

Module NXT

LEGO MINDSTORMS TUTORIAL

LEGO MINDSTORMS
TUTORIAL

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

Overview LEGO©MINDSTORMS©NXT from a community and user standpoint

Hardware Description of NXT hardware from a developer perspective

Software Description of NXT software with the aim of performing a firmware replacement

Note 1: LEGO has sponsored a NXT 2.0 kit, which we will make a draw for at the end of the tutorial.

Note 2: LEGO® , MINDSTORMS® are trademarks of LEGO® . The tutorial contains pictures from Atmel ARM7 documentation, and from LEGO documentation.

Note 3: There is additional material included: Many slides are supplementary and included for future reference.



Module NXT

LEGO MINDSTORMS

LEGO MINDSTORMS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

Outline

LEGO MINDSTORMS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- Includes as USB, SPI, and I2C
- How the open source universe around Lego Mindstorms NXT is structured
- Talk about a world that stretches from high-schools to adults and from building bricks to programming in assembler
- Emphasis on the overview and the programming aspects
- Readers will be able to see where in the NXT programming spectrum their interest match most
- Use the tutorial to get started with open source programming on NXT





- Description of open source hardware and software
- LEGO business models are discussed
- The old RCX
- NXT in several (online) communities
- Overview of how NXT can be programmed in different environments

Source and Documents by LEGO

NXT Firmware Open Source Complete software for the NXT firmware, which can be compiled with a compliant C compiler

Software Developer Kit A kit with software for controlling NXT remotely from a host computer. It includes a description of the NXT virtual machine.

Hardware Developer Kit Description of the possible sensor and actuator interfaces for NXT. This is extensively used by thirdparty sensor vendors.

Bluetooth Developer Kit It includes the communication protocol for controlling NXT over a Bluetooth connection.





- Lego published the schematics
- Can see which pins of the ARM7 MCU is connected to what ports
- Lego uses Orcad for its schematics, which is a commercial program



- NXT features both analogue and digital interfaces to its sensors.
- Sensors are based on an analogue to digital conversion
- Signals are fed into the ARM processor for further processing
- Digital sensors are made with an I2C compliant protocol
- I2C is a two wire protocol that allows for up to 127 devices on the shared bus



- LEGO provides a proprietary programming language for LEGO MINDSTORMS NXT called NXT-G
- **G** implies it is a graphical programming language
- Places NXT-G programming blocks in a sequence that involves the usual programming constructs such as loops, branches and computations
- Tutorials and small videos with instructions for a given task

Show NXT-G.

- LEGO MINDSTORMS NXT comes with an open source operating system
- Operating system is written in ANSI C
- Source code fundamentally access the physical layer such as input and output ports of the ARM processor
- Another part provides an abstraction layer to these services
- Operating system is a traditional round robin scheduler
- Services each a number of modules





- Each module is centered on a logical or a physical unit in NXT
- Example: the display or the user interface
- The virtual machine a logical module
- Each time the virtual machine is serviced it will execute a number of bytecodes
- A 1 ms period (system tick) services all modules



- LEGO has released an interface both for the PC and the Macintosh
- Linux is not supported at this point
- Firmware image in NXT is updated via a USB cable connection
- NXT is programmed from NXT-G either over USB or Bluetooth

- LEGO earns money on NXT by selling the kit itself
- Licences to educational institutions are also revenue source
- First version of LEGO MINDSTORMS NXT was released in the beginning of 2007
- A second version appeared in the fall of 2009 (biannual release schedule:-)?
- Shipped with a different set of standard sensors (color sensor)



- LEGO partly produce and package NXT on their factory in Billund
- Staff around NXT development is not big
- Firmware development team is in control of the software development
- National Instruments (NI) designs the NXT-G environment
- NXT-G transferring its bytecode-compiled programs to NXT for storage and subsequent execution



LEGO Design Routines

- Lego has a group of people who has the privilege of talking directly to LEGO staff
- Under non-disclosure agreements (NDA)
- First group of MCPs (MUP) were a select group
- Used as a sounding board for the ideas that went into developing the first LEGO MINDSTORMS NXT.
- MCP 4 is commencing as we speak





- NXT is a complex product compared to other LEGO products
- Requires focus on quality
- Updated versions released on the official MINDSTORMS website
- First firmware had the version number 1.1 (or 1.0...) and the current released version for NXT 2.0 is versioned 1.28

Partners

- many institutions, companies, groups, and individuals have a stake in NXT
- National Instruments is presumably the most important
- MIT, Tuft, Carnigie Mellon, and others are also active
- Others such as *nxtprograms*
- Number of large events
- This includes `Brickfest`
- There has also been a MINDSTORMS NXT Sumo Competition



Sensor Partners and Independent Manufactures

- A limited number of third-party sensors exists
- Hitechnic, Mindsensors, and Vernier
- Hitecnic produces its sensors in a standard LEGO sensor housing
- Mindsensors make its sensors available in different shapes and forms
- Vernier specializes in natural sciences experiments
- DCP



Vernier®



Standard Sensors

- Input and output ports feature a 6-wire RJ12 connector
- Ultrasonic distance measurement sensor, a light intensity sensor, a sound sensor, a touch sensor, and motors



(a) Light



(b) Motor



(c) Sound



(d) Touch



(e) Ultra



(f) NXT-G

Standard Lego Mindstorms NXT sensors and a NXT-G block



Some Mindstorms History



- 10 years since Mindstorms was released
- First LEGO Mindstorms was the LEGO RCX: Successful
- LEGO intended it to be a closed source product, but...
- It was soon hacked:-)
- The open source strategy was pursued even more with the present LEGO Mindstorms
- A Goldplated NXT and a limited edition Black NXT was produced in relation to the 10 year anniversary

- Development of NXT started in the 2004
- Initial group was Ralph, Steven, John, and David
- First prototypes were circulated on a limited number basis inside the current MUP group and a select few outside that group
- MUP: Jan 2006, 5 people
- MUP II: Nov. 2005, 11 people
- MDP: 2006
- MCP II: 31 persons, 100 chosen from 10000 applicants
- MCP III: 2008-2009
- MCP IV: Commencing...
- NXTGCC was created during the fall of 2006





- Lego has a website for educational activities
- Strong focus on university segment
- LEGO MINDSTORMS NXT comes in educational sets
- Rechargeable battery and extra building pieces
- Sold with site licences that accompany the NXT

- MINDSTORMS community is large and diverse
- Blogs, forums, large events, books, the MCP program and more
- NXTStep and the NXTasy blogs
- Lugnet is a large LEGO fanclub



The screenshot shows the website nxtasy.org with a navigation bar (Home, Forums, Repository, Challenge, About) and a login link. The main content area features a large image of a LEGO Mindstorms NXT brick with a sun icon. Below the image is a blog post dated September 14th, 2009, titled "LEGO MINDSTORMS NXT installation on Snow Leopard". The post text discusses reports of installation issues on Snow Leopard (Mac OS 10.6) and provides instructions for users to manually install the software from the Mindstorms CD. A sidebar on the right lists the site's Editor-in-Chief (Guy Zin), Creator (Eric Salinas), and a list of contributors including Dave Atchley, Linus Storf, Claude Baumann, Daniele Scardicelli, Rachel Delorme, Rod Gilles, John Hansen, Joshua Henzl, Adam Parkes, Kasper Olsen-Pedersen, Mac Fuso, Eric Salinas, Jander Solleat, Dick Swan, Paul Tinger, Sven Tulevik, and Guy Zin. At the bottom of the sidebar is a search box labeled "Posts" with a search button.



Schools

- Many schools participate in First LEGO League
- A tournament where current theme is played out
- The idea is to have adults lead or mentor a team of children aged 9-14 (16 outside the US)
- NXT is used to complete a challenge put out by the FLL management



LEGO MINDSTORMS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



NXT Alternatives

- NXT is not completely alone...
- Sun Spots from Sun/Oracle
- Fisher-Technic offers a new robot controller which is based on an ARM9 processor running at 200MHz
- Various evaluation kits: mbed, IAR, etc.
- However, it is difficult to match dealer network, community, and history of LEGO

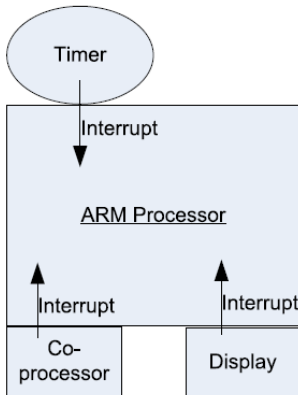




- A central micro processor controls NXT
- A second small micro processor assisting it
- Code is compiled to machine instructions which are executed one at a time
- NXT allows for firmware updates, which is the basis for alternative operating systems

NXT Firmware

- NXT actually consists of two different firmwares
- The main firmware controls the interaction with the user, and radio communication to other NXTs
- An ATmega48 generates the pulse width modulation (PWM) signals for the three motors
- It also provides input regarding the state of the push buttons on the NXT
- Host side SDK which allows for interaction with the NXT via a DLL



Firmware Enhancements



- Firmware enhancements are possible only because Lego released the software as open source
- IAR is a commercial compiler used by LEGO
- It was recently released in a free version for Mindstorms
- The other option for programming the existing firmware is to use GCC

Firmware Replacements

- A growing number of firmware replacements available
- Examples: lejos, NQC, pbLua, and RobotC
- A different example is TinyOS
- *leJOS* is based on a small virtual Java machine. It is a firmware replacement that allows for Java programming on Lego Mindstorms NXT. There is extensive support in forms of tutorials, well-developed APIs, and Netbeans/Eclipse programming IDEs at the leJOS website.
- Lua is a scripting language (see Lua website). It is said to perform faster than other scripting languages. Ralph Hempel has ported Lua to Lego Mindstorms NXT and labeled it *pbLua*
- *Not eXactly C* (NXC) is language similar to C. It is supported on Lego Mindstorms NXT by John C. Hansen and available from <http://bricxcc.sourceforge.net/nbc/>. It is built on top of the *Next Byte Codes* (NBC) compiler. It is a firmware replacement.



Hardware Developer Kit



- Lego provides the schematics and drawings for the NXT hardware
- Good for programming the ARM7
- Useful for developing a new sensor



- The free IAR compiler can be downloaded from IAR's homepage
- It is an industry strength programming environment called *Embedded Workbench*
- It interacts with a JTAG programmer connected to Lego Mindstorms NXT via the IAR C-SPY debugger
- The visualSTATE program interacts with the C-SPY debugger to give a graphical view of the debugging session

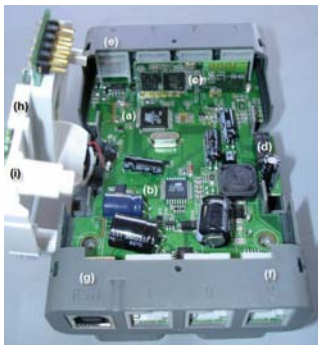
- The GCC compiler can be downloaded from the [nxtgcc](http://nxtgcc.sf.net) home page
- It is an *arm-elf-gcc* compiler
- There exists several good toolchains such as WinArm, CodeSurgery etc.



- NXT with touch, sound, light, and ultrasonic input sensors
- The three motors on the output ports are also shown
- NXT's main micro processor (MCU) is an AT91SAM7S256



- (a) ARM7 MCU
- (b) ATmega48 MCU
- (c) CSR BlueCore4 Bluetooth radio
- (d) SPI bus and touchpad signals
- (e) high-speed UART behind input port 4
- (f) output (generally motor) port
- (g) USB port, (h) four-button touchpad
- (i) 100x64 LCD display



NXT with important hardware highlighted



- Characterized by a number of modes, exception types, and interrupts
- Associated with each mode is a current program status register (CPSR)
- Each exception and interrupt type is associated with a priority
- 6 privileged modes named `system`, `svc`, `abort`, `fast interrupt`, `interrupt`, and `undefined mode`
- They can all modify the CPSR
- The unprivileged user mode cannot modify the CPSR



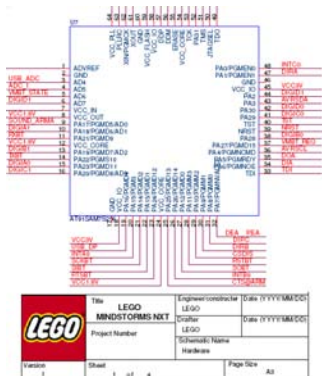
ARM7 Registers

- The CPSR defines the characteristics of the current mode
- It can mask (ie. not allow) interrupts and fast interrupts
- The CPSR includes a flag whether the ARM7 is executing thumb code or arm code
- There is a register file which includes 37 registers
- Each register is 32 bit wide
- Each mode uses 17 registers
- Registers R0-R12 are shared between `usr`, `sys`, `svc`, `abt`, `irq`, and `und` modes
- The `fiq` mode has its own banked version of the R8-R12 registers
- All modes have their own stack pointer and link/return pointer: R13 and R14
- The program counter, `pc`, is placed in R15



ARM and Thumb Instruction Sets

- The ARM instruction set is 32 bit
 - Thumb instruction set is 16 bit
 - Most of the code in NXT is compiled toward the Thumb instruction set
 - Thumb code is more dense than ARM
 - Exception and interrupt handlers are compiled to ARM code
- Demo ARM compilation.



NXT schematics

ARM Pins



PCB	Port/pin	ARM7 pin	ARM7 PIO
DIGIA0	1/5	15	PA23/PGMD11
DIGIA1	1/6	10	PA18/PGMD6/AD1
DIGIB0	2/5	38	PA28
DIGIB1	2/6	13	PA19/PGMD7/AD2
DIGIC0	3/5	41	PA29
DIGIC1	3/6	16	PA20/PGMD8/AD3
DIGID0	4/5	42	PA30
DIGID1	4/6	6	AD7

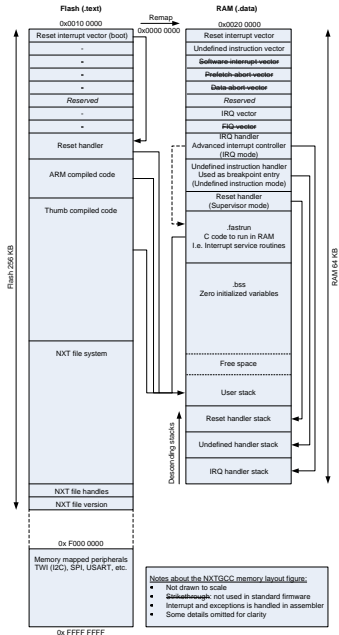
- Port 1 through 4 are mapped to general purpose input output pins
- Most ARM7 pins are multiplexed
- Pin 6 on inout port 1-3 both can serve as GPIO pins and as an analog to digital converter (ADC)
- The *Lowspeed* module use the GPIO pins to *bit-bang* an I2C protocol



- Flash memory is initially mapped to address 0x0000 0000
- Flash is also accessible at address 0x0010 0000
- NXT immediately branches to the reset handler that is defined in CStartup.S
- It uses a temporary stack in the RAM area
- The reset handler sets up the stacks for the interrupt handler (IRQ), the undefined instruction handler (UND) stack, and the user stack (USR)
- USR stack is used when the normal code executes
- The USR stack is used when the normal code executes
- IRQ stack is obviously used when the IRQ handler is invoked: it branches to the correct interrupt service routine (ISR)
- Controlled by writing the address of that ISR into the advanced interrupt controller (AIC) source vector register (SVR)
- Example: each of the three timers (TC0, TC1, TC2) have an ID, and that ID is used as index in the AIC SVR vector

Next: Memory Layout

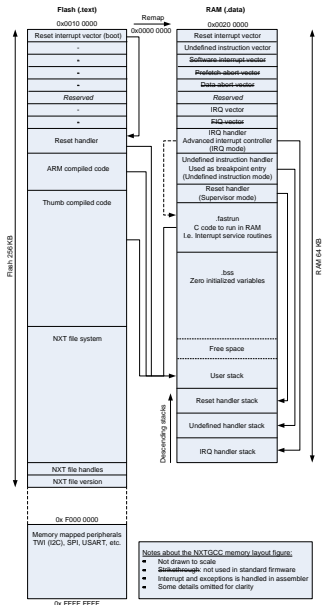
Memory Layout



Memory Layout



- The stacks on NXT (i.e. ARM7) are fully descending stacks
- Using the instructions STMFD and LDMFD
- Postfix FD depicts that we are working with a full descending stack



TODO: Show code with descending stacks

Undefined Instruction Stack Setup

- Undefined stack setup in Cstartup.S
- The illegal memory access (ABT) stack is set up before the undefined instruction stack

```
/* Set up Undefined Instruction Mode
   and set UND Mode Stack*/
.EQU UND_STACK_TOP, (ABT_STACK_TOP - ABT_STACK_SIZE)
msr    CPSR_c, #ARM_MODE_UND | I_BIT | F_BIT
mov    sp, r0 /* Init stack UND */
sub    r0, r0, #UND_STACK_SIZE
```



- Remapping of address 0x0000 0000 such that it is now the RAM and not the flash that is accessible from address 0x0000 0000
- Main reason for this is that we need to service interrupts even when the NXT flash-based file system is being written to
- Remapping the memory ensures that the interrupt vector table is access in RAM
- all interrupt service routines are linked into the .fastrun segment

Show segments in file.





- Mostly taking place in the Thumb compiled code
- The user code is executed in user mode [TODO: Tjek]
- Interrupt is asserted from one of the ARM7 peripheral sources then that interrupt service routine takes over
- Most peripherals in NXT are associated with a software module



- Each module is built from an header file, a C file, and .iom file
- Layered in a hardware physical layer (HPL), and a hardware abstraction layer (HAL)
- HAL files start with the letter c followed by an underscore, while the HPL files start with the letter d

NXT Software Modules and ARM7 HW Peripherals

LEGO MINDSTORMS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Name	ARM Peripherals	Note
Scheduler	-	Main control
Button	AVR via I2C/TWI	4 buttons
Display	SPI	LCD display
CMD	-	Virtual machine
Input	GPIO, PWM, AD	4 input ports
IOCtrl	I2C, AIC	AVR communication
Loader	MC	File management
Lowspeed	GPIO, PWM	I2C emulation
Output	AVR via I2C	Motor control
Sound	SSC	Loadspeaker control
UI	-	Menu system
Timer	PIT	Operating system tick
Bluetooth	SPI, GPIO	Communication control
Communication	UDP, UART, SPI	USB, RS485, and BT

Abbreviations: I2C:Inter-integrated circuit, TWI:Two-wire interface, PWM:Pulse width modulation, AIC:Advanced interrupt controller, MC:Memory controller, SSC:Synchronous serial controller, PIT:Periodic interval timer, SPI:Serial peripheral interface, UDP:USB device port, and UART:Universal asynchronous receiver tranceiver.

NXT Modules

Scheduler This is the main scheduler: `m_sched`. It works in a round robin fashion with a 1 ms system *tick*. It will call each of the modules listed below using function pointers. There is an initialization phase in which each module's `clnit` method is called. Then the system starts calling the `cCtrl` methods. Termination is done via calls to `CExit`.

Button It handles the four buttons on top of NXT. Principle: It uses the AD converter peripheral. Button 0 is read as? Button 1-3 is read as an AD converted value.

CMD This is the Lego virtual machine. It reads the executable files stored in flash and creates a RAM space for runtime information. The `rx` file contains bytecode instructions, bytecode scheduling, and run-time data.

Comm It takes care of the BT, USB and the Highspeed port. Peripherals used: SPI to BT, USB, and USART in RS485 mode.



NXT Modules

Display It is code for accessing the pixels of the 100 x 64 pixel display. It is possible to both set pixels, write characters, and strings (it is a matter of abstraction). There is a font included with the NXT. There are eight text lines on the display. It accesses the serial peripheral interface (SPI) hardware peripheral. The SPI works by writing data to a slave (the display in this case) over the master out slave in (MOSI) line. It receives data from the slave over the master in slave out line (MISO). The data flow is synchronous and is regulated by a serial clock (SPCK) line. The display receives a command from the ARM7 specifying which line is to be written, and then it receives the actual data for that line.

Input The input sensors are plugged into the 4 input ports on NXT. NXT will know which type of sensor is plugged into a port. It can be a temperature sensor, sound, lowspeed, and so on. The lowspeed sensor is an I2C/TWI compliant sensor. It uses two general purpose



NXT Modules

IOCtrl This module controls the booting and power down sequences of NXT. It will write to the AVR what the power should be and what the PWM for the motors should be. The IOCtrl module uses the I2C(TWI) hardware peripheral to relay this information to the AVR. It also uses the advanced interrupt controller (AIC), by having it call an I2C handler routine.

Loader File management is achieved by the loader module. It reads, writes, creates, and deletes files in the NXT flash. It resides in the upper part of the flash memory starting from the value of the memory address STARTOFUSERFLASH in d_loader.h. It uses the memory controller (MC) peripheral module of the ARM7. One special thing about the flash read and write is that writes are done on a per page basis, and it is not permitted to read from the flash memory during this process. Therefore, the methods that perform erase and write of flash pages are placed in RAM. This is achieved by marking the



NXT Modules

Lowspeed An I2C compliant device can be connected to one of the four input ports. The lowspeed module handles the communication and recognizes the connected sensor as either the standard ultrasonic device, or a custom device. It uses an IRQ handler that is invoked via the AIC hardware peripheral: It utilizes the pulse width modulation controller (PWM) peripheral to generate the clock. A four channel I2C compliant interface is controlled via this interrupt source. The Ultra sonic sensor is one such sensor.

Output Motors are connected to NXT through the three output ports: A, B, and C. The module communicates the motor output to the AVR via the I2C bridge. It is managed by struct, which is passed to the AVR from the ARM7 that controls this communication line. The information that controls the motors are an output mode and a PWM frequency for each motor. Equally important, the ARM7 uses the interrupts of the three timer channels to monitor each of the three



NXT Modules

Sound A loudspeaker with 8 bit resolution is controlled by the sound module. It plays sounds which are files residing in the NXT file system. Each time cCtrl method of the sound module is called, then it either plays a sound, continues to play a sound, or closes the sound file that has been played. The module utilizes the synchronous serial controller (SSC) hardware peripheral. With the SCC providing the AC signal ((SOUND_ARMA from PA17) to the SPY0030A audio driver, NXT can play sounds. It uses the peripheral DMA controller (PDC) to handle the transmission of the bitstream to the speaker.

UI The module manages the menu system, selected files, icons, animations, etc. on NXT. It responds when an user press on of the buttons on NXT.

Timer NXT is programmed around a system tick of 1 ms. This property is supported by the timer module that use the periodic interval timer (PIT) hardware peripheral to maintain a 1 ms counter.





- Changes needed to make the source code compile with an arm-elf based GCC
- See details in `ConvertNxtGcc.java` program
- First versions of NXTGCC were error-prone
- IAR supports, and Lego uses, nested flexible array members
- GCC does not support this
- Key was to change the nested flexible array to become a fixed size nested array

Nested Array



```
typedef struct
{
    UBYTE   FormatMsb;
    ...
    UBYTE   Data[];
}
BMPMAP;

typedef struct
{
    UBYTE   FormatMsb;
    ...
    UBYTE   PixelsY;
    /*nxtgcc converter*/
    UBYTE   Data[1];
}
/*nxtgcc converter*/
__attribute__((__packed__))
BMPMAP;
```



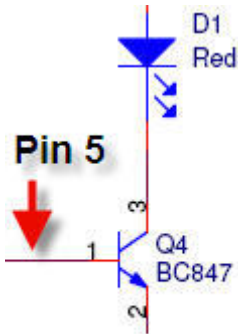
- Easiest way to debug NXT is to use the light sensor
- We can find the memory location that controls the GPIO pin
- Other choices is to use a JTAG debugger

Light sensor Port



```
//Turn on an input port
#define ON(port)
{
  *AT91C_PIOA_SODR = PORT_TO_PIN(port);
}

Turn on a light sensor on an input port for low-level
debugging
```



Source: LEGO

Light sensor schematics: Show how to turn on the light sensor

Example (from c_cmd.c):

```
{// LCD DEBUG START
  cnt++; // a static counter
  UBYTE DebugString[50];
  sprintf(DebugString, "Debug info: %d", cnt);
  if(cnt==1)
  {
    cDebugString(&DebugString[0]);
  }
} //LCD DEBUG END
```

Showing a debug string in the display



- We may think (initially) that we have stopped the processor
- A break point can be inserted into the code using the GCC inline assemble syntax

```
/*  
 * Trigger a breakpoint exception.  
 */  
__ramfunc void bsp_breakpoint(void) {  
    //asm volatile (".word 0xE7FFDEFE");  
    NXTGCCBREAK;  
}
```





- GDB is the debugging system that is connected to GCC
- GDB system offers a remote debugging system
- USB driver to use is the Fantom SDK
- Fantom C++ files, there is a section with extern C"
- Read and write methods for arbitrary bytes
- Allows for an implementation of the GDB remote debugging protocol
- Debugger is arm-elf-gdb



- It includes a one-byte checksum
- Comes after the # in the packet
- Problem is that the Fantom DLL does not like if we try to read from the USB if there is not something to read
- We read two characters (checksum)
- NXTGCC Fantom GDB wrapper (discussed later) writes and especially reads one character at a time to/from NXT, even though the USB endpoints are in bulk mode

Demo GDB here.

Hardware Debugging

- Open source project called OPENOCD
- It is possible to create one using a 1.27 mm pitch row
- Each of the eight pins from J17 can be connected to a standard 20 port JTAG like this:
1 → 9, 2 → 7, 3 → 13,
4 → 15, 5 → 5, 6 → 4,
7 → *not connected*, and
8 → 1.



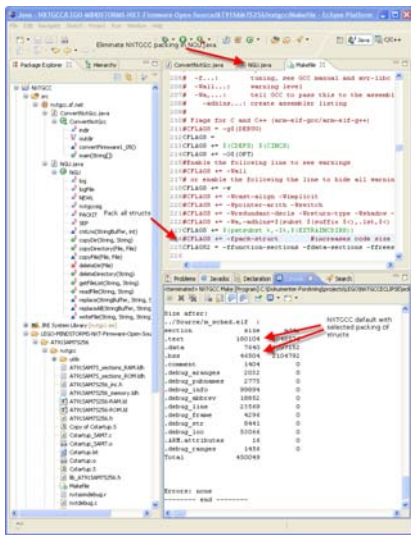
Using a JLink to flash an ARM7 Eval. board, while warming up for NXT to arrive in the summer of 2006.



Compiler Performance



- Eclipse is one possibility as an programming environment
- Output from compiling the firmware has been shown
- Size and memory addresses of the different sections are displayed after a successful build



The screenshot shows the Eclipse IDE with the NXTGCC compiler output. The 'Problems' window displays the following table:

File Name	Size	Address	Notes
..../Process_packed.o	6184	00000000	NXTGCC default with selected packing of structs
text	180104	00000000	
data	7648	00000000	
bss	48504	00000000	
..../nxtgcc.o	14074	00000000	
..../debug_ranges.o	2032	0	
..../debug_subroutine.o	2776	0	
..../debug_info.o	90096	0	
..../debug_decoder.o	18952	0	
..../debug_line.o	21568	0	
..../debug_EE.o	4296	0	
..../debug_PIC.o	8168	0	
..../debug_io.o	51064	0	
..../ARR_structures.o	16	0	
..../debug_ranges.o	1456	0	
Total	400248		

Reading the compilation statistics in Eclipse

Comparison of GCC and IAR Code Size

Compiler	Packing structs	Size of .text (bytes)	Size of .data (bytes)
GCC	All	202052	54548
NXTGCC	Selected	180104	54144
GCC	None	162136	54228
IAR	(speed)	127552	49831
IAR	(size)	122440	49763

GCC compiler (arm-elf-gcc version 4.2.2) that NXTGCC uses, creates a larger code footprint than the IAR compiler does. For the flash we can compare the GCC compiler's approx. 175 KB to the IAR compiler's approx. 120 KB. Regarding RAM, the picture is approx. 53 KB compared to approx. 49 KB in favor of IAR.





Comparison of GCC and IAR Code Speed

Compiler	Speed # loops
NXTGCC (selected structs)	615
IAR (size optimized)	640

TODO: Cite Benchmark



Lego, Atmel, and the NXTGCC project provide comprehensive information about NXT. This includes hardware schematics, explanation of important protocols, and documentation for programming NXT:

- **Lego:** Hardware Developer Kit
- **Lego:** Software Developer Kit
- **Atmel:** AT91SAM7S256 aka ARM7 hardware description
- **NXTGCC** <http://nxtgcc.sourceforge.net>



Features

- Incorporates the ARM7TDMI® ARM® Thumb® Processor
 - High-performance 32-bit RISC Architecture
 - High-density 16-bit Instruction Set
 - Leader in MIPS/Watt
 - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Internal High-speed Flash
 - 512 Kbytes (AT91SAM7S512) Organized in Two Contiguous Banks of 1024 Pages of 256 Bytes (Dual Plane)
 - 256 kbytes(AT91SAM7S256) Organized in 1024 Pages of 256 Bytes (Single Plane)
 - 128 Kbytes (AT91SAM7S128) Organized in 512 Pages of 256 Bytes (Single Plane)
 - 64 Kbytes (AT91SAM7S64) Organized in 512 Pages of 128 Bytes (Single Plane)
 - 32 Kbytes (AT91SAM7S321/32) Organized in 256 Pages of 128 Bytes (Single Plane)
 - Single Cycle Access at Up to 30 MHz in Worst Case Conditions
 - Prefetch Buffer Optimizing Thumb Instruction Execution at Maximum Speed
 - Page Programming Time: 6 ms, Including Page Auto-erase, Full Erase Time: 15 ms
 - 10,000 Write Cycles, 10-year Data Retention Capability, Sector Lock Capabilities, Flash Security Bit
 - Fast Flash Programming Interface for High Volume Production
- Internal High-speed SRAM, Single-cycle Access at Maximum Speed
 - 64 kbytes (AT91SAM7S512/256)
 - 32 kbytes (AT91SAM7S128)
 - 16 kbytes (AT91SAM7S64)
 - 8 kbytes (AT91SAM7S321/32)
- Memory Controller (MC)
 - Embedded Flash Controller, Abort Status and Misalignment Detection
- Reset Controller (RSTC)



AT91 ARM
Thumb-based
Microcontrollers

AT91SAM7S512
AT91SAM7S256
AT91SAM7S128
AT91SAM7S64
AT91SAM7S321
AT91SAM7S32



Periodical Interval Timer

The PIT timer generates the system tick.

AT91SAM7S Series Preliminary

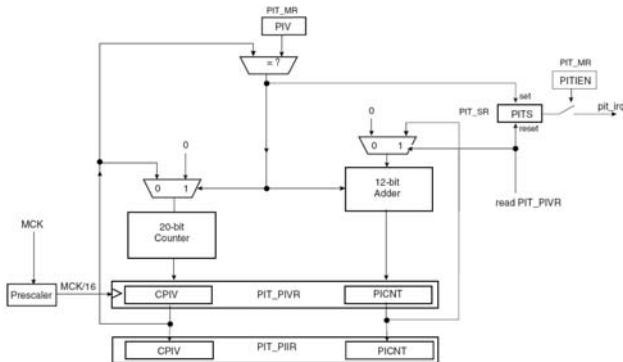
16. Periodic Interval Timer (PIT)

16.1 Overview

The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer



Memory Controller

The MC generates the abort exception (TODO: CHECK).



AT91SAM7S Series Preliminary

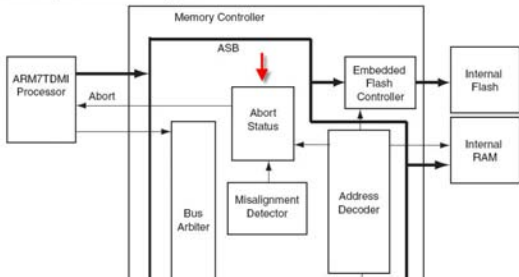
19. Memory Controller (MC)

19.1 Overview

The Memory Controller (MC) manages the ASB bus and controls accesses requested by the masters, typically the ARM7TDMI processor and the Peripheral DMA Controller. It features a simple bus arbiter, an address decoder, an abort status, a misalignment detector and an Embedded Flash Controller.

19.2 Block Diagram

Figure 19-1. Memory Controller Block Diagram



MCU Overview I

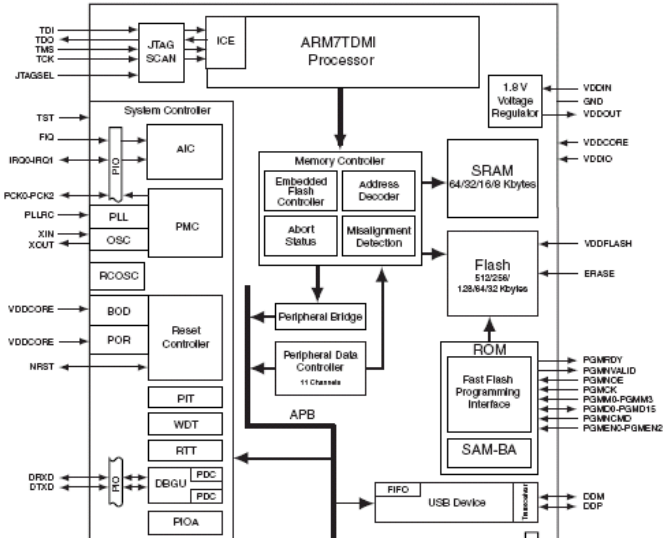
SAM-BA, SPI, TWI, AIC, SSC, etc.

LEGO MINDSTORMS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>

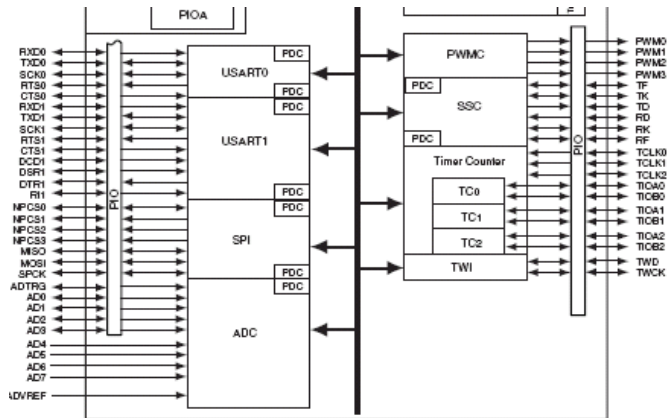


AT91SAM7S512/256/128/64/321 Block Diagram



MCU Overview II

SAM-BA, SPI, TWI, AIC, SSC, etc.



JTAG, Thumb, ARM



7. Processor and Architecture

7.1 ARM7TDMI Processor

- RISC processor based on ARMv4T Von Neumann architecture
 - Runs at up to 55 MHz, providing 0.9 MIPS/MHz
- Two instruction sets
 - ARM® high-performance 32-bit instruction set
 - Thumb® high code density 16-bit instruction set
- Three-stage pipeline architecture
 - Instruction Fetch (F)
 - Instruction Decode (D)
 - Execute (E)

7.2 Debug and Test Features

- Integrated EmbeddedICE™ (embedded in-circuit emulator)
 - Two watchpoint units
 - Test access port accessible through a JTAG protocol
 - Debug communication channel
- Debug Unit
 - Two-pin UART
 - Debug communication channel interrupt handling
 - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on all digital pins



256KB

8.2 AT91SAM7S256

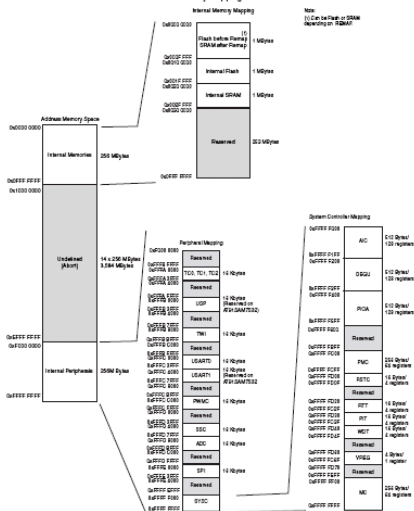
- 256 Kbytes of Flash Memory, single plane
 - 1024 pages of 256 bytes
 - Fast access time, 30 MHz single-cycle access in Worst Case conditions
 - Page programming time: 6 ms, including page auto-erase
 - Page programming without auto-erase: 3 ms
 - Full chip erase time: 15 ms
 - 10,000 write cycles, 10-year data retention capability
 - 16 lock bits, protecting 16 sectors of 64 pages
 - Protection Mode to secure contents of the Flash
- 64 Kbytes of Fast SRAM
 - Single-cycle access at full speed



ARM MCU Peripherals and Memory Mappings

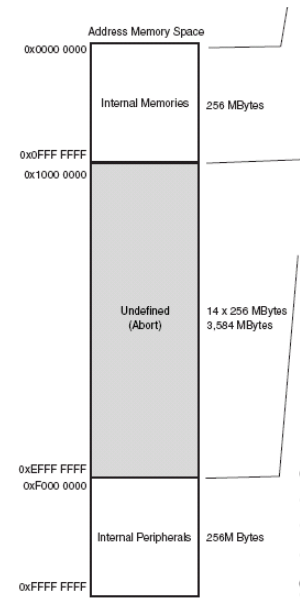
The Big (Small...) Picture

Figure 8-1. AT91SAM7S512/256/128/64/32/16 Memory Mapping



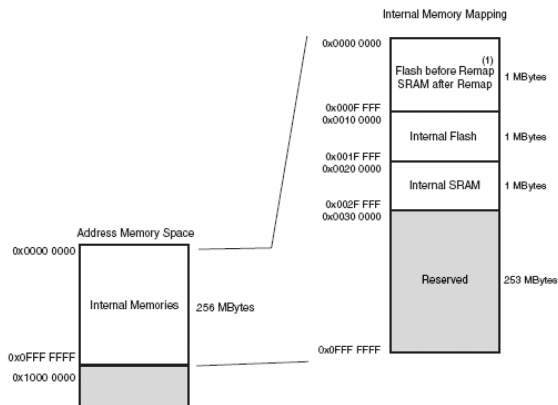
ARM MCU Peripherals and Memory Mappings

Main memory



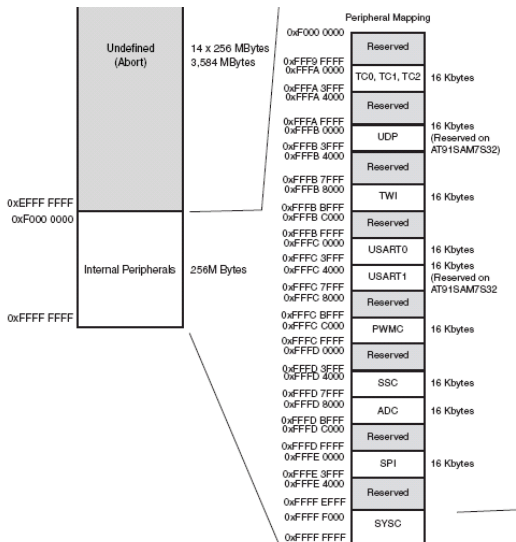
ARM MCU Peripherals and Memory Mappings

Main memory remapping



ARM MCU Peripherals and Memory Mappings

Peripherals such as timers



ARM MCU Peripherals and Memory Mappings

System Peripherals such as interrupt controller

System Controller Mapping

	0xFFFF F000	AIC	512 Bytes/ 128 registers
32 bytes	0xFFFF F100 0xFFFF F200	DBGU	512 Bytes/ 128 registers
32 bytes served on ISAM7532	0xFFFF F300 0xFFFF F400	PIGA	512 Bytes/ 128 registers
32 bytes	0xFFFF F500 0xFFFF F600	Reserved	
32 bytes 32 bytes served on ISAM7532	0xFFFF F700 0xFFFF FC00	PMC	256 Bytes/ 64 registers
	0xFFFF PC00 0xFFFF PD00 0xFFFF FDF0	RSTC	16 Bytes/ 4 registers
		Reserved	
32 bytes	0xFFFF FD00 0xFFFF FC20	RTT	16 Bytes/ 4 registers
	0xFFFF FD00	PIT	16 Bytes/ 4 registers
32 bytes	0xFFFF PC30 0xFFFF FD40	WDT	16 Bytes/ 4 registers
32 bytes		Reserved	
	0xFFFF FD60	VREG	4 Bytes/ 1 register
32 bytes	0xFFFF PC60 0xFFFF ED70 0xFFFF FEF0 0xFFFF FF00	Reserved	
		MC	256 Bytes/ 64 registers
	0xFFFF FFFF		





- NXTGCC is based on the LEGO source code. It was provided by LEGO to facilitate the GCC compiler use.



- LEGO uses the IAR compiler internally.
- It is a nice and easy programming tool to use. IAR also provides configurations for some Texas Instruments Zigbee chips.



- `binsert.c`
- `nxtgcc.rfw`
- `ConvertNxtGcc.java` and `NGU.java`
- `Source-xx` \implies Source copy
- `m_sched.map`



- NXTGCC note:
<http://nxtgcc.sourceforge.net/nxtgcc.pdf>

Module SW

Software

Software

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

Software

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- ① Main software parts
- ② LEGO OS
- ③ TinyOS parts
- ④ Low level SW





- AT48 initialization done over I2C from ARM7 (otherwise the ARM7 will be shut down)
- It is a string Let's SAMBA arm in arm...that is sent to the AVR
- One ms update round robin: Display->BT->Sensors->AT48 etc.



- Timers
- Interrupts
- Scheduler



- The timers in TinyOS are defined in the Timer Interface.
- **fired** is the main event in the Timer interface. A number of commands like **startOneShot** and **stop** controls the Timer
- The timer system in TinyOS is fairly complex... It uses almost all of the facilities of the nesC language to provide a virtual parameterized timer interface
- It provides up to 255 individual timers running on top of a few physical timer channels

Interrupts.

- If we start with the hardware, the advanced interrupt controller (AIC) handles the interrupts on the ARM
- Each peripheral (timers, spi, twi, etc) is associated with an id (AT91C_ID_TC0 = 12, AT91C_ID_SPI = 5, AT91C_ID_TWI = 9)
- This ID is used twice when programming an interrupt source in TinyOS. First, it is used when the interrupt interfaces are set up to identify the interface. Secondly, it is used when the interrupt is dispatched to the correct interrupt handler
- An interrupt is assigned a priority
- **nesC** provides language support for tagging interrupt context methods as asynchronous code, which then can use `atomic` blocks to protect state variables



- Scheduling in TinyOS is flexible. Out-of-the-box it provides us with a round-robin scheduler that runs tasks
- One task does not interrupt another task. I.e. each task runs to completion
- Tasks are defined in a nesC module and *posted* to the scheduler. An error is returned from the posting call if the task is already in queue
- Important: Tasks can post themselves, and this can be useful when working with algorithms that are computationally intensive



Low Level.



- The assembler startup file
- PlatformP (a TinyOS file) that initiates the hardware
- PlatformC which is used to wire to critical components
- In nxtmote we wire to a components that starts the system level timer (PIT timer) to fire each ms

Compilation.



- The compilation process is nesC code to C code and C code to assembler
- The nesc compiler places the C source in app.c and with the correct gcc compiler options we can have a look at the assembler code, corresponding C code statements, and nesC code
- Example: `2059 0bbc 0900A0E3 mov r0, #9`
- It will move the constant 9 into register r0
- `0900A0E3 = 1001 00 0 0000 0 1010 0000 11100011`



- LEGO MINDSTORMS Hardware and Software documentation. <http://mindstorms.lego.com/Overview/NXTreme.aspx>
- You can take a look at diku.dk/~leopold and see the TEP 999 draft. It discusses how to set up a new platform if that is part of your plans...

Module HW

Hardware

Hardware

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.



- 1 The main microprocessor in NXT: ARM7
- 2 The main communication protocols on the board
- 3 Components of NXT: Display, input/output ports etc.
- 4 Types of sensors
- 5 Relation between the coprocessor AT48 and the ARM7

Outline

Hardware

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- Startup
- Registers
- Modes
- Display
- Inputs ports
- Output ports
- BT



Startup Process.

- The system starts in an assembler file that sets up the interrupts
- The **main** method is in the RealMainP module
- Different initialization levels and the boot event
- After the boot procedure is done your application component will have the high level code that connect to various event handlers
- Example: If we have a bluetooth application then we will have a method for sending packets and on the receiver side an event that is called when the underlying bluetooth layers receive a packet



Registers, registers, registers...

- The ARM7 can be in a number of different modes
- The registers provide the interface to do all sorts of things with the processor. For example you can set up timers from registers
- **Example:** `AT91_Timer_Reg_Mode = TRIGGERONMATCH...`
- All registers are described in the PDF manual **AT91 ARM Thumb-based Microcontrollers** which covers the following list of microcontrollers: AT91SAM7S512, **AT91SAM7S256**, AT91SAM7S128, AT91SAM7S64, AT91SAM7S321, and AT91SAM7S32
- LEGO is using the MCU with 64 KiB RAM and 256 KiB Flash memory
- A register is a memory location in the ARM7 where we can read a register value
- Or write a value to the MCU register
- You can think of it as a method call with an argument (if you write) or a return value (if you read). Similar to getset methods in say Java



- An important register in the timer unit is the channel mode register: **TCCMR**
- Page 405 tells us that the TCCMR is offset 0x04 in the timer unit struct
- It is read/write (the write is what we will use)
- it is 32 bits and the first three bits are for selecting a clock to use. The ARM7 has 5 clocks to choose from
- A bit field named *RC Compare Trigger Enable* (**CPCTRG**) depicts if the counter should reset or continue to count when a compare match is encountered



Registers III



```
typedef struct _AT91S_TC_{
    unsigned AT91_REG TC_CCR; // Channel Control Register
    unsigned AT91_REG TC_CMR; // Channel Mode Register (Capture Mode / Waveform Mode)
    unsigned AT91_REG Reserved0[2]; //
    unsigned AT91_REG TC_CV; // Counter Value
    unsigned AT91_REG TC_RA; // Register A
    unsigned AT91_REG TC_RB; // Register B
    unsigned AT91_REG TC_RC; // Register C
    unsigned AT91_REG TC_SR; // Status Register
    unsigned AT91_REG TC_IER; // Interrupt Enable Register
    unsigned AT91_REG TC_IDR; // Interrupt Disable Register
    unsigned AT91_REG TC_IMR; // Interrupt Mask Register
} _AT91S_TC, *_AT91PS_TC;
```

Modes.

- Either the processor is executing the normal program flow or it is processing an interrupt
- The interrupt is triggered by some subsystem on the ARM7 such as the timer
- The ARM7 can branch directly to an interrupt handler based on the subsystem that triggered the interrupt
- There are two interrupt modes: fast and normal
- Important peripheral in this context is the Advanced Interrupt Controller (AIC)
- The AIC will take up to 32 sources and raise prioritized (user-defined) interrupts to the ARM processor

Note that this notion of normal program flow and interrupts carry over to TinyOS and the nesC programming language (as shall be discussed later).



Display.

- The display is 100x64 LCD
- It is controlled via SPI
- It works in a line mode. First a command is sent telling it which line is to be updated then the actual line data is sent
- NXTMOTE API exposes two versions of writing to the display: *Slow* updates that are serviced each ms, and *fast* updates that spin (waiting in a loop) the processor until the display is updated. The latter can be used if the display is used from an interrupt routine
- In the examples that we will look at later, we will see that `sprintf` is used to format a string and a call named `displayString` is used to show the string on the display
- A system timer runs each 1 ms to update the buffer to the display.



Input ports.

- Those four ports that are numbered 1-4 on NXT
- The sensors can be digital or active sensors
- Digital sensors work over a I2C protocol
- Active sensors have an analog value read. The value is then sent back to the ARM7 from the AT48
- Connected to port 4 is a high-speed UART that allows for sensors with higher bandwidth demands (cameras, radios etc.)
- For the exercises we are using the analogue ports from 1 to 3. They provide a value from 0 to 1023.



Output ports.



- The three ports named A-C are output ports
- Each port can provide a PWM signal
- The ports provide feedback the ARM7 to determine rotation direction



- A protocol named I2C is used. It is called TWI (two wire interface) in the Atmel world.
- In NXT is means that a struct is sent from the AT48 to the ARM7 every 2 ms and also from the ARM7 to the AT48 every 2 ms.
- In the struct that comes from the AT48, we find the ADC values from the input ports



- LEGO MINDSTORMS Hardware and Software documentation. <http://mindstorms.lego.com/Overview/NXTreme.aspx>

Module Radio Radio

Radio

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

Radio

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- ① Radio: Wireless Sensor Networks
- ② Standard NXT radio: Bluetooth
- ③ Active messaging

Overview

Radio

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>





- The idea is to use wireless communication as a replacement for cables
- Similar to a cabled IP network we have a number of motes sending packets over the air to one another
- Routing protocols exist within WSN networks as they do within IP networks
- Conservation of energy plays a role in WSN
- Dynamic (unpredictable) topology is a characteristic of WSN

Active Messaging.

Radio

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- A concept to dispatch messages in a network or parallel computer system
- A message id acts as a simple routing mechanism
- To be very specific: If you remember the nesC term *parametrization* then an interface can be parameterized according to an active message id

NXT Bluetooth.



- LEGO NXT uses a Bluetooth radio
- Bluetooth is a wireless short-range standard for communicating between small devices
- In its basic forms a small network with one master and a number of slaves
- For NXT, the Bluetooth radio forms a network with three slaves and one master
- Bluecore (the bluetooth radio) in NXT support the serial port profile

BT module.

- The Bluetooth module in NXT is probably the most complicated piece of software
- From a systems point of view the ARM7 chip controls the Bluecore chip
- A set of communications commands such as starting an enquiry (the way Bluetooth devices discovers neighbors)
- The Bluetooth module has its own software implementation (done by LEGO) running and it serves as the middle-man between two ARM7 processors on two different NXTs
- Raw data is transmitted in what is called "Btstream mode"
- We use the TinyOS message abstraction, but in essence we write bytes to the Bluetooth radio



NXTMOTE BT.

- The term *Active Message* is well-developed in TinyOS, and therefore the NXTMOTE BT offers a simplified version.
- Most (99% or so?) of NXTMOTE work with be done within a NXTMOTE network, so we can boil the 6 byte bluetooth address to the 2 bytes address used in the TinyOS active message stack
- A fairly safe approach would be to use a checksum of all the 16 bit parts of the BT address
- But simplicity is easier for most purposes, so we just use the first 16 bits of the BT address
- As the "active message" is about to be sent, we find the full BT address of the receiver and send it

Radio

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>





- TEP 111: message_t

Module Sensors

Sensors

Sensors

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.



- ① LEGO
- ② Mindsensor
- ③ Hitechnic

Gliederung

Sensors

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Lego Sensors.



- Light
- Touch
- Sound
- Ultra Sound

Standard LEGO MINDSTORMS NXT sensors and NXT-G block

Sensors

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



(a) Light



(b) Motor



(c) Sound



(d) Touch



(e) Ultrasound



(f) NXT-G block



Mindsensors.

- Camera
- Realtime Clock
- Acceleration (2,3, and 5 axis reading)
- Dual Infra Red Obstacle Detector
- Motor Multiplexer for NXT
- Magnetic compass
- Pneumatic Pressure Sensor
- High Precision Long Range Infrared distance sensor
- High Precision Infrared distance sensor (short, medium, long range)
- Various accessories: port splitter, cables, plugs, sensor kit, tools



Selected NXT sensors from Mindsensors



(g) Multi-axis acc.



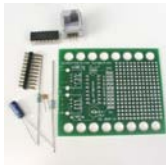
(h) Magnetic compass



(i) Pneumatic pressure



(j) Infrared distance



(k) Sensor building kit



(l) Temperature



- Prototype Board
- Gyro
- IR seeker
- Compass
- Color sensor
- Acceleration/tilt
- Sensor and motor multiplexer
- Accessories: Cables



Sensors from Hitechnic



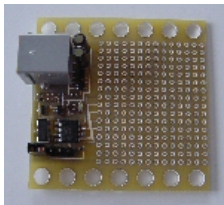
(m) Color



(n) Gyro



(o) Multiplexer



(p) Prototyping board

Sensors

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Sensors for exercises.



- Light sensor
- Touch sensor
- Microphone



- LEGO MINDSTORMS NXT Hardware Developer Kit.pdf and appendices 1-8.

Module NXTMOTE

NXTMOTE Project

NXTMOTE Project

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

- 1 Program
- 2 Vision
- 3 History
- 4 Structure
- 5 Links
- 6 Sourceforge
- 7 Contrib
- 8 Documentation
- 9 Wiki
- 10 TODO



Overview

NXTMOTE Project

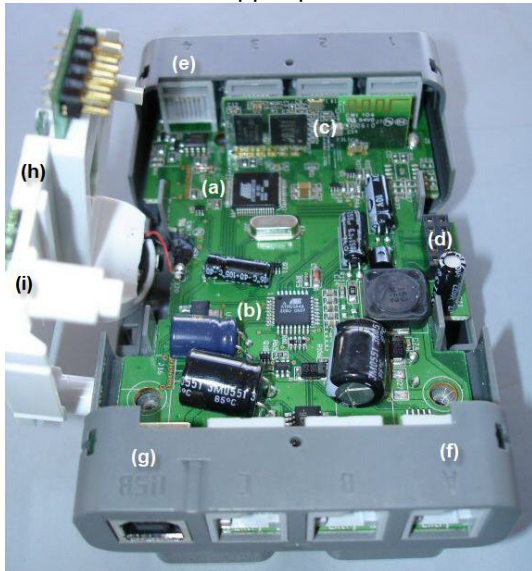
ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Taking a look at the hardware

Let's start with the fun part...

If we removed the upper part of NXT and looked inside...



The Idea with NEXTMOTE.



- To enable a larger group of people to explore what TinyOS is and what it can do
- To raise awareness of different operating system approaches by comparing the firmware replacements for NXT
- To be a flexible platform for embedded systems education and allow experimentation with a multitude of sensor setups

Timeline.



- ① 2006: Contact with LEGO and decision to start the project
- ② 2006: Convert LEGO NXT IAR compiler code to be gcc compatible and make first "Hello WorldTinyOS on NXT approaches by comparing the firmware replacements for NXT
- ③ 2007: Setting up various core TinyOS matters like interrupts, timers, sensors, initialization, radio
- ④ 2008: Starting to use NXTMOTE for TinyOS teaching purposes

More complete list at <http://nxtmote.sourceforge.net>.

Structure.

- The platform code is in the CVS repository provided by the TinyOS project. It is called `tinyos-contrib`. You can find it from the main TinyOS website <http://tinyos.net>
- Documentation is found at the companion website at sourceforge.net under the name **nxtmote**
- There is a wiki (provided by the TinyOS documentation group) which can be used for a NXTMOTE community to provide project links and additional documentation
- Suggestions, bugs, etc. can be directed to the project e-mail nxtmote@gmail.com



Links.



- There are some links that will be a good starting point
- <http://www.tinyos.net>
- <http://docs.tinyos.net>
- <http://mindstorms.lego.com>
- <http://nxtgcc.sourceforge.net>
- <http://nxtmote.sourceforge.net>

How can one contribute to the TinyOS community? Here is an example based on *nxtmote*

- The so-called TinyOS caretakers assure that a well-maintained project structure for contributed projects is in place
- NXTMOTE is one of these projects and the code is available via CVS
- The other sourceforge project named *nxtmote* provides access to the releases which are compressed files



The nextmote project is young still, and you are the first group to really try it. There is some way to go, but here is the intended structure of the documentation.

- The slides provide a first view to the various aspects of NXTMOTE and TinyOS technologies necessary to get started
- The wiki pages co-hosted under the TinyOS docs are dynamic in nature and for use by anyone using NXTMOTE
- The code is documented to some extent and the nesdoc API for NXTMOTE can be browsed from the project home page



Some ideas for you



- There is an idea list on the [nxtmote website](#) if someone is up for a project on their own
- Perhaps you: Develop an application and link to it from
- There is a sourceforge project named *nxtmoteprojects* where you can place a copy of [nxtmote](#) and start some student project. Someone is starting up on a student bluetooth project from a US university
- You can provide some project information in the [nxtmote wiki](#) at `docs.tinyos.net`



- R. U. Pedersen. Book chapter: TinyOS Education with LEGO MINDSTORMS NXT on NXTMOTE
- As MINDSTORMS NXT is based on an ARM7, I recommend that you go to the Atmel website and read the technical documentation on the processor.
- A first step would to read the TinyOS programming manual...

Module nesC

nesC

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

- 1 Introduction to nesC
- 2 Interfaces
- 3 Operators
- 4 Modules
- 5 Configurations
- 6 Parametrization
- 7 Default handlers

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Overview

1 Introduction

2 nesC Interfaces etc.

3 Further Reading

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Introduction to nesC

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- The language using when programming TinyOS
- Associated with some learning curve, but having nesdoc helps.
- nesdoc produce html files based on the nesc files
- It is documented in the nesC language specification and described in the *TinyOS Programming Manual*

Outline

Introduction

nesC

Interfaces etc.

Further Reading



- Operators: It contains the operators = and ->
- Examples: The operators wire modules together
- Important words: wire, pass-through wiring, modules, interfaces, configurations, parameterized, call, command, event, and signal
- More important words: unique, fan-in, fan-out, provides, uses, implementation

[Outline](#)

[Introduction](#)

[nesC](#)

[Interfaces etc.](#)

[Further Reading](#)

Various nesC code

Examples of pass-through wiring and normal wiring

```
configuration HalBtC {
  provides {
    interface HalBt;
  }
  uses {
    interface BTIOMapComm;
  }
}
implementation {
  components HalBtM, MainC;
  components BTIOMapCommP;
  components BC4ControlC;

  HalBt = HalBtM.HalBt;

  HalBtM -> MainC.Boot;

  ...
}
```

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

- nesC interfaces define the contracts for each module
- A interface can be parameterized to some type (advanced use)
- Each function in an interface is defined as an event or a command.
- It is also tagged as async (interrupt context) or sync (default)
- Can be parameterized based on a type



Interface Code

```
interface HalBt {  
  
    /**  
     * Send a message.  
     *  
     * @param msg Message to send.  
     */  
    command error_t sendMsg(uint8_t* msg);  
  
    ...  
}
```

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading



- nesC programs are composed of components, which are modules, configurations (and components)
- Modules contain the actual code
- Configurations wires it all to one application
- The `implementation` section contain task, command, event or straight C code.

[Outline](#)

[Introduction](#)

[nesC](#)

[Interfaces etc.](#)

[Further Reading](#)

Module code

```
module HalBtM {
    provides interface HalBt;
    uses interface HplBc4;
}
implementation {
    event void Boot.booted() {
        call BTIOMapComm.setIOMapCommAddr (&IOMapComm);
        call BTInit.init();
        cCommInit(NULL);
    }
    command error_t HalBt.sendMessage(uint8_t* msg) {
        error_t error = SUCCESS;
        return error;
    }
    ...
}
```

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Configurations

- Configurations connects the system to become an application
- There are two distinct ways a module work.
- It provides and uses interfaces just a module
- But more importantly it wires (using = or \rightarrow in its implementation section) to other modules or configurations

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Configuration code

```
configuration HalBtC {
  provides {
    interface HalBt;
  }
  uses {
    interface BTIOMapComm;
  }
}
implementation {
  components HalBtM, ...
  HalBt = HalBtM.HalBt;
  HalBtM -> MainC.Boot;
  ...
}
```

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Parametrization.

- It is possible that one interface can be used for different types. nesC allows for this.
- The timer interface is defined for various resolutions like milli second
- The parameter value can be used when a *call-back* is issued
- It makes use of the unique and uniquecount words

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

Default handlers.

- It is a mechanism to make wiring optional for signals or commands
- Examples: The operators wire modules together

nesC

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

Introduction

nesC

Interfaces etc.

Further Reading

- **TinyOS Programming Manual:**
<http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf>
- **nesC Language Reference Manual:** <http://nesc.sourceforge.net/papers/nesc-ref.pdf>



Module TinyOS

TinyOS

TinyOS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

TinyOS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- ① TinyOS
- ② What is the TinyOS community?
- ③ How to get started?
- ④ Conferences and journals



- Main web site at www.tinyos.net
- Mail lists for users and developers
- Core developers from US universities initially, but EU-based developers also represented as well as the rest of the world
- A rapidly growing community...
- Commercial companies are coming up like <http://tinyosmall.com/> and <http://www.xbow.com/>. But do not forget that you can get good generic platform in the form of MINDSTORMS...:-)



- TEP: Technical documents that are subject to a public review process. Examples include best practices for development, how to set up a new platform and many more.
- It is also supposed to be the way a contribution makes it way from the TinyOS contribution CVS to the core CVS
- Project listing on <http://www.tinyos.net>
- TinyOS contrib: A sourceforge repository with a process for contributing new projects
- NXTMOTE is one such project
- According to the contrib caretakers there were a huge increase in projects over the last year



What you need.

- Invest in a platform like mica or NXT for example
- There are mote platforms from TinyOS Mall, CrossBow, Intel, Sentialla (upcoming), LEGO MINDSTORMS, Sensinode, BTnut, SUN (Sun Spot) etc. etc.
- There are different operating systems available featuring various languages (nesC, Java, C, BTnut etc.)
- **nesC** provides good structure while still allowing low level HW access (preferred to some)
- If you are new to TinyOS it would make sense to get the LEGO MINDSTORMS platform. The software and hardware documentation provided by LEGO is very good. The NEXTMOTE project is an entry-level TinyOS platform with unlimited possibilities due to numerous affordable sensors.



What you can do.

- Do the tutorials. They are now places in the TinyOS wiki
- Read the *TinyOS Programming Manual* a few times
- Read the LEGO documentation
- Read the NXTMOTE manual and slides
- Use nesdoc to browse the code in HTML form
- Make some simple applications
- Sign up for the TinyOS mail lists (tinyos, devel, help, devel, tinyos-2-commits)



Conferences and Journals.

For graduate students and research staff there are numerous possibilities:

- Conferences: TinyOS Technology Exchange (TTX), Information Processing in Sensor Networks (IPSN), Embedded Networked Sensor Systems (SenSys), European conference on Wireless Sensor Networks (EWSN), IEEE Real-Time Systems Symposium (RTSS), International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), International Conference on Body Area Networks (BODYNETS), International Conference on Distributed Computing Systems (ICDCS), International Conference on Distributed Computing in Sensor Networks (DCOSS), International Conference on Networked Sensing Systems (INSS), Workshop on Embedded Networked Sensors (EmNets), etc.
- Journals: ACM Transactions on Sensor Networks (TOSN), International Journal of Distributed Sensor Networks, International Journal of Sensor Networks (IJSNet), etc.
- Other: European Conference on Machine Learning (ECML), International Conference on Machine Learning



Readings

TinyOS

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- <http://www.tinyos.net>

Module SVM

Support Vector Machine

Support Vector
Machine

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SVM

DSVM

SVM constraints

ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

- 1 Intro to machine learning
- 2 Classification, Clustering, Regression
- 3 Implementation issues
- 4 Support Vector Machine

Support Vector
Machine

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SVM

DSVM

SVM constraints

Overview

1 What is Machine Learning?

2 Types of ML

3 Implementation issues

4 Support Vector Machine

HW constraints

SVM basics

SMO

DSVM

SVM constraints

Support Vector
Machine

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SMO

DSVM

SVM constraints

Machine learning.

- It is the ability for a machine (ie. computer) to learn from past examples and continually adopt to new situations (information).
- Machine learning is currently a challenge (read: opportunity for research) in wireless sensor networks
- Challenges include battery constraints, radio communication, slow processors (compared to PCs), and limited memory available.

Support Vector
Machine

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SVM

DSVM

SVM constraints

Classification.

- Easy and popular algorithms such k -NN, which assigns the same class to a test point as the majority of its n nearest neighbors
- Neural networks were (and perhaps still are) very popular and has been subject to extensive research.
- More recent research is in the area of kernel learners such as the support vector machine.
- Clustering: K -means for example
- Regression analysis like least-square-regression in the simplest form



- The small micro processors does not include a floating point unit, meaning the float data type is a software implementation that is more expensive than usual
- The consequence is that battery, response time or other factors are affected.
- The benefits are of course ease of use and a less error prone implementation



Floating point and fixed point

- Float vs. FP
- FP operations
- Some fixed point operations are easy: addition and subtraction are like we are used to
- Multiplication and division involves bit shifting

Bitwise addition:

```
  0001
+0001
====
  0010
====
```



General Motivation.

- Learning and prediction is/should be an integral part of real-time embedded and distributed systems
- Taking a machine learning approach to intelligent solutions in embedded/distributed systems make design space tradeoffs more rigorous
- A certain class of machine learning algorithms is instance-based and formulated with mathematics/statistics/probability theory (Vapnik)
- Support vector machines are examples of such algorithms
- It is an emerging/maturing field and there are some interesting application areas:
 - Wireless sensor networks: reduce communication
 - Embedded real-time systems: WCET for non-linear predictions

Support Vector
Machine

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SMO

DSVM

SVM constraints

Objective

- Motivate the primary constraints of sensor network nodes
 - Battery
 - Memory
 - CPU
 - Radio
- Many ML algorithms can address constraints
 - Instance based: support vector machines etc.
 - Parametric: expectation maximization etc.
- Tutorial purpose: Use support vector machines as a method to see how one particular class of algorithms maps onto the problem
 - The embedded part
 - The distributed part
 - The (constrained) learning part



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SMO

DSVM

SVM constraints

- Linear and non-linear classification
- Instance-based (a good feature for wireless sensors): It allows selected data to be sent around in a sensor network. You can say that the algorithm is somewhat transparent to the existing application
- Instead imagine a neural network or some other model based algorithm. It would probably not be immediately clear which 10% of the data points to send from one node to the next



Outline

[What is Machine Learning?](#)

[Types of ML](#)

[Implementation issues](#)

[Support Vector Machine](#)

[HW constraints](#)

[SVM basics](#)

[SMO](#)

[DSVM](#)

[SVM constraints](#)

$$f(\mathbf{x}, \alpha, b) = \{\pm 1\} = \text{sgn} \left(\sum_{i=1}^l \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right)$$

- Parameter C controls misclassification behaviour/penalty
- Parameter α controls shape of separating hyper plane

Algorithm 1 Training an SVM

Require: X and y loaded with training labeled data, $\alpha \Leftarrow 0$ or $\alpha \Leftarrow$ partially trained SVM

- 1: $C \Leftarrow$ some value (10 for example)
- 2: **repeat**
- 3: **for all** $\{x_i, y_i\}, \{x_j, y_j\}$ **do**
- 4: Optimize α_i and α_j
- 5: **end for**
- 6: **until** no changes in α or other resource constraint criteria met

Ensure: Retain only the support vectors ($\alpha_i > 0$)



Outline

What is Machine Learning?

Types of ML

Implementation issues

Support Vector Machine

HW constraints

SVM basics

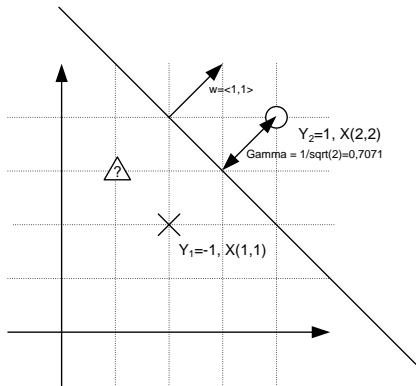
SMO

DSVM

SVM constraints

Margins in an SVM

- Introductory example
 - Binary classification
 - Training data in O and X point
 - Test data in ?
 - Real life test?



- ↔ Margin
- A Vector
- Class 1
- × Class -1
- △ Unknown Class



Outline

What is Machine
Learning?

Types of ML

Implementation issues

Support Vector
Machine

HW constraints

SVM basics

SMO

DSVM

SVM constraints

Sequential Minimal Optimization

Pseudo code for the SMO algorithm (for your information only):

- 1 Outer loop (first choice heuristic): Alternate with a scan of the full data set and multiple partial scans of the non-bound, NB , (not 0 or C) data set.
- 2 For each point—from either scan type—find those that violates the KKT conditions (greater than ϵ), and call inner loop for each such violating point. Terminate the outer loop when all points obey the KKT conditions (within ϵ)
- 3 Inner loop (second choice heuristic): SMO chooses the second point such that numerator in the calculation of α_2^{new} is likely to maximize the step size. For positive E_1 SMO chooses minimum E_2 . For negative E_1 , SMO chooses maximum E_2 .
- 4 If no progress is made from first and second choice heuristics then a new non-bound example is searched for, and if that also fails then an entire iteration is started.



Outline

[What is Machine Learning?](#)

[Types of ML](#)

[Implementation issues](#)

[Support Vector Machine](#)

[HW constraints](#)

[SVM basics](#)

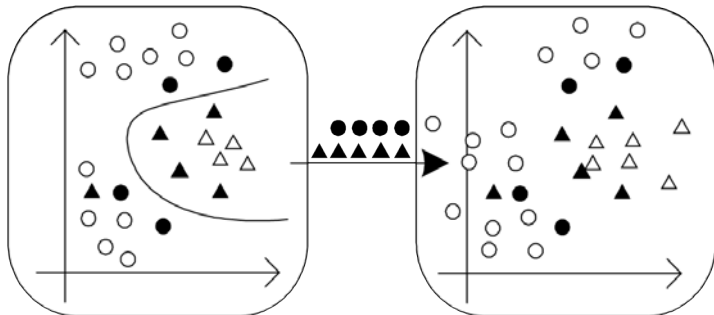
[SMO](#)

[DSVM](#)

[SVM constraints](#)

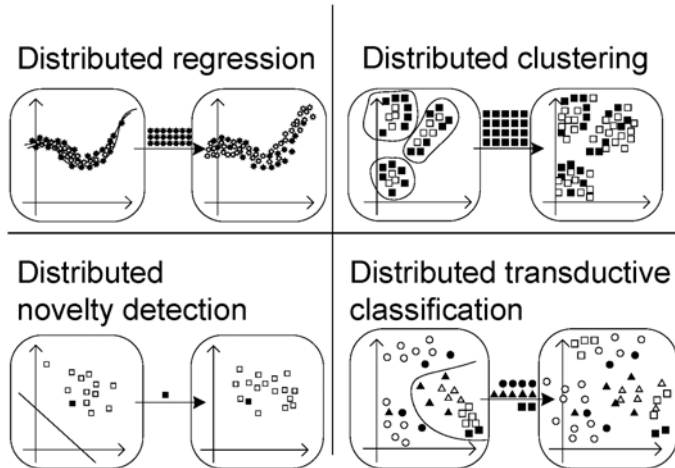
A Simple Distributed SVM

- Distributed classification
- Exchange instances according to some scheme: convex hull, support vectors, max. alpha etc.



More Distributed SVMs

- Distributed regression, clustering, novelty detection and transductive classification
- Exchange instances according to some scheme: convex hull, support vectors, max. alpha etc.



Mapping the SVM to the Constraints

- Battery
 - Small 8-bit CPUs are often used...
 - SVM: Local classification can save on radio bandwidth
- CPU
 - 4MHz compared 1500 MHz...
 - SVM: Switch to linear kernel, co-active training
- Memory
 - KB compared to GB...
 - SVM: You can scale from nomem. footprint to full kernel matrix
- Radio
 - Small bandwidth compared to wired networks. Expensive to use radio
 - SVM: Exchange *important* data with neighbors
- Accuracy
 - Hard-real time, soft-real time or ASAP...
 - SVM: Real-time enables on special operation system



Module EX

Exercises

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

- 1 NEXTMOTE
- 2 Hardware
- 3 Software
- 4 TinyOS
- 5 nesC
- 6 Sensors
- 7 Radio

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NEXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Overview

- 1 NXTMOTE
- 2 Hardware
- 3 Software
- 4 TinyOS
- 5 nesC
- 6 Sensors
- 7 Radio
- 8 Radio

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

- NXTMOTE
- Hardware
- Software
- TinyOS
- nesC
- Sensors
- Radio
- Radio

Preparing the PC

- Install cygwin
- Find the hadat.zip presentations at Sitescape LEGO folder
- Install Textpad
- Place the nesC.syn file in the Textpad systems folder
- Open Textpad and look at the files
- Rename c:/cygwin and unzip the one from LEGO folder to be the default
- Rename c:/cygwin/home/rup.inf to be the user-name of your machine
- Open Cygwin and change to directory \$NMA
- Change into the SVM directory and write `make clean` and `make nextmote`
- Look at the files being created

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Preparing NXTMOTE

- Put the sensors into NXT. Light sensor in port 1, touch in port 2, and leave the mic alone for now. You can use it later
- Put the battery in NXT and reset it. Resetting is done by pressing a little button
- Install the NXTMOTE USB driver when NXT is reset (see picture)
- The USB driver is in `C:/cygwin/home/nxtmote`
- In the SVM directory, write `fwflash.exe build/nxtmote/nxtmote.bin`



Registers

- Find the Timer/Counter block on page 4 of the AT91 manual (doc6175.pdf). How many timer channels are there?
- Look at the bullet "Compare RC Trigger". Where do we set the RC value?
- The RC value is set in `tos.chips.at91.timer.HplAT91OSTimerM` from the file `tos.chips.at91.timer.HalAT91AlarmM`. What is the value?
- The channel control register (see page 405) is set with `AT91C_BASE_TC0->TC_CMR = TC_CLKS_MCK2 | AT91C_TC_CPCTRG`. The `|` operator combines the two constants, but what does the constant `AT91C_TC_CPCTRG` mean?

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Build System

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- Find the file named PlatformP and open it. What is going on in this file?
- Find and analyze the build script named `nxtmote/support/make/at91/at91.rules`.
- What happens in the target named `exe0`?

Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Project overview

- Look around at `www.tinyos.net`. When you find the projects overview, try to think which projects can be inspirational for your own endeavors?
- See the `tinyos-2.x` directory (a few levels back from the `nxtmote` directory on your PC). Can you find the Timer interface (in some subdirectory)? What is the command to start a one-shot timer?

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio



- Try to change into the BTSVM directory and write `make nextmote docs`
- The documentation is placed in `.../nextmote/doc/nesdoc/nextmote`
- Try to browse it. Where is the BTSVM application?

Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

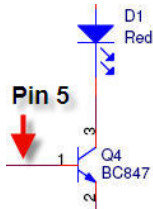
Radio

- The file `nxtmote/tos/chips/at91/timer/HplAT91OSTimerM.nc` contains timer code. Which interfaces are provided? Used?
- Use the `nesdoc.html` to see it
- What is the sequence of events when an interrupt fires and is dispatched by the fired event? Try to follow it in the code.
- It starts in `irqhandler()`
...`nxtmote/tos/chips/at91/HplAT91InterruptM.nc` so you can look there



Light sensor

- Please find the light sensor schematic PDF. It is called Appendix 3-LEGO MINDSTORMS NXT Light Sensor hardware schematic.pdf. Can you see what happens when we set pin 5? How does this relate to the `togglepin(1)` debug function in
- Also find the main schematic and find the two processors: ARM7 and AT48. Can you follow the analog input from the light sensor to the AT48 and then see the TWI (I2C) connection between the two processors?
- What is the struct called that holds the ADC light value (see the LEGO HW manual page 20).



Source: LEGO





- What happens in the message header file `nxtmote/apps/tests/bt/BtRadioCountToLeds.h`?
- Sent and received messages are routed internally based on the active message id. Where is that found in the header file?
- `nxtmote/apps/tests/bt/TestBtM.nc` is the module. Which methods are used when sending a packet? And receiving a packet?

Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Bluetooth radio overview

The goal of the exercise is look at the Bluetooth radio.

- First try to read a little bit about the BT radio in the LEGO pdf files.
- In the file `HalBt.nc` you should look at the contents.
- Look at the hardware specification. Can you see how the BT radio talks to the ARM? For example try to locate the pin that sets one of the modes?
- Then look carefully at the `cCommUpdateBt()` method.
- Try to map the states of this state machine to the LEGO pdf on the Bluetooth radio API.
- Can you see what the contents of the message between the ARM and the BT module is for the `MSG_GET_LOCAL_ADDR` message?
- Now open the `BtRadioCountToLeds.h` file. It shows how the message is structured.
- Try to make a slave by commenting out the `#define MASTER`

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Bluetooth radio application

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- Try to make a slave by commenting out the `#define MASTER` in `BT/TestBtM.nc` around line 48
- Start the slave
- Note down the last two bytes. If it says 5C.F then you write 0x5C0F in the `TestBtM.nc` at line 167
- Also de-comment the `#define MASTER` line
- Start the master

Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Bluetooth radio cont.

- Compile and load on a slave
- Now find the slave address in the code where the master looks: First line in `task void sendit()`
- Challenge: If you want to then try to send a variable (same type as the counters) along with the message. Can you get it to display on the other device? Hint: Change it in the `BtRadioCountToLeds.h` file and in sending and receiving places in `TestBtM.nc`.

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Controlling the Slave Display

- The format of the output is on one of the 8 lines. The lines are numbered 0-7.
- Look at code in the code in the `Receive.receive(message_t* msg, void* payload, uint8_t len)` code in the `TestBtM.nc` module.
- The sensor readings are named `xv1`, `xv2`, and `xv3`.
- The formatting of the string managed with `sprintf` (see <http://www.cplusplus.com/reference/cstdlib/sprintf.html>)
- Try to display the product (multiply) the two readings
- Try to format it so sensor is *Light*: and the other line is *Touch*:

Exercises

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



Outline

NXTMOTE

Hardware

Software

TinyOS

nesC

Sensors

Radio

Radio

Module OE

Support Vector Machine Demo (walk through)

Support Vector
Machine Demo (walk
through)

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



ESWEEK [DRAFT: Final PDF is uploaded after Sunday] 2009

<http://nxtgcc.sf.net>

Rasmus Ulslev Pedersen

rup.inf@cbs.dk

The presentation overview.

Support Vector
Machine Demo (walk
through)

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- 1 How to use the NXT for machine learning (or more correctly support vector machine)
- 2 Support Vector Machine. Simple simplified SMO (sequential minimal optimization)
- 3 Single node

Gliederung

Support Vector
Machine Demo (walk
through)

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



The Algorithm.

- Please have a look at the `TestTSVMM` and `TinySVMM` code. Can you see the similarity between the thesis SVM pages and this?
- Can you identify the place where most of the computation takes place?
- Look at thesis called *dsvm-thesis.pdf* around page 30



Early Stop.

- The TinySVM will sample max NUMDATA data points in NUMDIM dimensions
- The left button samples negative instances and the right button samples positive instances
- You can sample 1,2, or 3 dimensional data with the light, sound, and touch sensor.
- In `TinySVMM.nc` you enter training data in the method `prep()`. The `setVal` method automatically adjusts to the number of dimensions
- Try to make a sample similar to the one in the dsvm thesis
- Try to recompute the alphas. Does it correspond with the results on the screen
- Try to interpret the data on the screen



Early Stop.



- Try to make the SVM run to completion. How long (count kernel evaluations) did it take?
- Now try to enter a third training point? How many kernel evaluations did it take?
- Try to draw a solution on paper and verify it?

SVM Basics

The goal of the exercise is to understand the SVM a little bit.

- Find the SVM directory.
- Take a look at the data structure. You can find it in TestSVMM.nc. It is named `svmdatap` and it is defined in the `tosml.h` file.
- Try to open the `svm.h` file and see how the things are defined. Then try to open the DSVM thesis on page 18 to see what these variables mean:
 - x ?
 - y ?
 - a ?
 - e ?
 - w ?
 - b ?
 - f ?
 - ob ?
 - c ?
 - g ?



Sending/Receiving

The goal of the exercise is to send a packets from the master to the slave and let the slave classify the data point.

- Find the BTSVM directory.
- Then open the Makefile in that directory and comment out the line where MASTER is defined.
- Enter some data values in the `prep()` section for the SVM that makes *sense* for you in relation to the sensor readings.
- Now you can compile the slave by writing `make nextmote` on the command line
- After flashing the slave, then you can read the last two bytes of the BT address. Those two bytes has to be written in the Makefile for the `SLADDR` address. An example is `0xC47A`.
- Then compile the master and flash it. After the search period it will hopefully find the slave and start sending sensor readings to it.
- If you have entered training data correctly then you can create both negative function outputs and positive function outputs. Right?



Extra exercise

Support Vector
Machine Demo (walk
through)

ESWEEK [DRAFT:
Final PDF is
uploaded after
Sunday] 2009
<http://nxtgcc.sf.net>



- If time permits, you can try to implement it such that the kernel matrix for the training matrix is cached
- Try to create a data matrix in TinySVMM around line 19.
- Make a method called `initkernels()` and do the computation here.