

Chapter 2

Vision-enabled WSN Nodes: State of the Art

2.1 A Brief Introduction to WSNs

A *Wireless Sensor Network* can be defined as a set of autonomous sensor nodes which are densely and wirelessly deployed throughout a region of interest (Akyildiz et al. 2002). Each node, commonly referred to as a *mote*, includes sensing, data processing and communicating components. Its usual workflow consists of sensing data, processing it locally and communicating only the required information resulting from this processing. A remarkable feature is that the exact location of the nodes comprising the network does not need to be predetermined. Communication protocols and algorithms possess self-organizing capabilities. Typically, sensor nodes will be clustered. Within each cluster, a node acts as the head, routing data coming from the rest of sensors to a specialized node called sink, or base station. Multi-hop wireless communication is used throughout this process. End users can retrieve data and re-configure the network through the base station. A generic sketch of a multi-cluster WSN is depicted in Fig. 2.1.

Actual deployments of WSNs usually implement one of three general configurations (Kulkarni et al. 2010), namely: *periodic reporting*, *event detection*, and *database-like storage*. *Periodic reporting* is by far the most used application scenario: nodes sample regularly their environment, process the sensory data and store the result to be eventually sent to the base station. In *event detection*, nodes sense their environment and evaluate the data immediately for its usefulness. If an event of interest is detected, it is notified at once. In *database-like storage systems*, all sensory data (periodic sampling and/or events) is stored locally in the nodes. End users, by means of the base stations, search for interesting data and retrieves it directly from the corresponding nodes.

Networks, once deployed, must be able to collect data uninterruptedly for weeks, months or even years. Since energy consumption determines the node lifetime, sensors have limited computational and communication resources. They include lightweight embedded processors with a few MBytes of RAM. These processors usually run at 10–100 MHz at most, rather than typical GHz in current 64-bit CPUs. Their low-power radios can send to a few hundreds of kbps, rather than 802.11's tens of Mbits. As a result, software needs to be very efficient in terms of both CPU

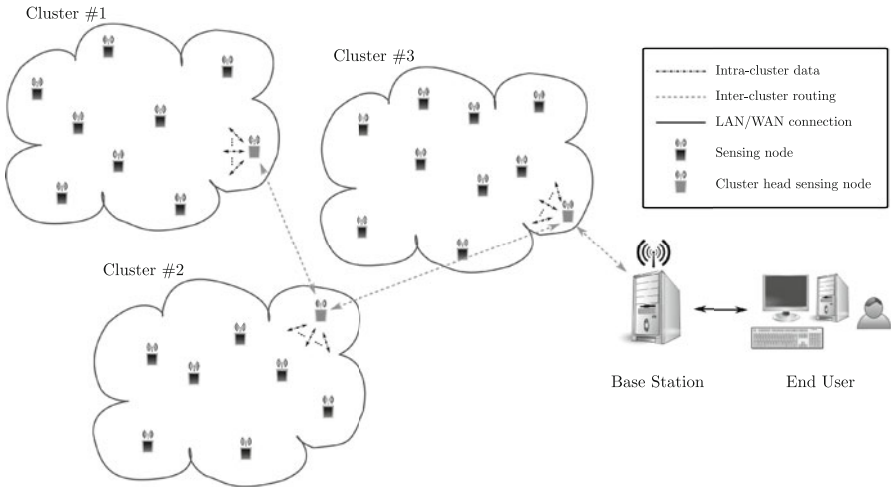


Fig. 2.1 Sketch of a multi-cluster WSN

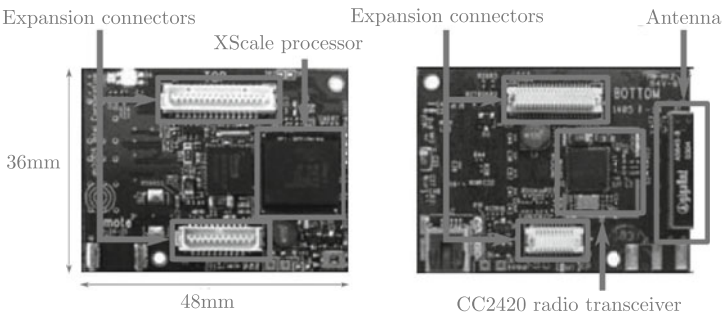


Fig. 2.2 Top view (*left*) and bottom view (*right*) of the *Imote2* platform

cycles and memory use. Consider Fig. 2.2 as an example. It shows a commercial WSN node, the *Imote2* platform (Imote2 data, MEMSIC). This node is built around a 32-bit ARM5 processor (PXA271, Marvell) which can operate in a low voltage (0.85 V), low frequency (13 MHz) mode, hence enabling very low power operation. The PXA271 is really a multi-chip module which also includes 256 kB SRAM, 32 MB SDRAM and 32 MB of FLASH memory. An 802.15.4-compliant radio transceiver (CC2420, Texas Instr.) is also integrated into the *Imote2* system. Note that external sensor boards can be connected through expansion connectors. In fact, the prototype vision chip reported in this book has been interconnected to the *Imote2* platform by using these connectors, as thoroughly described in Chap. 6.

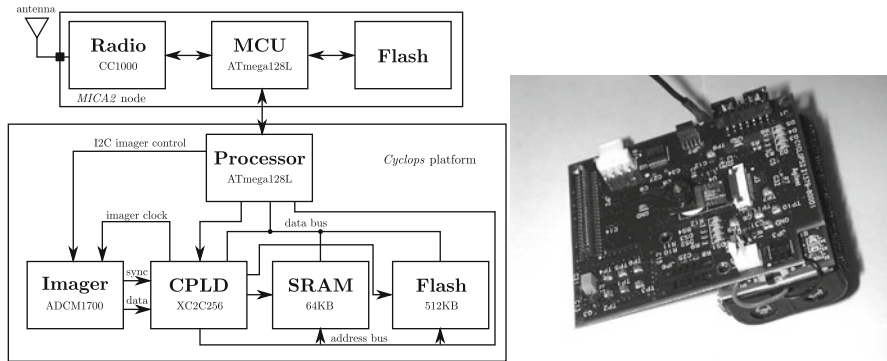


Fig. 2.3 Basic interconnection sketch of the *Cyclops* platform and a snapshot of the system. (With permission of the authors)

2.2 A Survey on Vision-enabled WSN Nodes

2.2.1 *Cyclops* (Rahimi et al. 2005)

The term *Cyclops* refers to an electronic interface between an off-the-shelf camera module and a lightweight wireless host. The philosophy behind this approach is to separate the computational resources used for imaging from those ones assigned to communication. Thus, *Cyclops* consists of an 8-bit RISC processor (ATmega128L, ATML), a CPLD, external SRAM (64 KB) and external FLASH (512 KB). The camera module is an ultra compact CIF resolution (352×288 pixels) CMOS imager (ADCM-1700, Agilent). The processor controls the communication with a wireless host (MICA2, MEMSIC), as well as the different parameters of the imager for frame capture. The CPLD provides high speed clock, synchronization and memory control. Both processor and CPLD can perform local computations on the image to produce an inference. The camera module contains a complete image processing pipeline for demosaicing, image size scaling, color correction, tone correction and color space conversion. It also implements automatic exposure control and automatic white balance. A basic sketch of the *Cyclops* platform and the interconnection of the different modules is depicted in Fig. 2.3 along with a snapshot of the system, reproduced here with permission of the authors. Note that the camera is embedded into the same PCB than the rest of elements comprising *Cyclops*.

For the sake of transparency in using resources such as the imager or the SRAM, firmware has been written in *nesC* (Gay et al. 2003), a programming language tailored for TinyOS (Levis and Gay 2009). One of the main advantages of having separated the computational resources for image processing (*Cyclops*) from those ones dedicated to wireless communication (*MICA2*) is flexibility to develop this firmware. The design principle is to support the long synchronous operations associated with image processing. Time-consuming computational blocks performed over

serialized images can be implemented without considering possible starvation for asynchronous requirements of the network. From the opposite point of view, image processing is speeded up on not having to deal with such requirements.

The *Cyclops* platform presents different power states according to the resources required to perform a certain operation. The worst case corresponds to permanent writing memory access, with a power consumption of 64.8 mW. For the image sensor, a typical consumption of 42 mW is reported. Finally, the *MICA2*'s MCU presents a power consumption of 24 mW. The radio module reaches 81 mW when transmitting with maximum power. Adding up all these figures, a simple application of capturing an image sequence, storing it and transmitting it would mean a power consumption of 211.8 mW, assuming that all modules are working concurrently.

Regarding the frame rate, the values reported are low due to the significant latency introduced by the pipeline and firmware layers. For image sizes of 32×32 px, 64×64 pixels and 128×128 pixels and room light conditions, the maximum frame rate achievable is, respectively, 12.5, 11.1 and 7.1 fps. If some processing is realized, frame size affects almost linearly the additional time required to complete the whole computation. For example, Sobel derivatization or background subtraction can be carried out with maximum frame rates of, respectively, 11.8, 8.7 and 4.3 fps.

Finally, two applications are presented, namely, object detection and hand posture recognition. The key point in the development of these applications is the progressive data reduction along the flow of execution. To start with, images are kept with the minimum size which the application can afford. Besides, the algorithms arrange their computations in such a way that data is rapidly cut down. Additionally, results of previous computations are reused multiple times, saving clock cycles and memory. Thus, the object detection algorithm runs at 10.3, 6.6 and 2.6 fps for frame sizes of 32×32 pixels, 64×64 pixels and 128×128 pixels, respectively. This algorithm constructs the stationary background by analyzing a certain number of images. Then, it generates a model of the foreground based on the absolute difference of the instantaneous image and the constructed background. Using a threshold filter and, subsequently, a blob filter, the object map is eventually obtained. The threshold value is constantly estimated as a factor of the background illumination. Regarding the hand posture recognition application, the algorithm runs below 2 fps for a frame size of 64×64 pixels. In this case, the algorithm tries to recognize static hand gestures. It transforms an image into a feature vector which is then compared to the feature vectors of a training set of gestures. This transformation is based on the calculation of the orientation histogram, which is in turn calculated by applying the Sobel derivative masks.

Therefore, as a summary, it can be said that the main feature of *Cyclops* is the total separation of the computational resources dedicated to image processing from those ones dedicated to network processing. This simplifies the development of firmware and speeds up the execution of vision algorithms. Concerning the strategies for energy efficiency, image processing is based on a constant data reduction and reuses computations in order to save clock cycles and memory accesses.

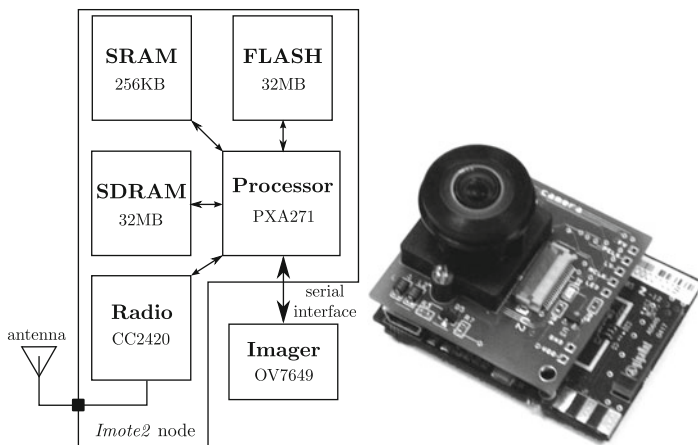


Fig. 2.4 Simplified interconnection scheme of the *Imote2*-based platform and a snapshot of the system. (With permission of the authors)

2.2.2 *Imote2-based Platform (Teixeira et al. 2006)*

This vision-enabled WSN platform is quite simple. It consists of a direct interconnection between the *Imote2* node, briefly described in Sect. 2.1, and a commercial CMOS imager (OV7649, Omnivision) through a serial interface. This imager can capture color VGA images (640×480 pixels) at 30 fps and QVGA (320×240 pixels) at 60 fps with a power consumption of 44 mW. Automatic exposure control, gain control and white balance are additional functionalities available. In fully-active mode, with *Imote2* working at 104 MHz and a frame rate of 8 fps, the system consumes 322 mW. A simplified interconnection scheme of this platform along with a snapshot of the system is depicted in Fig. 2.4.

An application to assisted living is presented (Teixeira and Savvides 2008). Six *Imote2*-based platforms are placed at different points of the ceiling of a room. They are a single hop away from their base station in a star topology. The nodes track the presence of people by analyzing images downsampled to 80×60 pixels at around 14 fps with the processor working at 208 MHz. Time-stamped information is then forwarded from each node to the base station, which aggregates it and reports accordingly to a gateway computer. The ultimate objective is to recognize unsafe and out-of-the-ordinary behaviors of the people walking around the room.

Due to the simplicity of this WSN node, there are two major parameters to be adjusted in order to reduce power consumption. First of all, the speed of the processor. For the application reported, the processor works at 208 MHz. A slower operation mode is possible by setting the speed of the processor to 13 MHz. The other possibility is to reduce the image resolution as much as the application under consideration can afford.

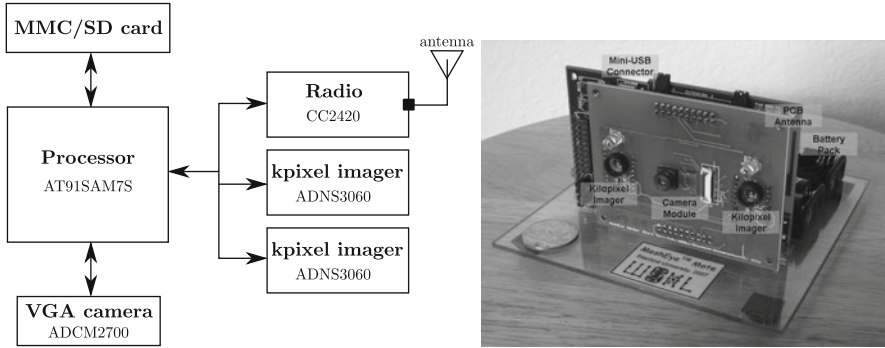


Fig. 2.5 Block diagram of the *MeshEye*TM platform along with a snapshot of the system. (With permission of the authors)

2.2.3 *MeshEye*TM (Hengstler et al. 2007)

This wireless smart vision node is an extension of the work introduced in (Downes et al. 2006). It is mostly oriented to applications within the field of distributed intelligent surveillance. Its design is characterized by the implementation of a direct interface between the embedded processor and several image sensors. No FPGA or CPLD device is used. The core of the system is a 32-bit ARM7 processor (AT91SAM7S, ATMEL). Up to eight kilopixel imagers can be hosted in the node. A VGA (640 × 480 pixels) camera module is also supported. The wireless connection to other nodes is established through an 802.15.4-compliant radio transceiver (CC2420, Texas Instr.). Finally, a MMC/SD flash memory card provides sufficient non-volatile memory for temporary frame buffering and even image archival. A block diagram of the *MeshEye*TM system is depicted in Fig. 2.5 along with a snapshot of the system, reproduced here with permission of the authors.

In a first implementation, two kilopixel imagers (ADNS-3060, Agilent) were used, with a VGA camera (ADCM-2700, Agilent) centered between them. Every kilopixel imager features a high-speed sensor with a resolution of 30 × 30 pixels outputting 6-bit grayscale images at up to 50 fps. The VGA module can deliver grayscale or 24-bit color images. All these pixel arrays are placed in parallel and face the same direction, focusing to infinity. They work collaboratively in the following way. One of the kilopixel imagers is continuously polling for moving objects entering its field of view (FoV). Once one or possibly more objects have been detected, position and size are determined for each object. Basic stereo vision of the two kilopixel imagers yields the distance to the object. This information allows the calculation of the region of interest (ROI) containing the object within the VGA camera's image plane. Thus, the processor triggers this camera in order to capture a high-resolution grayscale or color ROI which includes the detected object. Finally, this ROI is processed according to the specific target of the application. The reason behind using kilopixel imagers for stereo vision instead of simply two VGA camera modules is to reduce

energy consumption. Although downsampled frames could be delivered by these VGA modules in order to analyze kilopixel images, they still acquire the entire pixel array internally and downsample it digitally. Hence low-resolution frame capture with a kilopixel imager is more energy-efficient even if the power consumption per pixel is similar in both cases.

Image processing algorithms for object detection, stereo matching and object acquisition are presented. They are intentionally kept at low computational complexity in order to reach the corresponding application target at minimum energy cost. Object detection is performed over the left kilopixel imager through a very simple background subtraction on a frame-by-frame basis. Then, stereo matching correlates the result of this detection with the difference between the current frame and the background representation of the right kilopixel imager. If a positive match cannot be established, the left imager will continue detecting the object while it is within its FoV. This happens when the object lies outside the overlapping FoV of the two kilopixel imagers. Otherwise, knowing the object's position within both imagers and its size, the VGA camera module is triggered to acquire a high resolution snapshot of the object. A power model of this basic surveillance operation is reported. It also includes the power consumption of the radio transceiver communicating the result of intermediate level processing over the high resolution snapshot of the object. Two AA batteries power the whole system. Several frame rates below 2 fps are considered. As an example, the power consumption for 0.5 fps while no object is detected is about 7 mW. When objects with a size of around 64×64 pixels show up within the scene being surveyed, the power consumption reaches around 155 mW. Therefore, the activity within the scene governs the lifetime of the system. Frequent occurrences of objects reduce considerably its lifetime.

In a nutshell, *MeshEye*TM stands out for the direct interface between an embedded processor and vision devices. No FPGA or CPLD is included. Its design has been realized with a specific application in mind: distributed intelligent surveillance. For a basic surveillance task, it presents very competitive power consumption figures. However, its uptime is greatly influenced by the motion within the scene surveyed. Thus, *MeshEye*TM rapidly starves in scenes with much activity.

2.2.4 WiCa (Kleihorst et al. 2007)

Low-level image processing tasks (González and Wood 2002), also known as early vision tasks, are characterized by their regularity. They are equally defined for each pixel as a function of its own value and its immediate neighborhood. More importantly, the computation over each pixel is often independent from the result of the computations over the rest, opening the door for parallel processing. The *WiCa* platform takes advantage precisely of this potential parallel processing to reduce the number of memory accesses, clock speed and instruction decoding. The key component to achieve it is a high-performance digital processor called *Xetal-II* (Abbo et al. 2008). This processor is designed ad-hoc for frame-based real-time video analysis.

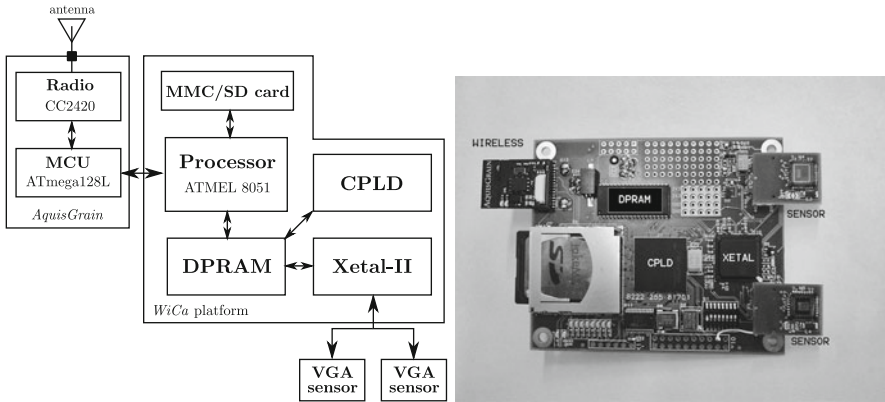


Fig. 2.6 Interconnection scheme of *WiCa* system and snapshot of the platform. (With permission of the authors)

It is based on the SIMD (Single-Instruction Multiple-Data) paradigm (Unger 1958), which fits perfectly with the characteristics of low-level image processing tasks just mentioned. *Xetal-II* incorporates 320 PEs which work in parallel to reach a measured peak performance of 107GOPS when operating at 84 MHz and 1.2 V, with a power consumption of 600 mW, that is, around 5.6 mW/GOPS. This number of PEs, 320, is an integral divisor of the image lines for most standard video formats, what makes it ideal for parallel processing no matter the format finally chosen. The chip is manufactured in 90 nm CMOS technology and occupies 74 mm². It also integrates a 10 Mbit memory allowing for energy-efficient inter-frame and intra-frame computations. The programming of *Xetal-II* is carried out by using an extended C programming language called XTC whose major feature is the introduction of a vector data type to represent 320-element variables. In addition to *Xetal-II*, *WiCa* includes a general-purpose processor (ATMEL8051, ATMEL) for control and medium- and high-level image processing. Both processors are coupled using a dual port RAM controlled by a CPLD that enables them to work in a shared workspace on their own processing pace. Finally, a MMC/SD memory card provides extra non-volatile memory. For radio communication, an external module called *AquisGrain* (Espina et al. 2006) containing a transceiver (CC2420, Texas Instr.) is connected to *WiCa*. Regarding vision sensors, one or two off-the-shelf VGA color image sensors (OM6802, Philips) can also be connected to the platform. These sensors work at 30 fps with digital RGB output. They can also be set at 60 fps in CIF resolution. The interconnection scheme of *WiCa* is depicted in Fig. 2.6 together with a snapshot of the platform, reproduced with permission of the authors.

No direct application of this version of the *WiCa* platform is presented. However, by making use of a previous version (Kleihorst et al. 2006), which includes the precursor of *Xetal-II* processor, applications like Canny edge detection (Geelen et al. 2009) or real-time gesture recognition (Zivkovic et al. 2008) have been reported. In

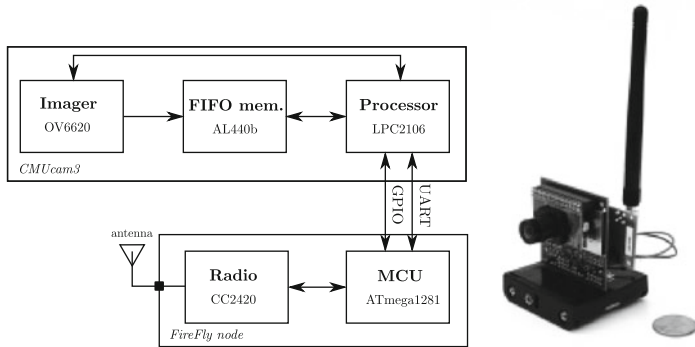


Fig. 2.7 Block diagram of the *FireFly Mosaic* platform along with a snapshot of the whole system. (With permission of the authors)

these two application examples, a power consumption of around 500 mW is achieved mainly by reducing the operation speed of the visual co-processor.

The *WiCa* platform can be considered as the first approach to enable vision into WSN nodes which implements an architecture adapted to the nature of visual processing. During the processing of an image sequence, the greatest computational effort usually falls on the so-called early vision tasks. These tasks feature a very regular computational flow which must be applied independently to each element of a large data set. The integration of *Xetal-II*, a dedicated SIMD-based visual co-processor designed ad-hoc to deal only with early vision, improves the efficiency of the system by carrying out a parallel implementation of these tasks. Unlike pipeline schemes, this processor achieves full-parallel operation. In other words, the result of applying a certain instruction to a set of pixels is output at the same time for every pixel of the set, speeding up greatly the completion of the corresponding task.

2.2.5 FireFly Mosaic (Rowe et al. 2007b)

This platform consists of a *FireFly* (Rowe et al. 2006) wireless node coupled with a *CMUcam3* (Rowe et al. 2007a) embedded vision processor. Its architecture is similar to that of *Cyclops*, described in Sect. 2.2.1. The computational resources for vision processing are again separated from those of wireless communication. *CMUcam3* gathers the components concerning vision processing, namely: a CIF CMOS color image sensor (OV6620, Omnivision), a FIFO memory for buffering these images and a 32-bit ARM7 processor (LPC2106, Philips), with 64 KB of on-chip RAM and 128 KB of on-chip FLASH memory. On the other hand, *FireFly* node manages the wireless communication by means of an 8-bit MCU (ATmega1281, ATMEL) and an 802.15.4-compliant radio transceiver (CC2420, Texas Instr.). The interconnection *CMUcam3-FireFly* is carried out through 5 GPIO ports and a serial port. A sketch of the platform is shown in Fig. 2.7.

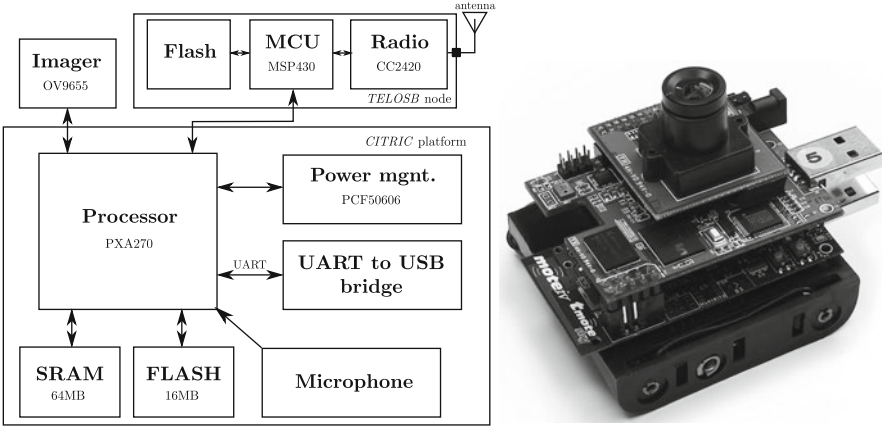


Fig. 2.8 Block diagram of the *CITRIC* platform along with a snapshot of the whole system. (With permission of the authors)

The main hardware difference between *Cyclops* and *FireFly Mosaic* is that the former needs a CPLD to complement the processor for image processing whereas the latter incorporates a more powerful processor which eliminates such a need. Another remarkable feature of *FireFly Mosaic* is the emphasis put on the problems related to transmitting information across a dense WSN. A time-synchronized link protocol for energy-constrained multi-hop wireless networks is implemented. Regarding the power consumption, a maximum value of 572.3 mW is reported when all the components are active simultaneously, being the FIFO memory the most power consuming module with 171 mW. The minimum value, with all the components sleeping, is 0.29 mW.

Finally, an interesting application involving eight nodes is presented. These nodes are deployed throughout a house in order to extract relevant information about the activity occurring inside, for example a person falling. This information is obtained by a background subtraction algorithm and subsequent region clustering. The result is sent to a PC for further analysis. The frame rate at each node is 5.2 fps, operating at approximately 20% duty cycle. This means, according to the authors, that the *motest* can run for just over 5 days from four AA batteries.

2.2.6 CITRIC (Chen et al. 2008)

The general configuration of this platform is also very similar to *Cyclops*, as can be seen in Fig. 2.8. There is a dedicated processor (PXA270, Marvell) for visual processing which receives images from a 1280×1024 pixels CMOS image sensor (OV9655, Omnivision). This processor directly communicates with a commercial

mote (TELOSB, MEMSIC), in order to endow the system with wireless communication. An interesting feature of *CITRIC* is the integration of a microphone. The information sensed by this microphone is also analyzed by the vision processor, which therefore becomes a multimedia processor. It makes the field of application of this platform much wider.

Different figures of power consumption are reported assuming four AA batteries connected to the system. First, with *CITRIC* running Linux but with no active processes (idle), the power consumption ranges from 428 mW to 478 mW, depending on the processor speed. Second, the same measurement is realized but now with *TELOSB* attached. No data is sent from *CITRIC* to *TELOSB*, although the latter is running an application that waits to receive any packets from the former and transmits them over the radio. The power consumption varies from 527 mW to 594 mW. Finally, the power consumption running a typical background subtraction function is also measured. The test utilizes all the components of the system and processes images with a resolution of 512×512 pixels. The image coordinates of the foreground are transmitted. For a processor speed of 520 MHz, the resulting power consumption is 970 mW. In this case, the expected lifetime of the node would be 16 hours under continuous operation.

Three vision applications are presented: image compression, target tracking and camera localization. In all the cases, the processor speed is set to 520 MHz. Image compression consists of JPEG compression of 512×512 pixels test images. The computation time required is 70 ms per image, that is, a maximum frame rate of 14.3 fps can be reached. The target tracking application is based on background subtraction via frame differencing. Each new frame is compared to a background image model, classifying the pixels with significant variation as part of the foreground. The foreground pixels are then processed for identification and tracking. Now, the execution time per frame is typically 0.2–0.4 s at a resolution of 320×240 pixels and 0.3–0.8 s at 640×480 pixels. The frame rate is not fixed due to the variable execution time of the algorithm depending on the number of foreground pixels. Finally, camera localization is the process of finding the position of the cameras as well as the orientation of each camera's field of view. In this experiment, two vision-enabled *CITRIC* nodes are positioned 2.5 m apart focusing an open area with people walking. Each camera mote runs background subtraction on its current image and then sends the bounding box coordinates back to a base station for each detected foreground object. The center of each bounding box is used to build object tracks over time on the base station computer. By using these tracks, the position and orientation of the cameras can be estimated.

As a summary, the *CITRIC* platform presents the incorporation of a microphone as a novelty, meaning the first attempt to integrate multimedia information—visual and audio processing—within the framework of WSNs. Another remarkable feature is the resolution of the CMOS image sensor, which is the highest one among the vision-enabled WSN nodes reported so far in the literature.

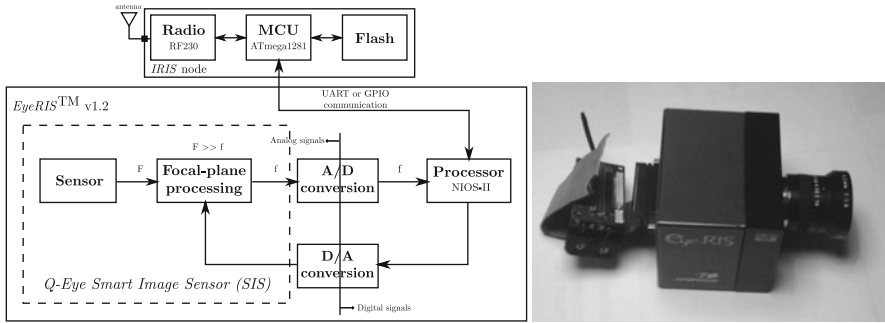


Fig. 2.9 Block diagram of *EyeRIS™*-based WSN node and snapshot of the platform. (With permission of the authors)

2.2.7 *EyeRIS™*-based Platform (Bakkali et al. 2010)

This platform takes further than *WiCa*—see Sect. 2.2.4—the adaptation of the architecture to the characteristics of visual processing. The vision capabilities of this WSN node are implemented by means of *EyeRIS™* v1.2 (Rodríguez-Vázquez et al. 2008), a general-purpose programmable autonomous vision system. This system employs an architecture in which image processing is carried out by two hierarchical stages. In a first stage, low-level tasks are performed by analog circuitry at the very focal plane. Thus, concurrently with the photosensing, interconnected elementary analog building blocks process the image just captured in a massively parallel way. Local memories are incorporated to every block in order to avoid large consumption overheads due to repeated memory accesses. The result is an array of 176×144 sensing-processing cells, called *Q-Eye*, which realizes very efficiently different kind of early vision tasks over the image. The accuracy of these operations is moderate (6-7 bits), but enough for the subsequent digital processor, which constitutes the second processing stage. This 32-bit RISC digital processor (NIOSt-II, Altera) performs higher abstraction tasks by making use of the pre-processed images coming from the sensor-processor. Complex algorithms with long and irregular computational flows leading to decision making are typical for this second stage. As a whole, the processing architecture implemented by *EyeRIS™* emulates the organization of the retina (Masland 2001). In natural vision systems, the visual information is not only acquired but also preprocessed in the focal-plane device, the retina, before being sent to the visual cortex through the optic nerve for further understanding. Interestingly, this pre-processing is performed in the analog domain by means of retinal cells organized in layers and interacting locally (Roska and Werblin 2001).

Regarding the wireless communication, a commercial node (IRIS, MEMSIC) is utilized. This platform includes an MCU (ATmega1281, ATMEL) and an IEEE 802.15.4-compliant RF transceiver (CC2420, Texas Instr.). The *EyeRIS™*-to-*IRIS* interconnection is carried out through either UART or GPIO pins available from both systems. A sketch of the resulting WSN node along with a snapshot, reproduced here with permission of the authors, are depicted in Fig. 2.9.

Three examples of possible applications are presented. The first one consists of broadcasting a motion-triggered alarm signal. A basic algorithm detects changes in an image sequence. When a change is detected, an alarm signal is transmitted to the *IRIS* through a GPIO pin in order to be broadcasted. The second example shows some of the possibilities of the *EyeRIS*TM system. The coordinates of a targeted ROI are extracted and delivered through UART to the *IRIS*, which broadcasts them. Finally, the result of an in-situ image flow analysis is also broadcasted. Specifically, a code indicating the sense of the major motion component within the sequence is delivered to the *IRIS* to be transmitted.

Summarizing, this vision-enabled WSN node achieves a very high degree of energy efficiency by implementing an architecture fully adapted to the features of visual processing. This adaptation does not only consist, like in *Xetal-II* processor described in Sect. 2.2.4, of taking advantage of the potential parallel processing of early vision tasks, which are now performed at the focal plane. In addition, *EyeRIS*TM system benefits from the moderate accuracy (Poynton 2007) required for such tasks. Thus, by making use of analog building blocks, not too accurate but very fast and energy-efficient, this platform exhibits a computational power of 250GOPS with a power consumption of 4 mW/GOPS. This figure improves the 5.6 mW/GOPS of *Xetal-II*. Nevertheless, despite the computational power and energy efficiency of *EyeRIS*TM, its power consumption, 1.5 W, is still too high to achieve long node lifetime in real deployments of WSNs.

2.3 Comparative Analysis

Some of the features of the vision-enabled WSN nodes summarized in the previous section have been gathered in Table 2.1. Its main objective is to have access to key aspects of the different platforms with a quick look. It also allows for a fast comparison of interesting data such as camera resolution or type of processor. Nevertheless, it is important to remark at this point that a fair comparison concerning the most important issue in a vision-enabled WSN node, that is, power consumption, is not possible. It is due to the heterogeneity of the application framework over which the different systems have been tested. As an example, consider the last two columns in Table 2.1. We have put together here a very general description of one of the applications reported for every platform along with the power consumption measured for that application, the resolution of the images processed and the frame rate reached. It can be seen that there is not a single match, even though we have chosen the most similar set of applications. This absence of a common application testbench for vision-enabled WSN nodes prevents us from establishing an adequate comparison of the energy efficiency of the different approaches. Such a testbench should define a prescribed vision algorithm to be run under prescribed light conditions. It should also define the size of the images to be processed as well as the frame rate. Under these circumstances, each platform could be adjusted to fairly compete with the rest for delivering the targeted result with minimum power consumption. As an attempt to provide a FOM for a comparison of this set of applications, the last column also

Table 2.1 Comparison of vision-enabled WSN platforms

Platform/year	Imager/ resolution (px)	Processor/ architecture	Transceiver/ Freq. (GHz) @ Bitrate (kbps)	Application	Power consumption (mW)/ Resolution @ Frame rate (fps)/ Energy per pixel (μ J/px)
<i>Cyclops</i> 2005	ADCM-1700 352 \times 288	ATmega128L 8-bit RISC	CC1000 in <i>MICA2</i> 0.9 @ 76.8	Object detection via background subtraction	211.8 128 \times 128 @ 2.6 4.97
<i>Imote2</i> -based 2006	OV7649 640 \times 480	PXA271 32-bit ARM5	CC2420 in <i>Imote2</i> 2.4 @ 250	Pattern recognition via motion detection	322 80 \times 60 @ 8 8.38
<i>MeshEye</i> TM 2007	ADNS-3060 (2) 30 \times 30 ADCM-2700 640 \times 480	AT91SAM7S 32-bit ARM7	built-in CC2420 2.4 @ 250	Object detection via stereo matching	155 64 \times 64 @ 0.5 75.68
<i>WiCa</i> 2007	OM6802 (2) 640 \times 480	<i>Xetal-II</i> +8051 SIMD PE array + 8-bit CISC	CC2420 in <i>AquisGrain</i> 2.4 @ 250	Pattern recognition via background subtraction	500 320 \times 120 @ 30 0.43
<i>FireFly Mosaic</i> 2007	OV6620 352 \times 288	LPC2106 32-bit ARM7	CC2420 in <i>FireFly</i> 2.4 @ 250	Region clustering via background subtraction	114.5 352 \times 288 @ 5.2 0.27
<i>CITRIC</i> 2008	OV9655 1280 \times 1024	PXA270 32-bit ARM5	CC2420 in <i>TELOS B</i> 2.4 @ 250	Target tracking via background subtraction	970 320 \times 240 @ 3.3 3.83
<i>EyeRIS</i> TM -based 2010	<i>Q-Eye</i> 176 \times 144	Q-Eye+NIOS-II SIMD PE array + 32-bit RISC	RF230 in <i>IRIS</i> 2.4 @ 250	Object detection via motion detection	1500 176 \times 144 @ 15 3.94

includes the energy per pixel required by each platform to complete the corresponding algorithm. But we must insist about the weakness of such a comparison.

Another noteworthy detail clearly highlighted by Table 2.1 is that applications tend to process images with very low resolution, specially when compared to the maximum possible resolution of the associated imager. By working with low resolution images, the power consumption can be significantly reduced for a prescribed frame rate. Or, from the opposite point of view, for a prescribed power consumption, a higher frame rate can be obtained. The fact is that currently the integration of a high-resolution imager could be counterproductive because of the extra power consumption associated to the capture of the entire pixel array.

Finally, notice that all platforms but *Cyclops* and *WiCa* make use of 32-bit processors. Apparently, 8-bit processors are lighter and therefore consume less power. However, 32-bit processors are better suited for image processing than their 8-bit counterparts. In (Downes et al. 2006), an interesting experiment is carried out. The time needed to perform operations such as 2-D convolution on an 8-bit processor clocked at 4 MHz is 16 times higher than with a 32-bit ARM7 device clocked at 48 MHz, while the power consumption of the 32-bit processor is only six times higher. Hence, an 8-bit processor turns out to be slower and less energy-efficient. In fact, *Cyclops* needs a CPLD to be able to reach acceptable frame rates for the applications considered. On the contrary, *WiCa* can afford a less powerful processor due to its visual co-processor performing the most computationally heavy tasks.

2.4 Guidelines for a Power-aware Vision-enabled Wireless Node

WSNs can be considered as an essential building block for the paradigm of *ubiquitous computing*. Its field of application is currently very extensive, but it can be significantly boosted by the incorporation of multimedia sensing within the catalog of sensing capabilities of the nodes. Specifically, the integration of vision has become a burning issue within the research community. In this chapter, a survey of different approaches reported to carry out this integration has been realized. Most of them implement conventional processing architectures. Images are captured by a sensor and immediately converted into the digital domain. The resulting serialized digital flow is usually handled by a single processor, which performs all the processing required by the corresponding algorithm. However, the special characteristics of this digital flow—massive and multidimensional—as well as of the low-level tasks commonly applied during early stages of image processing—regular and potentially parallel—have led to more efficient alternative approaches. The *WiCa* platform represents the first attempt to address such special characteristics differently by making use of a high-performance parallel digital co-processor. This co-processor, dedicated only to vision, takes advantage of the regularity and parallelism of early image processing to reduce the number of memory accesses, clock speed and instruction decoding. The *EyeRIS*TM-based platform goes further in the adaptation of the architecture to image processing. In this case, low-level tasks are carried out in a massively parallel way at the very focal plane. The imager does not already consist only of photosensing

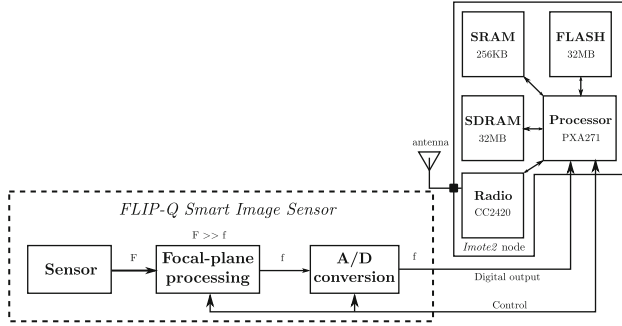


Fig. 2.10 Basic sketch of the vision-enabled WSN node proposed

devices and some additional readout circuitry. Now, it becomes a really smart camera where distributed PEs influence the image capture in order to deliver a simplified representation of the scene. These PEs, which work very efficiently in analog mode, reach equivalent resolutions of about 6-7 bits. It is enough for the subsequent digital processor, which now receives pre-processed images over which apply medium- and high-level processing. As a whole, the *EyeRIS*TM-based platform exhibits a better performance than *WiCa*. The main drawback of the *WiCa* and *EyeRIS*TM-based platforms is their high power consumption. Despite their computational power and energy efficiency, their design is not specifically oriented to WSNs. On the contrary, their scope of application is very wide thanks to the great deal of functionalities and prominent features available. But these functionalities and prominent features also mean that the power consumption of the system is higher. Indeed, too high for the real requirements of vision-enabled WSN applications.

All in all, our proposal is based on a processing architecture fully adapted to the special characteristics of WSNs. Thus, we address the VLSI implementation of a prototype vision chip delivering a set of focal-plane processing primitives which has been selected with the requirements of WSN nodes in mind. Minimum power consumption and image simplification represent the driving forces for the design of the chip. As a result, the prototype outputs a digital data flow containing reduced representations of the scene obtained at ultra low energy cost. This energy efficiency together with the programmability of the focal-plane processing have enabled the direct integration of the chip with *Imote2*, the commercial node already mentioned at different points of this chapter. It is not necessary now the separation of the computational resources assigned for image processing from those assigned for communication. Under normal circumstances, the PXA271 processor of *Imote2* will be able to deal with both image processing and asynchronous requirements of the network as its computational load has been greatly alleviated by the WSN-oriented smart vision chip. A basic sketch of the resulting system is depicted in Fig. 2.10. As a first step for its comprehensive description, we next analyze and justify the set of focal-plane processing primitives chosen to be implemented in the prototype while developing the mathematical framework which supports them.

<http://www.springer.com/978-1-4614-2391-1>

Low-Power Smart Imagers for Vision-Enabled Sensor
Networks

Fernández-Berni, J.; Carmona-Galán, R.;

Rodríguez-Vázquez, A.

2012, XXIV, 156 p., Hardcover

ISBN: 978-1-4614-2391-1