Versatile Sensor Node (VSN) platform design of hardware and software

Marko Mihelin



Introduction

- A new WSN node The Versatile Sensor Node
- Designed and developed in collaboration with ISOTEL d.o.o.
- Multiple purpose design, can be used in many different applications
- Open source software







Concept of the Versatile Sensor Node

- Modular base, which can be quickly adapted to various applications
- Powerful micro-controller with a ARM Cortex-M3 core
- Supports basic digital interfaces such as I²C, SPI, UART, IrDA and analog interfaces ADC and DAC
- Dedicated radio module that supports ISM band frequencies (315 MHz, 433 MHz, 868 MHz, 915 MHz and 2400 MHz), Xbee and Bluetooth modules
- Various options for powering the nod







VSN Modules

- VSN comprises of at least two different modules
- The core module VSC supports the micro-controller and power supply
- The radio module VSR houses the low power radio interfaces
- The expansion modules VSE



 The power module – VSP provides additional power supply options







VSC Module

- Used micro-controller is STM32F103 with 512 kB flash, 64 kB RAM and maximum clock of 72 MHz
- Additional memory 128 kB FRAM, miniSD card
- Supported interfaces USB, RS232, UART, IrDA, SPI, I²C, 12 bit ADC, DAC, instrumental differential amplifier with settable gain
- Selectable system voltage (3,3 V, 3 V, 2,3 V)
- Battery charger and solar battery charger with MPPT (Maximum Power Point Tracker)







VSC Module power states

- The micro-controller has a few standard low power states
- Supported low power modes: Sleep, Stop and Standby
- Additional low power mode : Deep Hibernation
- Power consumption in deep hibernation is below 7 μA
- The node can be woken from deep hibernation using RTC alarm
- System state can be recalled from external high speed non volatile memory







VSC module block schematic



VSE Module

- Communication expansion module which supports:
 - Serial to Ethernet (Xport, Xport Pro, XportAR, Connect-Me) converters from Lantronix and Digi with PoE power supply
 - Serial to Wi-Fi (Connect-Wi-Me) converter from Digi
 - Telit GM862
 - Xbee or Bluetooth modules
- Development module with a standard 20 pin JTAG connector, and small protoboard

Full size protoboard for applications itesting Department of Communication Systems

Communication VSE module block schematic









VSR Module

- Low power radio module
- Supports ISM band frequencies from 315 MHz to 2,4 GHz
- Sub GHz frequencies CC1101 from Texas Instruments
- 2,4 GHz frequency band CC2500 from Texas Instrument
- Optionally a Xbee or Bluetooth module can be attached











VSR Module block schematic









Tools and principles used in VSN design process

- For developing two layer PCBs we used CadSoft Eagle Student edition
- For hardware simulation we used SpiceOpus, a free Spice package
- Most of circuits can be designed with the help of application notes and fine tuned with simulation and trial and error







Software development tools

- Professional
 - Keil, full development system
- Open Source
 - Development environment: Eclipse IDE
 - Tool-chain: Sourcery G++ Lite
 - Cygwin, Linux environment for Windows
 - JTAG server: OpenOCD
 - JTAG hardware interface: Olimex ARM-USB-OCD







Eclipse IDE C/C++

C/C++ - VSNDrivers/VSNDriversDemo/main.c - Eclipse

<u>File Edit Source Refactor Navigate Search Project Run Window H</u>elp

🖕 Project Explorer 🛛 📃 🚺	main.c 🔀	street_led.c	💧 Makefile	street_led.h						- (3)
□ 😫 🐨 ▽		debug_str("\r\	nVSN Driver	Test");						1	
> VSNDrivers 79 [VSNDrivers, Branch		/*=====TEST /* SHT11 test	FUNCTIONS==				*/				I
Archives		SHT11_Read_Tem	p_and_Hum(&	SHT11_Tempera	ture[0],&SHT	11_Humidity[0]);				
and Includes		debug_str("\r\	niemperatur	13:"); debug	g_str(SHIII_ tr(SHT11_Hum	idity):					
> OpenOCD 42		acbag_bor((r)	annunitat og 1.	. ,, acbug_	ci (oniii_nuu	idicy/,					
SNDriversDemo 79		/* TSL2561 tes	t				*/			_	
📷 > VSNLib 75		illuminance =	measure_ill:	uminance();						E	
🕞 > FWLib 40		vsn_ledBusyOve	<pre>erride();</pre>								
🔁 include 75		snprintf(fstri	.ng, 40, "\r	nIlluminance	is %f", ill	uminance);					
Car SGLib 18		debug_str(fstr	ing);								
📷 > src 75		vsn_ledkelease	:();								
stm32f10x_it.c 60		/* TTS331 tost					+ /				
stm32f10x_it.h 53	11	van ledBusvOve	rride():								
/ libvsn.a					m						
A PEADAG 18	•	(a) (-	~								4
E Sample Deputet 19	Problems	🖉 Tasks 📮 Console	e 🖾 📃 Prope	erties 🕸 Debug					🐥 🗘 🔄 🛄 👪 🐘 🛛 🖛 🗖 🗸	<mark>° </mark>	
STM32 128K 20K FLASH Id 1	Build [VSND	rivers]									
STM32 256K 48K FLASH Id 1	Internet	rning.	main.out >	main.11St							*
STM32 512K 64K FLASH.Id 1	MS-DOS s	style path deter	sted: E:\IJS	VSN\FW\VSND	rivers						
STM32 COMMON.Id 18	Preferre	ed POSIX equiva	lent is: /cv	gdrive/e/IJS	VSN/FW/VSNDr	ivers					
STM32 SEC EXT.Id 18	CYGWIN e	environment var:	iable option	"nodosfilew	arning" turns	off this war	ming.				
STM32_SEC_FLASH.Id 18	Consult	the user's guid	de for more	details abou	POSIX paths	:					
STM32_SEC_RAM.Id 18	http:/	//cygwin.com/cyg	gwin-ug-net/	using.html#u	sing-pathname	3					
STM32_SEC_RAMonly.ld 18	m-none-e	aabi-objcopy -S	-Rinput-	target=elf -	-output-targe	t=srec -v mai	n.out main.s1	.9			
GODENOCD-ARM-USB-OCD VSND	opy from	`main.out' [elf	f32-littlear	m] to `main.	19' [srec]						
OpenOCD-Signalyzer VSNDrivers	m-none-e	eabi-objcopy -5	-Rinput-	target=elf -	-output-targe	t=binary -v m	ain.out main.	bin		-	-
UCNIDeivere Dahus Jaunch 10	opy from	main.out' [el]	132-11ttlear	mj to main.	oin. [pinarð]					, l	2
						1	1	The second second	3		<u> </u>
						Writable	Smart Insert	149:26	Building Targets: (0%)	10	
								1.			

Communication Systems

- d X

Project build system

- Fully customizable build system based on makefiles
- Requires knowledge of how the building system works
- Provides full control of the build process



http://imgs.xkcd.com/comics/compiling.png







The ST Standard Peripheral Library

- We use the ST Standard Peripheral Library V3.1.2
- The library deals with register and bit peripheral access so we don't have to
- The library provides macros for various register addresses and states
- All other software including the device drivers are build on top of this library







Driver development general pointers

- The STM32F103 Reference manual is your best friend
- The sensor/device datasheet is your second best friend
- If your best friends fail to provide you with enough information ask Google
- Have patience







Driver development procedure

- 1) Identify the sensor/device interface
- 2) Connect the sensor/device to appropriate pins on VSN
- Initialize the identified clocks and peripherals of the microcontroller
- 4) Write a test driver to check if all is working correctly
- 5) Complete the driver, use interrupts and DMA to maximize the performance







Thank you for your attention.

Questions?





