Development of sensor drivers on Tyndall Wireless Platform

Wassim MAGNIN

Project presentation - 1st July 2009

Supervisor: **Dr. Essa Jafer**

Master 2 Electronique des Systèmes Communicants

www.tyndall.ie

Table of Contents

Residential building energy monitoring systems are based on non-residential buildings
=> unsophisticated, high cost

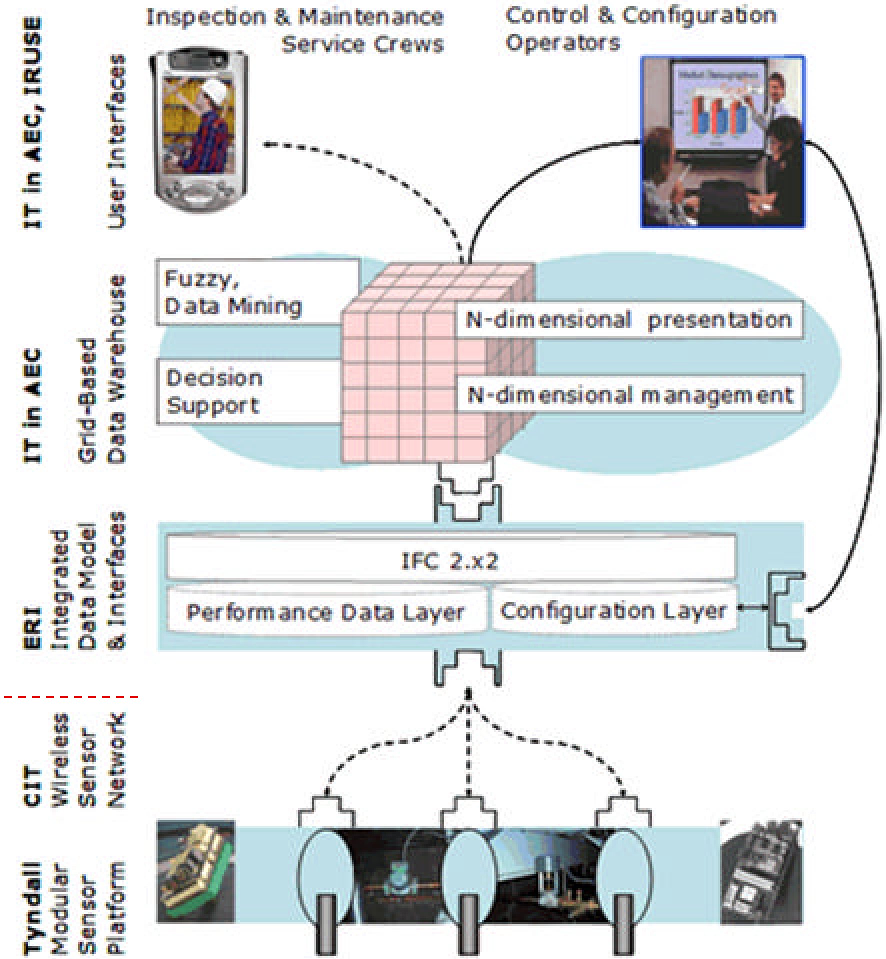
Goal: specify, design and validate a data management technology platform
=> integrated environmental energy management in buildings

Combination of holistic environmental and energy management scenarios

- integrated building information model
- data mining methods and technologies
- wireless sensor network technologies

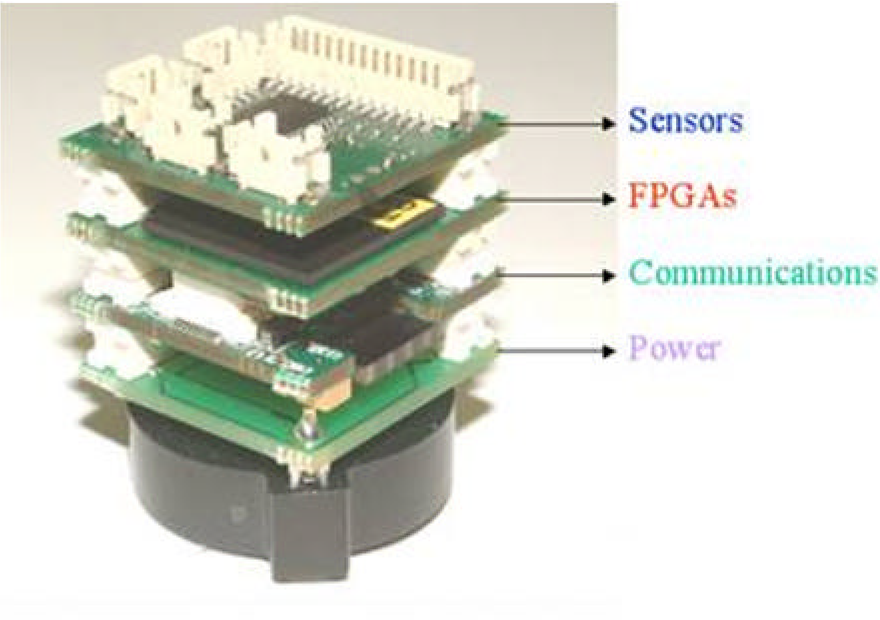
- ***Project Partners***
- **UCC Environmental Research Institute**
  IRUSE: Informatics Research Unit for Sustainable Engineering
- **UCC Department of Civil and Environmental Engineering**
  IT in AEC: Chair of Information Technology in AEC
- **Cork Institute of Technology**
  Centre for Adaptive Wireless Systems

Tyndall Platform development and implementation



II. Tyndall Wireless Platform

- 25 25 mm
- Microcontroller ATMEL Atmega 128L
  - 128K Bytes of In-System reprogrammable Flash
  - 4K Bytes EEPROM
  - 4K Bytes Internal SRAM
- Chipcon CC2420 2.4GHz transceiver
- Multi-sensors layer
- Power monitoring layer
- Coin cell battery



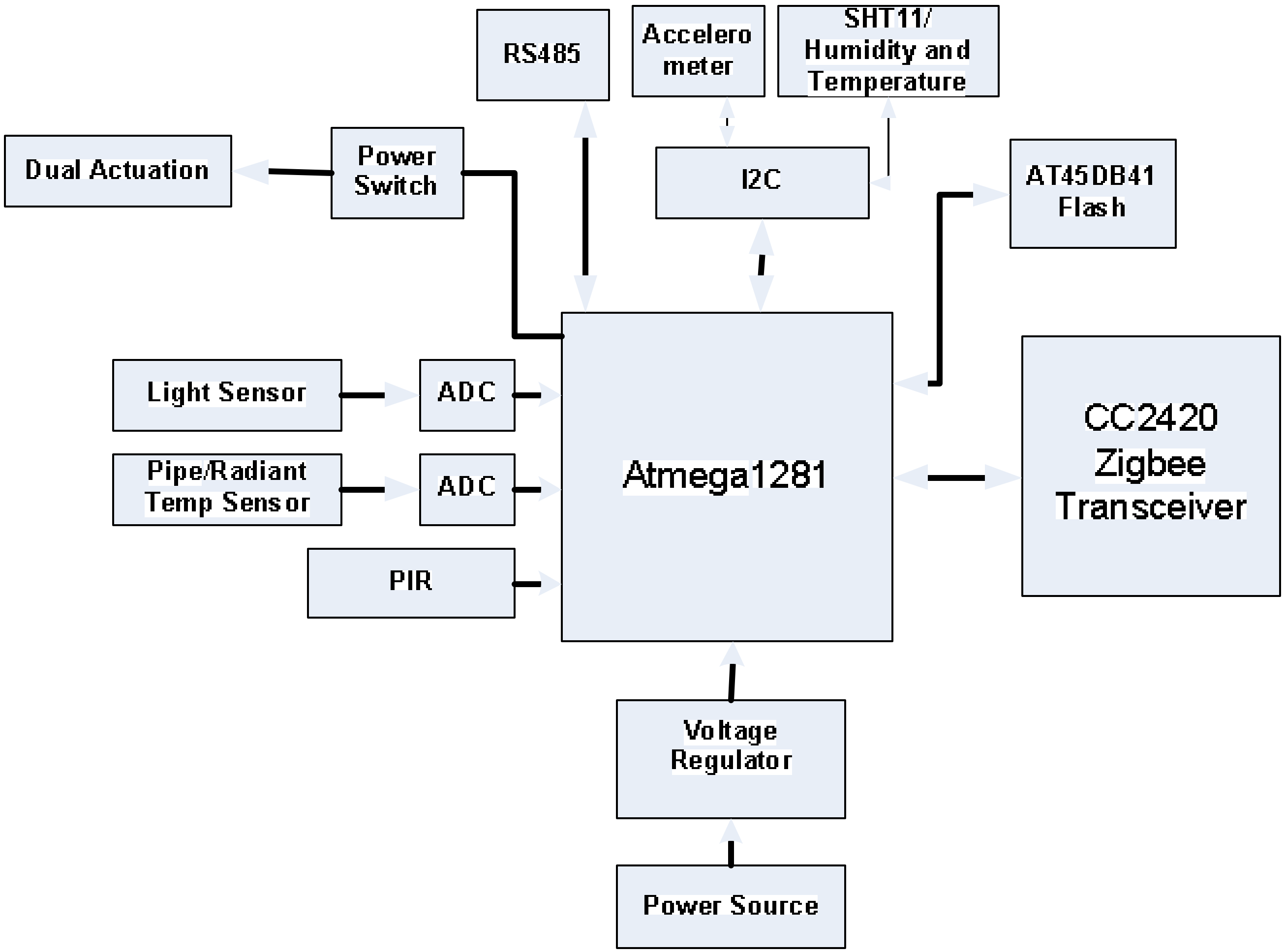


Figure 1: Tyndall mote architecture

➢ III.1 TinyOS

- *originally developed as a research project in Berkeley*
- *open-source operating system designed for wireless embedded sensor networks*
- *component-based architecture* **=> minimize code size**
- *event-driven execution*
- *designed for limited resources (8KBytes of program memory,512 bytes of RAM)*

NesC

III.2 NesC

- *Programming language structured for TinyOS concepts*
- *Java, C,C++ mix*
- *Separation of construction and composition (component concept)*
- *Thread of control passed through interfaces*

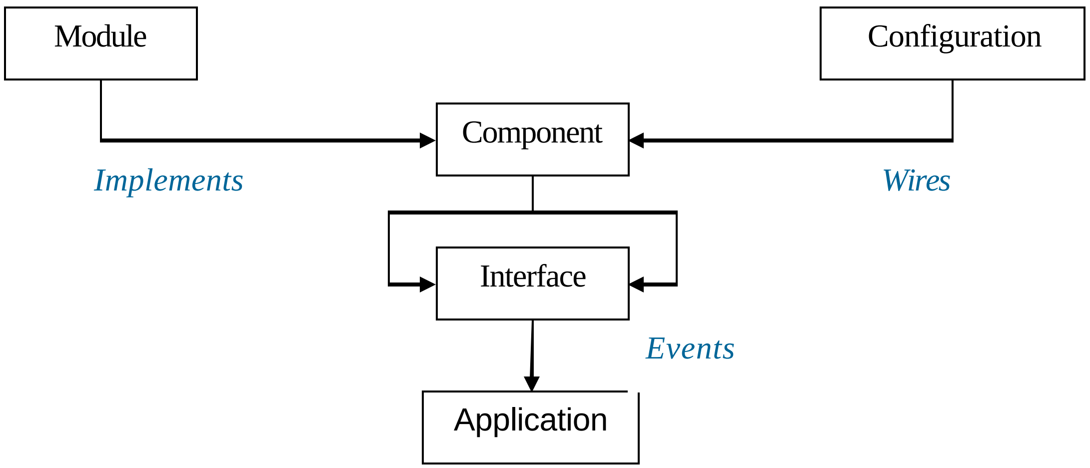


Figure 2: NesC architecture

- **Interface**

  Defines a set of functions and events

  - **Interface provider**

    Implements functions and signals events

  - **Interface user**

    use functions and received events interrupts

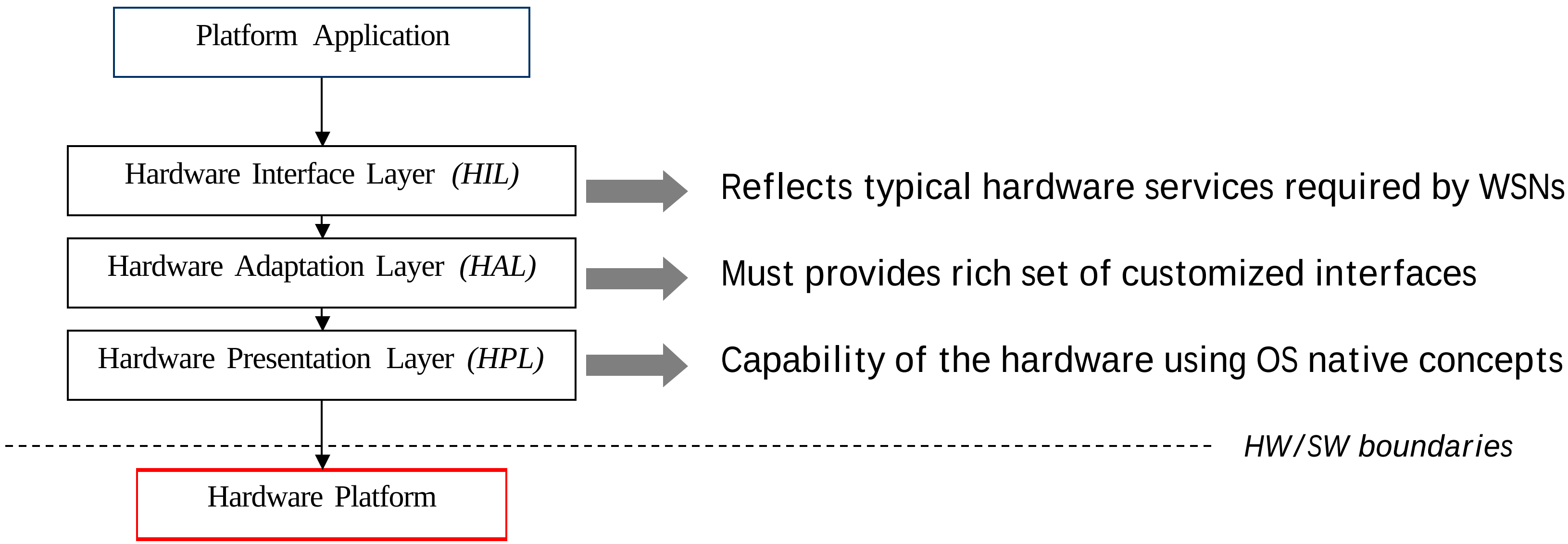
Component 1		Component 2			Component 3
Provides	readByte writeByte readByteDone	Uses	Provides	WriteWord ReadWord readWordDone	Uses

+ Hardware independence

Figure 3: NesC: Component concept

III.3 TinyOS hardware abstraction

- ***3 distinct layers of components***
- ***Each layer dependant of interfaces provided by lower components***



- **Xubuntu OS**
- **Linux-like environment**
- **TinyOS 2.1**
- **Virtual Machine**

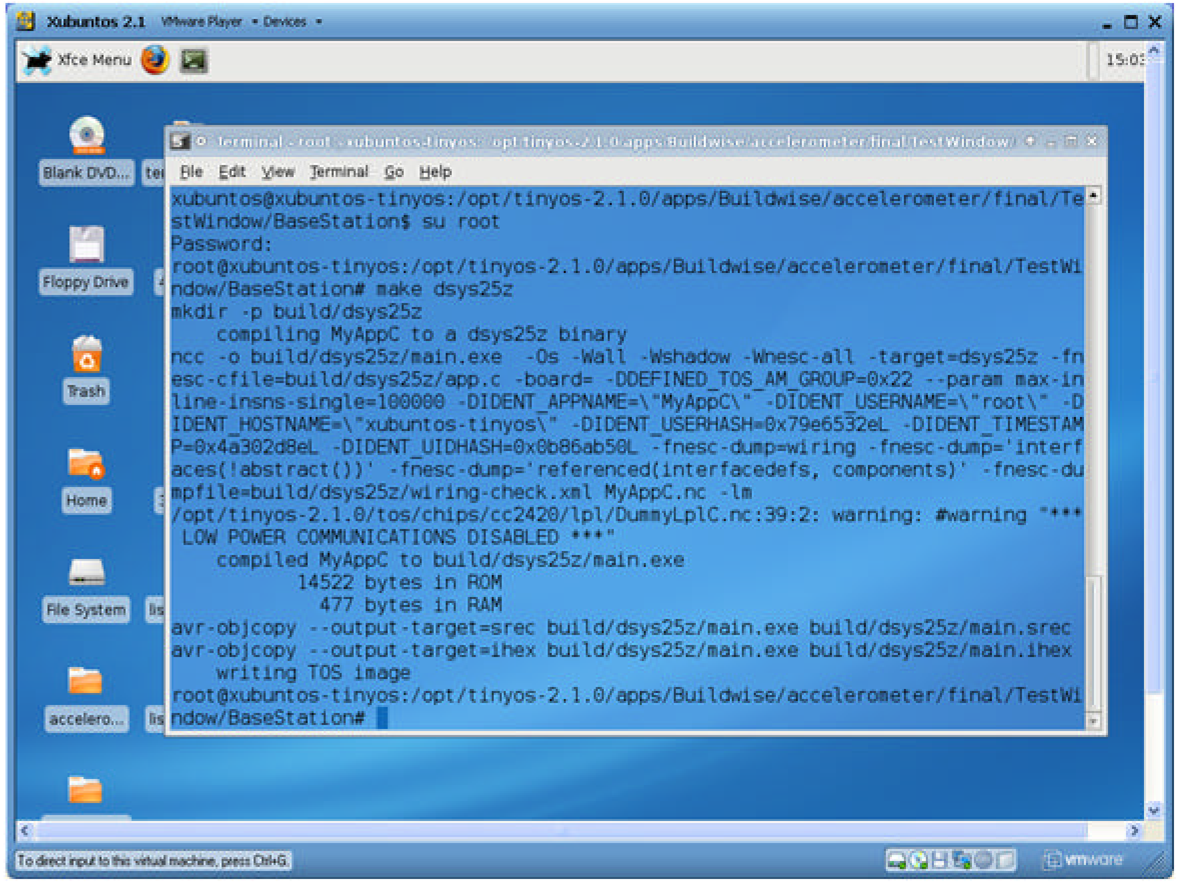
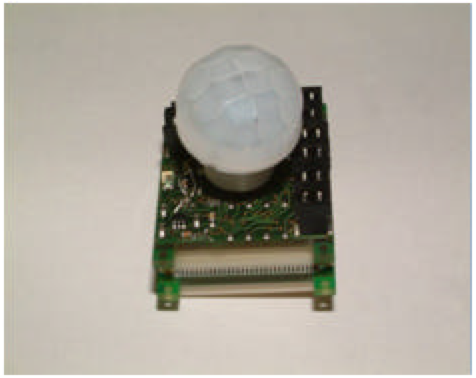


Figure 4: XubunTOS interface

- **Tyndall sensor board**
  - **Onboard light sensor**
  - **Onboard Temperature and Humidity sensor**
  - **Onboard PIR *(Occupency detector)***
  - **Miscellaneous purposes RS485 interface**
  - **Onboard 3-axis accelerometer**
  - **Interface for water pipe and radian temperature sensor *(external sensor)***



V.I Light sensor

- Example of driver using Atmega ADC
- Use AdcReadClientC component =>
- Concepts of generic component and resources

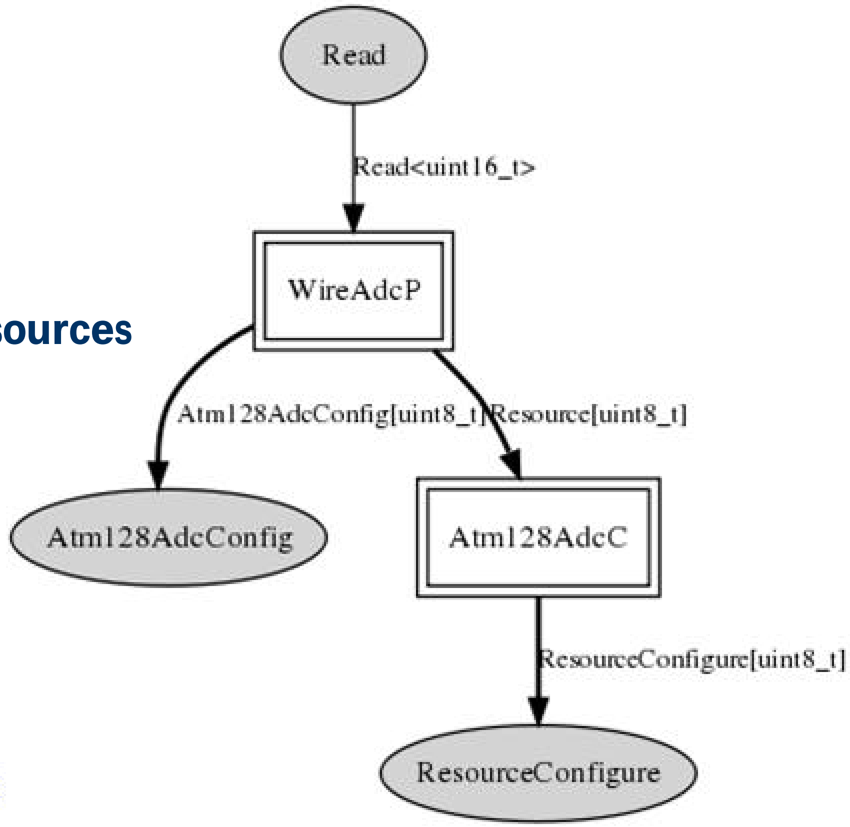


Figure 6: ADCReadClientC wiring

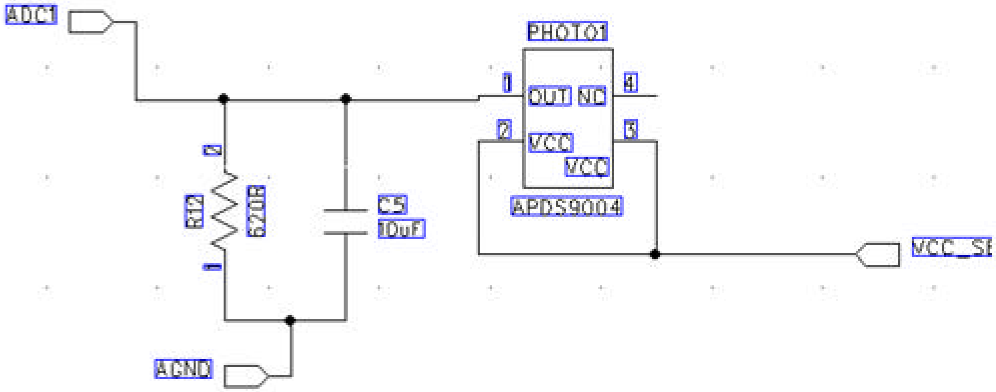


Figure 5: Light sensor schematics

- **V.2 I²C protocol (Temp/Hum sensor and 3-Axis accelerometer)**

devices connected to the ATmega128L 2 wires interface (TWI)

SCK: clock

SDA: data

driver follows I²C protocol with specific features depending on the hardware

- **start sequence**
- **reset sequence needed…**

Master	ST	SAD+W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

Figure 7: Write one byte using I²C protocol

➢ V.2 I²C protocol (Temp/Hum sensor and 3-Axis accelerometer)

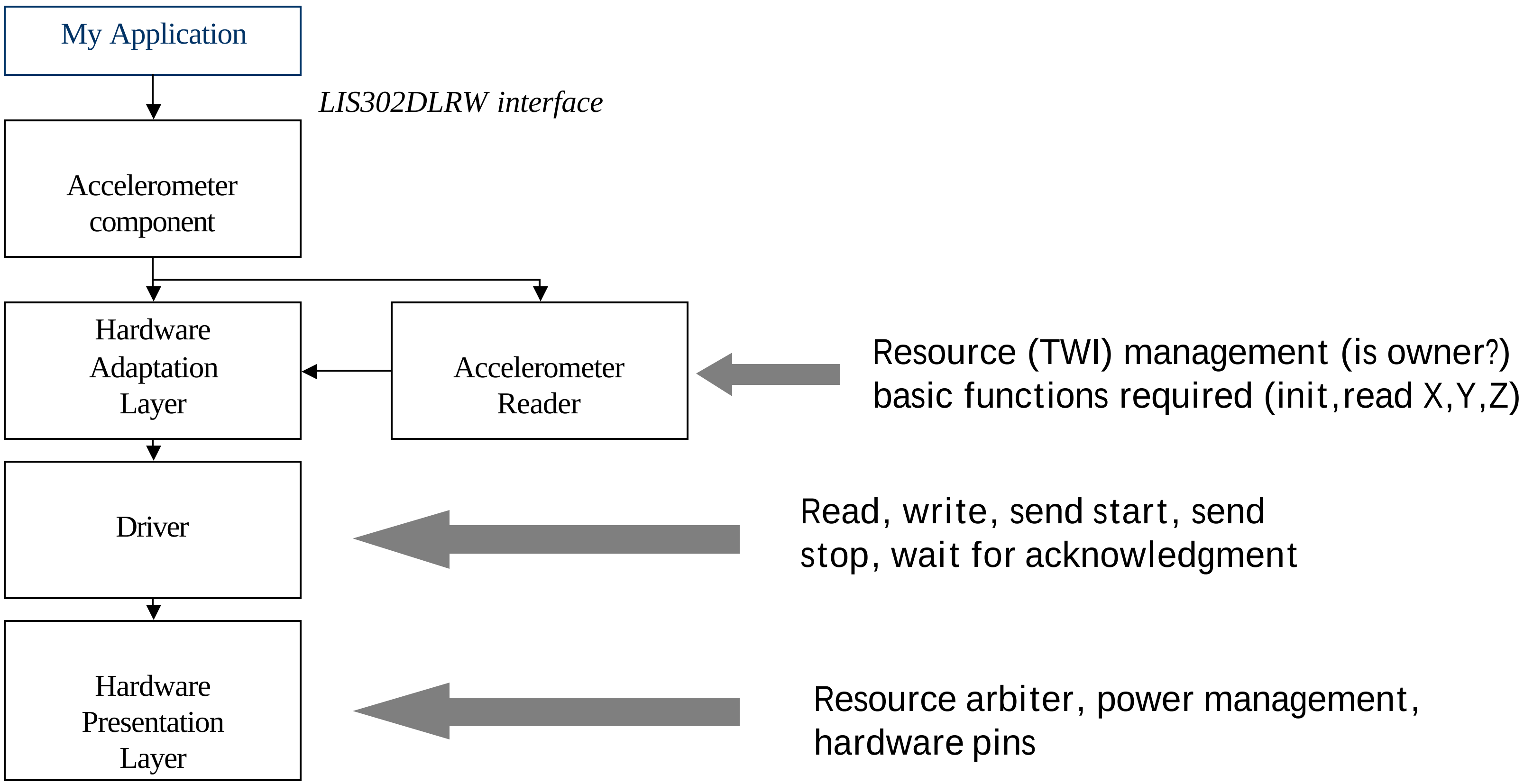


Figure 8: 3-Axis Accelerometer components stack

V.3 Modbus protocol

- External Water flow meter
- RS485 interface (EIA-485)
- Modbus ASCII mode:

Start	Address	Function	Data	LRC	End
1 char	2 chars	2 chars	0 up to 2 252 char(s)	2 chars	2 chars CR, LF

- **Frame to send:** ***Address: 0x01*** ***Function: 0x03*** ***Data: 0x05***
- **Modbus Frame:** **3A 30 31 30 33 30 35 D7 46 37 0A**

F7

V.3 Modbus protocol

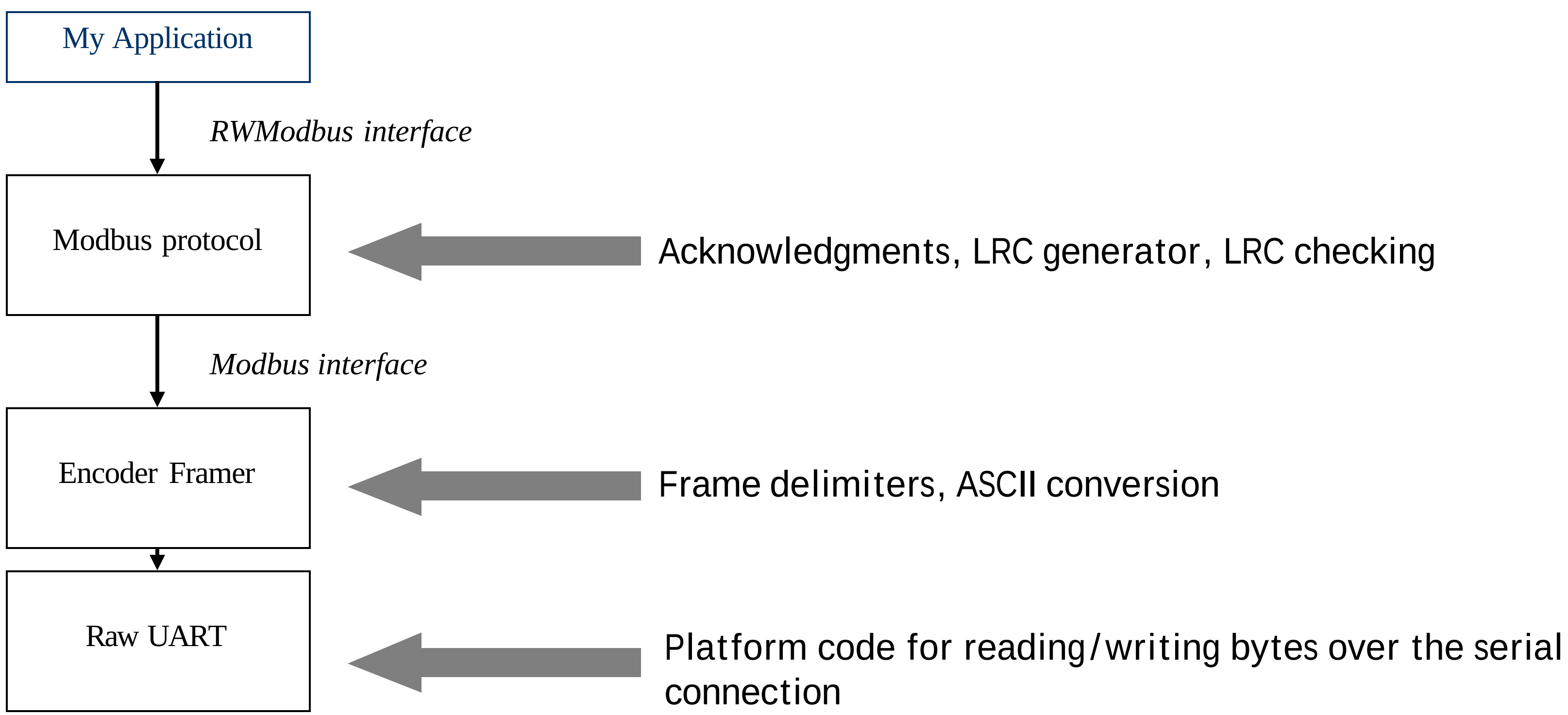


Figure 9: Modbus components stack

- **Using Labview to display data**

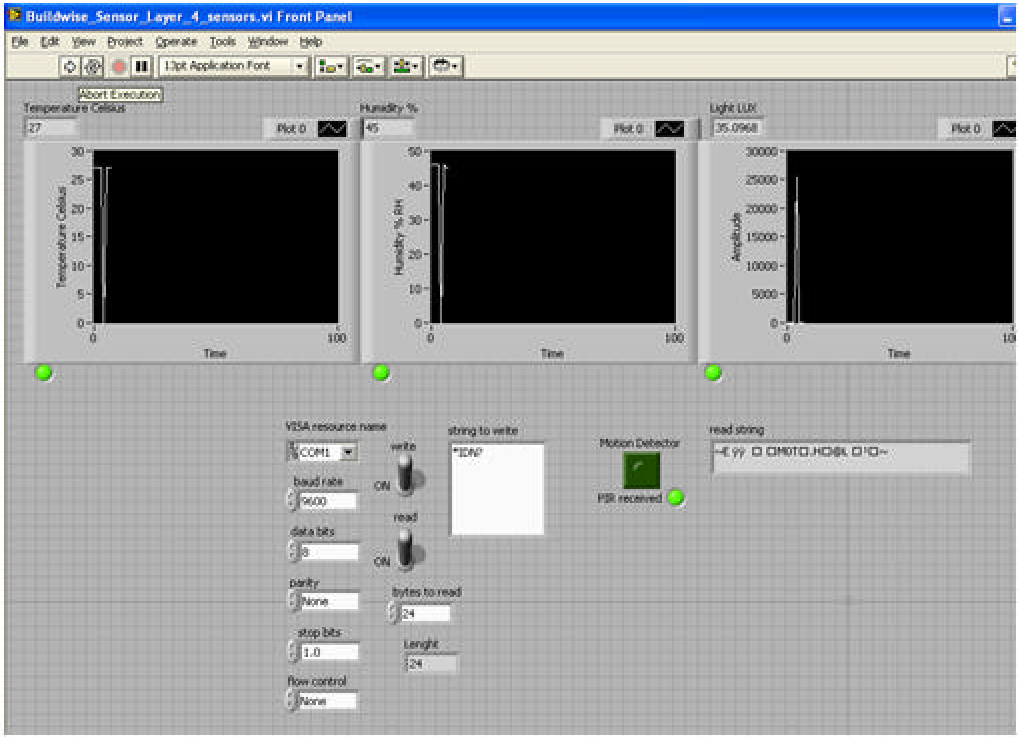


Figure 10: Temp/Hum/Light/Motion sensors

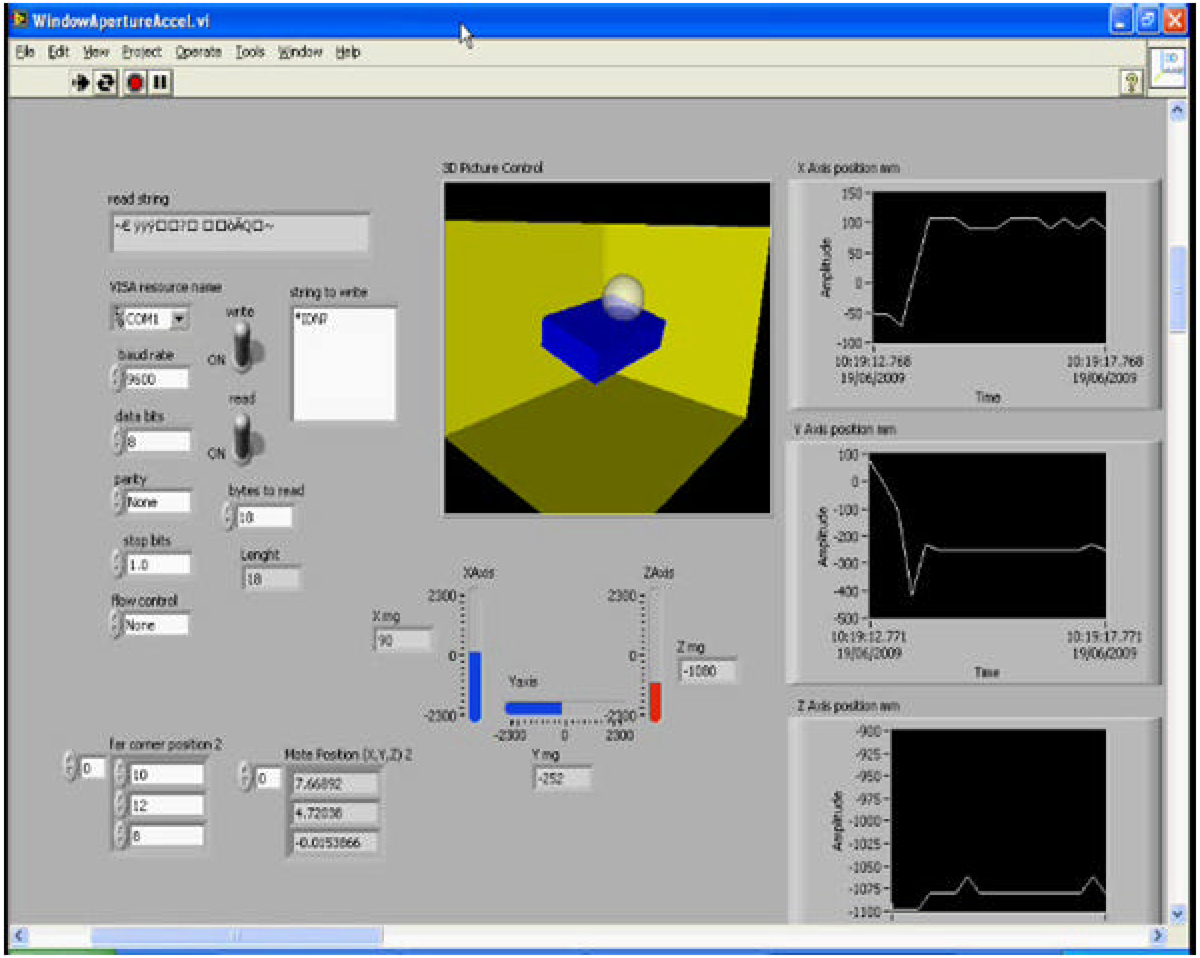
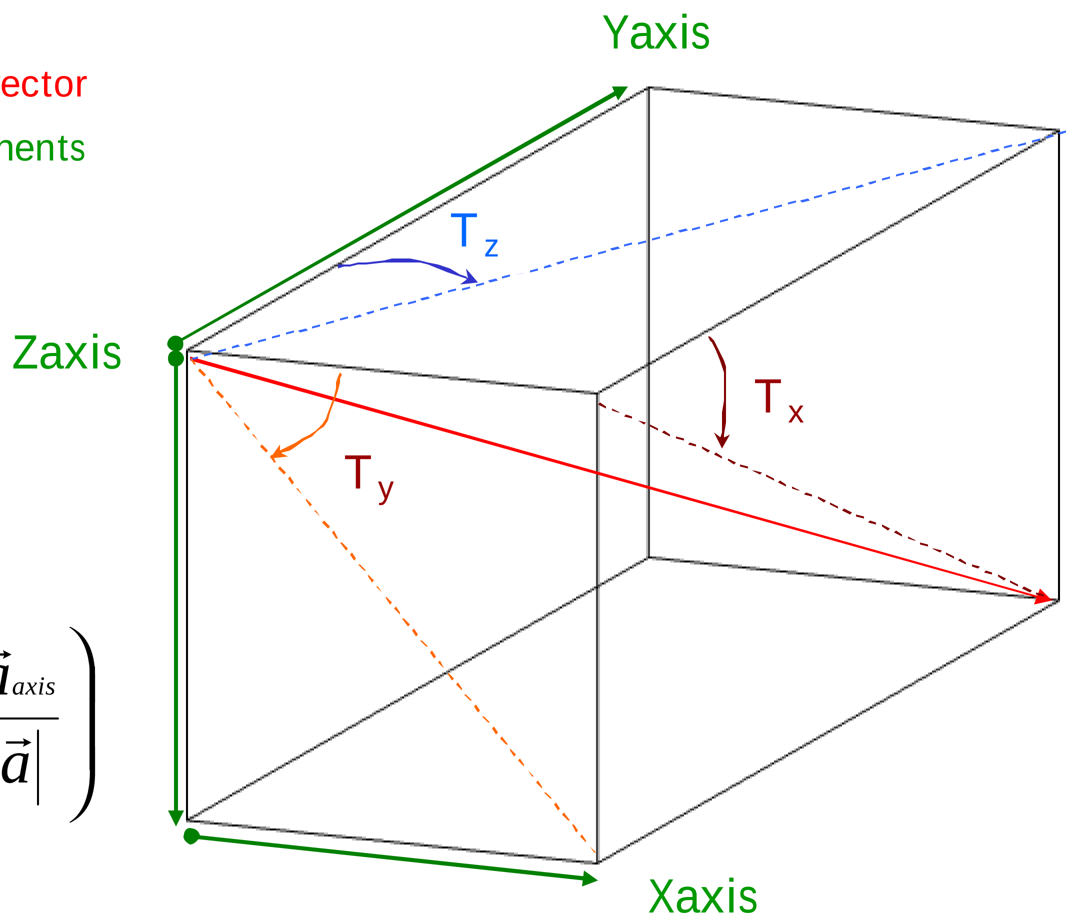


Figure 11: Accelerometer GUI

Figure 12: Accelerometer demonstration

Acceleration vectors computation:

- Acceleration vector
- Vector components



$$\Theta = \arcsin\left(\frac{\vec{a}_{axis}}{|\vec{a}|}\right)$$

- **NAP 217 purpose**
  - **Create a java mote using SQAWK Java Virtual Machine (JVM)**
  - **Based on the SUN Microsystems eSPOT architecture**
  - **Provides an high level programming interface for Java developers**
  - **Power management layer**
  - **CLDC compliant (J2ME)**
  - **PADS design (PCB design software)**

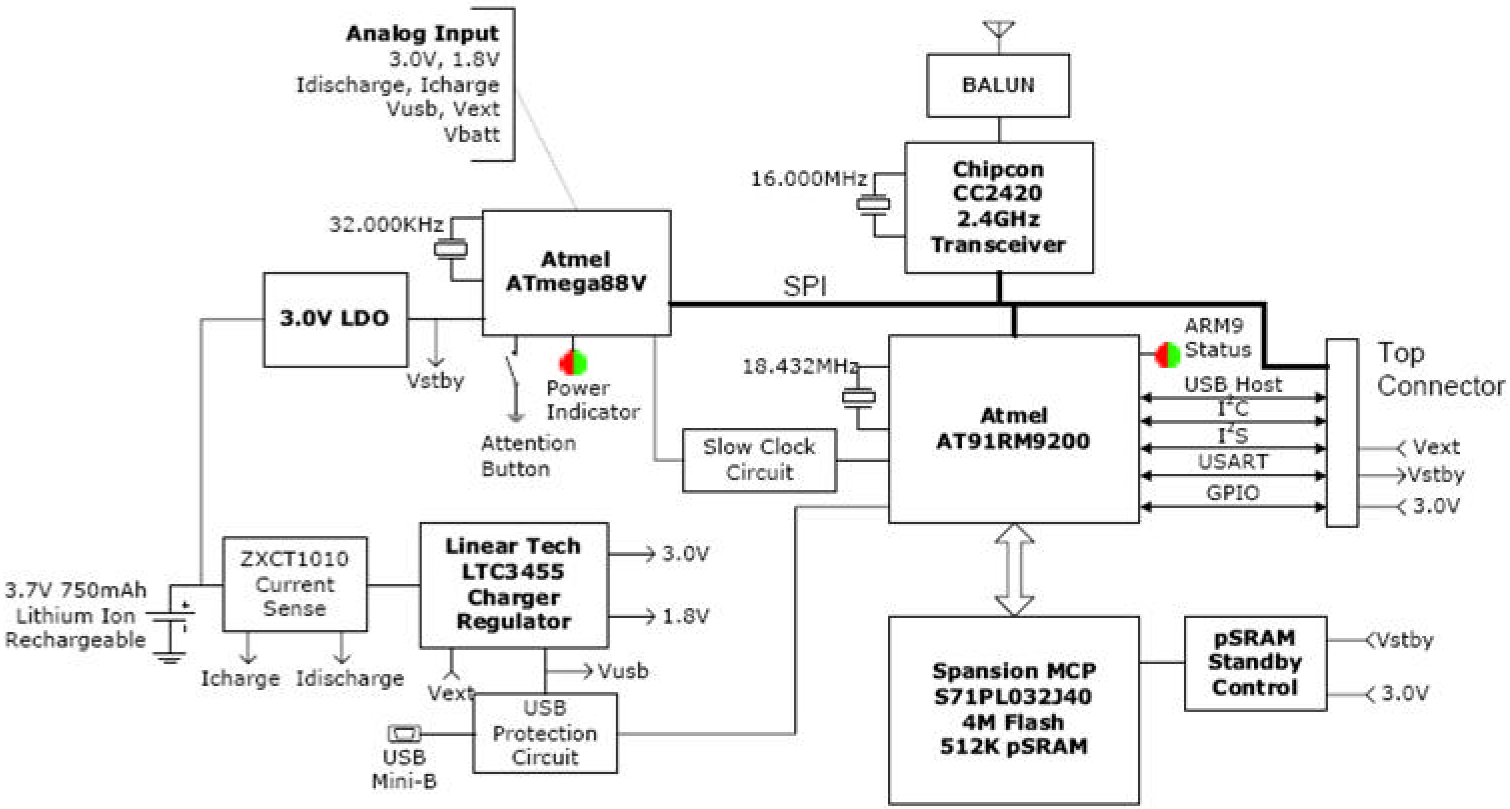


Figure 13: eSPOT architecture

- **Java SQUAWK layer integration**

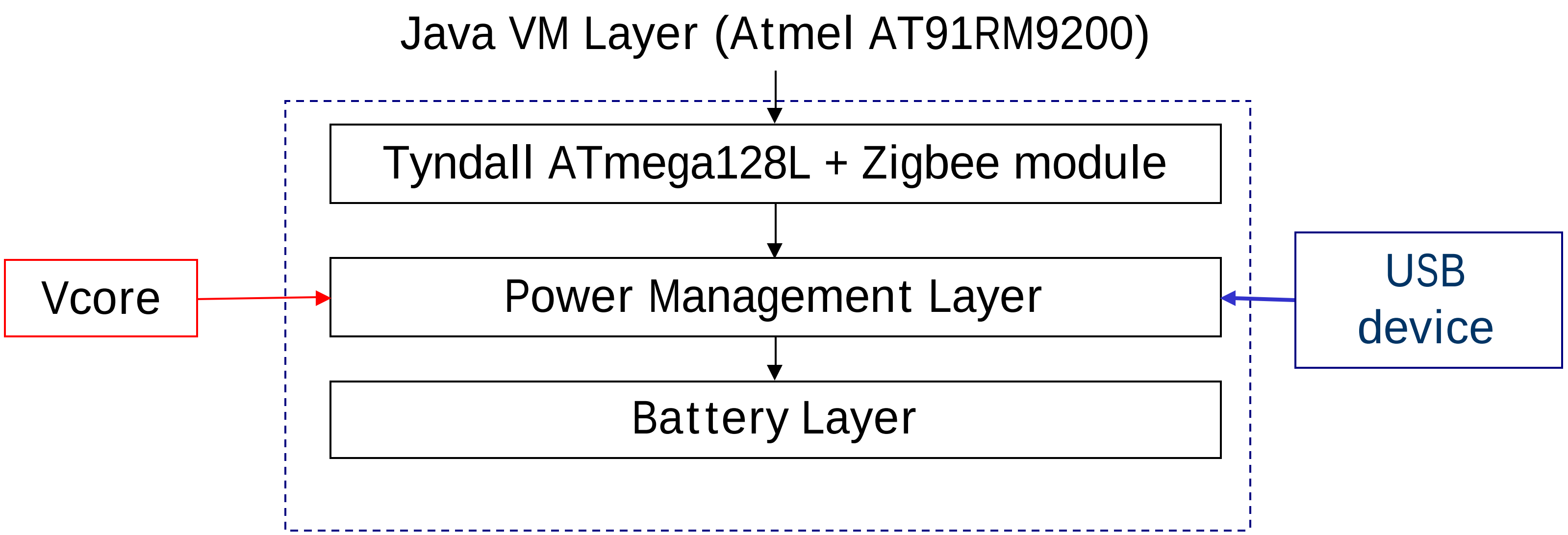


Figure 14: Tyndall SQUAWK mote architecture

- Get experience with TinyOS and embedded systems
- Deal with hardware and software issues
- Interesting background
- Labview programming
- Work with the latest technologies for communication
- Partnership with other companies research centres
- Experience abroad

IX. Sources

- **Tinyos website:** http://www.tinyos.net/
- **Avrdude compiler:**http://www.nongnu.org/avrdude/user-manual/avrdude_4.html
- **Modbus protocol:** http://www.modbustools.com/modbus.asp
- **I2C protocol:** http://en.wikipedia.org/wiki/I%C2%B2C
- **XubunTOS:** http://toilers.mines.edu/Public/XubunTOS
- **Wikipedia:**www.wikipedia.org

QUESTIONS?

Tyndall
National Institute
Institiúid Náisiúnta

www.tyndall.ie