HL. HPL elaborates this primitive operation into serial transmission by providing sending and receiving API for it in a bytes

stream. The HAL is used to represent the actual piece of hardware such as button, sensors and LEDs in an abstract manner by using the layers of abstraction provided by WiseOS which in turn making it independent from the hardware.

4.2 Event-Based Programming

WSN application is expected to operate for a long time without human intervention. To prolong wireless sensor network life-time expectancy, the node must be put into sleep state to reduce power consumption when there are no events. Since WSN only active when there are events, event based programming is very useful in order to achieve power usage efficiency. Event based programming is efficient in handling high level of concurrency using a very small amount of memory space. Thread based software architecture on the other hand requires a separate stack space for each execution context. For radio communication with a baseband processing of 19.2Kbps, the context switch rates of thread based software architecture require about 40000 switches per second [30]. This switch rates is significantly higher than event based architecture where only single stack is required and each context execution run until completion. Event based programming is expected to have a more stable performance compared to threaded program under heavy load [31].

WiseOS respond to event immediately by instructing the system to execute its context. The context is returned to the system only when the event processing is completed. Each event signaled by lower layer may have a task associated with it. The tasks will be handled after event is signaled. After all events and tasks are fully processed, the CPU is idle and thus can be put into low power state. Polling and blocking code which constantly looking for interesting events which waste CPU cycles are eliminated when using event based approach. Figure 11 illustrate the events and tasks processing in event-based programming.

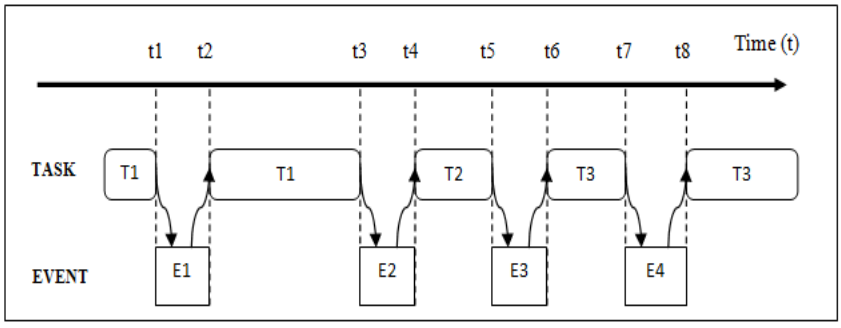

Figure 11 Events and tasks processing

Figure 11 illustrate the two level scheduling mechanisms in WiseOS. In this scheduling scheme, the event has the higher priority than the task. Event can be generated by hardware or software interrupt. Task T1 in Figure 11 is being executed by the scheduler but preempted at t1 by event E1. E1 would be run till completion before resuming task T1 at t2. As illustrated, T1, T2 and T3 are being executed in order and can only be preempted by an event. When there are no tasks or events to be processed, the mote can be put to sleep mode to conserve power consumption. Task operation is explained in details in the next section.

4.3 Tasks

If an event handler executes long running codes, other events may not be processed in time (overload). To overcome this limitation, WiseOS provides a mechanism called tasks. A task basically is a deferred procedure call. Task execution runs in the background without interfering with the system events. A task can be scheduled arbitrarily anywhere in the program as shown in Figure 12 but only get CPU time when all hardware interrupt has been processed.

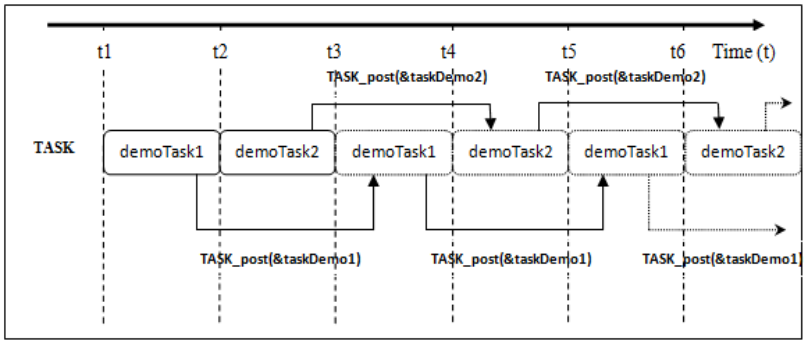

Figure 12 Emulating concurrency using tasks

This mechanism allows low-level events handler to have a minimal code which is time critical to be executed immediately while the rest of the code which are more computational intensive to be scheduled and processed later. Tasks may be preempted by the hardware interrupt but may not be preempted by another tasks and each tasks run to completion as shown previously in Figure 11.

WiseOS scheduler is being implemented using FIFO technique to schedule tasks. This technique is efficient in WSN environment and furthermore it is easy to implement. Tasks with priority may be implemented, however it is not a major problem in WSN where event is expected to be detected sporadically and most of the time the node is inactive.

4.4 WiseOS Modules

Modules in WiseOS have been designed to be modular to provide easy component integration for application developer. Event driven operating system requires each module to have a set of event/command interfaces. Application developer only needs to interconnect the modules using the interfaces to produce a complete system.

WiseOS modules have four main components, a set of commands, a set of event handlers, private data and tasks/functions associated with that particular module. When these modules are being interconnected to each other, a layered system composition can be easily derived as shown in Figure 13.

A Command illustrated by arrow pointing inside the block in Figure 13 may provide a feedback (arrow pointing out) to its caller to indicate the status of the command e.g. failed, success or buffer overrun and etc. The feedback gives important information to the caller about the lower layer current state for error handling in case it is needed. Each blocks on each layer in WiseOS are independent of each other which translated to each commands on each blocks are independent with each other. This non-blocking scheme is realized using task, where every command are a deferred procedure called which enable the developer to call arbitrary commands from arbitrary blocks successively without having to wait the previous command to finish first. Commands and events provide a seamlessly instantaneous state transition for each module and tasks simulate concurrency for event-driven model to provide an arbitrary computation.

4.5 WiseOS Evaluation

To evaluate WiseOS, the key criteria are small footprint and have an efficient modularity. Small footprint is desirable in TelG mote given the size of the RAM is only 4 KB. Efficient modularity plays a major role in application development that would enable rapid prototyping of WSN applications.

Figure 14 shows WiseOS compilation output of its kernel (scheduler and timer) that display the usage of the read only memory (ROM) and RAM.

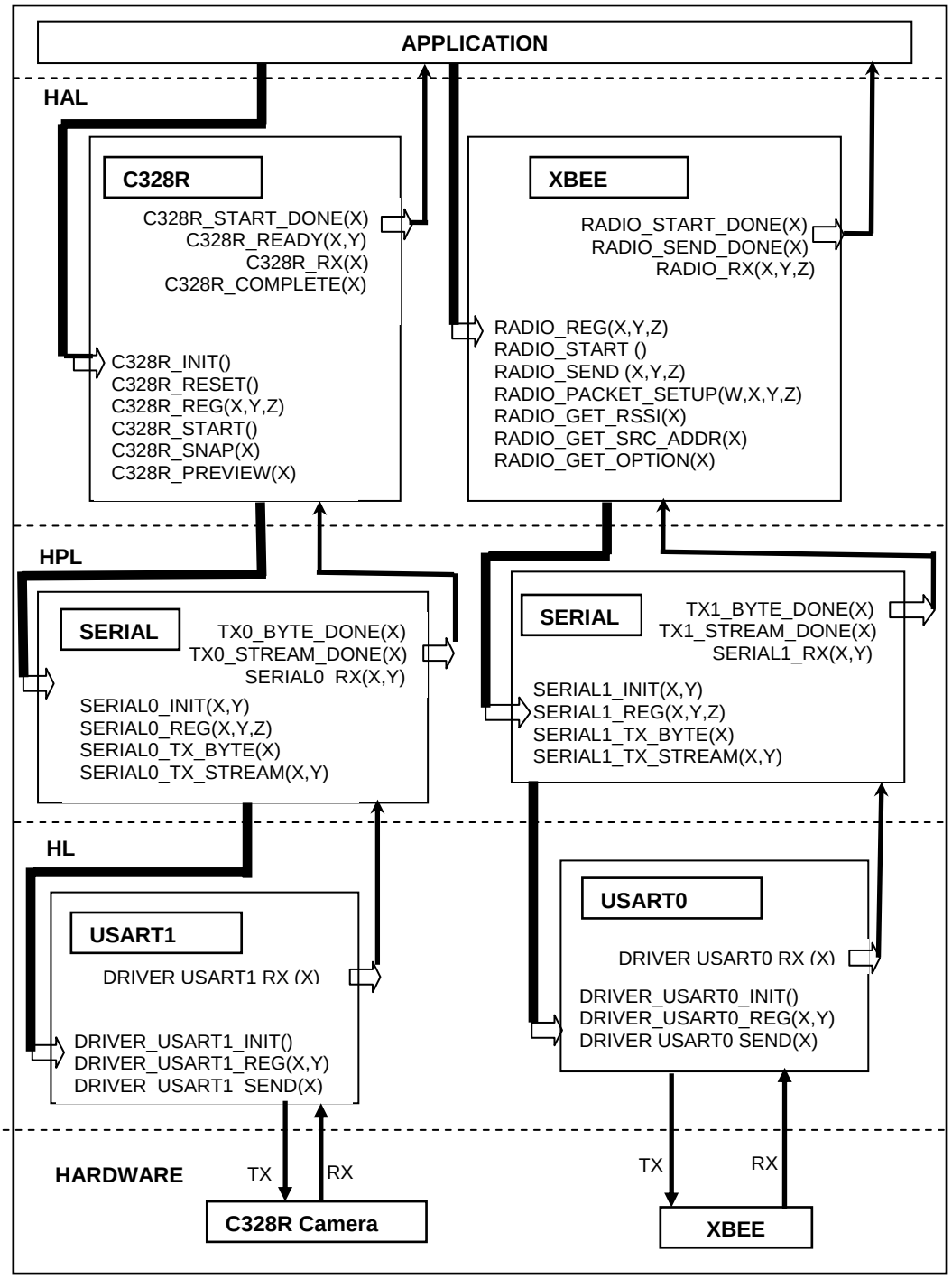

Figure 13 Functional block diagram of WiseOS call graph

```
Size after:
AVR Memory Usage
----------------
Device: atmega644p

Program:    1632 bytes (2.5% Full)
(.text + .data + .bootloader)

Data:         21 bytes (0.5% Full)
(.data + .bss + .noinit)
```

Figure 14 WiseOS kernel compilation report

As shown in Figure 14, WiseOS kernel only occupies 2.5% of the Flash memory (Program) and 0.5% of the RAM (Data). The ROM usage most of the time is not a major concern compared to the usage of RAM. As reported, 0.5% of RAM usage is indeed small which leave 95.5% of the RAM to be used by the application developer.

WiseOS has been developed in modular manner where a modules from several layer can be snapped together easily to compose a complete system. When porting WiseOS to another platform that conform to American national standards institute (ANSI) C language, only hardware layer of the WiseOS need to be rewritten and the application program can be used with little or no modification.

Figure 15 shows a sample of image captured by TelG mote and transferred to sink node (PC) and Table 3 summarizes the performance for single hop communication at different image resolutions.

Figure 15 JPEG images at 320x240 resolution for single hop communication

Table 3 Image transfer performance for single hop communication at different image resolutions

Resolution	80 x 64	160 x 128	320 x 240	640 x 480
Delay (s)	0.61	1.46	6.66	17.22
PRR (%)	100	100	100	100
Throughput (kbps)	18.51	15.44	17.15	16.98
Corrupted packets (%)	0.003	1.25	1.02	1.00
Frame rates (fps)	1.63	0.68	0.15	0.06

From Table 3, it is clear that the delay to transfer a single frame of image would increase as the resolution of the image increases given that the maximum speed of the communication is fixed at 115.2kbit/s. For a single source, single hop communication where the source and the sink node are placed together at a close proximity to each other, packet collision does not occur which resulted in a 100% PRR. The throughputs attainable from the experiments are statistically consistent for each image resolution with the variance only valued at 1.58. The attainable throughput is only 15% of the maximum throughput provided by the communication devices mainly due to the processing delay of the CMOS camera and communication delay.

5.0 CONCLUSION

A typical WSN applications requirements are low rate, low power and only accommodate scalar data. To accommodate multimedia data into the conventional WSN many challenges need to be overcome. This presents the development of WMSN motes namely TelG mote equipped with operating system called WiseOS and multimedia sensor consisting of a CMOS camera. TelG mote is built using Atmel's microcontroller as its CPU, XBEE radio module as the transceiver and CMOS camera from CoMedia as its visual sensor. Atmel's ATmega644PV is chosen mainly because of its low power profile, rich in peripherals and have sufficient size of RAM (4KB) and ROM (64KB) for WSN applications. XBee radio module is based on the famous IEEE802.15.4 standard and provides a user friendly API for application developer and also using USART interface. C328R CMOS camera provides an easy to use API to access the camera with built in JPEG image compression, automatic packet fragmentation and using the same USART interface as radio module. TelG mote is design for the development of WMSN testbed.

Embedded operating system for wireless sensor network platform often comes in package such as TelosB and MicaZ uses TinyOS and Contiki for Sensinode. WiseOS has been developed for TelG mote. Operating system is very important and crucial in managing mote resources such as power and memory. The

proposed facilitate mote operation such as communication and reading sensors data. WiseOS has been developed using TinyOS as the framework and guideline. WiseOS implements an event-driven type operating system, which is very suitable for WSN environment where the motes only respond to events instead of polling and sleep during idle time.

In general, the work has successfully achieved the target objective that to develop and design a platform for WMSN. The development of the hardware, which includes TelG mote and the operating system which is the WiseOS and the experimental WMSN test bed has proven the practicality of the proposed WMSN system for surveillance, monitoring and detection. The WMSN can be tailored for application including environmental monitoring, human and device surveillance system and many other applications that require visual information.

6.0 FUTURE WORKS

The proposed WMSN using TelG mote, WiseOS and low rate image or video streaming application demonstrate that many other applications can be deployed for various surveillance and monitoring purposes. However, the proposed WMSN can be further improved by using enhanced devices for TelG mote, efficient algorithm on the embedded operating system and efficient routing protocol for multimedia data transfer. Below are the recommendations for future works;

- The processing unit of TelG mote can be improved by using a digital signal processing (DSP) coprocessors to process the multimedia data.
- TelG mote may be further improved by using high-end microcontrollers such as the 32-bit Intel/Marvell PXA255 processor that provides a high speed processing as well as a higher degree of parallelism.
- Using multiple processor approach dedicated to process multimedia data on the same platform connected to a multipurpose low-end processor for interfacing with the transceiver and the imaging unit. The trade-off between power consumption and processing capabilities must be considered in such system.
- System software can be developed to provide a basic set of interface that can be accessed easily using APIs. WiseOS is highly efficient for WSN environment but its flexibility and interoperability can be further improved.
- Sensing WMSN needs further improvement to avoid redundancy in sending the same information by multiple sources. Signal processing

technique like edge detection to avoid sending raw data over the wireless link can be included.

- Using CMOS camera with varying resolution can make use of the wireless link condition by adaptively switching its resolution.

REFERENCES

[1] Ian, F. A., T. Melodia, K. R. Chowdhury. 2007. A Survey on Wireless Multimedia Sensor Networks. *Computer Networks (ScienceDirect)*. 51(4): 921-960.

[2] Jennifer, Y., B. Mukherjee, Dipak, Ghosal. 2008. Wireless Sensor Network Survey. *Computer Networks*. 52(12): 2292-2330.

[3] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, K. Frampton. 2004. Sensor Network-Based Counter Sniper System. Proceedings of the Second International Conference on Embedded Networked Sensor Systems (Sensys). November 03-05, 2004. Baltimore, MD, USA: ACM Press. 1-12.

[4] Ian, F. A., T. Melodia, K. R. Chowdhury. 2008. Wireless Multimedia Sensor Networks: Applications and TestBeds. *Proc. of the IEEE*. 96(10):1588-1605.

[5] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, M. Srivastava. Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks. Proc. Of the ACM Conf. on Embedded Networked Sensor Systems (SenSys). November 02 - 04, 2005. San Diego, CA: ACM. 2005. 192-204.

[6] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, D. Culler. The NesC Language: A Holistic Approach to Network Embedded System. 2003. Proc. Of the ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation (PLDI). June 09 - 11, 2003. San Diego, CA, USA: ACM. 1-11.

[7] A. Rowe, A. Goode, D. Goel, I. Nourbaksh. 2007. *CMUcam3: An Open Programmable Embedded Vision Sensor*. Pittsburgh, PA: Technical Report.

[8] J.Bruce, T. Balch and M. Veloso. 2000. Fast and Inexpensive Color Segmentation for Interactive Robots. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. 2000 (IROS 2000).Oct 31-Nov. 5, 2000. Takamatsu , Japan: IEEE. 2061-2066.

[9] G.D. Hager and K. Toyama. 1998. The Xvision System: A General Purpose Substrate for Real-Time Vision Applications. *Computer Vision and Image Understanding*. 69(1): 23-37.

[10] I. Horswill. Polly: *A* Vision-based Artificial Agent. 1993. The Proceeding of Eleventh National Conference on Artificial Intelligence. July 11–15, 1993. Washington DC: AAAI Press.. 824-829.

[11] R. Sargent, B. Bailey, C. Witty and A. Wright. 1997. Dynamic Object Capture Using Fast Vision Tracking. *AI Magazine*. 18(1). 65-72.

[12] I. Ulrich,and I. Nourbakhsh. 2000. Appearance-Based obstacle Detection with Monocular Color Vision. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. July 30 - August 03, 2000. Austin, Texas, USA: MIT Press.. 866-871.

[13] Crossbow Technology Inc. 2004. *Stargates Developer's Guide (Rev. A)*. San Jose California. User Manual.

[14] C. Lynch and F. O. Reilly. 2005. Processor Choice for Wireless Sensor Networks. June 20-21, 2005. Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Stockholm, Sweden: Sensornets. Session 5.

[15] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler and K. S. J. Pister. 2000. System Architecture Directions for Networked Sensors. *ACM SIGPLAN Notices*. 35(11): 93-104.

[16] I. Downes, L. B. Rad and H. Aghajan. 2006. Development of a Mote for Wireless Image Sensor Networks. Proc. Cogn. Syst. Interact. Sensors (COGIS). March 16, 2006. Paris, France: SEE.

[17] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson and R. Han. Mantis OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. *Mobile Networks and Applications*. 2005. 10(4):563-579.

[18] C. L. Liu and J. W. Layland. Scheduling Algorithm for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*. 1973. 20(1): 46-61.

[19] J. P. Lehoczky, L. Sha and Y. 1989. Ding. The Rate Monolithic Scheduling Algorithm: Exact Characterization and Average Case Behavior. Proceedings Real Time Systems Symposium. Dec. 05-07, 1989. Santa Monica, CA , USA: IEEE Xplore. 166-171.

[20] L. Sha, R.Rajkumar and J. Lehoczky. 1990. Priority Inheritance Protocols: An Approach to Real Time Synchronization. *IEEE transactions on Computers*. 39(9): 1175-1185.

[21] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. D. Kim, B. Zhou and E. G. Sirer. 2002. On The Need for System-Level Support Ad-Hoc and Sensor Networks. *ACM SIGOPS Operating Systems Review*. 36(2):1-5.

[22] A. Dunkels, B. Gronvall and T. Voigt. 2004. Contiki – A Lightweight and Flexible Operating System for Tiny Networked Sensors. Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks. November 16 - 18, 2004. Washington DC, USA: IEEE Xplore. 594-601

[23] C. C. Han, R. Kumar, R. Shea, E. Kohler and M. Srivastava. 2005. A Dynamic Operating System for Sensor Nodes. Proceedings of The 3rd International Conference on Mobile Systems, Applications, And Services. June 06 - 08, 2005. Seattle, Washington: ACM portal. 163-167.

[24] J. Lifton, D. Seetharam, M. Broxton and J. A. Paradiso. 2002. Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks. In: Mattern, Friedemann and Naghshineh, Mahmoud. *Pervasive Computing*. Berlin / Heidelberg: Springer. 605-614.

[25] J. Yannakopoulus and A. Bilas. 2005. Cormos: A communication-Oriented Runtime System for Sensor Networks. Proceeding of The Second European Workshop on Wireless Sensor Networks (ESWN 2005). 31 Jan.-2 Feb. 2005. Istanbul, Turkey: IEEE Xplore. 342-353.

[26] P. Levis and D. Culler. 2002. Mate: A Tiny Virtual Machine for Sensor Networks. Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems. October 05 - 09, 2002. San Jose, CA, USA: ACM Portal. 85-95.

[27] S. Dulman and P. Havinga. 2002. Operating System Fundamentals for The EYES Distributed Sensor Network. Proceeding of Progress 2002, Utrecht. Netherlands, October.

[28] J. Ousterhout. 1996. Why Threads are a Bad Idea (For Most Purposes). Invited Talk at 1996 USENIX Technical Conference. Real-Time Computing and Communications Lab., Hanyang University.

[29] R. von Behren Jeremy Condit and E. Brewer. 2003. Why Events are a Bad Idea (For High-Concurrency Servers). Proceedings of HotOS IX: The 9th Workshop on Hot Topics in Operating Systems, May 18–21, 2003, Lihue, Hawaii, USA: ACM Portal. 4-4.

[30] S. Hong and T. Kim. 2003. SenOS: State-Driven Operating System Architecture for Dynamic Sensor Node Reconfigurability. In International Conference on Ubiquitous Computing, October 12-15, *2003*. Seattle, Washington: Springer. 56-60

[31] Dabek, F., Zeldovich, N., Kaashoek, F., Mazières, D., and Morris, R. 2002. Event-Driven Programming For Robust Software. In Proceedings of the 10th Workshop on ACM SIGOPS European Workshop. July 01, 2002. Saint-Emilion, France: ACM. 186-189.

[32] J. Lifton, D. Seetharam, M. Broxton and J. A. 2002. Paradiso. Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks. In: Mattern, Friedemann and Naghshineh, Mahmoud. *Pervasive Computing*. Berlin / Heidelberg: Springer. 605-614.

[33] Atmel Corp. 2009. 8-bit Microcontroller with 16/32/64K Bytes in-System Programmable Flash. San Jose California: Datasheet.

[34] MaxStream Inc. 2005. XBee™/XBee-PRO™ OEM RF Modules. Lindon UT. Product Manual.